RFID-Based Real-Time Production Monitoring in a Variant Production Environment

Von der Graduate School of Excellence advanced Manufacturing Engineering
der Universität Stuttgart
zur Erlangung der Würde eines Doktor-Ingenieurs (Dr.-Ing.)
genehmigte Abhandlung

von

Bilal Hameed

aus Karachi, Pakistan

Hauptberichter:	Prof. Dr. rer. nat. Dr. h. c. Kurt Rothermel
Mitberichter:	Univ.-Prof. Dr.-Ing. Prof. E.h. Dr.-Ing. E.h. Dr. h.c. mult Engelbert Westkämper i.R.
Mitberichter:	Prof. Dr. Winfried Lamersdorf, Universität Hamburg

Tag der mündlichen Prüfung: 9. Dezember 2016

Institut für Parallele und Verteilte Systeme (IPVS)
der Universität Stuttgart
2016

Acknowledgments

Firstly I would like to thank Allah for giving me the strength, courage, intelligence and most of all the patience to undertake this endeavour i.e. the Phd Project. The past five years of my life have been a humbling experience for me.

I would like to specially thank Prof. Dr. Kurt Rothermel for selecting me to conduct research under his supervision. For Prof. Dr. Kurt Rothermel, many candidates might have fulfilled the research position, but for me, this might have been the only opportunity. So I owe him more gratitude and thanks then I could write in words here. The selection was not the only thing Prof . Dr. Kurt Rothermel did for me, his guidance, foresight and support played an equally important role in the identification of the research problems and development of their potential solutions during the course of my doctoral studies. Moreover, I would like to thank Prof. Dr. Engelbert Westkamper for taking the time to review my dissertation.

I specially want to thank my mother, Nusrat Naheed, who bore the pains and sorrows of living so far apart from his son, who sacrificed so much so that I could be what I am today and what I will be tomorrow. I also want to thank my father, Hameed Alam, who raised me to be what I am, a person with an extremely soft and receptive heart towards the sufferings and ailments of others by always putting my needs in front of his own. I also want to thank my wife, Dr. Erum Saba, who suffered both emotionally and physically as a result of living apart from me for these four and a half years, just so that I could grow a little more intellectually. I am extremely thankful to her for taking such good care of my kids, while I was in Germany working on my doctoral project. I am also thankful to my brother, Dr. Sufian Hameed, who has always been there to share the sorrows and joys of my life, and who is also responsible for reviewing this dissertation.

Apart from my family, continual moral and technical support of several individuals has made this dissertation possible. I am inclined to name some of these wonderful individuals below.

Dr. Frank Durr, my project supervisor. If I would only have to say one sentence, I would say that this work would not have been possible if not for Dr. Frank. He has

always been available for discussions, whether it was a research dialogue, a programming tip, a documentation hint, a resource allocation problem like hiring a hiwi, a paper review and so on. In short, Dr. Frank was my go to guy for everything inside the department.

Throughout my doctoral studies, I have had the opportunity to work with some wonderful people as colleagues. The support that I received from the following people has made me think of them as friends rather than just co-workers. Dr. Faraz Memon and Dr. Adnan Tariq, who have been my friends for almost ten years now guided me throughout my stay at the department of distributed systems, or throughout their own stay - as is the case with Dr. Faraz Memon. Brilliant programming and problem solving skills of my Master thesis students Farhan Rashid and Eid Badr, and that of Imran Ahmed Khan, who was both a friend and a Hiwi, helped me immensely in my research work. Gerald Koch, Marco Volz and Stefan Foll provided their valuable feedbacks on the research papers that I submitted for review at the international conferences.

To all the people mentioned above, and to the unmentioned ones who helped me in any way, I am thankful from the bottom of my heart.

In the end I want to dedicate this dissertation to my teacher and mentor Moulana Muhammad Akram Awan, for his teachings and guidance have transformed my personality and character and have made me the individual that I am today. To him, I owe eternal gratitude and thankfulness.

This research was supported by Graduate School for Advanced Manufacturing Engineering (GSaME). GSaME is a central scientific institution of the University of Stuttgart and has been funded by DFG and industrial partners under the German Excellence Initiative since 2007.

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

AutoID	Automatic Identification
RFID	Radio Frequency Identification
CE	Complex Events
CEP	Complex Event Processing
VR	Virtual RFID Reader
PR	Physical RFID Reader
PubSub	Publish Subscribe
EPC	Electronic Product Code
DB	Database
ONS	Object Name Service
DNS	Domain Name Service
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
GUI	Graphical User Interface
DoD	United States Department of Defense
PBL	Performance Based Logistics

List of Symbols

pr	Physical Reader
vr	Virtual Reader
e	Read Event
o	Product Part
$p(e)$	Probability of Read Event
$p(pr_{act})$	Actual Probability of a Physical Reader
$p(pr_{est})$	Estimated Probability of a Physical Reader
DPL	Detected Parts List
ps	Partial Sequence
es	Extended Sequence
PPL	Product Part List
PSL	Partial Sequence List
PSL_g	Global Partial Sequence List
ESL	Extended Sequence List
ttl	Time to Live
$p(ps)$	Probability of Partial Sequence
$p(es)$	Probability of Extended Sequence
ppo	Product Part Order
PPO_{pln}	Planned Product Part Order
PPO_{act}	Actual Product Part Order
PP_{pln}	Planned Product Parts
$OOPL$	Out of Order Parts List
MPL	Missed Parts List
FPL	False parts List
ce	Complex Event
$p(ce)$	Probability of Complex Event
ce_{seq}	Sequence Error Complex Event
ce_{syn}	Synchronization Error Complex Event
ce_{del}	Delay Error Complex Event
ce_{mpp}	Missing Part Error Complex Event
ce_{ipp}	Incorrect Part Position Error Complex Event
$SeqList$	Sequence List
$SynList$	Synchronization List
$DelList$	Delay List
$AsmList$	Assembled Part List
$IppList$	Incorrect Part Position List
$p(ce_{seq})$	Probability of Sequence Complex Event
$p(ce_{syn})$	Probability of Synchronization Complex Event
$p(ce_{del})$	Probability of Delay Complex Event
$p(ce_{mpp})$	Probability of Missing Part Complex Event
$p(ce_{ipp})$	Probability of Incorrect Part Position Complex Event
ce_{thd}	Complex Event Threshold

Abstract

Manufacturing organizations around the globe are now leaning towards just-in time or customized production in order to gain competitive advantage in an ever more intense marketplace. The problem with customized or variant production is that, organizations have to track the different product parts precisely in order to ensure that the right part is at the right place in the right amount at the right time, to deliver customized products to their customers. Due to this reason, companies are now increasingly using RFID technology to track production in real-time.

The wide spread deployment of RFID technology for production monitoring and tracking is impeded by the fact that the technology is inherently unreliable and RFID readers suffer from errors such as duplicate, false, missed and out of order readings. In this work we have presented algorithms to a) enable real-time tracking and monitoring of product parts on the production lines, b) provide probabilistic guarantees to the real-time product parts that are being tracked c) enable the RFID readers to self-calibrate their reader probabilities so that the readings that they generate are highly reliable at all times d) generate complex manufacturing events and provide probabilistic guarantees for the accuracy of these complex manufacturing events. In particular the following contributions are made in this thesis.

As a first contribution, we developed a consistency stack that conceptually divides the different consistency/reliability issues in production monitoring into separate layers. In addition to this we have built a consistency management framework to ensure consistent real-time production monitoring, using unreliable RFID devices. Secondly, we deal with the problem of detecting object sequences by a set of unreliable RFID readers that are installed along production lines. We propose a probabilistic sequence detection algorithm that assigns probabilities to objects detected by RFID devices and provides probabilistic guarantees regarding the real-time sequences of objects on the production lines.

Thirdly, we developed a probabilistic model to assign probabilities to the RFID readers and to the product part detections. We also present a probability self-calibration algorithm that automatically adapts the probabilities of RFID readers to better reflect

their performance at current instance of time. This would ensure that unreliable RFID devices would have little or no say in the overall production monitoring and tracking within the production environment.

The use of RFID technology in manufacturing and production is still limited because of the non-availability of middleware solutions to transform raw RFID data into higher level meaningful information. So as our fourth contribution, we present a complex event processing framework that can be deployed in manufacturing environments. The framework is capable of processing raw RFID events to generate complex manufacturing events that are of relevance to the production operations. The framework assigns probabilities to each complex event, which are continuously updated as more information is made available regarding these events. This provides a measure to the higher level applications about how accurate or inaccurate a certain complex event really is.

Zusammenfassung

Die Just-in-time-Produktion und die Fertigung kundenspezifischer Produkte sind wichtige Anforderungen an Unternehmen, die in einem hoch kompetitiven globalen Markt agieren. Die variantenreiche Serienproduktion erfordert die präzise Verfolgung des aktuellen Ortes von Werkstücken und Produktteilen während des Produktionsprozesses um sicherzustellen, dass die richtigen Teile zum richtigen Zeitpunkt in der richtigen Menge am richtigen Ort sind. Daher setzen Firmen zunehmend RFID-Technologie ein, um Teile in Echtzeit während der Produktion zu verfolgen.

Der weitverbreitete Einsatz von RFID-Technologie fÃ¼r die Produktionsüberwachung und Verfolgung von Teilen wird durch die technologiebedingte Unzuverlässigkeit der funkbasierten RFID-Sensoren erschwert, die zu Fehlern wie die mehrfache Detektion von Teilen (Duplikate), nicht detektierte Teile, falsch erfasste Teile (z.B. von benachbarten Produktionslinien) und Reihenfolgefehlern führt. In dieser Arbeit stellen wir Algorithmen zur Behandlung dieser Fehler und Verbesserung der Zuverlässigkeit vor, die (a) eine Verfolgung von Teilen entlang von Produktionslinien in Echtzeit mit Hilfe von RFID-Lesern erlauben; (b) probabilistische Aussagen bezüglich der durch eine Menge von RFID-Sensoren gemachten Beobachtungen ermöglichen; (c) RFID-Leser automatisch kalibrieren, um so deren Zuverlässigkeit zu ermitteln und durch redundante Beobachtungen die Qualität der Beobachtungen des Systems zu verbessern; (d) die Ableitung komplexer Produktionsereignisse aus einfachen Beobachtungen ermöglichen und für diese komplexen Ereignisse probabilistische Aussagen zu deren Korrektheit geben. Im Einzelnen leistet diese Arbeit die folgenden Beiträge.

Als erster Beitrag wird ein Schichtenmodell (Konsistenz-Stack) vorgeschlagen, das die verschiedenen Zuverlässigkeits- und Konsistenzaspekte der Produktionsüberwachung in aufeinander aufbauende Schichten einteilt. Des Weiteren wird ein Rahmenwerk fÃ¼r die Produktionsüberwachung mit Hilfe unzuverlässiger RFID-Sensoren entworfen, das konsistente Beobachtungen ermöglicht.

Als zweiter Beitrag werden Konzepte und Algorithmen zur verteilten Beobachtung von Objektsequenzen entlang von Produktionslinien mit Hilfe einer Menge unzuverlässiger RFID-Sensoren vorgeschlagen. Hierzu wird ein probabilistischer Algorithmus zur Erfas-

sung der Sequenz von Objekten entworfen, der einzelnen Beobachtungen Wahrscheinlichkeiten zuweist und aus mehreren Beobachtungen die Reihenfolge der Objekte ableitet, zusammen mit einer Wahrscheinlichkeit für die Korrektheit der ermittelten Sequenzen.

Drittens wird ein probabilistisches Modell zur Modellierung der Zuverlässigkeit von RFID-Lesern und der ermittelten Objektpositionen vorgeschlagen, sowie ein Verfahren zur automatischen Kalibrierung dieser Wahrscheinlichkeiten. Hierdurch soll insbesondere der negative Einfluss einzelner unzuverlässiger RFID-Leser vermindert und somit die Konsistenz der Beobachtung mit Hilfe mehrerer Leser verbessert werden.

Als vierter Beitrag wird in dieser Arbeit ein Rahmenwerk fÃ¼r die Ableitung komplexer, aussagekräftiger Produktionsereignisse aus Sensorbeobachtungen entwickelt, das insbesondere einzelne Beobachtungen von RFID-Sensoren zu komplexen Ereignissen verknüpft, um somit die Fertigungssteuerung zu unterstützen. Komplexe Ereignisse, die aus unzuverlässigen Sensorbeobachtungen abgeleitet werden, werden dabei wiederum mit Wahrscheinlichkeiten verknüpft, um Anwendungen die Bewertung der Korrektheit erkannter komplexer Ereignisse zu ermöglichen.

Chapter 1

Introduction

Industry is the cornerstone of growth and development in the modern world and production plants and factories form the basic building blocks on which the industry stands. Manufacturing has gone through a lot of improvements during the past 100 years. In the early nineteenth century when Henry Ford developed the first assembly line [For07], everything on the line was done manually. Recent advancements in technology have also made their way into factories and as a result new and innovative tools and ideas have revolutionized modern production environments.

Due to ever increasing competition, manufacturing organizations are trying to gain competitive advantage by either positioning themselves as a niche producer or by developing and designing mass market and yet custom-made products. These build-to-order products allow the users to tailor the products to their hearts content. Several manufacturers have now made the transition from making batch products to developing build-to-order variant products. Some of the prominent variant manufacturers include BMW, Porsche, Toyota and Daimler in the automotive sector and HP, Dell, and IBM in the PC and consumer electronics sector.

Dell has setup an online portal, where anyone can go and custom built its own PC, which would then be assembled and ship by Dell in almost the same time as it would take to buy a generic PC or laptop from any other vendor. Even among the mass market competitors vast differences exist in the number of product variants that the company is able to offer to its customers. As an example, Toyota offers 448 different variants for Toyota Yaris as compared to 176,576 variants offered by VW for VW Polo. Porsche on the other hand offers its customers 10^7 different variants to chose from [Kho11]. Once a customer precisely selects what he needs in his car, he would have to wait for around three months for Porsche to deliver his car.

In order to ensure correct, optimized, and error free variant production, Porsche is using a set of error prevention techniques during both the production planning and assembly process. During the planning phase similar and closely related cars are arranged to-

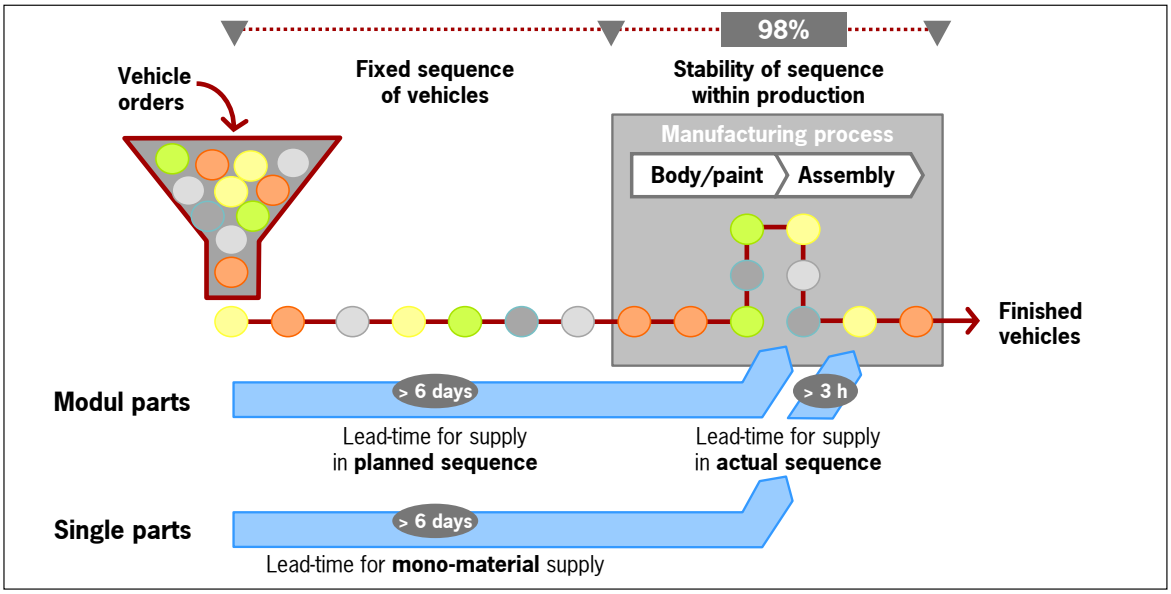


Figure 1.1: The Build-to-Order production system of Porsche based on the pearl-chain principle [Kho11]

gether. This ensures that the highly variable custom requests have a certain amount of discipline and order. This pre-assembly phase greatly simplifies the assembly process, as workers would now have to carry out similar tasks on a series of semi-assembled cars instead of performing different tasks on each and every car.

During the assembly process, Porsche utilizes barcode technology and a customized lighting system to prevent assembly errors. There are numerous assembly points within the factory. At each and every assembly point, certain defined parts are assembled together with the semi-assembled vehicles. Each semi-assembled vehicle has a barcode tag attached to it. The parts that are to be assembled are arranged on shelves (or come in through conveyor belts/production lines). Every shelf has a variably coloured light bulb attached to it. When the semi-assembled car reaches the assembly point, a worker uses a barcode reader to read the tag on the semi-assembled car. Since the specification for the car is written on the barcode, the colored bulb on a specific shelf lightens up to indicate to the worker that the vehicle be assembled using a part from this specific shelf. This barcode based system is aimed at preventing assembly errors from happening.

In addition to this, Porsche is also using techniques to discover errors within a maximum of five minutes after an assembly procedure. At certain assembly points, the parts to be assembled are placed within special carts before the arrival of the semi-assembled cars. These carts should become empty i.e. all items within the cart should be assembled with the incoming car. In case a cart is not empty after five minutes i.e. items in it were not consumed, an error message is generated which would inform the factory that

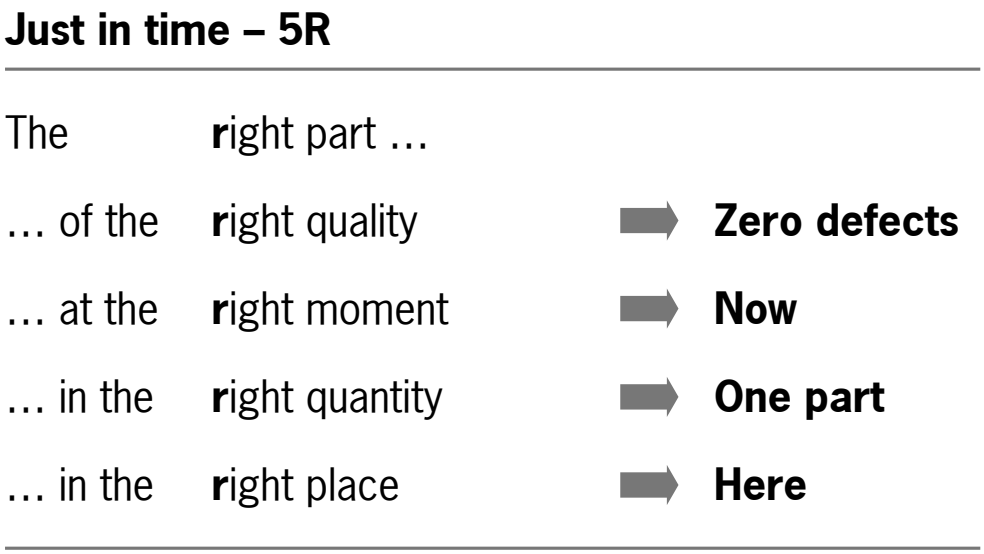


Figure 1.2: The Just-in Time or Build to Order Vision [Kho11]

an incorrect or incomplete assembly has taken place within the last five minutes at that specific assembly point. Figure 1.1 shows the production system used by Porsche in a graphical manner.

Even with the utilization of technologies such as barcode, workers get physically and mentally stressed out due to the nature of repetitive tasks. Moreover, owing to the huge variety of products, it becomes difficult to ensure 5Rs (i.e. to have the right part, of the right quality, at the right moment, in the right quantity, in the right place) for variant or just-in-time production (cf. Figure 1.2). Implementing the 5Rs rule for just-in-time production is easier said than done. Various errors emerge during assembling parts to their respective products, such as sequence and synchronization errors. In a sequence error, product parts move on the assembly line in an incorrect order where as in a synchronization error the semi-assembled cars reach the assembly point but the parts that have reached the assembly point belong to other semi-assembled cars. The large number of variants makes managing the state of the product really tough. Furthermore, a worker might forget to scan the product part, or he might forget to press the button to indicate the completion of assembling a specific part.

Although barcode technology provides certain advantages when it comes to monitoring the production processes, it has a lot of limitations such as the need for extensive human interaction and line of sight readings, which are also limited to single reads at a time.

Due to the limitations posed by barcode technology, manufacturers are now switching over to radio frequency identification (RFID) systems. RFID technology can be used to detect the presence or absence of objects and to know their relative positions within the production environments. The system generally comprises of two main components: a

transponder (tag) and an interrogator (reader). To make an object identifiable, a tag is attached to it. Tags are of two types: passive and active. A passive tag does not need a power source. It uses a coupling element to obtain power from the reader. An active tag on the other hand has its own voltage supply (a battery) and gets power from it. It also contains an electronic microchip that is used for processing and storage. The RFID reader reads identification data from tags and could also modify the data currently stored on these tags. The utilization of RFID systems has been picking pace gradually. The market size for passive RFID technology was $874 million in 2011 and is expected to reach $3.9 billion by the end of 2016, which would represent a compound annual growth rate (CAGR) of 29.3% from 2011 to 2016 according to a recent market research [Inc11]. The prime reason for the exponential growth in the passive RFID market segment is the expectation that the price of a passive RFID tag would soon be reduced to 1 cent per tag.

The main advantages of RFID over barcode technology in variant manufacturing is that RFID provides features such as contactless and non-line of sight identification, multiple tag reads, seamless interfacing with higher level processing devices (PCs) etc., that are essential for real-time production tracking. Due to these advantages, Ford [Joh02], BMW and Vauxhall [BL97] [ZGP04] have deployed RFID based systems, whereby they place a programmed RFID tag on a vehicle skid. During each production step, the tag is read to determine what needs to be done and the updates are automatically written to the tag. This results in eliminating paper work and the associated human errors.

The existing RFID deployments in industry only track the products and not the individual parts. Due to this, the manufacturer is able to find out about an error (such as an incorrect product assembly) but remains oblivious to the reasons for the error (delays, product parts out of sequence etc). In the sections below we would take a detailed look at some of the specific problems facing the manufacturing industry regarding variant production monitoring and the solutions that we propose to address these issues.

1.1 Research Objectives

This thesis focuses on three problem areas in factories that are an impediment to variant production.

1.1.1 Consistent Real-Time Production Monitoring

In customized or variant production environment, the production process remains the same. However, the parts moving through the production lines vary, which results in different variants of the same product. In such an individualized production scenario

it is essential to ensure that product parts move through the production lines in a desired sequence. A disruption in the correct sequence of these parts will produce an undesirable final product, which will not be consistent with the production plan.

RFID readers can be used for real-time production monitoring by putting RFID tags on the product parts and installing RFID readers on the production lines. However, RFID readers have a reliability of 80-90% [AKFR07], which is further affected by environmental factors such as presence of metal objects, interference from multiple readers, or the presence of multiple tags. In case a reader attempts to detect more than five tags, its reliability drops to 70% [PSR+06], [Vio05], [HC06].

Missing out on product parts will result in different readers on the same production line detecting different product part sequences. As an example, consider that three product parts o_1, o_2 and o_3 pass through the production line with o_1 being the first product part and o_3 being the last. An RFID reader missing out on o_3 will assume that the product part sequence is o_1, o_2, whereas another RFID reader that misses out o_2 will assume that the product part sequence is o_1, o_3. Missing out on detecting the product parts is not the only consistency issue. RFID readers can detect product parts incorrectly, overshoot and detect product parts passing through neighboring production lines, and can have duplicate readings. One of the many focus(es) of our work is to deal with these consistency issues that arise from unreliable RFID readers in real-time production monitoring.

In recent times, the RFID community has proposed several middleware systems for deploying RFID devices. Savant [CTCC03], HiFi [FJK+05], RFIDStack [FL05] are primarily focused on efficient handling and querying of large amounts of RFID data. WinRFID [PSR+06] on the other hand is concerned with data availability.

The BRIDGE project [IAM09] has proposed concepts for supply chain data consistency and use rule-based data analysis techniques to monitor the RFID data generated throughout the supply chain for inconsistencies. RF2ID [AKFR07] tries to address the unreliability of RFID devices by deploying multiple RFID readers along the path of object movement. Bauer et al. [BJS04] have proposed a solution to monitor the location of specialized mobile tools, which must be fitted on manufacturing machines for assembling customized products. However, till now there exists no solution to monitor the state of production, and detect the product parts and their sequences as they move through the production lines.

We propose a system tailored to the monitoring of product part sequences by a set of unreliable RFID readers (cf. Chapter 5). In order to accomplish this we have designed a novel *production path* abstraction, which models the flow of product parts on the production lines. Several RFID readers are deployed on each production path, which detect product parts as they move through these paths. The redundancy in RFID reader deployments together with the knowledge of production paths and product part routes is used to detect the consistency issues in production monitoring.

1.1.2 Self-Calibration of RFID Reader Probabilities

In our initial approach we detect product parts and the sequence of these product parts as they move across the production path. The product part and sequence detections also have associated probabilities. However, our initial approach assumed that RFID readers would have fixed probabilities of correct readings which are known a-priori. This does not reflect reality, since the reliability of an RFID reader can change over time or can differ from the manufacturer specified reliability levels even at the deployment time.

As a matter of fact our experiments revealed that RFID readers of the same model, developed by the same manufacturer deployed under same physical conditions perform differently from one another.

A lot of scientific effort has been made in order to improve RFID reader reliability. The inability of an RFID reader to detect all the tags occurs due to collision between multiple tags, all of which try to send data to the RFID reader at the same time. Several techniques have been developed overtime to solve this issue. These techniques can be broadly divided into two categories: probabilistic identification methods [Sch83], [EPC04], [Vog02], [ZKS04], [CK05], [FW06], [Flo06], [BKS05], [LJL05] and deterministic identification algorithms [CCK04], [SRSMN06], [CS88], [NC05], [ZCJ+04], [HW98], [BR05b], [LLS00], [ML05], [GFL87], [PFP04]. In probabilistic identification, all the tags transmit at the same time, whereas in deterministic identification the reader pre-determines a certain number of tags to send their ID to the reader. This pre-determinism leaks a lot of information and hence is not preferred due to security reasons.

We have tried to address the same issue, i.e. to increase the reliability of RFID readers, albeit from a different angle (cf. Chapter 6). Instead of ensuring that a reader reads all the tags, we assign probabilities to RFID readers and then calibrate these probabilities over time to reflect how reliably the reader reads all the tags. So overtime an RFID reader's probability would tell us how many tags it would detect and how many tags it would miss. This real-time measure of the reliability of RFID readers allow us to assign appropriate weightage to the readings of all RFID devices deployed within the production environment.

1.1.3 Reliable Complex Manufacturing Event Processing

Another important aspect of reliable variant production is the ability to detect manufacturing errors in real-time. By doing so the manufacturing organization is quickly able to identify whether or not the products are being assembled according to the customers' orders.

Till now, we only focused on detecting basic RFID events and ensuring that these events are as accurate as possible. However, such basic events are of little or no benefit in a

variant production environment as it provides no higher level meaning and awareness about the state of production. In order to resolve this issue, we identified complex higher level events/errors that are of importance to a production environment. These complex errors include: a) sequence errors, b) synchronization errors, c) delay errors, d) incorrect part position errors, and e) missing part errors.

Several RFID based CEP systems [ZZ08], [HYHZ08], [WDR06], [WLLB06] have been proposed by the research community. These systems differ from our endeavour in two important ways. Firstly, the prime objective of these systems was to develop a CEP based system for processing RFID streams. Although we are also detecting complex events, these events cant be detected using off the shelf complex event processing systems. The first and foremost contribution was to develop algorithms to detect the complex manufacturing events mentioned in the previous paragraph. In addition to developing algorithms to detect these complex manufacturing events, we also designed a system to assign probabilities to these complex events/errors so as to provide probabilistic guarantees to these complex events. As of now, ours' is the only work that endeavours to provide reliability guarantees for complex events.

The information regarding the accuracy of complex manufacturing events can then be used by higher level applications to decide whether to act upon a certain event or not.

1.2 Contribution

The main contributions of the work performed as a part of this PhD thesis are the following:

1. We have presented several case studies regarding how manufacturing organizations are transitioning from developing products to delivering services. After presenting these case studies, we took a detailed look at some of the different components or building blocks that need to be developed and deployed in order for a variant production environment to have reliable and error free variant production [HMW+11]. These building blocks essentially serve as requirements for our work.

   With respect to [HMW+11], my colleagues from GSaME: J. Minguez, M. Wörner, P. Hollstein, S. Zor, and S. Silcher contributed the writeup of their respective research projects which form different components within the smart factory. The abstract, introduction, related work and discussion of my personal work has been my personal contribution.

2. We propose a consistency stack for RFID based production monitoring middlewares. The consistency stack is conceptual in nature and categorizes the different consistency issues into separate layers. By presenting a consistency stack we have formalized the consistency issues, such as duplicate readings, missed readings, and

false readings etc., that need to be considered by almost all RFID applications. In addition to the RFID consistency issues, the stack also formalizes the production consistency issues, such as sequence and synchronization errors [HKDR10].

3. We have designed a production path/line based system model to provide probabilistic guarantees regarding real-time production monitoring. In addition to this, we have also developed an algorithm to detect object sequences and assign probabilities to these sequences. Furthermore, we exploit redundancies in the deployment of RFID readers to increase the confidence in the detected sequences [HKDR10].

   With respect to [HKDR10], Imran Ahmed Khan worked as a Hiwi student on the simulation prototype to evaluate the concepts. The design, and refinement of the concepts and the research paper that was eventually published as a result of this effort has been my personal contribution. During the paper review phase, Dr. rer. nat. Frank Dürr provided valuable feedback that greatly helped in finalizing and refining the probabilistic model. In addition to this, Dr. rer. nat. Faraz Ahmed Memon and Gerald Koch reviewed the research paper and provided valuable feedback.

4. We also designed and developed an efficient and scalable self-calibration algorithm. The self-calibration algorithm automatically calibrates the probabilities of RFID readers deployed in the factory to reflect the reliability with which these readers are detecting product parts at any given instance of time [HRDR12].

   With respect to [HRDR12], MSc. Farhan Rashid contributed towards implementing the prototype to evaluate the concepts of the self-calibration algorithm within his masters thesis. The design and development of concepts and the research paper that eventually came out of this effort was my personal contribution. During the paper review phase, Dr. rer. nat. Frank Dürr provided valuable feedback and also presented the work at the Pervasive conference which was held in United Kingdom.

5. We identified several complex events that are of interest in a manufacturing environment. After that we designed and developed algorithms to detect each of these complex manufacturing events. In addition to this, we formulated a probabilistic model to assign probabilities to the detected complex manufacturing events so that the factory workers and staff can know with certainty how reliable a certain complex manufacturing event is. The complex event processing system is not limited to a single source for receiving data and can fuse data coming from a diverse array of sensors and generate complex manufacturing events reliably.

   Philipp Riffelmacher provided detailed overview on the working of the Lernfabrik [Rif13]. Later on Max Dinkelmann provided valuable feedback on the concepts for complex manufacturing events and supported the evaluation of RFID readers within the Lernfabrik by scheduling access time to the Lernfabrik for me and

my master's thesis student Eid Badr. Without the valuable support of Philipp Riffelmacher and Max Dinkelmann, the RFID tests conducted at the Lernfabrik would not have been possible.

1.3 Structure of the Thesis

The rest of the thesis is structured as follows. In Chapter 2 we discuss the notion of product service systems and show how a variant production environment can actually provide certain informational services to both the producers and the consumers. We also highlight the different components that need to be developed in order for the variant production environment to offer the services under discussion. Chapter 3 addresses the system model on which the entire real-time production monitoring infrastructure was built. Chapter 4 presents the Lernfabrik, which is a production testbed that was used to evaluate the reliability of RFID readers. Chapter 5 presents the RFID based consistency management framework that can be used to monitor production in factories in real-time. Chapter 6 describes the self-calibration algorithm, which is used to self-calibrate the probabilities of RFID readers that are deployed to monitor production processes in a factory. The complex manufacturing events are presented in Chapter 7 along with the algorithms to detect these events. Finally, the summary of our contributions and an outlook onto future work is presented in Chapter 8.

Chapter 2

Case Studies: From Production based Manufacturing to Services based Manufacturing

For most of the twentieth century, a high quality product or lower prices were sufficient to differentiate amongst competitors. However, due to globalization, tear down of import duties, and saturation of markets, the possibility to distinguish on the basis of a product alone has become impossible. Products have converged remarkably during the past decade in terms of quality, technology and pricing. Due to this reason, companies are now facing ever increasing competitive pressures and decreasing profit margins [BLE+07].

One way out of this competitive situation is to integrate products with services and offer them together as Product Service Systems (PSS). The bundling of products and services can result in fulfillment of customer needs and establishment of intensive long term relationships with the customers at the same time. Such a scenario creates competitive advantage and additional barriers for competitors to encroach on a company's customers. In addition to this, research has proved that a PSS offering can extend the product life cycle considerably, resulting in continued revenues and more sustainable operations [SD03].

The most common example of a PSS is a leasing service that companies offer to their customers. The service allows the customers to lease the product (laptop, automobiles, etc.) instead of paying cash out right for the product. The leasing service has become such an integral offering especially in the automobile industry that almost all major automobile manufacturers now have a financial arm that offers leasing service for customers [Vol94], [Toy00], [GM92], [For59].

However, product services are not limited to financial services alone. Informational services form the other main component of services that companies frequently offer to the customers. Examples of information services include: information of ingredients mentioned on food items, and constant tracking of parcels that is available for postal

services like DHL and FedEx etc. Product warranties are another example of service offered to customers, which allow users to buy products with an assurance that if anything goes wrong in a certain defined time-frame, the customer wont lose out on his investment in the product.

When it comes to product service systems, much of the attention has been focused on bundling services with the product itself, whereas the entire process of manufacturing has remained a black box inaccessible to the customers. Once a customer orders a product, he has no way of finding out if his product has been planned, and how long it will take for his product to be assembled. If the customer is able to get real-time information about the product that he has ordered, the entire process of manufacturing would itself be transformed into an information service for the customer. We believe that today's state of technology can transform factories into smart environments that can provide real-time production information not just to the producers but to the consumers as well.

In this chapter, we will discuss the different building blocks (such as real-time production monitoring, RFID based complex event processing, etc.) of a smart variant production environment that would be capable of informing its customers about the status of their products in real-time. In addition to discussing the building blocks of the factory we would also highlight the different services that such a futuristic factory can offer to both the company itself and its customers [HMW+11].

The rest of the chapter is structured as follows: in Section 2.1, we take a look at different services case studies in the manufacturing domain. In Section 2.2, the notion of an RFID-based smart variant production environment is presented along with the different components that are needed to create such a smart manufacturing environment. The different services that can be offered by such a smart variant production environment are then discussed in Section 2.3. In Section 2.4 we have summarized the numerous requirements for creating a smart variant production environment.

2.1 Product Service Systems: Case Studies

The entire premise of this chapter is that factories should not just build and deliver products but instead should take a leap forward and deliver services as well. Before dwelling into what services facories can deliver and how they can deliver them, it might be pertinent to have a look at some case-studies where-by product offerings were transformed into service offerings or normal product buying agreements were modified to depict sales of services. In Section 2.1.1 we would take a look at how the US Department of Defense DoD is now structuring its normal procurement contracts to be performance-based or service-based contracts. Section 2.1.2 takes a look at the Power-by-Hour contracts of Rolls-Royce, which is one of the leading aircraft engine manufacturers in the world. The section will also explain how the company has transformed

itself from selling aircraft engines to selling services that ensures engine availability and uptime for aircrafts. Section 2.1.3 takes a look at Car2Go, a car sharing service that is literally transforming Daimler AG from a producer of cars to a provider of mobility services.

2.1.1 Performance Based Contracting at U.S. Department of Defense

The US Department of Defence (DoD) is now increasingly shifting towards Performance based Contracting (PBC) [KCN07], whereby the department would only be paying for the quality or availability of a certain service. The basic idea of performance based contracts is that you pay for the performance outcome and not the individual parts and repairs that are delivered to you. Instead of buying a certain product and a set of spare parts, a certain no of repairs etc, a buyer buys a predetermined level of availability of the product.

In layman's terms you pay for an equipment when it works and dont pay when it doesn't. If the idea catches fire, we might soon be living in a world where we would pay for our car, only when it works. The same would be true for all other commodities that we use but do not consume such as the Television or the Toaster.

Prior to PBC, the DoD mostly relied on fixed-contracts - whereby a buyer pays a fixed pre-determined price to purchase spare parts and support and maintenance services or cost-plus contracts - whereby a supplier repairs the product that he has sold and charges the entire cost plus a certain premium as service charges.

The problem with both of these models is that both of these contracts do not align the objectives of the buyer with the objectives of the supplier. As an example: when a part of a product breaks, there is no incentive for the supplier to repair it at the lowest possible cost, since everything would be reimbursed. Furthermore, the supplier also has no incentive in repairing the product in the least possible time frame.

The US Department of Defence initiated 57 PBL programs in 2002. Because of the availability guarantees that are inherent in performance based contracts, these contracts were an astounding success over the previous prevailing models. In August 2004, the DoD mandated that all future support contracts would be structured around the performance based contracting model. As a result of this by 2005, the number of PBCs climbed to 92.

2.1.2 The Rolls-Royce Engine Case Study

Till now we have dicussed performance based contracts from military's stand-point. However, such contracts and services have been used by the industry for a much longer time. Rolls-Royce was the first company to offer performance based contracts to commercial airlines for Rolls-Royce engines and other aviation products.

The problem with the airline industry is that aeroplane availability is of utmost importance to airline operations. A typical commercial airline does 6-8 short hops, or a minimum of 2 domestic long haul flights or typically 1 international flight per day per aeroplane. Such an operational schedule means that an aeroplane is in constant use, while only the operational staff such as the pilots and air hostesses change over.

Engine availability is almost synonymous to aeroplane availability in the airline industry. Due to this critical nature of availability of engines to the operations of the commercial airlines, Rolls-Royce started to offer the aviation industry performance based contracts. According to these contracts, the commercial airlines such as Lufthansa, Emirates etc would not buy the Rolls-Royce engine outright but would only pay for the number of hours for which the Rolls-Royce engines fitted in the aeroplanes in their fleet would be used. Rolls-Royce coined the term "Power by the Hour" for such contracts and the airlines started paying Rolls-Royce for the number of hours for which the engines performed.

Prior to "Power by the Hour" contracts once an engine was sold, it was in the interest of the engine manufacturer or supplier that the engine malfunctions so that the manufacturer/supplier could generate extra income from repairs and services. The more the engine malfunctioned, the better for the supplier and worst for the commercial airliner. Needless to mention that the after the initial engine sales were made, the fortunes of both the engine manufacturers and the airline operators were inversely proportional.

However, with "Power by the Hour" contracting, Rolls-Royce only gets paid for the number of hours for which its engines perform. So when an engine malfunctions, its in the interest of Rolls-Royce to repair the engine as quickly as possible. Furthermore, Rolls-Royce does not get paid for the repairs, but only for the operational hours of the engine.

The contracts became such a resounding success, that in 2011 Rolls-Royce generated an annual revenue of 11.3 billion pounds out of which more than half came from "Power by the Hour" contracts [Rel12].

Today performance based contracting is a norm and not an exception in the commercial airline industry and most airlines only pay for the number of hours for which they use the engines. All engine manufacturers such as General Electric and Pratt & Whitney etc now offer performance based contracts to their customers. The crux of this discussion is that in airline industry, aircraft engines are not sold as a product but are rather offered as a service.

2.1.3 Car2Go

Car2Go [Con08] is a car rental service of Daimler AG. The service was initially launched on experimental basis in Ulm, Germany in 2008. For much of 2008, the service was

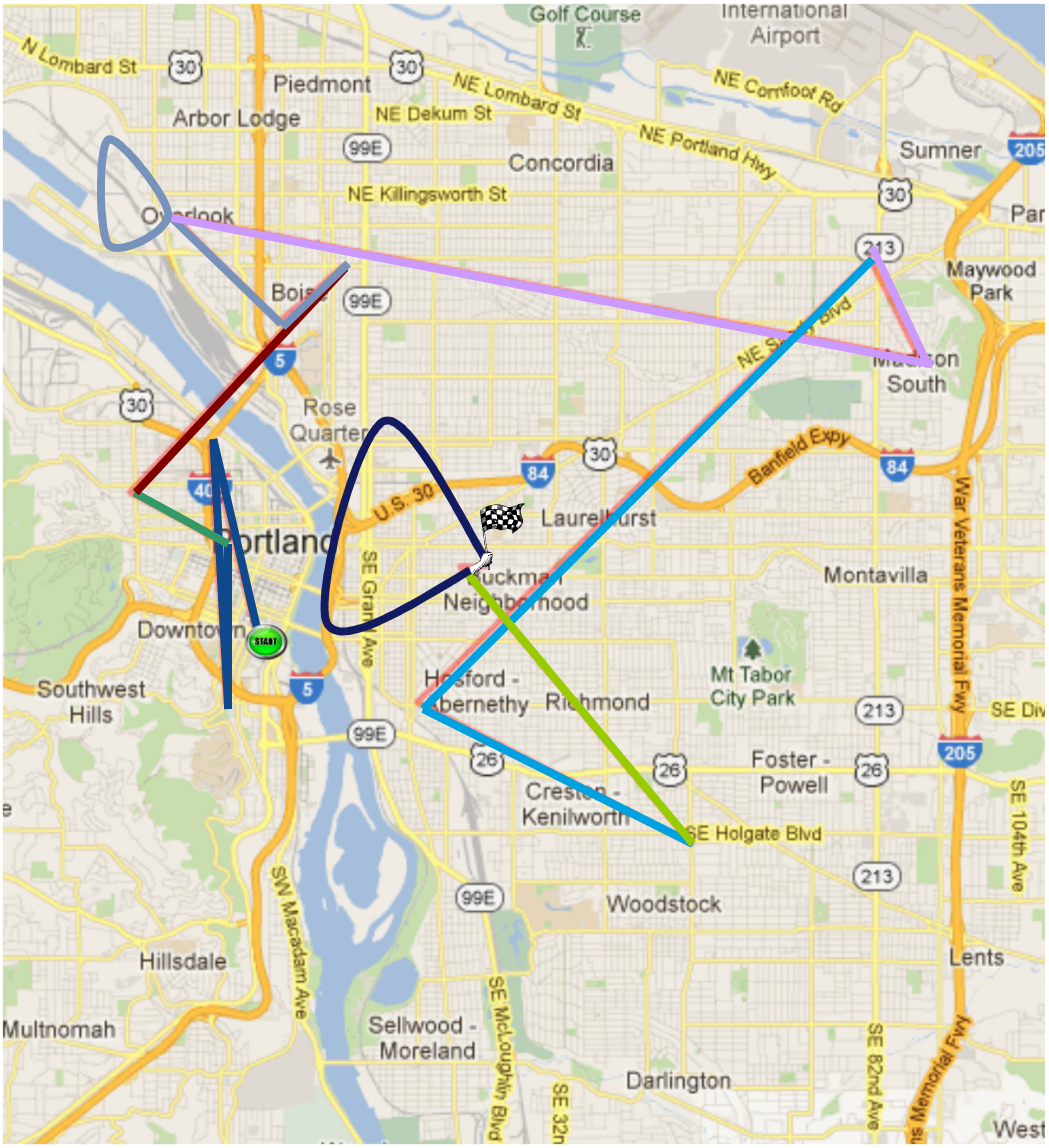


Figure 2.1: Typical Car2Go Daily Utilization [Car12]

test-marketed exclusively with Daimler employees. However, it is now available for general public as well.

The Car2Go service is revolutionary in many ways. To start with, this is the first time an automobile manufacturer is trying to shift the conversation in the automobile sector from products (cars) to services (mobility). Price of an average car produced by German automobile manufacturers ranges between 20,000-25,000 euros. In comparison, an average car produced by Chinese auto-manufacturers costs around 12,000-15,000 euros. However, there is much more to owning a car, then the initial purchase price. This is

why consumer organizations have now come up with a concept known as TCO (Total Cost of Ownership) [Rep12] which takes into account factors such as depreciation, fuel, interest, insurance, maintenance, and taxes on a particular car to compute the real cost of owning a car for 1-5 years. When we take all these factors into account. The real cost of a car is not its initial purchase price, but rather the total cost of ownership/km over 1-5 years.

In order for customers to see cars not in terms of a product but rather in terms of the service they provide (i.e. mobility), auto-manufacturers must come up with offers that charge for a service (mobility) and not the product (a car). Daimler's Car2Go service is an effort by Daimler to do just that.

With Car2Go, Daimler is not just competing with cheaper and inferior quality cars, it is competing with the entrenched car rental and taxi services as well. Upon registration with the service, a customer gets an electronic seal. The seal can be used to open up any available car. Car2Go has specially developed a smartphone app to help its users locate available cars in the vicinity. Once a user has reached his destination, he can park the car at any public parking space. Users can rent the car for as little as a few minutes to as long as they desire.

Car2Go charges a per minute rate and offers fixed discounted rates for hourly and daily usage. The rates are all inclusive of fuel, insurance, maintenance, rent and parking. The per minute charging model of Car2Go is a deviation from the normal car rental services, that allow users to rent cars for a minimum of 1 day to begin with.

After 5 years of its initial test launch, Car2Go service has turned out to be a phenomenal success. As of May 2013, the service operates over 7,300 smart two person vehicles in 25 cities around the world serving more than 375,000 customers [Rep13].

A cursory look at the statistics mentioned above show that Car2Go has around 51 registered members per car. Figure 2.1 shows typical Car2Go daily utilization. Each line is a new trip undertaken by a different user. Car2Go commissioned an internal study in the city of Seattle, which revealed that [Car12]:

- Each Car2Go vehicle is estimated to take an average of 15 private vehicles off the road.
- 30% of households that joined a car sharing service sold a car and others delayed purchasing one.
- 20% of Car2Go members are either very likely or somewhat likely to reduce the number of vehicles owned.
- 67% of Car2Go members hold a transit pass to utilize public transportation.

Car2Go plans to launch in a further 25 cities and take the tally of total cities to 50 within the next 5 years. In addition to this, the service plans to have a joint information

and payment system to enable an interface with the public transport in the cities in which it is operating.

The Car2Go service should not just be seen as an effort by Daimler AG to compete with low end cars produced by companies in China and India. The real prize could actually be much bigger than that. Support and maintenance services constitute a significant part of the world economy, and often generate twice as much profits as the sales of the original products. A study conducted by Accenture in 2003 [DK03] revealed that General Motors booked profits of $2 billion on after-sales services of $9 billion. This is a much higher rate of profit compared to what General Motors' $150 billion in car sales generated over the same period. The study showed that automanufacturers generate 40%-50% of their total profits from after-sales services which form only 25% of the over-all revenue stream. What Daimler is really trying to do is enter a services market that is much more profitable than the products market.

2.2 Requirements for a Smart Variant Production Environment

The idea behind a smart variant production environment is to create a context sensitive factory that could not just build products but also deliver services to both the producers and the consumers alike. The factory would be capable of monitoring production processes in real-time so that any disruptions in production can be detected and rectified immediately. Based on an extensive survey of variant production organizations, we have drafted the following requirements for a smart variant production environment:

1. It should be able to monitor/track production in real-time. Monitoring should not be limited to monitoring a specific part or process, instead all parts and mobile machine tools should be monitored in real-time and at all times.
2. It should be able to present the results of production and asset monitoring to higher level management and consumers alike in a meaningful way.
3. It should be able to adapt to process changes and system changes in a graceful manner.
4. It should be able to respond to changes (intended and unintended) during manufacturing in real-time. An intended change can be a product configuration change request from a consumer in the midst of production, whereas an unintended change can be a product identified as not being assembled according to a plan.

In the sub-section below, we present the different requirements for building a smart variant production environment.

2.2.1 Real-Time Production Monitoring:

The first requirement for a smart variant production environment is to monitor and track all production processes in real time. By deploying a real-time production monitoring system, factories would be able to monitor all aspects of production immediately as product parts are being made and assembled on the shop floor. Data updates are instantaneous and continual, and would not require any human intervention. Critical data such as parts produced, production time, down time, incorrect parts, work-in progress products, and finished product counts can be compiled and made available within seconds for the production managers to analyze and decide upon.

Real-time production monitoring would enable factories to have cost-effective and centralized control over plant operations, which would result in improved efficiency, greater visibility, and higher levels of productivity.

This requirement formulated the basic component of our research effort. Real-time production monitoring can be carried out in a factory by deploying RFID devices. In order to accomplish this, we developed an RFID-based real-time production monitoring framework (cf. Chapter 5). Furthermore we also developed several algorithms to ensure that the production monitoring is reliable (cf. Chapter 5 and 6).

2.2.2 RFID-based Complex Event Processing Framework for Manufacturing:

Once an RFID-based production monitoring component is deployed, it will enable a factory to monitor the product parts moving across the production lines in real-time. However, raw RFID data poses additional challenges that need to be addressed in order to effectively use this real-time RFID data.

First and foremost, the raw RFID events have implied meaning which needs to be transformed and aggregated with other events to obtain the semantic meaning of those events in the context of manufacturing. Consider for example that we have an RFID event "part p1 at production line 1 at time t" and another RFID event "part p2 at production line 1 at time t". Now from our production database we would know that part p1 needs to be assembled together with part p2 and since our RFID-based production monitoring framework is telling us that it has read both the parts at the same time at the same place, it means that the two parts have been assembled together. This task of deriving high level complex events from low level raw RFID data is referred to as complex event processing (CEP).

The second problem is that this raw RFID data is temporal in nature and is generated in high volume, so the processing of this data needs to be done in an automatic manner and in real-time. Therefore in order to address the second requirement of the smart variant production environment i.e. to present the results of production monitoring to producers and consumers in a meaningful way, we would need to have an RFID based

complex event processing framework for the manufacturing domain. This component of the smart variant production environment has also been developed and presented in Chapter 7.

2.2.3 Realiability Framework for Production Monitoring:

In production environments accuracy and precision is of paramount importance. In a factory setting merely having a real-time production monitoring and complex event processing framework would not be enough if these components wont have a high degree of reliability. Consider for example that we have developed and deployed the RFID-based real-time production monitoring framework. Now, our framework tells us that product part o_1 is missing on production line 1. Does this mean that the product part o_1 is actually missing or is it that the RFID reader deployed on production line 1 missed out on detecting the product part.

In short, both the real-time production monitoring and complex event processing components would only be half good if we don't have a reliability component or mechanism to ensure that the product parts monitored and the complex events detected have actually occured in reality. Therefore, a smart variant production environment must also have a reliability framework to ensure this. We have also designed and implemented this component of the smart variant production environment and provide probabilistic guarantees for both product part and complex event detections that these detections actually match reality (cf. Chapter 5 and 7).

2.2.4 Real-Time Networked Manufacturing:

Once a real-time production monitoring component and the complex event processing component are deployed, we can monitor the production process in real-time and gain meaningful insight into the production process by viewing high level information such as information about the work-in-progress products in real-time. However, in case there is a discrepancy the system would merely report the inconsistency.

In order for the factory to react to this newly generated RFID data (which is also the fourth requirement for a smart variant production environment), we would have to integrate this data with all the other systems running within the factory, such as the Manufacturing Execution System (MES), or Materials Control System (MCS). This integration will enable the relevant systems within the smart variant production environment to get RFID data and respond to changes in the production process in real-time.

As an example consider that a product has been fitted together with an incorrect part. Once this information is made available, it makes no sense for the assembly of this product to continue. Therefore, the MES can update its schedule and also let the MCS

know that it should not deliver the remaining parts of this product to their respective assembly points. Such a real-time removal of faulty products from the production line would save the factory from complete assembly of a faulty product which in any case would have to be disassembled and then re-assembled again.

The integration of all the existing systems and components within a factory was beyond the scope of our work. However, systems and technologies are now available that can ensure that the different components within a factory can be networked with each other and respond to changes in real-time.

2.3 Service Offerings of a Smart Variant Production Environment

In this section we would take a look at some of the services that can be offered once a smart variant production infrastructure is in place. It is worth noting that since the smart variant production environment would be able to generate immense amount of additional contextual data as compared to a traditional factory, most of the services that can be offered by establishing a smart variant production environment are also informational in nature.

2.3.1 Production Monitoring for Consumers

In the past, the postal service was a black box process for the customers. Customers used to send post and after doing so have had no information about what happened to their post. Whether the post was delivered successfully was only known to them once they received a response from the person to whom the post was sent.

This lack of information was such a big market opportunity that saw the rise of companies like FedEx that strived on nothing but speed and informational transparency. Users of FedEx can find out exactly where the post is during each step of the delivery process and can find out at what time the post was delivered and who exactly received it.

Today's factories work in exactly the same way as the postal service used to work in olden days. Once a product is ordered, the user has no way of finding out if the product is planned for production, when it will be produced, and at what stage of production it currently is. This lack of information is especially troublesome for products that have a long manufacturing cycle, such as automobiles or aeroplanes.

The real-time product tracking component of a smart variant production environment can not only be of use for the producers but can also be offered as an informational service to the customers, thereby allowing them to access real-time information about the state of assembly of their product. We firmly believe that access to this particular information will have a similarly disruptive affect on the manufacturing segment as the access to postal information had on the postal service.

2.3.2 Dynamic Product Re-configurations

Integration of RFID technology for real-time production monitoring with networked manufacturing can significantly reduce response time for both the producers (reaction to production errors) and consumers (changes to previous orders). Let us consider the assembly of a laptop at a computer manufacturer. The customer usually specifies the processor, the RAM, the hard disk, graphics card, screen size and other accessories. With networked manufacturing and production monitoring service for consumers (cf. Section 2.3.1), customers would have the flexibility to change their order specifications midway through manufacturing. At any moment of time, the customer can track what parts of the laptop has been assembled together and what is still to be done and hence can change the specification of the parts that are still to be assembled together. The factory would respond to these changes in a similar way as it would respond to production errors and will consider the previous customer order to be incorrect and would change the production process such that the new and correct specification is assembled. Since the production environment will be closely integrated there won't be any paperwork and human interference in order to carry out these changes. In this way, a company would be able to provide customer satisfaction through tailor-made products supplied reliably and efficiently with exceptionally short response times.

2.3.3 Production Data Traceability for Producers

The services mentioned above are the ones that can be offered to the customers. However, a smart variant production environment can also offer production data traceability to the company itself. Production data should be traced in order to enhance operational efficiency and to infer the quality of different product parts. As a matter of fact, RFID technology has already been used to achieve the above mentioned objectives. In the sections below, we would take a look at two possible ways in which traceability information could benefit a company.

Data Traceability for Operational Efficiency and Asset Visibility:

Ngai et al [NCL+07] have reported a research study about deployment of an RFID-based traceability system in an aircraft engineering company based in Hong Kong. The company is responsible for maintaining and repairing aircrafts produced by both Boeing and Airbus for different airlines. The study concluded that RFID technology was able to efficiently and effectively track both assets (machines and tools required for maintenance) and repairable items (aircraft parts) during the maintenance cycle. Due to this, the said company was able to increase operational efficiency and enhance asset visibility.

Data Traceability for Inference of Product Quality:

One of the ways in which cost reductions can be achieved in supply chain management is by managing information about a physical object throughout its life cycle [Ina09]. This

is one aspect for which RFID technology is being used in supply chain management by retailers like Wal-Mart and Tesco [Rob05], [Pro03]. One side affect of managing the information about a physical object throughout its life-cycle is that this makes it feasible to maintain the traceability information of the object and hence can be used to infer the quality of the product in which that physical object is used. Foods, pharmaceuticals and the airline industry are prime examples of areas that require maintenance of traceability information about the objects that constitutes the products [KSCB03], [Jon06], [PTMS07].

True traceability would require that products should not just be monitored once they are produced but should be monitored even during production along with the parts that are used to manufacture these products. Once a factory would have this traceability data, it can have operational efficiencies and asset visibility and can use this information to infer product quality if and when required. In addition to this the production data traceability service would enable a smart variant production environment to enhance customer experience by providing traceability information about the product and its parts throughout the life-cycle of the product.

2.4 Summary

In this chapter we presented several case studies to show that both commercial and military organizations are now moving away from procuring and manufacturing products to assembling and acquiring services. The case studies further showed the advantages of acquiring services over typical products in certain manufacturing scenarios. We then move on the present requirements for a smart variant production environment and finally described several services that can be provided by the smart variant production environment to its customers.

The real-time production monitoring, complex manufacturing event processing and reliability framework that are the basic building blocks of a smart variant production environment form the basis of the work we have undertaken in this research project. Each of these tasks are detailed in separate chapters later on in this dissertation.

Chapter 3

System Model

In this chapter we discuss the system model and assumptions that form the basis of the entire dissertation. The chapter is divided into several sections. Production Monitoring System (cf. Section 3.1) present the different architectural and design components of our system. After that we go on to present the Formal System Model (cf. Section 3.2). The Event Model (cf. Section 3.3) discusses the basic events that are generated within our system. Next we go on to discuss the different RFID Reader Failures (cf. Section 3.4) that we consider in our work. The discussion of RFID Reader Failures is succeeded by System Failures and Assumptions (cf. Section 3.5) in which we talk about the different system failures and our assumptions regarding those failures. Lastly, the chapter is concluded with a short summary (cf. Section 3.6).

3.1 Real-Time Production Monitoring System

The Real-Time Production Monitoring System can be divided into three broad categories: production system, monitoring framework, and the applications (cf. Figure 3.1). The Production System (cf. Section 3.1.1) is the factory environment that is to be monitored and supplies the Real-Time Production Monitoring System with real-time production data. The Monitoring Framework (cf. Section 3.1.2) consists of infrastructure that is needed to collect and make sense of the real-time production data which is supplied by the production system. The Applications (cf. Section 3.1.4) in turn rely on the real-time data provide higher level services.

3.1.1 Production System

In factories, product parts moving through production lines meet at certain assembly points. It is at these assembly points that these product parts are processed or assembled. In our framework, production lines are modelled as **production paths**.

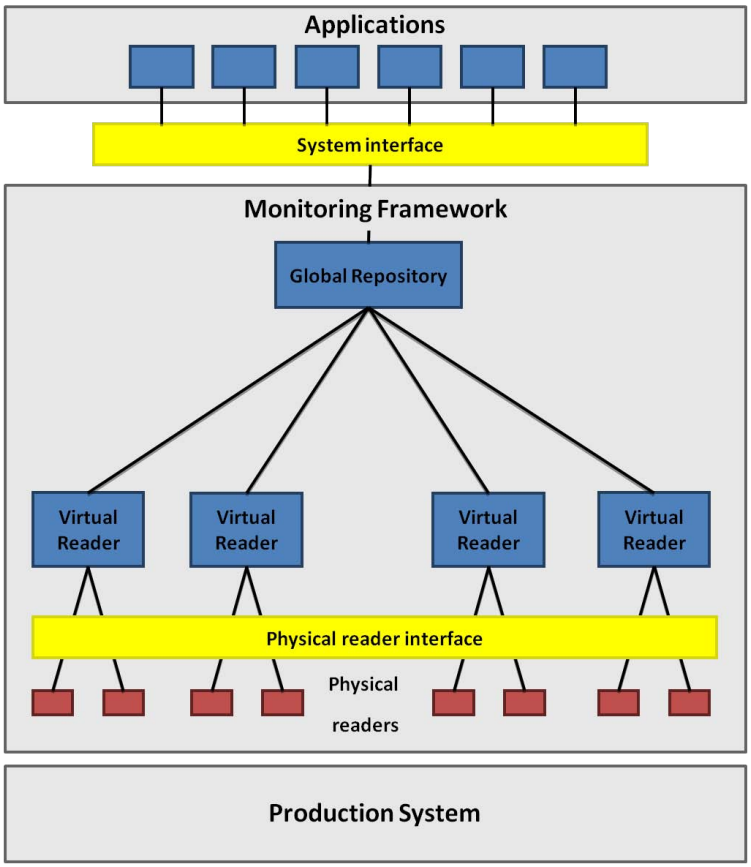


Figure 3.1: System Architecture

Production paths can run in parallel, converge at an assembly point, or diverge from an assembly point. The movement of product parts on production paths is sequential and uni-directional. Since the topology of these production lines and assembly points vary from one factory to the other, the box representing Production System in Figure 3.1 is empty. However, a potential factory deployment scenario along with the different Monitoring Framework components is shown in Figure 3.2.

3.1.2 Monitoring Framework

The monitoring framework (cf. Figure 3.1) is comprised of a global repository, several virtual readers *vr*s and a large number of physical RFID readers *pr*s. Figure 3.2 shows a potential factory deployment scenario of the *vr*s and *pr*s in our production monitoring framework. The discussion of important components and concepts of the monitoring framework follows:

Physical Reader: Physical RFID readers, denoted as *pr*s, are deployed on the production lines. Each product part is tagged with an RFID tag, and as these product

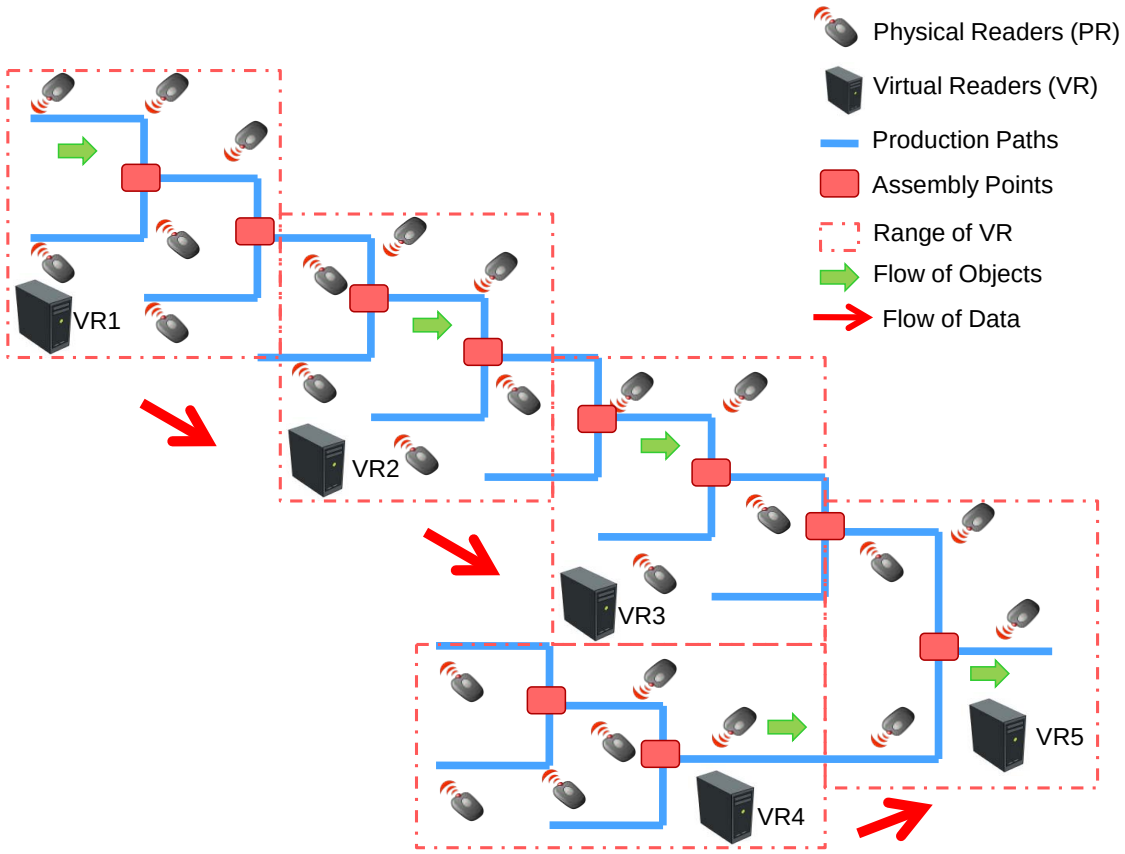


Figure 3.2: Sample Factory Deployment

parts move across the production lines, they are continuously tracked by the *pr*s. A *pr* can read several RFID tags within its sensing range.

The physical readers *pr*s are placed on the production paths and detect the product parts moving through these paths. A read event e by a *pr* is defined as, $e = (o, pr, t)$; where t is the time at which product part o was detected by physical reader *pr*.

Since the location of each *pr* is already known via the global repository, we can deduce the location of product part o from the location of the *pr* that detected product part o. Each read event e is correct with a certain probability, $p(e)$. For the purposes of Chapter 5 we assume that every *pr* correctly detects the product parts with the same probability. However, this assumption would be dealt upon in Chapter 6, in which we would present a system to continuously update the probabilities of RFID readers according to the accuracy with which they detect the product parts.

Virtual Reader: The task of detecting hundreds of product parts and thousands of potential complex events requires extensive computations. In order to achieve this objective, we have designed a distributed and scalable abstraction called the virtual

reader vr. The vrs distribute the workload and process the data close to the sources prs.

Each pr is assigned to exactly one virtual reader vr and covers a contiguous part of the production path. The set of prs corresponding to a virtual reader vr_i is denoted by $PR_i = \{pr_{(i,1)}, pr_{(i,2)}, \ldots, pr_{(i,m)}\}$. In production environments since the topology of the manufacturing plant does not change very often, the pr to vr mapping can be done during system initialization.

The manufacturing plant is divided into *topological regions*. The regions are strictly non-overlapping. Every region is associated with exactly one vr such that the vr can process the events generated by the prs in that region. Our framework can monitor product parts and detect complex manufacturing events in factories irrespective of the plant topology.

A vr gathers raw data from a set of prs in its region. This data is then used to track product parts and deduce sequences of product parts moving through the production paths. Since a large number of product parts move through the production paths during assembly, the inherent inaccuracies of the RFID readers lead to many possible product parts sequences. To overcome this issue, the deduced sequences are assigned probabilities that represent the likelihood for that sequence to exist.

Global Repository: The repository has knowledge of the entire factory layout, production modules, the locations of the physical readers prs, virtual readers vrs, and the assembly points. The precise information stored in the repository is listed below:

- Plant Topology: The layout of the factory including the location of production lines and assembly points and their respective order.
- Physical Readers: Location of physical readers and their mapping with the Virtual Readers.
- Virtual Readers: The layout of the Virtual Reader deployments, that is information such as which VR succeedes and preceeds which other VR.
- Production Plan: Information regarding the production plan, such as what products would be assembled on a given day, which product parts would be a part of what products, and what product parts would move through which production lines. Production plan is discussed in detail in LernFabfrik (cf. Chapter 4.2)
- Production Data Archives: Historical production data, such as what products were assembled during what day, and which parts were used to assemble specific products.

The repository does not perform any computations. In fact, it provides other components with the above mentioned information that is necessary for processing data or responding to queries.

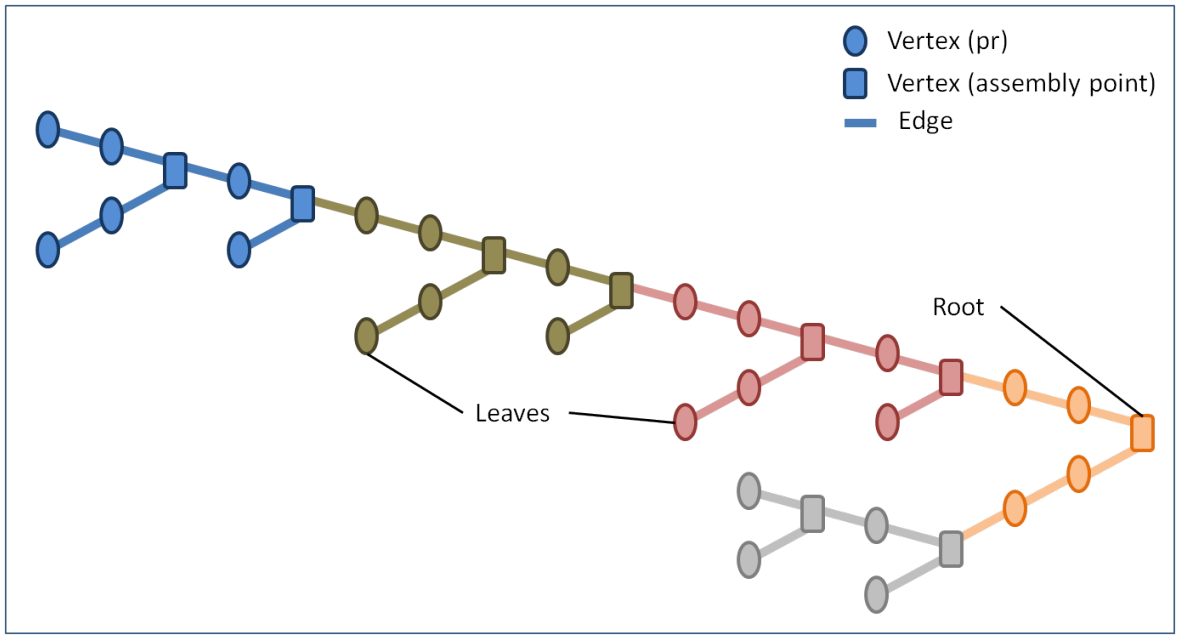


Figure 3.3: Polytree/DAG Model of Sample Factory Deployment

3.1.3 System Interface

The System Interface provides an API using which the higher level applications can query the Monitoring Framework and access useful information regarding the state of production of both the products and the production process. The methods listed below are only a representation of the kind of information that can be accessed via the API and is in no way an exhaustive list of the API functions.

getPartPosition(productpart id): returns the last sensed position of a product part.

getProductStatus(product id): returns the last updated production status of the product.

getErrorPosition(product id): returns the position at which the product had an error.

isProductFaulty(product id): returns true if the product has a production error, false otherwise.

isProductionOnTrack(product id): returns true if the assembly of the product is going according to the production plan and on schedule, false otherwise.

getErrorImpact(error id): returns the number of products affected by the error.

getErrorProbability(error id): returns the probability of the error.

3.1.4 Applications

The design an development of applications that would use the data generated by real-time production monitoring system is beyond the scope of this work. However, we would still like to elaborate some potential applications that can be built on top of our system.

Production monitoring: Production monitoring would naturally be the first application that can be built for both the producers and customers. The application would be capable of providing real-time information about the status of production of individual products.

Inventory Monitoring: An inventory monitoring application would inform the factory manegement about the status of inventory of different components. In a normal inventory system, a component is removed from inventory when it leaves the warehouse. However, the real-time inventory monitoring could track the component even during production and would consider the part used/consumed when it either leaves the plant as part of a product or is destroyed or gets damaged.

Fault Monitoring: A fault monitoring application would inform the factory management about the faults/errors that may arise during production in real-time. In addition to this, the management could be apraised about the delay in shipment of products as a result of these faults.

Production Data Traceability: A traceability application can be built that can be used to find out traceability information about an individual product such as the date on which the product was built, the different components that were used to build the product, if there were any production issues during the production of that specific product and so on.

3.2 Formal System Model

We model a factory or production environment in our framework as a *polytree*, which is basically a *directed acyclic graph* whose underlying undirected graph is a tree. Both the assembly points and the physical readers *pr*s deployed on the production paths are designated as vertices on the polytree, where as the production paths are modeled as the edges of the tree. Figure 3.3 is a Polytree/DAG representation of the sample factory deployment shown previously in Figure 3.2.

Our polytree is always a *rooted tree*. The root vertex is a vertex at which the product is completely assembled and hence moves out of the factory. The leaf nodes or vertices are the assembly points at which the production process initiates. Leaf nodes are vertices of degree 1 or in other words are *terminal vertices*. Production paths normally have multiple terminal vertices from where multiple components may start initial production.

The edges in the polytree have a natural orientation towards the root. In terms of the production environment this means that product parts would actually move from the initial assembly point towards the terminal assembly point or from leaves towards the root.

The set of all vertices within the polytree is denoted by $V = \{a, b, \ldots, n\}$. The set of all vertices representing the physical readers *pr*s is denoted by $PR = \{c, d, \ldots, m\}$, where $PR \in V$. The set of all vertices representing the assembly points is denoted by $AP = \{e, f, \ldots, o\}$, where $AP \in V$. PR and AP are disjoint that is no member of one set is contained in the other, in other words $PR \cap AP = \emptyset$. The set of all edges within the polytree are denoted by $EG = \{a, b, \ldots, n\}$.

The set of *vr*s in the production monitoring framework is denoted by $VR = \{vr_1, vr_2, \ldots, vr_n\}$. Each vr_i knows its predecessor(s) vr_{i-1} and successor(s) vr_{i+1}. A *vr* at the start of manufacturing plant does not have a predecessor vr_{i-1}, where as a *vr* at the end of manufacturing plant does not have a successor vr_{i+1}. Each *vr* is configured to receive information from its predecessor vr_{i-1} and sends information to its successor vr_{i+1}.

$(pr_i \sqsubset pr_j)$ is the **production path order**, which implies that pr_i is deployed before pr_j on the production path i.e. product parts will first be read by pr_i and then by pr_j.

Since physical readers *pr*s are modeled as vertices in our framework, the production path order $(pr_i \sqsubset pr_j)$, can also be depicted as $u \leq v$, where pr_i is represented by vertex v and pr_j is represented by vertex u.

The polytree has a topological ordering called the *tree-order*. The *tree-order* is a partial ordering on the vertices of the tree with $u \leq v$ if and only if the unique path from the root to vertex v passes through vertex u. In general, this ordering is not unique; a polytree has a unique tree order if and only if it has a directed path containing all the vertices. In this case the ordering is the same as the order in which the vertices appear in the path.

The polytree in our framework is a *labeled graph* i.e. all of its vertices and edges are given a unique label. So, if there would be n vertices in the tree, the labels would be 1, 2, ..., n. Edges of the tree are also labeled similarly.

Both the vertices and edges of the tree have associated weights, with W_v depicting the weight of vertex v and W_{xy} used to depict the weight of edge (xy).

The weight of each vertex denotes the time in seconds it takes for the production process at that assembly point to be completed, whereas the weight of each edge denotes the amount of time it would take for an object to traverse the production path depicted by the edge.

We have already mentioned that both the assembly points and physical readers are modeled as vertices. Both of these vertices have weights. Figure 3.4 shows a Polytree having vertex and edge weights. The circular nodes represent the physical readers,

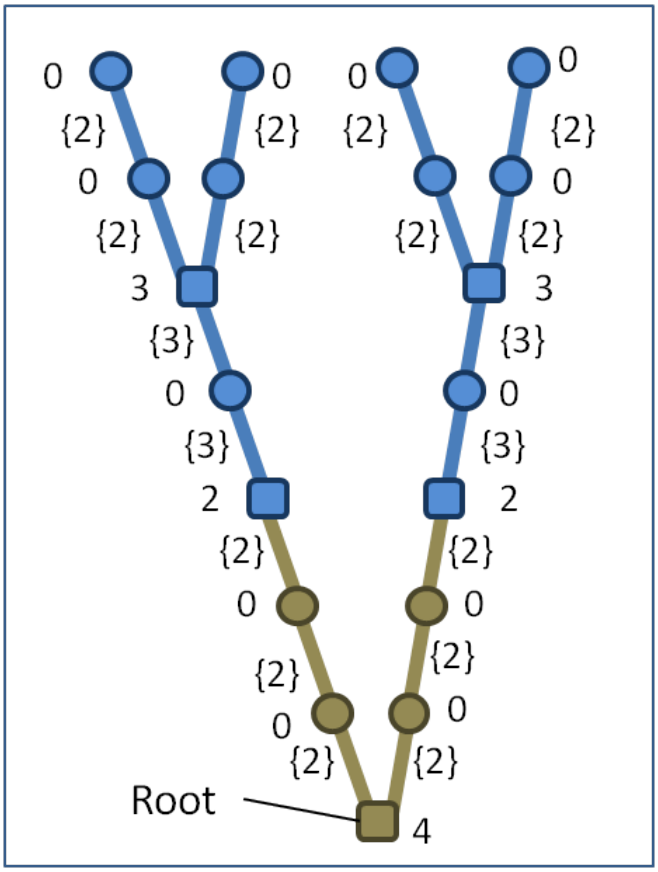


Figure 3.4: Polytree/DAG with Vertex and Edge Weights

where as the square vertices represent the assembly points. However, this depiction is only graphic, since we different between the two types of vertices because of their membership in their respective sets PR and AP.

Vertices denoting the assembly points have positive real integer weights, where as the vertices denoting the physical readers have null weights. The weights for vertices depicting physical readers are null/zero because it takes no time for a physical reader to perform its task. In other words the physical readers are able to read product parts moving on the production paths without interfering with the movement of these product parts. On the other hand, the processing at each assembly point requires some time. The weight of each assembly point depicts the amount of time it takes for a product part to move through the assembly point. The actual unit of time could be seconds, minutes or even hours depending upon the production environment. For the rest of this dissertation we would assume that the time unit is 1 second, unless otherwise stated.

During any specific production run, the product parts move on the production lines in a cyclic manner i.e. when production is in progress a new product part enters the

production line every 1 second. This is refered to as the arrival rate of new product parts. The arrival rate varies from one production environment to the other. Product parts would keep entering the production line in a cyclic manner until the production is halted or is successfully completed. The concept of cycles in the simulations that we have conducted to evaluate our algorithms have been further elaborated in Chapter 5.3.1

In addition to assigning weights to the edges we also perform *edge coloring* i.e. assign different colors to all the edges of the polytree. A key property of edge coloring described in literature is that no two adjacent edges should have the same color. In our model, we purposely violate the "no two adjacent edges are similarly colored" property. Adjacent edges in our polytree can and do have the same color. The similarly colored adjacent edges and vertices in our polytree depict the region/range of a virtual reader vr.

If v is reachable from u, then u is a predecessor of v and v is a successor of u. If there is a path from u to v, then u is a direct predecessor or parent of v, and v is a direct successor or child of u.

The *distance* $d_G(u, v)$ between two vertices u and v in the polytree is the length of a shortest path between them. When u and v are identical, their distance is 0. When u and v are unreachable from each other, their distance is defined to be infinity ∞. As an example the distance between the root and the leaves of the polytree in Figure 3.4 is 21.

3.3 Event Model

This Event Model section is divided into two sub-sections: Raw RFID Events (cf. Section 3.3.1), and Primitive Events (cf. Section 3.3.2). The raw RFID events section discusses the raw RFID events and explains the techniques that we use to transform these raw events into primitive events. The primitive events section discusses the primitive events which are filtered events that do not contain any duplicates.

3.3.1 Raw Read Events

Raw events are events detected by RFID readers. These raw events contain noise in the form of false positives, and duplicates. Another type of noise is false negatives, which are RFID events that a reader fails to read although their corresponding tags are within the reading range of the reader.

A raw event is defined as:

$e_{raw} = (o_i, pr_i, t_i)$; where t_i represents the time at which product part o_i was detected by physical reader pr_i.

e_{raw} are raw events such that there exists other raw events e_{raw} having the same o_i and pr_i. In other words there exists other detections of the same product part o_i by the physical reader pr_i

3.3.2 Primitive Read Events

Primitive read events are de-noised or non-duplicate physical readings. We operate the RFID readers in continuous read mode. In this mode, the readers read RFID tags continuously as long as the tag is within the read range of the reader. In order to filter out the noise, we insert each RFID tag detected by an RFID reader into a queue. If the same tag is detected again, a counter associated with the queued tag is increased. Once the counter reaches a certain threshold we transform that read event into a primitive event. In this way all the duplicate reads for the tag are eliminated. In case the counter of the raw RFID event fails to reach the threshold value, they are discarded as being false positive events.

A primitive event is defined as:

$e = (o_i, pr_i, t_i)$ such that there exists no other primitive event e having the same o_i and pr_i.

The *probability of a primitive read event* $p(e)$ denotes the probability with which a product part o was correctly read by a pr. In the rest of this dissertation, we would use primitive read events in our algorithms and would refer to them as read events.

3.4 RFID Reader Failures

In this section we will go through the different errors that affect the reliable operations of the RFID devices. RFID reader errors result from the operations of RFID devices and can be broadly divided into four distinct categories; duplicate reads, false reads, missed reads, and out of order reads.

3.4.1 Duplicate Reads

Duplicate reads occur when a pr reads the same product part o multiple times. This happens because we operate the RFID readers in continuous mode. In this mode, the readers continuously read RFID tags. Since, it takes some time for a product part to move across the read range of an RFID reader, the product part is normally read multiple times during that single pass across the reader. We filter out the duplicate reads as is explained in Primitive Read Events (cf. Section 3.3.2).

3.4.2 False Reads

False reads occur when a physical reader pr reads a product part o_i incorrectly or overshoots and reads other product parts that are not on the production line, but may come within the sensing range of the pr. Let $PP_{pln_{pp_i}} = \{o_a, o_b \ldots, o_n\}$ be a set of all product parts that are supposed to pass through production path pp_i. Then a false read is defined as a read event $e_{false} = (o_i, pr_i, t_i) \mid o_i \notin PP_{pln_{pp_i}}$. $PP_{pln_{pp_i}}$ is obtained from the global repository (cf. Section 3.1.2), which has information about each specific product part that is to pass through a specific production path.

3.4.3 Out of Order Reads

Out of order reads occur due to the read range of the prs, similar to false reads. An out of order read occurs when a pr detects product parts in the order $(o_j < o_i)$, when the actual product part order PPO_{act} on the production path was $(o_i < o_j)$.

3.4.4 Missed Reads

Missed reads occur as a result of unreliability of prs. Some of the common causes of missed readings include metallic interference and batch tag reads. If a product part o_i passes through the production line on which physical reader pr_i is deployed but is not detected by the physical reader pr_i, it is considered a missed read.

We find out missed reads of a particular physical reader pr_i using the following method. Let $PPL_{vr} = \{o_a, o_b \ldots, o_n\}$ be a set of all detected product parts at a vr and $PPL_{pri} = \{o_a, o_b \ldots, o_m\}$ be a set of detected product parts by physical reader pr_i. Then the complement of PPL_{vr} with respect to PPL_{pri} ($PPL_{vr} \backslash PPL_{pri} = \{x : x \in PPL \mid x \notin PPL_{pri}\}$) gives us the set of missed readings $PPL_{pri_{missed}}$ of pr_i. The cardinality of set $PPL_{pri_{missed}}$ gives the total number of missed reads of physical reader pr_i.

3.5 System Failures and Assumptions

We assume that the smart variant production environment in which our system would be deployed would have a local area network connected via ethernet [MB76], with TCP/IP based communication protocol [CI05] for the higher layers. TCP/IP is a reliable protocol that ensures that data is sent reliably and efficiently over the network.

In addition to this, we assume a fault tolerant system in which there are no omission (for e.g., crash failures, failing to receive a request, or failing to send a response etc.) or commission failures (e.g., processing a request incorrectly, corrupting local state, and/or sending an incorrect or inconsistent response to a request etc.). Several algorithms and

protocols have been proposed to make a system tolerant to crash faults. Protocols such as Q/U [AEMGG+05], HQ [CML+06], Zyzzyva [KAD+10], ABsTRACTs [GKQV10], Aardvark [CWA+09] and RBFT [AMQ13] can be readily deployed in order to achieve fault tolerant node replication with low costs, high performance and robustness.

Furthermore, we assume the clocks between the different nodes to have high accuracy (in sub-microsecond range). Protocols such as Network Time Protocol (NTP) [Mil91] can not provide such accuracies, however IEEE precision time protocol [LEWM05] can achieve sub-microsecond level accuracies between system nodes in a local area network. For our system, we need accuracies within the sub-second range, so the IEEE precision time protocol is an adequate solution for our system.

3.6 Summary

In this chapter we have presented the different system components that form the basis of our dissertation. These components include the virtual reader, physical reader, higher level applications and the interfaces between these components. We then move on to present a formal system model for the real-time production monitoring framework. In addition to this, we also formally presented the basic RFID events and presented the different RFID reader failures such as duplicate readings, false readings, out-of-order readings and missed readings that must be considered by any reliable RFID framework. Finally, we discuss our assumptions regarding network and node failures.

Chapter 4

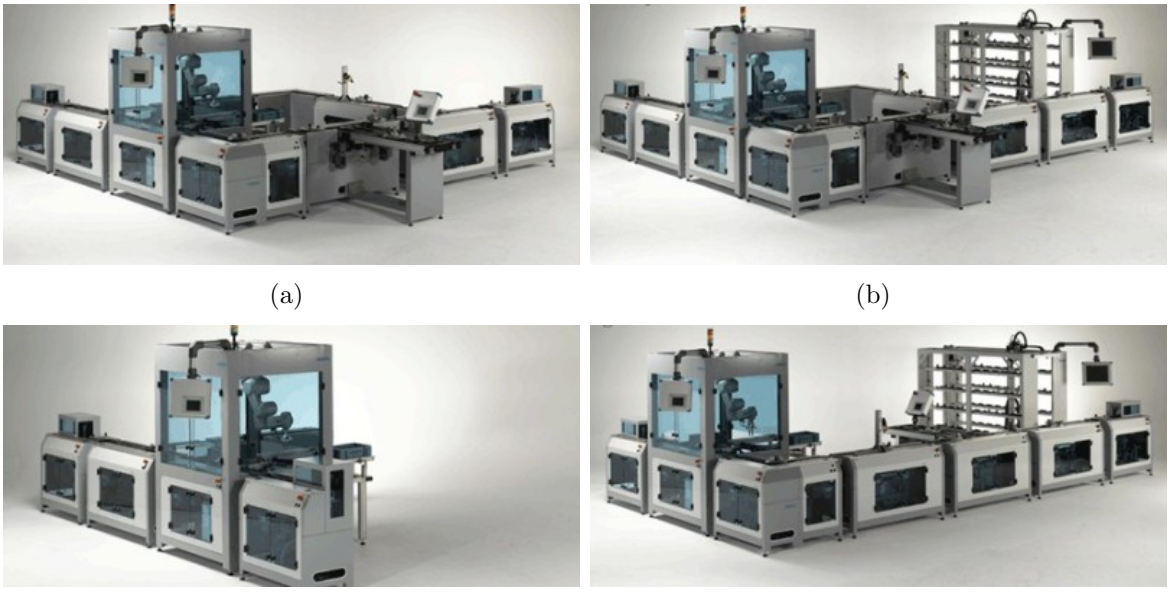
LernFabrik

The Lernfabrik [LCW08] (german for teaching/learning factory) is a modern factory infrastructure that is established at the University of Stuttgart for conducting research on manufacturing processes using real products and production lines. We conducted experiments at the Lernfabrik in order to find out the reliability of off-the-shelf RFID readers. Our experiments showed that the accuracy of RFID readers is not just a number but is dependent on many factors such as the position of the RFID reader, the distance between the reader and the tag, the distance between two adjacent tags, the number of tags that the reader is trying to detect simultaneously and so on.

The rest of the chapter is divided into several sections. Section 4.1 introduces the Lernfabrik and discusses the design of the factory and the different types of products that it produces. Section 4.2 discusses at length the production planning process and finally presents the production plans that are created as a result of this process. Section 4.3 goes on to present the results of RFID reader experiments that we conducted in the Lernfabrik. Finally Section 4.4 provides a brief summary of the chapter.

4.1 LernFabrik Introduction

The Lernfabrik has several modules which can be configured into various topologies. Figure 4.1 shows the various deployment topologies of the lernfabrik. The Lernfabrik modules are of two types active modules such as the robotic arm or the assembly points on which some activity takes place and inactive modules such as the production lines and the storage area. The production process involves product planning and product assembly. Production planning involves plans for the type of products that would be built, the parts that would belong to each product and so on. Planning for a single type of product is a simplified process, however, with variant production, this process becomes quiet extensive.



(a) (b)

(c) (d)

Figure 4.1: LernFabrik Deployment Topologies

The Lernfabrik produces a stationary box that has three parts: a large cup, a small cup and a thermometer or a hydrometer. Each of these three parts has several different variants such as a large cup without hole, a large cup with hole, a large cup with stripes, etc. The lernfabrik can produce around two dozen product variants. The product parts are assembled on a baseplate, which in turn is mounted on a tray that moves on the production line. The baseplate has three sockets (placeholders) for the parts, and every part can be placed on any of the three sockets on the baseplate. The assembly of each new part on the baseplate is carried out by a human worker or the robot at the assembly point. When the baseplate reaches the assembly point it stops for a few seconds, the worker places a new part on the plate and updates the terminal which enables the baseplate to move again.

Each baseplate moves through three assembly points and three cameras that are fitted after each assembly point. A correct assembly involves the placement of the right product part on the right position on the baseplate.

We used the Lernfabrik as a test environment to evaluate the reliability of RFID readers in a real production environment. The rest of this chapter is divided into two main sections Production Planning (cf. Section 4.2) and RFID Reader Evaluations (cf. Section 4.3). In the production planning section, we explain how the production is planned within the Lernfabrik and also show the production plans that are generated as an outcome of this process. The RFID reader evaluations section discuss the RFID reader evaluations under different settings.

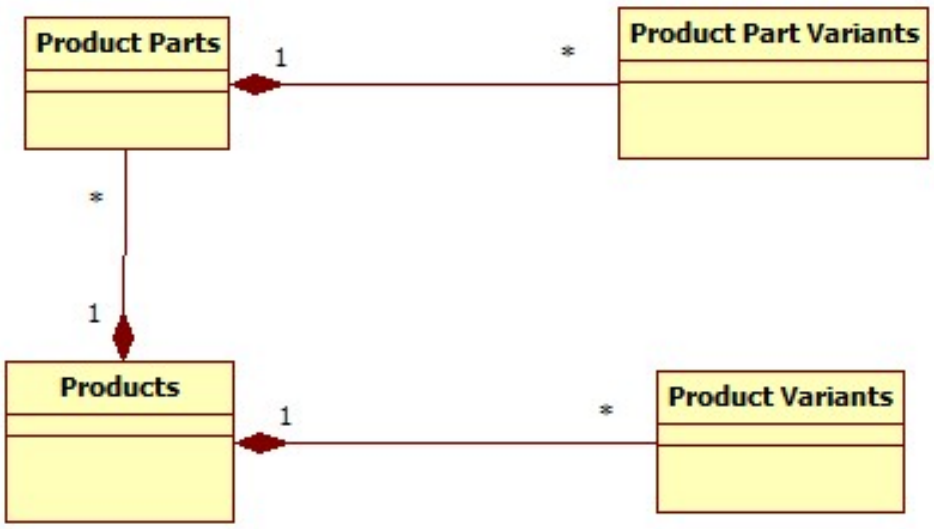


Figure 4.2: ER Model of Production Database

4.2 Production Planning

Production planning is arguably one of the most important and crucial steps in production and specially in variant production. Once the factory has received orders to build a certain number and type of products, it starts to plan the actual production. This involves planning for each of the sub-parts for all these products to be available in the inventory and then be available on the production lines in the desired order and sequence at the desired time.

In order to explain production planning, we would walk you through the different production planning steps taken in the lernfabrik in order to produce the different variants of the stationary box day in and day out. Figure 4.2 shows the ER model for the key elements of the lernfabrik production database. The database consists of four basic entities or tables namely: product parts, product part variants, products and product variants. Each product (i.e. stationary box) consists of many different product parts. Each of the product parts in turn have many different variants, which results in the final products being built having different variants of their own. Table 4.1 lists all the different product part variants along with their respective codes that are used to produce stationary boxes at the lernfabrik. Figure 4.3 in turn shows some of the different variants of stationary boxes that are produced using the different product part variants at the lernfabrik.

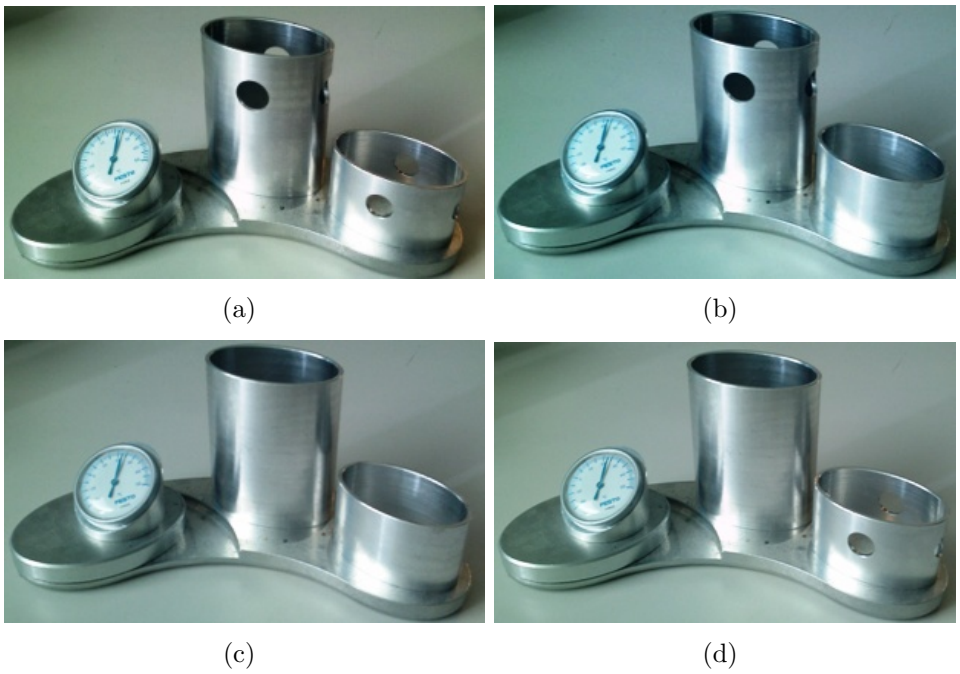
At the start of each day, the production manager at the lernfabrik receives a production plan which details the different product variants that are to be produced during that

Table 4.1: Product Part Variants

Product Part Code	Product Part
ppv_1	Large Cup
ppv_2	Large Cup with Hole
ppv_3	Large Cup with Stripes
ppv_4	Small Cup
ppv_5	Small Cup with Hole
ppv_6	Small Cup with Stripes
ppv_7	Thermometer
ppv_8	Hydrometer

day. Table 4.2 shows a sample production plan from the lernfabrik, the production plan lists the different products that are to be built during that specific day along with precise specification of the different product part variants that would become a part of each of these products.

Before the production plan is taken to the factory floor to carry out production, it is processed. This step involves sorting the production plan for the different product part variants such that the same product part variants are next to each other. Table 4.3 shows the initial production plan for a certain day, after it has been processed. In our scenario, we sort the production plan with respect to the product part 1, and then go



(a) (b)

(c) (d)

Figure 4.3: LernFabrik Product Variants

Table 4.2: Production Plan for Day X

Product Code	Product Part 1	Product Part 2	Product Part 3
pv_1	Large Cup	Small Cup	Thermometer
pv_2	Large Cup with Hole	Small Cup with Stripes	Thermometer
pv_3	Large Cup	Small Cup with Hole	Hydrometer
pv_4	Large Cup with Stripes	Small Cup with Stripes	Hydrometer
pv_5	Large Cup	Small Cup	Hydrometer
pv_6	Large Cup with Hole	Small Cup with Hole	Thermometer
pv_7	Large Cup with Hole	Small Cup with Stripes	Hydrometer
pv_8	Large Cup with Stripes	Small Cup with Hole	Hydrometer
pv_9	Large Cup	Small Cup with Stripes	Hydrometer
pv_{10}	Large Cup	Small Cup with Hole	Thermometer
pv_{11}	Large Cup with Stripes	Small Cup with Stripes	Thermometer
pv_{12}	Large Cup with Stripes	Small Cup	Thermometer
pv_{13}	Large Cup	Small Cup with Stripes	Thermometer
pv_{14}	Large Cup with Stripes	Small Cup	Hydrometer
pv_{15}	Large Cup with Hole	Small Cup	Thermometer
pv_{16}	Large Cup with Hole	Small Cup with Hole	Hydrometer
pv_{17}	Large Cup with Stripes	Small Cup with Hole	Thermometer
pv_{18}	Large Cup with Hole	Small Cup	Hydrometer
...	...	...	...

on to sort the production plan with respect to product part 2 and finally with respect to product part 3.

It can clearly be seen that in the processed production plan, all the products having Large Cups, would be produced one after the other, and then the products having Large Cup with Holes would be produced and so on.

The core idea of processing is to make the production plan resemble a batch production operation as much as possible so as to eliminate the complexity of assembling different product part variants on the production floor. If our stationary box would have consisted of only one part (i.e. product part 1), the processing step would have ensured that first all the Large Cups are produced and then the Large Cups with holes and then the ones with stripes. In essence, the production process would have mimiced a batch production process for each specific variant of product part 1.

All companies that carry out variant manufacturing process their production plans to minimize complexity during the actual production process. Automobile companies for example arrange similar and closely related cars together to simplify the assembly process as much as possible.

From the processed production plan (cf. Table 4.3), we can easily deduce the planned

Table 4.3: Production Plan for Day X after Processing

Product Code	Product Part 1	Product Part 2	Product Part 3
pv_1	Large Cup	Small Cup	Thermometer
pv_5	Large Cup	Small Cup	Hydrometer
pv_{10}	Large Cup	Small Cup with Hole	Thermometer
pv_3	Large Cup	Small Cup with Hole	Hydrometer
pv_{13}	Large Cup	Small Cup with Stripes	Thermometer
pv_9	Large Cup	Small Cup with Stripes	Hydrometer
pv_{15}	Large Cup with Hole	Small Cup	Thermometer
pv_{18}	Large Cup with Hole	Small Cup	Hydrometer
pv_6	Large Cup with Hole	Small Cup with Hole	Thermometer
pv_{16}	Large Cup with Hole	Small Cup with Hole	Hydrometer
pv_2	Large Cup with Hole	Small Cup with Stripes	Thermometer
pv_7	Large Cup with Hole	Small Cup with Stripes	Hydrometer
pv_{12}	Large Cup with Stripes	Small Cup	Thermometer
pv_{14}	Large Cup with Stripes	Small Cup	Hydrometer
pv_{17}	Large Cup with Stripes	Small Cup with Hole	Thermometer
pv_8	Large Cup with Stripes	Small Cup with Hole	Hydrometer
pv_{11}	Large Cup with Stripes	Small Cup with Stripes	Thermometer
pv_4	Large Cup with Stripes	Small Cup with Stripes	Hydrometer
...	...	...	...

product part order PPO_{pln} in which each of the product parts should move on their respective production lines. Table 4.4 shows the planned product part order PPO_{pln} in which the product parts 1, 2 and 3 should move on their respective production lines. Table 4.4 has been created by replacing the names of the product parts in the planned product part order PPO_{pln} (cf. Table 4.3) with the respective codes for each of these product parts (cf. Table 4.1).

The planned product part order PPO_{pln} is then used during the production process to ensure if the product parts are moving on the production lines in the appropriate and planned order.

4.3 RFID Reader Evaluations

In this section we will describe the different rfid evaluations that we conducted in the lernfabrik. The purpose of these evaluations was to have a benchmark for the performance of these readers in the lernfabrik under different conditions and settings.

We configured the lernfabrik and deployed seven passive modules (production lines) and three active modules (assembly points) to build our production environment. We

Table 4.4: Planned Product Part Order for Day X

Product Part 1	Product Part 2	Product Part 3
ppv_1	ppv_4	ppv_7
ppv_1	ppv_4	ppv_8
ppv_1	ppv_5	ppv_7
ppv_1	ppv_5	ppv_8
ppv_1	ppv_6	ppv_7
ppv_1	ppv_6	ppv_8
ppv_2	ppv_4	ppv_7
ppv_2	ppv_4	ppv_8
ppv_2	ppv_5	ppv_7
ppv_2	ppv_5	ppv_8
ppv_2	ppv_6	ppv_7
ppv_2	ppv_6	ppv_8
ppv_3	ppv_4	ppv_7
ppv_3	ppv_4	ppv_8
ppv_3	ppv_5	ppv_7
ppv_3	ppv_5	ppv_8
ppv_3	ppv_6	ppv_7
ppv_3	ppv_6	ppv_8
...	...	...

configured the production lines in a straight line (one after the other), with an assembly point deployed after every two production lines. We deployed three *vr*s and six RFID readers *pr*s. The *vr*s were Lenovo T61 laptops with 4GB RAM, 100GB harddisk and 2.50 GHz Intel Core 2 Duo processors, whereas the RFID readers *pr*s were Volaré UHF USB readers [Vol13].

We placed RFID tags on each and every product part and then studied the effects of tag placement, multiple tag reads, tag orientation, RFID reader power, and reader interference on the accuracy of RFID readers. The metrics that we evaluated are more or less standard metrics to measure the performance of RFID readers [LW09]. Our studies showed that the accuracy of RFID readers changes with a change in the parameters. As an example the accuracy of the readers decreased if the distance between the reader and the production line was increased, or if the reader power was decreased and so on. In addition to this our studies revealed that two RFID readers from the same manufacturer does not have the same reliability under perfectly similar conditions. The evaluations therefore further highlighted the need to calibrate the RFID reader probabilities, a task that has been discussed at length in Self-Calibration of RFID Reader Probabilities (cf. Chapter 6).

For each of the following experiment we assembled 100 products. The different factors

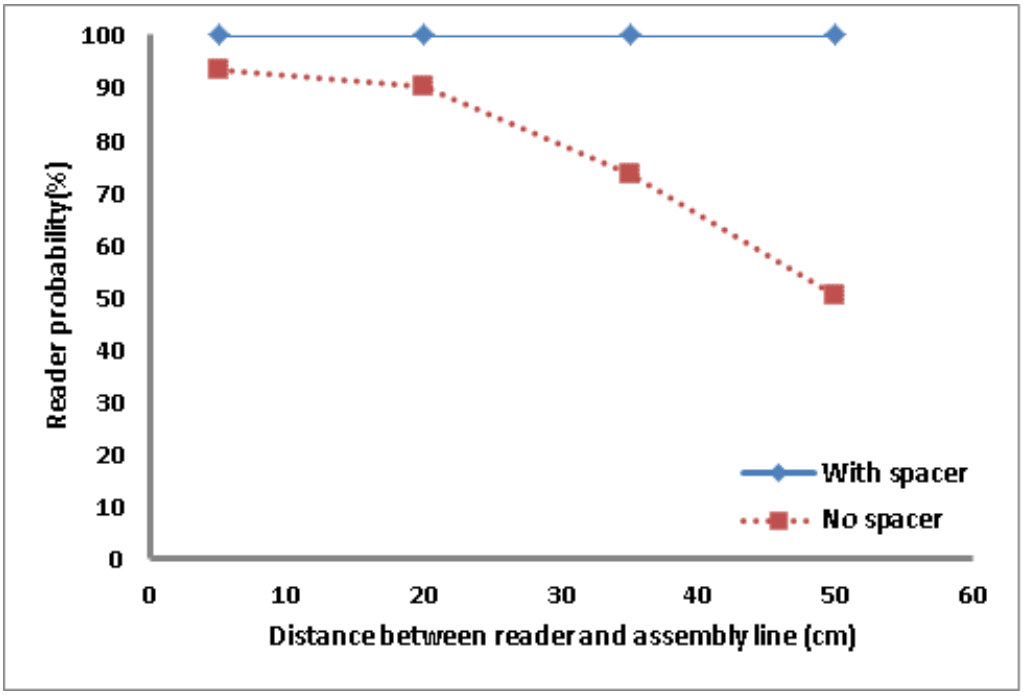


Figure 4.4: Effect of Tag Placement on False Negatives

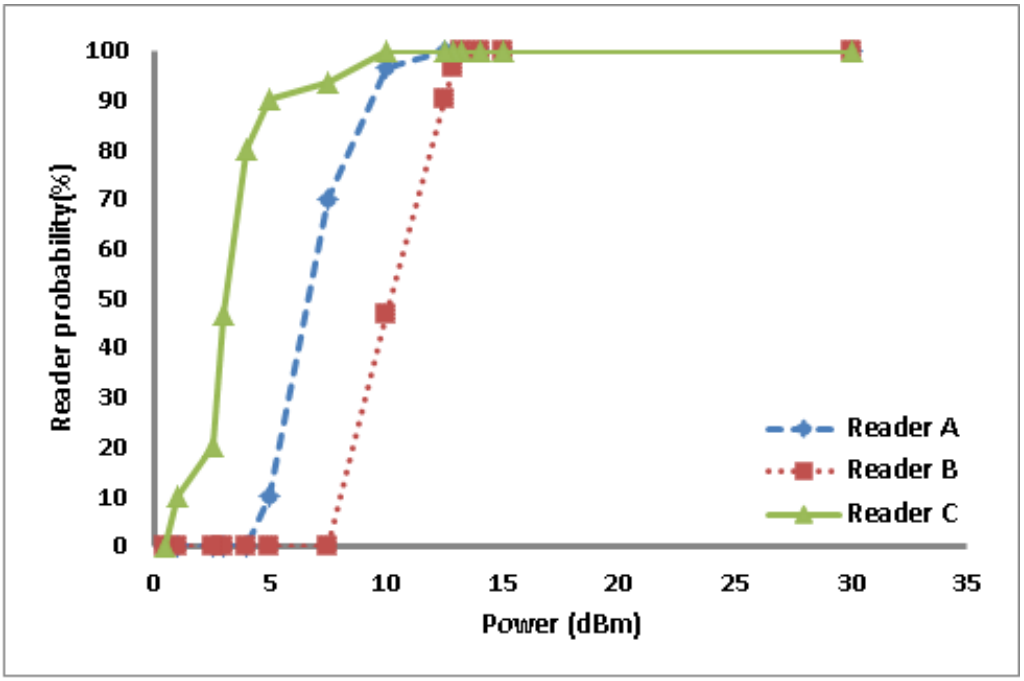


Figure 4.5: Performance of Different Readers when Varying Power Value

that affect the reliability of the physical readers *pr*s have been evaluated below:

Tag Placement: We tested two tag placement settings: using a spacer, which was an insulation material between the tag and the metallic product part, and attaching part of the tag to the product part, while leaving some part unattached. Figure 4.4 shows the effect of each setting on the number of false negatives. As you can see the number of false negatives decreases dramatically when the tags were attached to the items using spacers. For scenarios, where we attached the entire tag to the metal parts, the detections were not possible. In each of the subsequent experiment, we used spacer between the parts and the RFID tags.

Differences among Readers: In this experiment, we tried to test the performance of different *pr*s. Figure 4.5 shows the differences among the *pr*s when the power is varied. The number of missed tags differs from reader to reader. For values less than 13 dBm, RFID Reader C has the lowest number of false negatives for the same power value. These experiments reveal that even RFID readers of the same model and from the same company vary in reliability even when they are operating under exactly the same conditions.

Multiple Object Reads: In this experiment, we evaluated the reliability of RFID readers for multiple product part detections. Figure 4.6 shows the performance of the RFID reader when it has to read one product part as compared to three product parts at the same time. When a single part was used, the number of false negatives was slightly less than the number of false negatives when three parts were used. The number of false negatives could have been higher, if the readers had to detect even more tags simultaneously. However, since in the lernfabrik the product consists of only three parts, we sufficed ourselves to these values. This experiment showed one of the most common and pressing issues with RFID readers i.e. the reliability of RFID readers decreases when the RFID reader has to read multiple RFID tags simultanesouly.

Tag Orientation: In this experiment, we studied the effect of tag orientation on out-of-order readings. We used two different tag orientations. In one setting the product parts were placed on the base plates such that when they move through the production line, the RFID tags would directly face the antenna of the RFID readers deployed on the production line. In the second setting, the product parts were placed randomly on the base plates. The results (cf. Figure 4.7) show that random placement of product parts on the base plates caused a higher number of out-of-order readings, this was because RFID readers read tags more reliably if the tags are facing the readers.

Reader Power: In this experiment we studied the effect of RFID reader power on out-of-order readings. The results (cf. Figure 4.8) show that reducing reader power to 14 dBm eliminated all out-of-order readings. In order to further examine the effect of varying power on the number of out-of-order readings, we used three product parts per baseplate and tried to detect these product parts under four different power settings. Figure 4.8 shows that reducing the RFID reader power reduces the number of out-of-

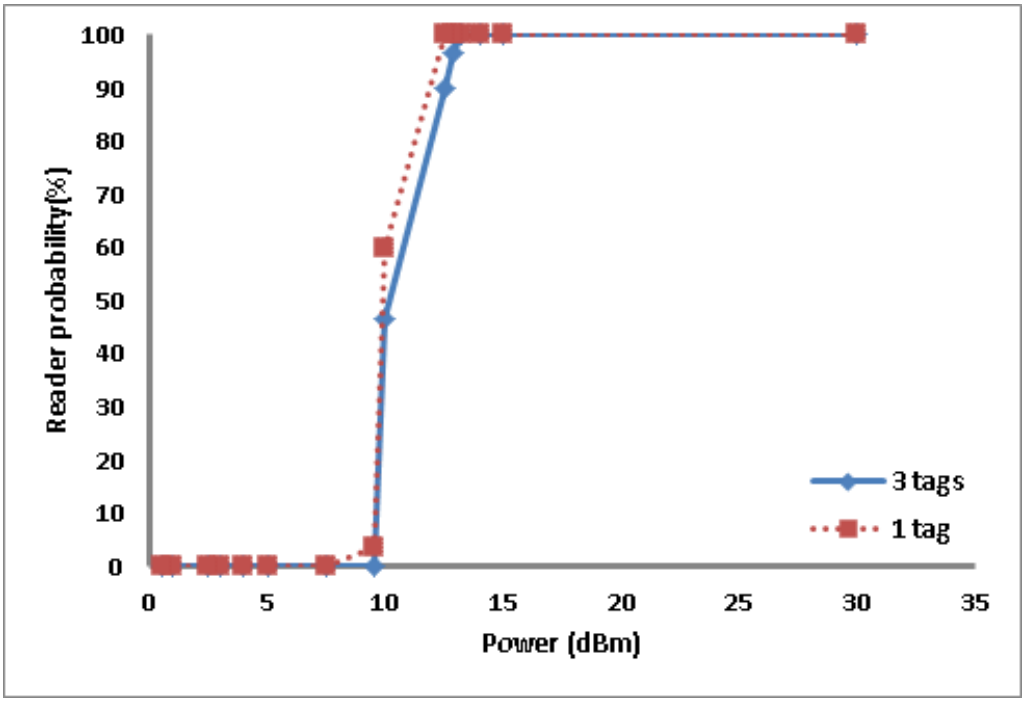


Figure 4.6: Effect of Number of Objects on the Reader's Performance

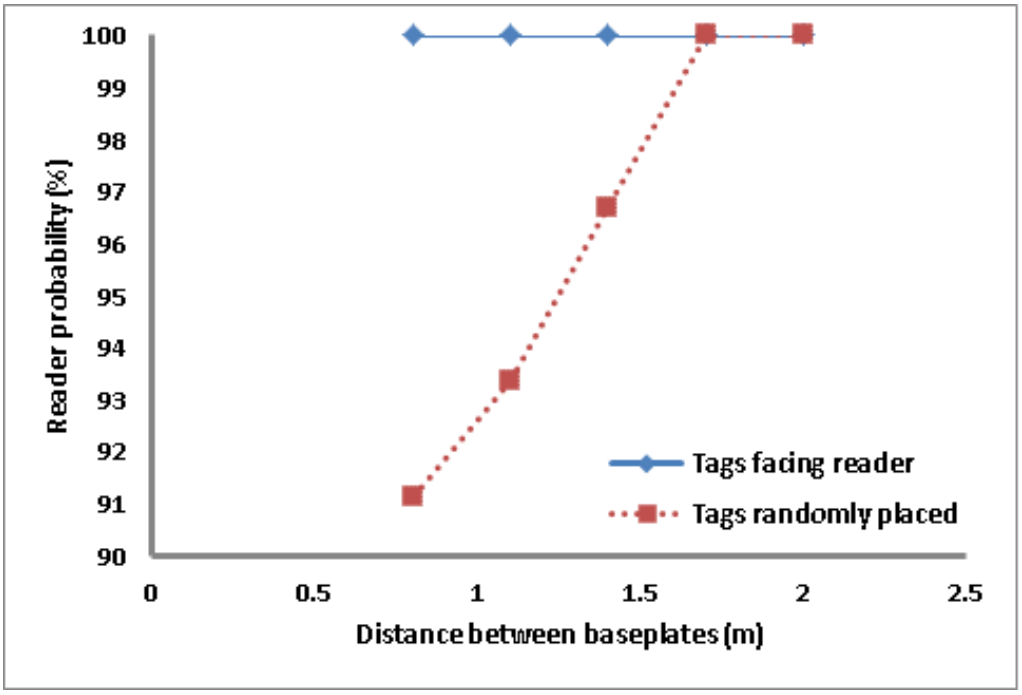


Figure 4.7: Effect of Tag Orientation on the Out-of-Order Readings

order readings. This is because, by reducing the power of an RFID reader its sensing range is reduced, which in turn decreases the number of out-of-order tag reads.

Interfering Readers: In this experiment we studied the effect of an RFID reader within the sensing range of another reader on out-of-order readings. Figure 4.9 shows the results when reader B is deployed before reader A on the production line i.e. ($readerB \sqsubset readerA$). This means that product parts would first be read by reader B and then by reader A. The results show that the reader that is deployed after another reader has a lesser number of out-of-order readings and hence a higher reliability. This is because when a reader (which in this case is reader A) is deployed within the sensing range of another reader, the sensing range of the later reader (i.e. reader A) decreases. The decrease in the sensing range of reader A, in turn has the same effect on out-of-order reads as is observed when we reduce the power of a RFID reader.

Discussion: From the lernfabrik experiments we can conclude that on the one hand we have to increase the reader power in order to avoid missed reads, and on the other hand we must reduce it to minimize the number of out-of-order reads. From the results discussed above, it is obvious that it is hard to find a setting without having some type of errors. The results of these experiments make it abundantly obvious that we need some sort of a middleware to solve the issue of unreliablity in RFID readers. This is exactly what we have endeavoured to do in the subsequent chapters of our work.

4.4 Summary

In this chapter we described the lernfabrik at length. This description also included a discussion of how production is actually planned. In addition to this we also presented the results of RFID reader evaluations that were conducted at the lernfabrik.

The findings of the lernfabrik evaluations are summarized below:

- Tag Placement: We placed RFID tags on metallic parts both directly and using a spacer between the tag and the metallic part. The evaluations showed that tags placed with a spacer have had far higher read accuracy then the ones placed directly on the metal objects.
- Differences among Readers: In this study we tested three RFID readers under the same physical conditions. The results showed that even RFID readers of the same model and from the same manufacturer have different accuracy even when they are operating under the same conditions.
- Multiple Object Reads: In this study we observed that the accuracy of RFID readers was slightly lower when they had to read 3 simultaneous tags, as compared to a setting in which readers had to read just 1 RFID tag at a time. Experiments conducted by other researchers [PSR+06], [Vio05], [HC06] show that this accuracy fall sharply when readers have to read more than five tags simultaneously.

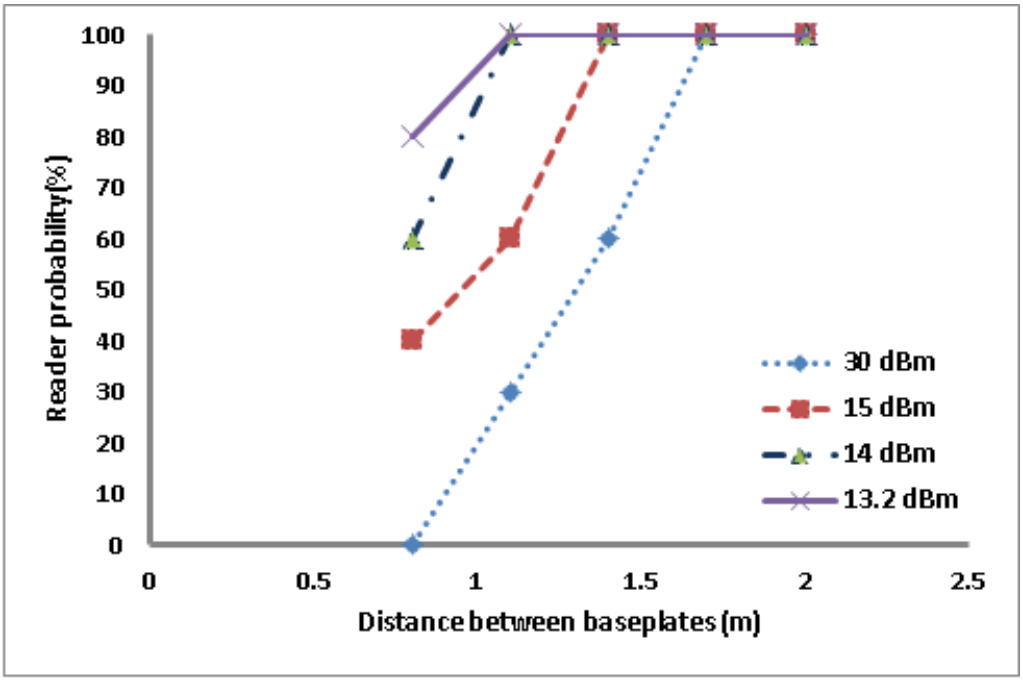


Figure 4.8: Reader Power: Multiple Product Parts

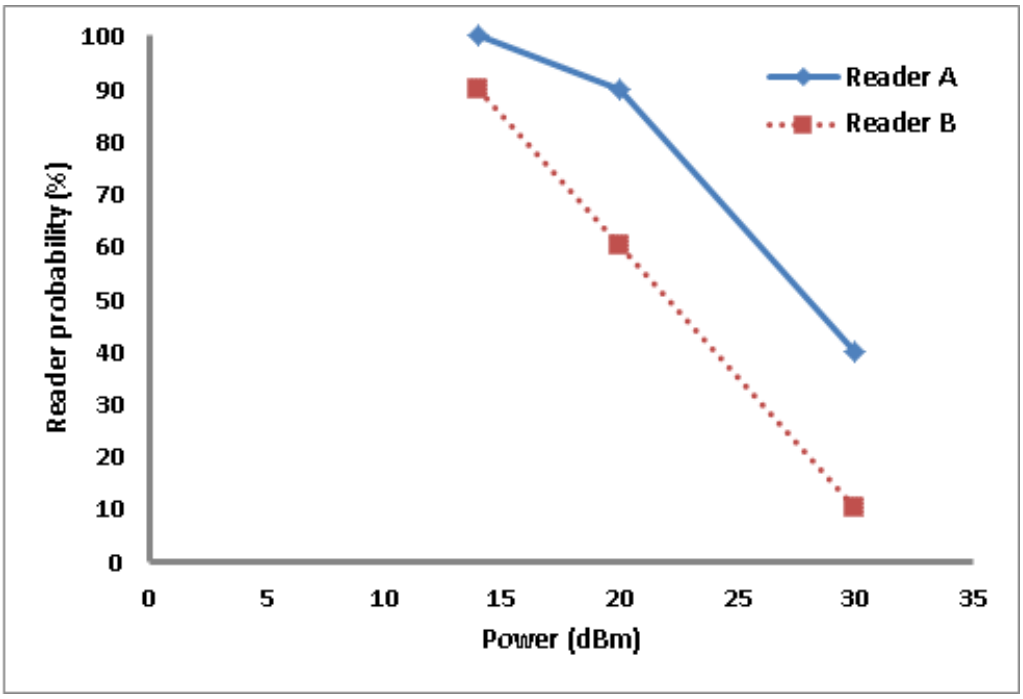


Figure 4.9: Reader B ahead of Reader A

- Tag Orientation: In this study we evaluated two different tag orientation settings. Firstly, the tags were placed on the product parts so that they would face the readers while passing in front of them. Secondly, the tags were placed randomly on the product parts. The evaluations showed that the RFID readers have far higher accuracy when they have to read tags which are directly facing them.
- Reader Power: The reader power evaluations revealed that reducing the reader power reduces the out-of-order readings. This is quiet intuitive as well, since with higher power the read range of an RFID reader increases which results in the reader reading tags out of order.
- Reader Interference: The experiments for reader interference revealed that if two readers are deployed within the read range of each other. The out-of-order readings of the later reader is less than that of the one deployed before it. This is because the signals of the initial readers interferes with the second reader and decreases the read range of the later reader.

The results of RFID reader evaluations provides further motivation for a real-time monitoring framework that not just monitors the production environment but also provide guarantees regarding the reliability of RFID devices.

Chapter 5

RFID Based Consistency Management Framework for Production Monitoring

Manufacturing organizations are now increasingly deploying RFID based systems to gather information regarding the state of production in real-time. Toyota (South Africa) for example has tagged its carriers to streamline manufacturing and vehicle tracking. The tags are intended to remain with the vehicle throughout its life and hold its maintenance history. Harley Davidson has implemented process automation by tagging bins carrying product parts to provide instructions to employees at each stage of the process. Johnson Controls has started using RFID to track cars and truck seats throughout the assembly process. TrenStar tracks their beer kegs to improve demand forecasts and increase efficiency. International Paper tracks their paper roll to reduce lost or misdirected rolls. Gap tracks its denim apparel to improve customer service through better inventory management. Raxel has started tagging reusable plastic biohazard containers to avoid contamination. Michelin on the other hand tags its tyres to comply with the TREAD act and recall management and so on [BR05a]. All the RFID deployments mentioned above were carried out in order to improve shop floor inventory tracking and automate warehouse operations.

The problem with majority of the existing RFID systems is that they merely deal with increasing the efficiency of manufacturing operations by eliminating manual work and replacing it with automatic accounting and inventory management. None of the RFID deployments that we have just discussed, tries to track each and every product part throughout the different stages of production and in doing so detect different production errors as soon as they arise.

Such real-time production monitoring is of utmost importance in a variant production environment and is vital to ensure that variant products are produced correctly and efficiently and production errors are detected as quickly as possible. In this chapter, we present a:

- consistency stack for RFID based production monitoring middlewares/frameworks. The consistency stack is conceptual in nature and categorizes the different consistency issues into separate layers. The basic purpose of presenting the consistency stack is to make an effort to formalize the different consistency issues, such as duplicate readings, missed readings, and false readings etc., that need to be considered by almost all RFID applications.
- probabilistic model for sequence detection. The model assigns probabilities to the product part and product sequence detections and hence provide a measure of how accurate these detections are.
- algorithms and system to improve the accuracy of sequence detections through redundant readings.

The rest of the chapter is structured as follows. The consistency stack is described in Section 5.1. In Section 5.2, the probabilistic sequence detection algorithm is presented followed by its evaluations in Section 5.3. Section 5.4 provides an overview of the related work. Finally, we conclude the chapter with a short summary in Section 5.5.

5.1 Consistency Stack

We have categorized the different consistency issues that may arise in manufacturing environments into a layered model called the **consistency stack**. The consistency stack is comprised of two distinct sub-stacks, the RFID consistency substack and the production consistency substack (cf. Figure 5.1).

- RFID consistency substack (cf. Section 5.1.1) deals with consistency between physical world (for e.g. actual location of a product part) and the world model (i.e. observed location of the product part by our monitoring framework).
- Production consistency substack (cf. Section 5.1.2) deals with consistency between world model (i.e. observed location of the product part) and the production plan.

The RFID consistency substack lies below the production consistency substack because it is necessary to have reliable RFID readings before comparing that data with the planned product part order PPO_{pln}. The RFID consistency substack has been implemented and is deployed on the virtual readers vrs, where as the implementation and evaluation of the production consistency substack is discussed in Chapter 7.

5.1.1 RFID Consistency Substack

The RFID consistency substack addresses the consistency issues of RFID devices that hamper the correct perception of the physical world. These consistency issues arise as a result of the inherent unreliability of RFID readers. The consistency issues within

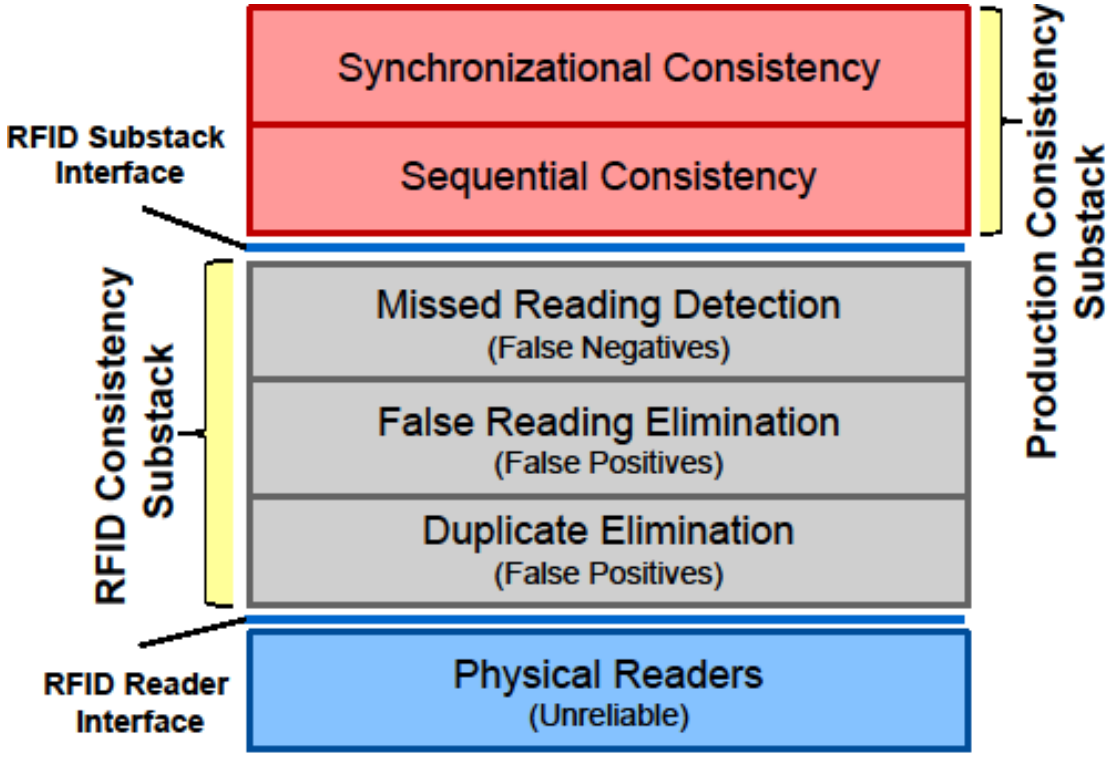


Figure 5.1: Consistency Stack for RFID Deployments in Production Environments

this substack are systematically eliminated using contextual knowledge. In particular, we will show later (cf. Section 5.2) how the knowledge about production paths can be used to derive accurate sequence information.

Duplicate Elimination: RFID readers can have duplicate readings. Duplicate elimination is now often performed by the RFID readers, however [BWL06], [BWL+07] provide techniques to eliminate duplicate readings at the system level. The duplicate elimination layer filters out duplicate readings.

False Reading Elimination: RFID Readers *pr*s can report false readings due to a number of reasons such as incorrect product part detections, and overshooting (reading product parts moving on neighboring production paths) etc. In manufacturing environments, product parts cannot change their production paths i.e. randomly jump from one path onto another. We use this knowledge of production paths and product part routes to eliminate false readings.

Missed Reading Detection: RFID readers are unreliable and hence may miss out on detecting certain product parts. In our implementation, we deploy multiple *pr*s on a production path. Since product parts can not skip a location on a production path, even if a *pr* misses out on a product part, it is detected by other *pr*s deployed on that production path. Section 5.2 describes in detail how missed product parts are detected

and probabilities are assigned to product part detections.

RFID Substack Interface: The RFID substack interface supports location queries such as location of a product part on the production path, location at which the sequence of product parts got changed and so on. The interface masks physical errors that may arise as a result of unreliable RFID devices and provides probabilistic results.

5.1.2 Production Consistency Substack

The production consistency substack addresses the consistency issues that result in improper products being manufactured. As a result of these issues the products being assembled differ from the planned product specifications. These consistency issues may arise as a result of mechanical or human faults. The consistency issues within this substack can be detected using domain knowledge, such as production plans, product specifications, planned product part order etc. Unlike the RFID consistency substack that eliminates the issues that arise as a result of RFID devices, the production consistency substack merely detects and reports issues to the applications, and the actual task of resolving the issues is left to the applications.

The production consistency substack is further sub-divided into two layers: the sequential consistency layer and the synchronizational consistency layer. The sequential consistency layer lies below the synchronizational consistency layer, because product parts that are not in desired sequence can never be synchronized with other product parts. In other words, sequential consistency is needed to achieve synchronizational consistency.

Sequential Consistency: During manufacturing, product parts should move through the production paths in certain defined sequences in order to be assembled with other product parts correctly. Deviation of actual product part sequence PPO_{act} which is the sequence of product parts on the production line from the planned sequences PPO_{pln} is referred to as sequential inconsistency. Sequential inconsistencies can arise as a result of mechanical or human faults, that may alter the sequence of objects. The sequential consistency layer uses information from the production plan to detect sequential inconsistencies.

Synchronizational Consistency: During manufacturing, product parts should arrive at the assembly points within certain defined time spans in order to be correctly assembled with other sub-components i.e. product parts should reach assembly points in sync with their corresponding product parts. Mechanical failures or human errors may cause delays in some areas of a factory, which results in synchronizational issues at some assembly points. The synchronizational consistency layer ensures that product parts moving through the production paths are in sync and will reach the assembly points within the required time span to be assembled together with their respective sub-components.

In the rest of this chapter we would discuss the specific functionality of RFID consistency substack (i.e. reliable detection of product parts and product part sequences). To accomplish this we have designed a sequence detection algorithm that eliminates false positives and negatives in RFID readings based on a probabilistic model (cf. Section 5.2). The functionality of the production consistency substack (i.e. reliable detection of production issues and errors) is discussed in the RFID-based complex event processing chapter (cf. Chapter 7)

5.2 Probabilistic Sequence Detection

In this section we present the concepts for probabilistic sequence detection of product parts moving on the production lines. First we present the probabilistic sequence model, that is used to detect the product part sequences (cf. Section 5.2.1). Then we provide an overview of our approach (cf. Section 5.2.2). After that we describe the actual probabilistic sequence detection algorithm (cf. Section 5.2.3). Finally, we explain how we detect changes to product part sequences on the production lines (cf. Section 5.2.4).

5.2.1 Probabilistic Sequence Model

Due to the inherent unreliability of RFID devices, there is a possibility that different *prs* may deduce different product part sequences. In order to resolve this issue, we assign probabilities to the deduced sequences, which serve as a measure of confidence in the correctness of deduced sequences.

A *partial sequence ps* ($o_i < o_j$) implies that o_i is directly ahead of o_j on the production path such that there exists no $o_k : (o_i < o_k < o_j)$. In contrast an *extended sequence es* ($o_i << o_n$) implies that o_i is ahead of o_n on the production path such that there exists an $o_k : (o_i < \ldots o_k \ldots < o_n)$. *Non-unique partial sequences* are partial sequences of the form ($o_i < o_j$) and ($o_i < o_k$), which implies that o_i is directly ahead of both o_j and o_k. Since product parts move in sequence on the production paths, non-unique partial sequences cannot exist in reality. However, they can arise as a result of different inconsistent readings, which have already been explained previously (cf. Section 3.4). One of the goals of the sequence detection algorithm is to resolve such inconsistencies using multiple readings for partial sequences, and assigning probabilities to detected sequences.

The *probability of a sequence* denotes the probability with which a deduced sequence (partial or extended) matches reality. The probability of a deduced partial sequence ps is denoted by $p(ps)$, whereas the probability of a deduced extended sequence is denoted by $p(es)$

The goal of our algorithm is to determine the sequence of product parts ($o_i < \ldots o_k \ldots < o_n$) moving through the production paths. Each sequence has an associated probability, which enables applications to query the real-time production monitoring framework for the most probable current sequence.

5.2.2 Overview

In this section we will give a brief overview of our approach. The core steps of the probabilistic sequence detection algorithm are as follows:

1. Detect all possible partial sequences ps. The outcome of this step are multiple possible ps, which are stored in a partial sequence list PSL, which contains all the possible ps detected at a particular virtual reader vr. If there are n product parts on the production line, we would have $n-1$ partial sequences in reality. As an example, for four product parts o_1, o_2, o_3, and o_4, we would have three partial sequences $(o_1 < o_2)$, $(o_2 < o_3)$, and $(o_3 < o_4)$.
2. Assign probabilities to the detected partial sequences. The probability assignment in our algorithm works in such a way that the probabilities of partial sequences that exist in reality would increase over time, whereas that of false or non-unique partial sequences would decrease over time.
3. Detect all possible extended sequences es. The outcome of this step are multiple possible es. Unlike partial sequences, only one possible extended sequence can exist in reality. As an example, for four product parts o_1, o_2, o_3, and o_4, we would have only one extended sequence in reality i.e. $(o_1 < o_2 < o_3 < o_4)$.
4. Assign probabilities to the detected extended sequences. Due to unreliability of RFID readers, we would have non-unique partial sequences, which in turn would lead to multiple extended sequences. However, over time the probability of the extended sequence that would match the actual sequence of product parts on the production line would be higher then all other extended sequences. Hence, we would eventually have the most probable extended sequence of product parts moving on the production line.

5.2.3 Sequence Detection Algorithm

Sequence detection algorithm is divided into three sub-sections. In the data structures section, we describe all the data structures that are used in the algorithm. Then we go on to describe the partial sequence detection algorithm and finally we conclude by explaining the extended sequence detection algorithm.

Data Structures:

- event $e_i = (o_i, pr_i, t_i)$: Read event e_i by pr_i.

- $events_{pr_i} = (e_1, e_2, ... e_n)$: Set of read events of pr_i.
- $events_{vr_i} = (e_1, e_2, ... e_m)$: Set of read events at vr_i.
- *productpart* $o_i = (o_i, ttl)$: Product part o_i is associated with its id and a time to live ttl variable.
- Product part list $PPL = (o_1, o_2, ... o_n)$: Product part list contains all the product parts detected at the vr_i.
- Partial sequence $ps_i = ((o_i < o_j), events, p(ps_i), ttl)$: Each partial sequence ps_i contains a partial sequence of the form $(o_i < o_j)$, the events that led to the detection of the partial sequence, the probability with which the partial sequence was detected and a time to live ttl variable that shows how recently the sequence was detected.
- Partial sequence list $PSL = (ps_1, ps_2, ... ps_n)$: Partial sequence list PSL contains all the partial sequences ps detected at the vr_i.
- Extended sequence $es_i = ((o_i < o_j < o_k ... < o_n), PSL_e, p(es_i), ttl)$: Each extended sequence es_i contains the extended sequence, the partial sequences ps that led to the deduction of the extended sequence stored in PSL_e, the probability with which the extended sequence was detected and a time to live ttl variable for the extended sequence.
- Extended Sequence List $ESL_{[n]} = (es_1, es_2, ... es_m)$, where n denotes the length of the extended sequence: Extended sequence list ESL contains all the extended sequences that were detected at the vr_i. We maintain a separate extended sequence list for sequences of a specific length. As an example, extended sequences of length 3 would be stored in $ESL_{[3]}$, where as extended sequences of length 4 would be stored in in $ESL_{[4]}$

Partial Sequence Detection:

The sequence detection algorithm runs on the vrs. After every read event e by a pr, we try to determine the partial sequence ps of the product part o that was read. If a ps is deduced, we compute the probability of the ps and try to extend the ps to create an es. The es is also assigned a probability.

Partial sequence ps detection involves detecting sequence duplets of the form $(o_i < o_j)$. A ps is detected using the following three rules:

1. Rule 1: If pr detects a product part o_i at time t_i and then detects product part o_j at a later time t_{i+k}, then o_i is ahead of o_j on the production path, i.e. $(o_i < o_j)$.
2. Rule 2: If pr_i is deployed before pr_j on the production path i.e. $(pr_i \sqsubset pr_j)$, then a product part o_i detected by pr_j at time t_i is ahead of product part o_j detected by pr_i at the same time t_i, i.e. $(o_i < o_j)$.

Algorithm 1 Partial Sequence Detection Algorithm

Let $e_i = (o_i, pr_i, t_i)$ be a read event detected at vr_i
PPL;
PSL;
ESL;
while pr_i detects o_i **do**
  ps = PartialSequenceDetection(o_i);
end while
for function ps PartialSequenceDetection(o_i) **do**
  if o_i != new productpart **then**
    $PPL.o_i.ttl == 1$;
  else {o_i == new productpart}
    $o_i.ttl == 1$;
    $events_{pr_i} \leftarrow o_i$;
    $PPL \leftarrow o_i$;
    Detect partial sequence ps using sequence detection rules
    if partial sequence ps_i is deduced **then**
      if ps_i == new sequence **then**
        $ps_i.ttl == 1$;
        $ps_i.p(ps) \leftarrow$ **ComputePSProbability(**ps_i**)**;
        $PSL \leftarrow ps_i$;
        return ps_i;
      else {sequence ps_i != new sequence}
        there exists a $PSL.ps_h$ such that $PSL.ps_h == ps_i$;
        $PSL.ps_h.ttl == 1$;
        $PSL.ps_h.o_1.addevents(ps_i.o_1.readevent)$;
        $PSL.ps_h.o_2.addevents(ps_i.o_2.readevent)$;
        $PSL.ps_h.p(ps) \leftarrow$ **ComputePSProbability(**ps_h**)**;
        return $PSL.ps_h$;
      end if
    end if
  end if
end for
for function $p(ps)$ ComputePSProbability(ps) **do**
  $p(ps) = (1 - [(1-p)^{ps.ne_{o_1}} + (1-p)^{ps.ne_{o_2}} - (1-p)^{ps.ne_{o_1} + ps.ne_{o_2}}])$;
  return $p(ps)$;
end for

3. Rule 3: If pr_i is deployed before pr_j on the production path i.e. ($pr_i \sqsubset pr_j$), then a product part o_i detected by pr_j at time t_i is ahead of product part o_j detected by pr_i at a later time t_{i+k}, i.e. ($o_i < o_j$).

Whenever a product part o_i is read by pr_i, it is placed in the respective $events_{pr_i}$. The product part o_i is also added to the *product part list* (PPL). Each product part has an associated time to live (ttl) variable, which shows how recently the product part was detected. Whenever a product part is detected its ttl is refreshed and set to '1'. This ttl is then decreased over time. If the product part is not detected again before its ttl becomes 0, it is removed from the PPL. If the product part is detected before its ttl becomes 0, its ttl is refreshed to '1' once again. (cf. Algorithm 1 line 8-14).

For every read event of a product part, we try to deduce the partial sequence ps of the product part with the previously detected product parts using the rules discussed above.

Each deduced partial sequence ($o_i < o_j$) is associated with an event list that contains the read events that led to the deduction of the ps, along with the probability with which this partial sequence was deduced and a time to live ttl variable that shows how recently the ps was deduced. So basically we have two ttl variables, one associated with product parts and the other with the deduced partial sequences. Once a ps is detected, it is added to the *partial sequence list* (PSL), which contains all the ps detected at a vr. (cf. Algorithm 1 line 15-21).

If the newly read product part is already present in the PPL, the ttl of the product part is refreshed to '1' (cf. Algorithm 1 line 9-10). Similarly if the deduced ps is already present in the PSL, the ttl of the ps is refreshed to '1'. In addition to this the events that led to this recent deduction of the ps are added to the event list of the ps, and the probability of the ps is re-computed (cf. Algorithm 1 line 22-32).

The probability of a partial sequence ($o_1 < o_2$) is computed as:

$p(ps) = 1 - [(1-p)^{ne_{o_1}} + (1-p)^{ne_{o_2}} - (1-p)^{ne_{o_1}+ne_{o_2}}]$

where ne_{o_1} is the number of read events for o_1 and ne_{o_2} is the number of read events for o_2. As long as at least one read event for o_1 and one read event for o_2 is correct, we can make a statement about the partial sequence ($o_1 < o_2$). Only if, either all read events for o_1 are incorrect $(1-p)^{ne_{o_1}}$ or all read events for o_2 are incorrect $(1-p)^{ne_{o_2}}$, we cannot derive a sequence. In order to not count the incorrect read events twice, we subtract $(1-p)^{ne_{o_1}+ne_{o_2}}$. This gives us the probability that of no correct reading for ($o_1 < o_2$). Subtracting this probability from 1 gives us the probability of all the cases where at least one correct reading for o_1 and at least one correct reading for o_2 is included in the set of readings.

A product part o_i is removed from PPL if it times out, i.e. if its ttl becomes 0. Once a product part o_i is removed from the PPL, all the ps in which product part o_i was participating are also removed from the PSL. In addition to this, a ps is also removed from the PSL, if it times out, i.e. if the ttl of the partial sequence ps becomes 0.

After every cycle, a vr_i sends its PSL to vr_{i+1}. Upon receiving the PSL from vr_{i-1}, vr_i merges the PSL of vr_{i-1} with its own PSL. The ps deduced at vr_{i-1} that are not deduced at vr_i as yet are discarded because the PSL of vr_i should only contain the ps deduced within the vicinity/range of vr_i. The ps deduced at vr_{i-1} that are also present in the PSL of vr_i are merged together by combining the event lists of the two ps, re-computing their probability and setting the ttl of the ps to the recent value.

Extended Sequence Detection:

Whenever a ps is added to the PSL, we try to extend it to create an extended sequence es. Partial sequences can be extended if they are transitive i.e. for ps $(o_i < o_j)$ and $(o_j < o_k)$ we can create an extended sequence es $(o_i << o_k) = (o_i < o_j < o_k)$ (cf. Algorithm 2 line 18-34).

Upon determination of transitivity in partial sequences, the extended sequence es is added to the *extended sequence list* (ESL). Whenever an extended sequence is updated as a result of transitivity determination, we compare it with all the other extended sequences within the ESL to determine if a transitivity relationship now exists between this recently updated extended sequence and the other extended sequences in the ESL. If transitivity is determined between two es, they are removed from the ESL and the new es created by extending the two transitive extended sequences is added to the ESL (cf. Algorithm 2 line 3-16).

The PSL can contain non-unique partial sequences of the form $(o_i < o_j)$ and $(o_i < o_k)$. However, once transitivity is determined between $(o_i < o_j)$ and $(o_j < o_k)$, the partial sequence $(o_i < o_k)$ is removed from the PSL. Whenever a ps is removed from the PSL as a result of being timed out or determination of non-uniqueness, all the es containing the ps are also removed from the ESL.

Each *extended sequence list* (ESL) has an order denoted as $(ESL_{[n]})$, where [n] depicts the length of the sequences stored in that particular ESL. As an example $(ESL_{[3]})$ will only contain extended sequences of length 3, where as $(ESL_{[4]})$ will store extended sequences of length 4. When a new extended sequence is created, it is stored in the ESL corresponding to the length of this new extended sequence.

An *extended sequence list* $(ESL_{[n]})$ of a particular length can contain multiple extended sequences. As an example consider *extended sequence list* $(ESL_{[3]})$ having two sequences es_1 $(o_1 < o_2 < o_3)$ having a probability of 0.9 and es_2 $(o_1 < o_3 < o_2)$ having a probability of 0.5. When an *extended sequence list* has multiple extended sequences, the sequence with the highest probability is considered to be the extended sequence on the production line.

The probability of an extended sequence es of three product parts o_1, o_2, o_3 $(o_1 << o_3) = (o_1 < o_2 < o_3)$ is computed as:

$p(es) = 1 - [(1-p)^{ne_{o_1}} + (1-p)^{ne_{o_2}} + (1-p)^{ne_{o_3}} - (1-p)^{ne_{o_1}+ne_{o_2}} - (1-p)^{ne_{o_1}+ne_{o_3}} - (1-p)^{ne_{o_2}+ne_{o_3}} - 2 * (1-p)^{ne_{o_1}+ne_{o_2}+ne_{o_3}}]$

Algorithm 2 Extended Sequence Detection Algorithm

1: PSL;
2: ESL;
3: **for procedure** ExtendedSequenceDetection(ps_i) **do**
4: **while** vr_i deduces a partial sequence ps_i **do**
5: **for all** ps in PSL **do**
6: **if** $PSL.ps.o_2 == ps_i.o_1$ **then**
7: $es_i \leftarrow PSL.ps.append(ps_i)$;
8: $es_i.p(es) \leftarrow$ **ComputeESProbability(**es_i**)**;
9: ComputeESTransitivity(es_i);
10: **else** $\{ps_i.o_2 == PSL.ps.o_1\}$
11: $es_i \leftarrow ps_i$.append($PSL.ps$);
12: $es_i.p(es) \leftarrow$ **ComputeESProbability(**es_i**)**;
13: ComputeESTransitivity(es_i);
14: **end if**
15: **end for**
16: **end while**
17: **end for**
18: **for procedure** ComputeESTransitivity(es_i) **do**
19: **for** every es_n in ESL **do**
20: **if** $ESL.es_n.o_n == es_i.o_1$ **then**
21: $es_{new} \leftarrow ESL.es_n.append(es_i)$;
22: ESL.remove(es_n);
23: $ESL \leftarrow es_{new}$;
24: $ESL.es_{new}.p(es) \leftarrow$ **ComputeESProbability(**es_{new}**)**;
25: **else** $\{es_i.o_n == ESL.es_n.o_1\}$
26: $es_{new} \leftarrow es_i.append(ESL.es_n)$;
27: ESL.remove(es_n);
28: $ESL \leftarrow es_{new}$;
29: $ESL.es_{new}.p(es) \leftarrow$ **ComputeESProbability(**es_{new}**)**;
30: **else** $\{es_i$ not transitive with any es_n in $ESL\}$
31: $ESL \leftarrow es_i$;
32: **end if**
33: **end for**
34: **end for**
35: **for function** $p(es)$ ComputeESProbability(es) **do**
36: $p(es) = 1 - [\sum_{i=1}^{N}(1-p)^{es.ne_{o_i}} - (\sum_{j=1,k=1}^{N}(1-p)^{es.ne_{o_j}+es.ne_{o_k}} - (N-1) * (1-p)^{\sum_{l=1}^{N} es.ne_{o_l}}]$;
37: **return** $p(es)$;
38: **end for**

where ne_{o_1} is the number of read events for o_1, ne_{o_2} is the number of read events for o_2 and ne_{o_3} is the number of read events for o_3. As long as at least one read event for o_1, o_2 and o_3 is correct, we can make a statement about the extended sequence $(o_1 < o_2 < o_3)$. We can not derive a sequence, if all the read events for o_1 are incorrect (i.e. $(1-p)^{ne_{o_1}}$) or all the read events for o_2 are incorrect (i.e. $(1-p)^{ne_{o_2}}$) or all the read events for o_3 are incorrect (i.e. $(1-p)^{ne_{o_3}}$).

In order to not count the incorrect read events twice, we would have to subtract the cases where both the read events for o_1 and o_2 were incorrect $(1-p)^{ne_{o_1}+ne_{o_2}}$, the case where both the read events for o_1 and o_3 were incorrect $(1-p)^{ne_{o_1}+ne_{o_3}}$, the case where both the read events for o_2 and o_3 were incorrect $(1-p)^{ne_{o_2}+ne_{o_3}}$ and the cases where the read events for all of o_1, o_2, o_3 were incorrect $2*(1-p)^{ne_{o_1}+ne_{o_2}+ne_{o_3}}$.

This gives us the probability of no correct reading for $(o_1 < o_2 < o_3)$. Subtracting this probability from 1 gives us the probability of all the cases where at least one correct reading for o_1, o_2 and o_3 is included in the set of readings.

Similarly the probability of an extended sequence of n objects $(o_1 << o_n)$ is computed as:

$p(es) = 1 - [\sum_{i=1}^{N}(1-p)^{ne_{o_i}} - (\sum_{j=1,k=1}^{N}(1-p)^{ne_{o_j}+ne_{o_k}} - (N-1)*(1-p)^{\sum_{l=1}^{N} ne_{o_l}}];$

$\forall j,k \in [1,N]$ and $j \neq k$, and $j \neq z$ where $z \in [1,j]$

5.2.4 Change of Sequence Detection

Product part sequences can change due to human intervention or other mechanical reasons. The objective of change detection is to reflect changes in deduced sequences as quickly as possible, once they occur. A sequence can change as a result of *product part removal* or *sequence reversal*. In product part removal scenario a product part is removed or gets dropped off the production path resulting in $(o_i < o_j)$ getting changed to $(o_i < o_k)$. In sequence reversal scenario a human picks up a product part and places it ahead or behind other product parts. This intervention results in one or more *ps* getting reversed i.e. $(o_i < o_j)$ getting changed to $(o_j < o_i)$.

If a product part is removed from the production path, its *ttl* will decrease over time. Once the product part times out (i.e. its *ttl* becomes 0), it will be removed from the PPL. Once the product part is removed from the PPL, all the *ps* in which the product part was participating are also removed from the PSL, along with corresponding extended sequences from the ESL.

If a sequence is reversed, the old *ps* will not be deduced again and will time out (i.e. its *ttl* will become 0), whereas the probability of the newly deduced *ps* will increase over time. Once the old *ps* times out, it is removed from the PSL and all corresponding *es* are removed from the ESL as well.

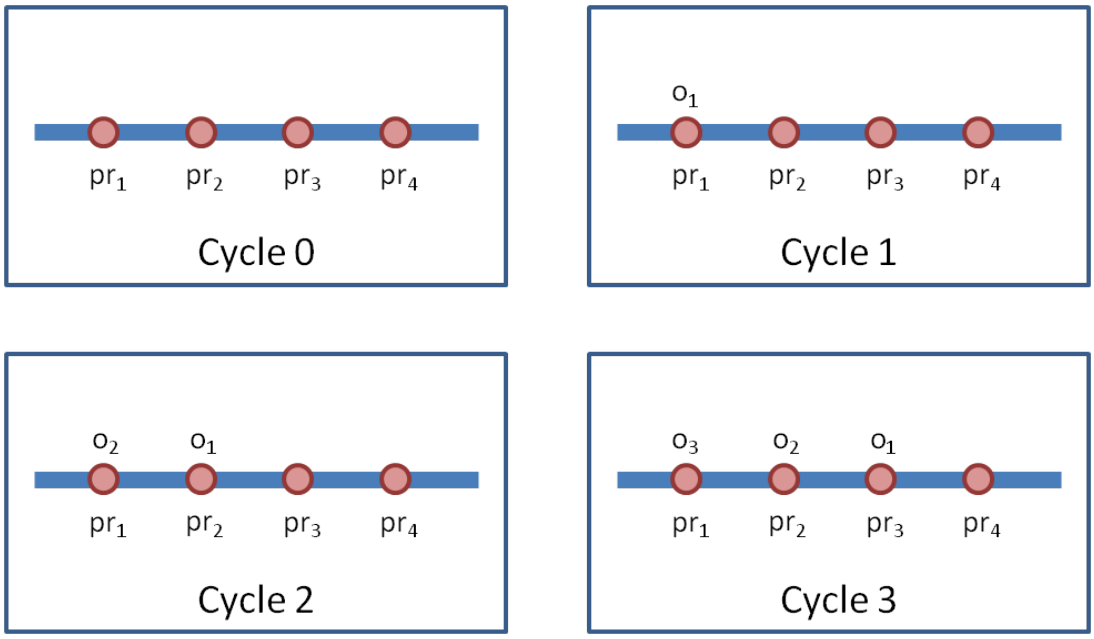


Figure 5.2: Simulation Cycle

5.3 Evaluations

In this section, we evaluate the performance of the probabilistic sequence detection algorithm with respect to the accuracy with which the algorithm detects partial and extended sequences. Simulations were performed using PeerSim [JMJV09], a large-scale P2P discrete event simulator.y

5.3.1 Simulation Setup

Accuracy in our context is defined as the number of sequences correctly determined in the presence of missed readings. Missed readings indicate that a reader is unable to detect a product part. All the simulations are performed with 8,192 physical readers *prs* distributed across 1024 nodes, where each node in our context is a virtual reader *vr*. The primary performance metric for the evaluations is the time (in cycles) it takes for the algorithm to detect a certain percentage of the total deduced sequences with a probability of 90% or more.

Simulation Cycle: Before moving on to discuss the different evaluations, it is pertinent to first describe a simulation cycle. We have already explained in the system model (cf: Chapter 3.2) that during any production run, the product parts move on the production lines in a cyclic manner i.e. when a production is in progress a new product part enters the production line every x seconds. The Peersim simulator can be operated in cycle or event mode. In the event mode, computations are performed

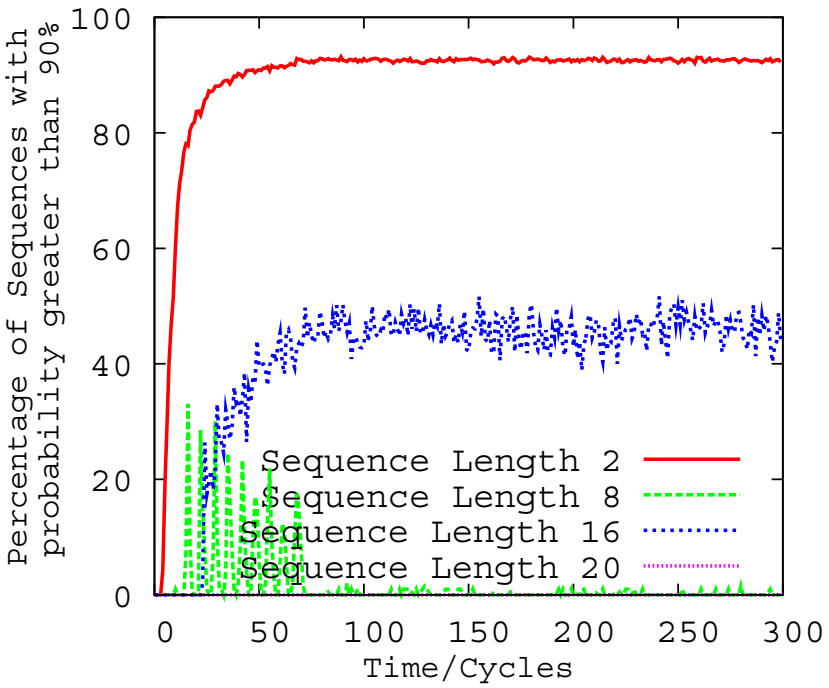
whenever an event is triggered, where as in cycle mode, computations are performed on each an every cycle. All of our experiments in this dissertation have been carried out on Peersim with the simulator being run in cycle mode.

A simulation cycle in our experiments is the time it take for a product part to move from one physical reader onto the next. Furthermore, during every cycle the product parts already on the production line move ahead by one physical reader respectively. In order to further explain this, lets consider a concrete example. Figure 5.2 shows a production line with four physical readers deployed on it. Before the simulation starts, there are no product parts moving on the production line. But as the production run starts, during the first cycle, product part o_1 is in front of physical reader pr_1. During the second cycle, o_1 moves forward and is now in front of physical reader pr_2, where as a new product part o_2 comes onto the production line and is now in front of physical reader pr_1. Similarly during the third cycle, both o_1 and o_2 moves forward and a new product part o_3 comes on the production line and is now infront of physical reader pr_1.

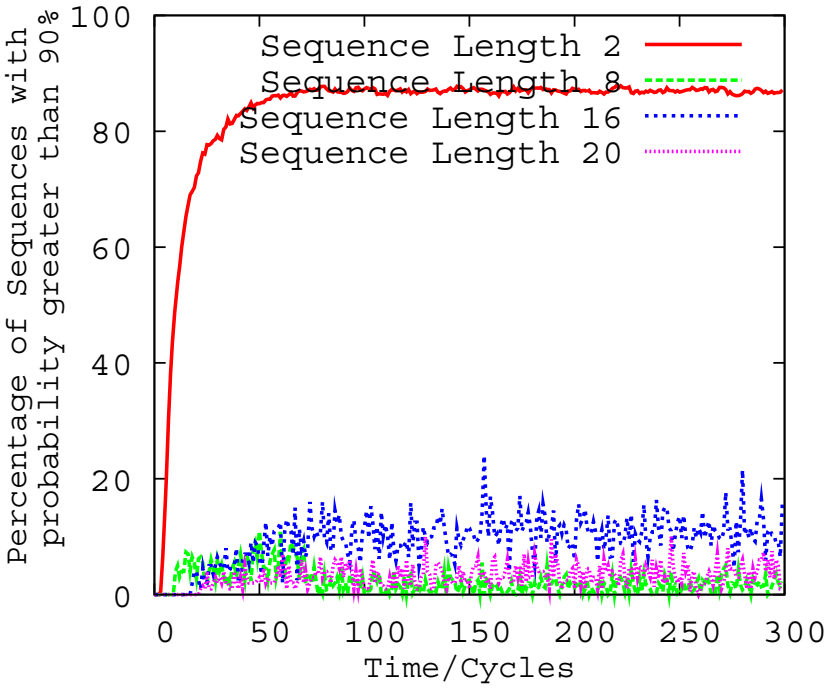
5.3.2 Impact of Physical Reader Distribution

In this scenario, the impact of distribution of prs within the vrs is evaluated. The read probability $p(e)$ of each pr is set to 0.7. Figure 5.3(a) shows the performance of the algorithm, when prs are distributed uniformly amongst the vrs i.e. each vr contains 8 prs, whereas Figure 5.3(b) shows the performance of the algorithm when prs are distributed amongst the vrs using a Zipf-like distribution with $\alpha = 1.0$ i.e. 80% of the prs are distributed across half of the vrs, whereas the remaining 20% of the prs are distributed across the remaining half of the vrs.

Uniform Distribution of Physical Readers: In uniform distribution scenario (cf. Figure 5.3(a)), more than 90% of the ps i.e. sequences of length 2 reach a probability of 90% or above in less than 12 cycles. 100% of the ps can never have a probability of 90% or above because we always have a small percentage of non-unique partial sequences, which have low probabilities. The percentage of ps having a probability of 90% or above stablizes at the 90% mark around 50^{th} cycle indicating that the remaining ps in the system are non-unique partial sequences. The es of length 8 and 16 are detected late because a new product part enters the production line on each cycle, therefore o_{16} enters the production line on the 16^{th} cycle. The first es of length 8 and 16 are deduced with a probability greater than 90% around the 9^{th} and 18^{th} cycle. The es of length 8 has very few deductions after 60^{th} cycle. This is because es of length 8 are consumed i.e. removed from ESL, since they start participating in the deduction of es of longer lengths. The longest es in the ESL has a length of 16, this is because earlier objects get timed out as a result of leaving the vr. Only around 40% of the es in the system reach a probability of 90% or above. This is in line with reality because unlike ps, only a few es are probable and only one can exist in reality.



(a) Uniform Distribution



(b) Zipf Distribution

Figure 5.3: Impact of Physical Reader Distribution

Zipfian Distribution of Physical Readers: In the zipfian distribution scenario (cf. Figure 5.3(b)) the probabilities of *ps* and *es* are far lower than the uniform distribution scenario. This is because due to the skew in distribution of *pr*s, some *vr*s have more *pr*s than are needed to accurately deduce sequences, while the other *vr*s may not have enough *pr*s to accurately deduce sequences.

5.3.3 Probabilistic Detection of Partial and Extended Sequences

Previous scenario (cf. Section 5.3.2) showed that the probabilistic sequence detection algorithm works better with *pr*s uniformly distributed across *vr*s. In this scenario, we keep the distribution of *pr*s uniform across the *vr*s, and further evaluate the confidence with which the probabilistic sequence detection algorithm deduce partial and extended sequences. The read probability $p(e)$ of *pr*s was set to 0.7.

For the partial sequences (cf. Figure 5.4(a)), our algorithm was able to deduce more than 90% of all *ps* with a probability of 90% or above. Whereas for the extended sequences of length 16 (cf. Figure 5.4(b)), the sequence detection algorithm was able to deduce around 80% of the *es* with a probability of 50% or above, around 60% of the *es* with a probability of 70% or above, and around 40% of the *es* with a probability of 90% or above. In other words, as the probability increases, the number of extended sequences decreases. This is perfectly inline with reality since only one extended sequence actually exist on the production line.

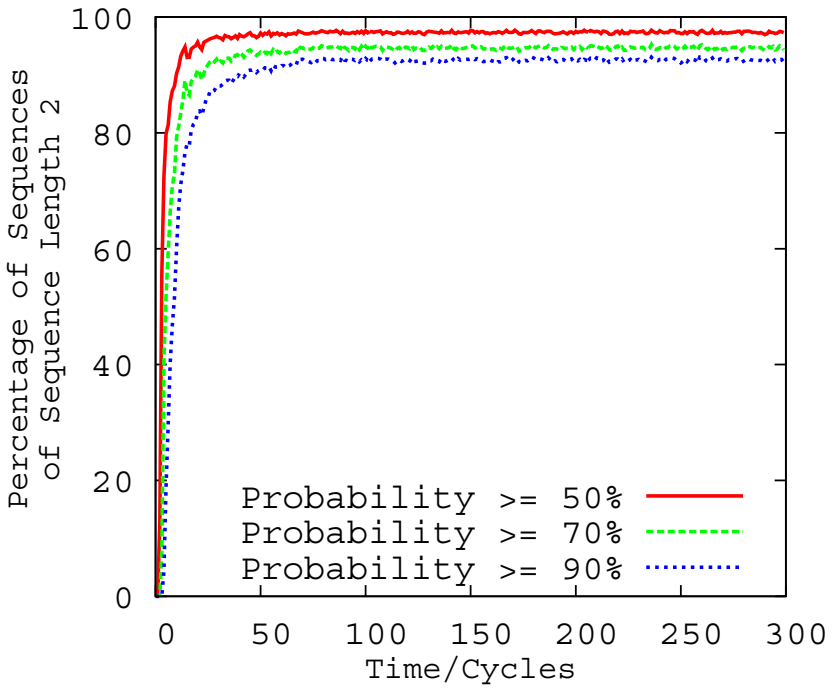
5.3.4 Influence of Physical Reader Reliability

In this scenario, we evaluated the influence of physical reader reliability on the probabilistic sequence detection algorithm. The *pr*s were uniformly distributed amongst the *vr*s and the evaluations were conducted by setting the read probability $p(e)$ of *pr*s to 0.5, 0.7, and 0.9.

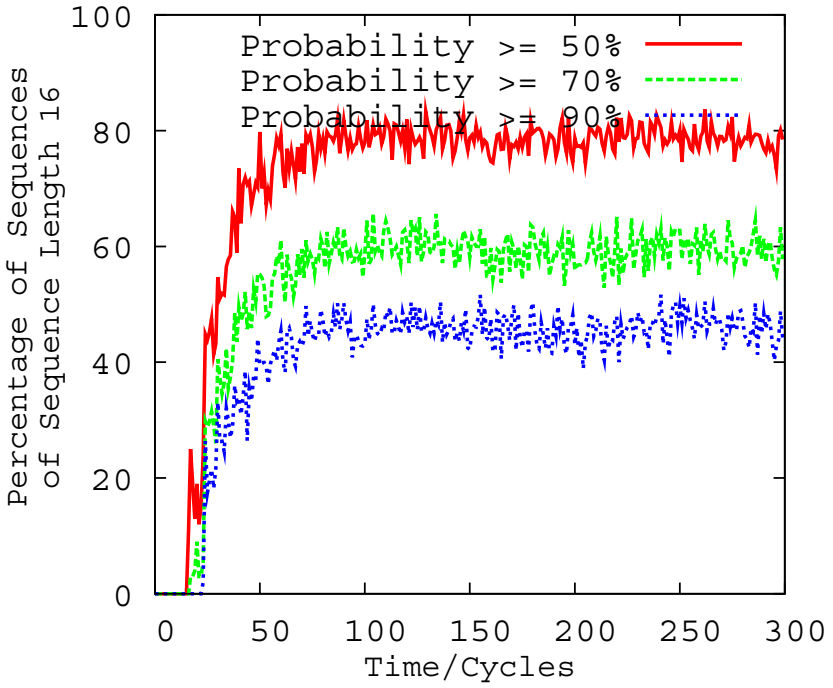
For *pr* accuracy/reliability of 0.7 and 0.9, the sequence detection algorithm was able to deduce more than 90% of all partial sequences *ps* with a probability of 90% or above (cf. Figure 5.5(a)). However, with a *pr* accuracy/reliability of 0.5, only around 60% of the *ps* were deduced with a probability of 90% or above.

5.3.5 Change Induction with Time to Live (TTL)

In this scenario, we have evaluated the ability of the probabilistic sequence detection algorithm to handle changes in sequences and have also used *ttl* to time out both the product parts and the sequences. The evaluations were conducted by setting the *ttl* to 10, 8, and 6 cycles. The *pr*s were uniformly distributed amongst the *vr*s and a change

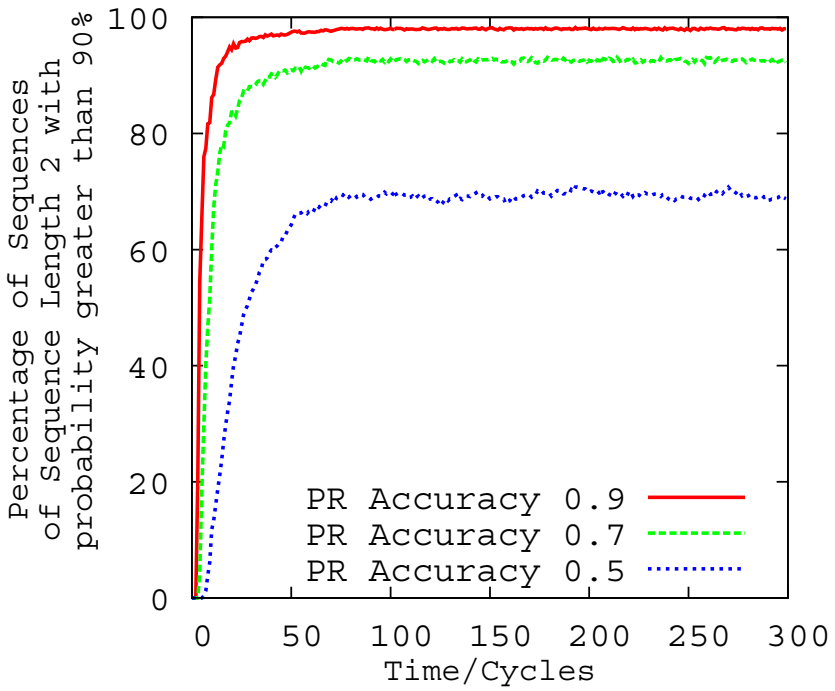


(a) Partial Sequences

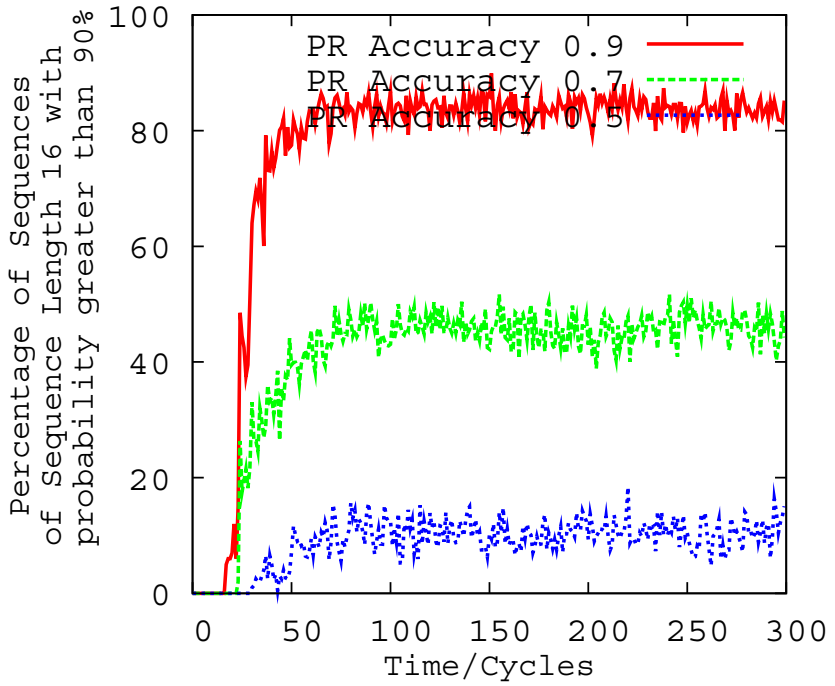


(b) Extended Sequences

Figure 5.4: Probabilistic Detection of Partial and Extended Sequences



(a) Partial Sequences



(b) Extended Sequences

Figure 5.5: Influence of Physical Reader Reliability

is induced on the 100^{th} cycle by removing a product part from the production path to mimic a product part removal scenario.

When sequences were not timed out (cf. Figure 5.3(a)), the algorithm was able to deduce more than 90% of the *ps* with a probability of 90% or above. When the sequences were timed out (cf. Figure 5.6(a)), the algorithm deduces around 60% of the *ps* with a probability of 90% or above for *ttl* of 6 cycles, around 40% of the *ps* with a probability of 90% or above for *ttl* of 8 cycles, and deduces around 30% of the *ps* with a probability of 90% or above for *ttl* of 10 cycles for product part sequences. Furthermore, with *ttl* the longest *es* is of length 12 (cf. Figure 5.6(b)) instead of 16 (cf. Figure 5.3(a)).

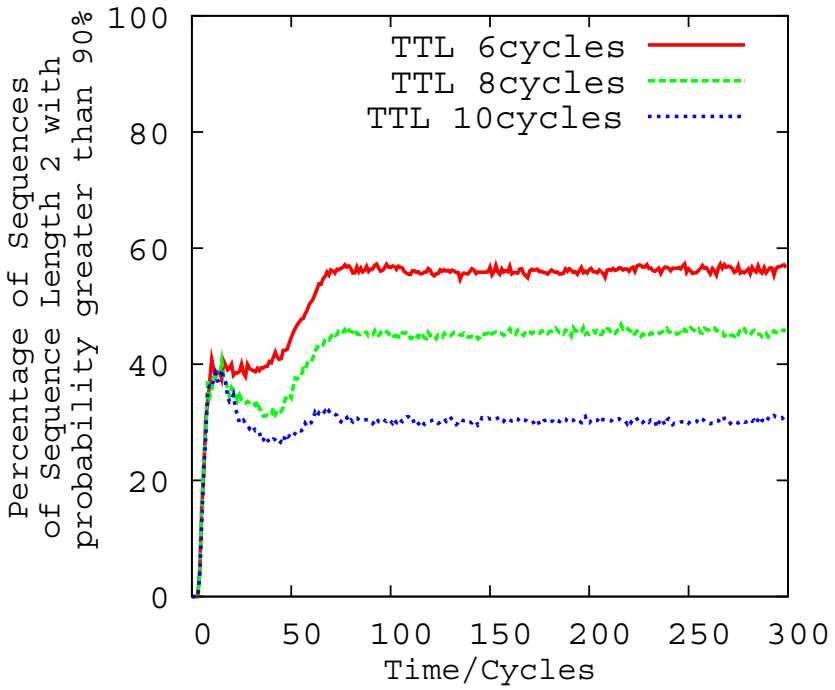
The percentage of *ps* having a probability greater than 90% has decreased. This is because due to timing out sequences, the stale but correct *ps* are quickly removed from the PSL. Stale *ps* are those *ps* that were correct but can no longer be deduced because their respective product parts have left the *vr*. The quick removal of stale *ps* additionally result in *es* of lesser lengths.

A good probabilistic sequence detection algorithm would have a large percentage of *ps* and a low percentage of *es* with a very high probability. There can be a lot of *ps* because of the large number of product parts moving through the production lines, therefore a large percentage of *ps* with a very high probability will indicate that the system has a high degree of confidence in a large percentage of *ps*. In contrast with the *ps*, there can only be one correct *es*. Therefore, the percentage of *es* having a high probability should always be very low. The removal of stale *ps* through using time to live mechanism, results in only around 5% of *es* having a probability of 90% or above (cf. Figure 5.6(b)) as compared to around 40% of the *es* with a probability of 90% or above when the time to live mechanism was not applied (cf. Figure 5.4(b)).

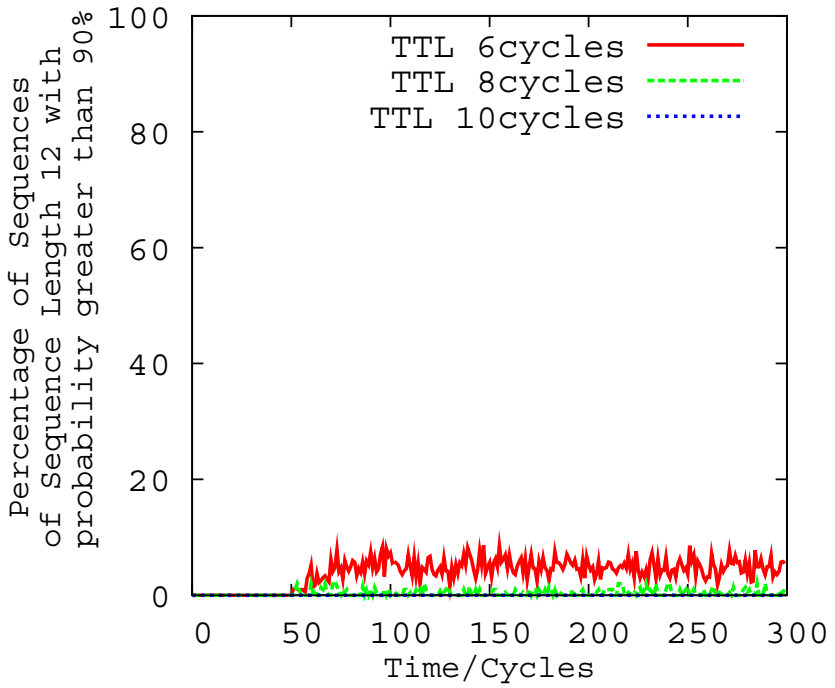
The probabilistic sequence detection algorithm also handles changes very robustly and the induction of changes does not have a dramatic affect on sequence deductions. This is evident from the performance of the algorithm after the 100^{th} cycle (cf. Figure 5.6(a) and Figure 5.6(b)), when a product part was randomly removed from the production lines to mimic product part removal scenario.

5.4 Related Work

In this chapter we have proposed an RFID-based data management framework to track products and product parts in real-time in production environments. Research work that has previously been done in this area can be broadly divided into RFID data management frameworks, RFID-based object tracking and RFID deployments in manufacturing. In order to properly address the topic, the related work has also been divided into the aforementioned categories.



(a) Partial Sequences



(b) Extended Sequences

Figure 5.6: Change Induction with TTL

5.4.1 RFID Data Management Frameworks

Since RFID readers can produce a huge amount of data, we need to have sophisticated techniques and/or infrastructure to manage this data. HiFi (High fan-in) system [FJK+05] is one such system which consists of a large number of receptors (RFID sensors) to collect raw data readings. These sensor are deployed at the edge of the system/network. The data produced by these edge devices is aggregated locally with data from other nearby devices. The locally collected data is then aggregated within a larger area, and so on. Data requests within the system are specified by using a stream-oriented query language. The language is used at each level to query data flowing from the edges of the network towards the interior of the system. HiFi uses a five stage process to convert raw receptor readings into useful data. These stages of processing are cleaning, smoothing, arbitrating, validating and analysing and are collectively termed as CSAVA. CSAVA uses a window of data items to process the data from each receptor. This windowed-based query processing has inherent delays which causes a significant data lag from the time when a receptor reads a data value to the time when data is available at the output.

Another important RFID data management framework is RF2ID (Reliable Framework for Radio Frequency Identification) [AKFR07], which was primarily developed to address the reliability issues in RFID deployments. The system tries to address the reliability issues by deploying multiple RFID readers along the path of object movement. RF2ID employs an abstraction called the Virtual Reader. Data aggregated by multiple readers is sent to a certain virtual reader which is responsible for processing this data and generating a more reliable sensor data from the raw readings. The virtual reader is able to generate a more reliable sensor data since it has access to redundant RFID reader data.

WinRFID [PSR+06] is yet another RFID-based middleware framework. The prime objective of the framework is to have extensibility and scalability for RFID data management. The system has divided RFID related management tasks into five separate layers : RFID hardware, protocols, data processing, XML framework and data representation. The RFID hardware layer is responsible for managing physical hardware like readers, tags, and other sensors. The protocol layer deals with reader-tag protocols. The data processing layer deals with the processing (filtering, aggregating etc) of the data streams generated by the RFID readers. The raw tag data from the physical layer data stream can be formatted in a variety of ways to a high-level XML based representation. These formatting tasks are performed at the XML framework layer. The final data representation layer is responsible for presenting data to be consumed by different enterprise level applications such as database servers, portals, mobile infrastructure etc. In a nutshell WinRFID has tried to incorporate the layered approach of OSI reference model [Zim80] into RFID data processing so as to have an extensible and scalable architecture.

RFID Stack [FL05] is an RFID framework that was designed to address the issue of restricted bandwidth available to RFID devices. Since RFID data is eventually consumed by a diverse set of applications RFID Stack proposes the use of an event-based middleware (publish-subscribe system) to decouple RFID readers from applications. Publish Subscribe systems are broadly divided into two categories: Content-based publish subscribe systems [CRW01], [JCL+10], [TKK+11], [TKKR12] and Topic-based publish subscribe systems [BBQ+07], [CMTV07], [MZV07], [CJV10]. Content-based publish subscribe systems deliver messages to the subscriber if the attributes or content of those messages match the constraints defined by the subscriber. In topic-based publish subscribe systems, messages are sent to "topics" or named logical channels. Subscribers in a topic based publish subscribe system would receive all messages published to the topics to which they subscribe. In addition to this, all subscribers to a topic will receive the same messages.

RFID Stack uses a content-based publish subscribe system, Elvin [SAB+00] to deliver messages generated by RFID readers to the subscribed applications. The obvious advantage of using a publish-subscribe system for RFID data management in RFID Stack is that RFID readers do not have to track the applications that are interested in a certain message. Similarly, applications consuming RFID data do not have to maintain communication channel with individual RFID readers, and can just specify the events in which they are interested by submitting their subscriptions to the messaging system. This results in less bandwidth requirement for disseminating useful messages to subscribing applications.

Discussion: HiFi (High fan-in) system [FJK+05] was mainly developed to create an infrastructure to manage huge amount of raw data produced by RFID deployments. WinRFID [PSR+06] tries to do something similar and has tried to create an extensible and scalable architecture for managing RFID data. RFID Stack [FL05] on the other hand was proposed to address the issue of restricted bandwidth available to RFID devices. The system tries to do that by using a publish-subscribe system to disseminate RFID events to applications. The main objective of RF2ID however was to address the reliability issues in RFID deployments specially the ones in logistics and warehousing domains.

In our work, we have also tried to address the issue of reliability. However, reliability takes on an altogether new meanings in production environments. In logistics and warehousing applications, ensuring reliability suffices by ensuring that all the objects have been accounted for. However, in production environments product parts/objects should not just be properly accounted for, they should also have a pre-determined order/sequence and should be present at critical locations at certain defined time-spans. None of the systems mentioned above accomplishes what is actually the prime requirement in production environments.

5.4.2 RFID-based Object Tracking

RFID technology has been used in the industry for robustly tracking the parts or part carriers during production, storage, distribution and ultimately the supply chain. Although the technology simplifies the process of object tracking, the unreliability of RFID devices and tags leads to uncertainty in the location of the objects that are being tracked. In order to increase the reliability with which objects are tracked, several methodologies and frameworks have been proposed. In the sections below, we would take a brief look at some of these frameworks and techniques.

Pradip et al [DBD04] proposed an architecture to track the mobility of physical objects tagged with RFID labels or EPC codes [Bro01]. RFID readers form the leaves of the topology. These readers continuously detect RFID tags attached to the mobile objects. Since a lot of RFID readers are reading the tags, which results in a large data that needs to be managed, a certain number of RFID readers are assigned to a data routing server which resides at the next level of the architectural hierarchy. These data servers form a distributed hierarchical structure amongst themselves for the purpose of managing the data flow. The job of the data servers is to smoothen and aggregate the data coming from the RFID readers. In addition to this, these servers which are referred to as Savants also correct errors introduced during the process of data capture at the RFID reader level. The Savants themselves are also organized in a tree-like structure, with the leaf nodes being called Edge Savants and the internal nodes being labelled as the Internal Savants.

All the information associated with the RFID tag is archived in Physical Markup Language (PML) [BMKL01] which is designed as a common language for describing physical objects. A sample PML file contains both static and dynamic data related to a certain object. These PML files are stored on a specialized server appropriately named as the PML server. The PML server resides one level above the Savants in the architectural hierarchy with multiple savants being connected to a PML server. The static data pertaining to an object would include EPC class-level manufacturer's data etc, whereas the dynamic data related to an object would consist of the mobility information like its location centric data. The static data is stored in a database which is called Information Database (IDB) whereas the dynamic mobility information is stored in mobility management registers. Similar to mobile cellular systems, every EPC is associated with its Home Location Register (HLR) [RR02]. The HLRs are maintained at the corresponding Home PML of the object and keeps track of all the pointers to its information database. When an object leaves its current location, its HLR information which keeps a history of the PML servers visited by the EPC is updated. The association between the EPC and the HLR is permanent for as long as the EPC stays within its home. When the EPC is assigned a new Home PML, HLR of the EPC is changed to the new owner's HLR.

Similar to the HLR, a dynamically changing Visitor Location Register (VLR) [RR02]

register is maintained at each PML server. This VLR keeps information about all the visiting objects. When a new object is detected by an RFID reader, the corresponding PML Server makes a copy of all the relevant EPC information about this RFID tag from its HLR at the H-PML and stores it in its local VLR. Typically in an organization, each PML server would have an associated HLR and VLR.

In order to query information about any object, its H-PML should first be identified. Due to this reason, there exists a mapping between the IP address of the H-PML of an object and the EPC of that object. This mapping is performed by an Object Naming Service (ONS), which is a similar concept to the Internet Domain Name Service (DNS) [Moc83]. For any information update regarding an EPC, the ONS is queried to retrieve the IP address of the H-PML after which the update required is sent to the H-PML.

Yap et al [YSM05] argued that humans are powerful sensors who can better locate objects based on cues and identifiable landmarks and hence should be involved in the task of tracking objects. One advantage of bringing humans into the loop is that it reduces the complexity of the system and eventually results in a simple and scalable architecture. Yap et al [YSM05] propose a system called MAX in which they assume that all physical objects can have a wireless tag attached to them, with each tag containing information about the physical object with which it is attached. These physical objects can be spread over a vast geographical area, which in turn could be divided into sub-structures such as rooms, hallways etc which in this work are called localities.

Within each locality, MAX has a three-tiered architecture which consists of base stations that are tied to a locality; sub-stations that are mainly tied to static objects (e.g., chairs, tables, shelves); and finally the RFID tags that are attached to the physical objects (e.g., keys, books, phone, documents).

Kumar et al. [KAE00] evaluated two approaches to track objects by attaching networking devices to them. The first approach which is referred to as SCOUT-AGG uses aggregation along with the sensor hierarchy. In this approach the sensors are organized in an hierarchy such that the higher level sensors have more concise and aggregate information about the objects sensed by low level sensors. Remote sensors then use this aggregate information at the high level sensors to direct queries towards the low level sensors having more precise information about a certain object.

However, the aggregation based approach suffer from inefficiencies due to the fact that aggregation typically results in a loss of information. As a result of this, a parent sensor does not know for certain if an object is located in a certain child branch or not and hence the query has to be forwarded up the hierarchy (until the root) to reach all other relevant branches as well in order to ensure a response. This issue can lead to flooding in the worst case.

The second approach uses indirection with the hierarchy and is referred to as SCOUT-

MAP. In this approach, sensors use a hash function to map an object name to a sensor address that becomes the locator sensor for the object. This approach was first presented in Landmark routing by Tsuchiya et al [Tsu88]. The address of the sensor monitoring the object and the object location is stored at the locator sensor for the object. Whenever a query is made, it first performs the same algorithmic mapping to the object name and obtains the locator sensor address. The locator sensor is then contacted for the object location or for the address of the sensor monitoring the object.

Indirection based approach also suffers from inefficiencies relative to SCOUT-AGG when either the mobility of objects or network dynamics exceeds a certain threshold. The locator sensor needs to be updated in SCOUT-MAP whenever the object moves. Similarly, the excessive rate of network dynamics can also make indirection inefficient as many objects would have to be re-mapped to different locator sensors after the topology has been changed.

James Brusey and Duncan McFarlane [BM05] have tried to examine the impact of RFID technology for part tracking in the industry. They presented an approach to integrate RFID sensor data with a representation of the state of the manufacturing system and a model of how that state is changed. The aim of there approach is to enhance the accuracy with which the parts are identified thereby improving the robustness of the over all manufacturing system. The central idea of the Brusey approach is that parts are rarely seen in isolation, but often travel together.

A common example that comes to mind is that of pallets and cases containing many parts that are to be tracked. Two cases on the same pallet will tend to both be detected by RFID sensors at around the same time. Similarly, the pallet will be detected along with the two cases. All together they form an aggregate. Aggregated objects provide an opportunity to improve the reliability of RFID information. The authors developed a containment relationship, which defines a container and the contained objects. In the example given above the pallet would be the container and the parts on it would become the contained parts in the logical model. Once a containment relationship is established for a certain container and its constituent parts. It is not necessary to track all the parts and hence tracking the container would lead to the tracking of all the contained parts.

Wang et al [WLWT07] proposed a system to track objects moving on the assembly lines. The objective of the work is to find out the location of a specific object on the assembly line so as to perform some tasks on it. For the system, Wang et al [WLWT07] deployed a certain number of RFID readers along the assembly lines in a factory. These readers were interconnected wirelessly to form an RFID grid. This RFID network grid establishes a monitoring region on the assembly line. Each object, with an RFID tag attached, moving through this monitoring region will be detected by the readers. Since each tag has a unique id, the moving objects are thus uniquely identified and located on the assembly line.

In addition to tracking objects on the assembly line, the authors further employed non-linear Bayesian tracking method to forecast the object's moving direction and future location by recursively calculating the previous probability density function (pdf) of the tag's position in the region. This forecasting information can then be used by the production facility to coordinate its production tools at the location on which it wants to process the said object.

Discussion: Pradip et al [DBD04] proposed an architecture to track the mobility of physical objects tagged with RFID labels or EPC codes [Bro01]. The system maintains an object naming service similar to how the names are managed by Internet Domain Name Service (DNS). However, the object tracking concepts used by Pradip et al are similar to the ones used by RADAR [BP00], SpotON [HVBW01] and LANDMARC [NLLP04]. RADAR recordes and processes signal strength information at multiple base stations which are positioned so as to provide overlapping coverage in the area of interest. SpotON uses an aggregation algorithm to perform a three dimensional location sensing based on radio signal strength analysis on the RFID devices. LANDMARC on the other hand, uses active RFID tags for locating objects inside buildings. They use reference tags within the building and location of each RFID tag then is computed with reference to these tags.

Yap et al [YSM05] have argued that humans are powerful sensors who can better locate objects based on cues and identifiable landmarks. So they have tried to perform object tracking by having humans involved in the loop. Kumar et al. [KAE00] however have tried to track objects without having humans in the loop and by using unattended methods. The approaches presented in SCOUT either use aggregation or maintains hierarchies to route queries to lower level sensors, which are located near the actual physical objects.

James Brusey et al [BM05] presented an approach to integrate RFID sensor data with a representation of the state of the manufacturing system and a model of how that state is changed. The aim of there approach is to enhance the accuracy with which the parts are identified thereby improving the robustness of the over all manufacturing system. Similarly, Wang et al [WLWT07] proposed a system to track objects moving on the assembly lines. The objective of Wang's work is to find out the location of a specific object on the assembly line so as to perform some tasks on it.

Bursey and Wang's work is highly related to our approach. Bursey [BM05] tries to increase the reliability of detections by forming aggregates and containment relationships etc. However, what sets us apart is that we have developed a model to assign probabilities to these detections and hence can argue with mathematical precision about the reliability of our detections. Wang et al has presented a solution to track RFID tagged objects moving within a certain geographical area. The technique allows knowing the relative location of objects. Again, this is something that we also try to accomplish. However, we go a step further and also try to find out if the part is in a correct position with regards to other moving parts or not i.e. if its in moving through the production

lines in a correct sequence or not.

5.4.3 RFID in Manufacturing

RFID applications are closely tied to manufacturing executive system which are used to control the production process. RFID technology may support many of the control functionality in a manufacturing environment like operation scheduling, labour management, maintenance management, data collection, quality management and performance analysis. Some of applications and deployments of RFID technology in manufacturing are discussed briefly in this section.

In [BR05a], the authors presented a detailed discussion on applications of RFID in manufacturing. Major applications include warehouse management, manufacturing engineering, and mistake-proofing of mixed-flow assembly. The authors also reviewed a number of internal needs driven applications. Toyota (South Africa) for example has tagged its carriers to streamline manufacturing and vehicle tracking. The tags are intended to remain with the vehicle throughout its life and hold its maintenance history. Harley Davidson has implemented process automation by tagging bins carrying parts to provide instructions to employees at each stage of the process. Johnson Controls has started using RFID to track cars and truck seats throughout the assembly process. TrenStar tracks their beer kegs to improve demand forecasts and increase efficiency. International Paper tracks their paper roll to reduce lost or misdirected rolls. Gap tracks its denim apparel to improve customer service through better inventory management. Raxel has started tagging reusable plastic biohazard containers to avoid contamination. Michelin on the other hand tags its tyres to comply with the TREAD act and recall management. All the RFID deployments mentioned above were carried out in order to improve shop floor inventory tracking and automate warehouse operations.

Automobile manufacturers have also successfully deployed RFID based systems in their production facilities. As an example Ford Motor Company uses RFID technology at its automated production lines in Cuautitlan, Mexico in order to improve product quality [Joh02]. The Cuautitlan facility produces 300,000 to 400,000 cars and trucks each year. Each of these vehicles are built using just-in-time assembling where suppliers supply parts on an as-needed basis. Due to this, it is crucial that the inventory and production in the plant are precisely tracked.

Prior to the deployment of RFID technology, Ford used a manual coding system to track automobile and truck frames as they went through the final assembly, paint and body shop areas of the production line. However, manual tracking was ineffective due to frequent errors and costs associated with resulting production oversights.

With the RFID technology deployed, an RFID tag is placed on a vehicle skid, and then programmed with a serial number that is used to reference the vehicle in Ford's operating system. As a vehicle passes through the different stages of production, different

parts of the 23 digit serial number are referenced, indicating what needs to be done at each station. The manual system required an identification sheet to be manually updated at each step, whereas with the RFID deployment the updates are automatically written to the tag. In this way, the risk of operator errors in updating the sheets to determine what needs to be done is totally eradicated.

BMW and Vauxhall [BL97], [ZGP04] have also made use of the RFID technology to accurately customize the cars according to the customer specifications. A read/write smart RFID tag is programmed with the customer specification and is then placed on the car. The tag moves with the car during the production process and at each step is read to determine exactly what needs to be done and which equipment needs to be fitted into this specific car. This tracking ensures that the car is manufactured with the correct color, model, interior, and other user specifications.

Similarly [VPR09] has also inspected several applications of RFID in manufacturing and explored their perceived benefits in the field. The inspected applications spanned areas as diverse as production activity control, inventory management, quality control, plant maintenance, and tracking new product development.

Zhekun et al [ZGP04] have presented the concept of having smart and intelligent parts in the production environments. The idea is that each part should be smart in a sense that it should have a unique itentity and be able to communicate with other parts and production tools for flexible manufacturing. In addition to this, the authors have also presented the notion of concurrent intelligent manufacturing [LLGF08]. The idea behind concurrent manufacturing process is that it would be a process in which both the design of the product and its manufacturing run concurrently. The design in this scenario does not indicate the design of each individual part, but rather entails the specification of the specific parts that should be used for the manufacturing of the overall product. In this scenario the customer typically acts as the designer and indicates the specifications of the product. The idea is that the customer specification should be programmed on an RFID tag that should then be read at each stage of assembly to determine what tasks need to be performed. If during the manufacturing the customer decides to change his/her specification, the tag information should be updated accordingly. Since, each assembly point carries out the processing after reading the RFID tag, the updated specification would ultimately result in a modified product.

Tan et al [TWLW08] developed an RFID based stacking and sorting system for a Tobacco manufacturing plant. The system uses RFID readers deployed along the production line. These readers are connected to the control machines (e.g. sorting and stacking machines) in the plant. An RFID tag is attached to each cigarette box and contains information about the box such as the brand of cigarettes and how it should be handled and stored. An RFID reader, reads tags on the cigarette boxes moving on the line and sends this information to a sorting controller which in turn determines where to route the box. A stacker crane robot stacks the forwarded boxes on pallets according to their brand. Pallets also have RFID tags. At the end of the assembly

line another RFID reader updates the RFID tags on the pallets to contain information regarding what type of cigarette boxes are contained within each specific pallet. The authors also conducted a comparative study with the existing barcode based system. The study concluded that the RFID based system enhances time management and reduces the number of damaged products because it does not need manual scanning of tags. In addition to this, the higher memory of RFID tags results in a more efficient management of the products in the warehouse.

Fagui et al [LM06] developed an RFID based Activity Monitoring System (RAMS) for a bath-tub manufacturing plant. The plant produces two types of bath-tubs a normal and a specialized one. Each of these tubs has a different production plan. Customers can give their preferences when ordering the special bath-tub. The system tracks the activity being performed at each step. This activity monitoring results are then used by managers to follow up with the orders for each specialized bath-tub. Through the deployment of this system, the authors have tried to rectify some commonly occurring problems such as: missing components, workers moving products to the wrong place or forgetting to move them to the desired location on time. The deployment of RAMS resulted in a 5% increase in the service level of the plant and a 1.5% annual increase in sales. In addition to this, it resulted in a 60% reduction in delayed production and subsequently accounted for a 0.5 million US dollar reduction in the inventory costs per annum.

Latham et al [LWG08] have presented an RFID based time tracking system for moisture sensitive devices that was deployed at Universal Avionics. In the production of avionics systems and equipments, Universal Avionics uses Moisture Sensitive Devices (MSD). The exposure time of these devices and components should be accurately tracked so that the operators can be notified when the moisture from the air has accumulated to a critical level. A part that accumulates more moisture then is desirable would ultimately cause a critical part failure later on in the manufacturing process, and hence would result in a loss of time and resources. The system developed by Latham et al tracks the time period during which trays containing MSDs are exposed to open air. Each tray is attached with an RFID tag. When a tray comes in i.e. is read by the RFID reader, the system starts logging its exposure time. The system generates a warning message when the exposure time is about to reach the maximum threshold limit. This notifies the operators to take preventive measures. Prior to this, Universal Avionics managed this activity manually, whereby the workers would record the time on sheets and would then perform manual calculations to find out the exposure times. This manual process consumed 3 hours of an employee each day.

Working worker assembly island is a form of manufacturing in which the products and their sub-parts are placed at fixed positions on the shop floor while the workers move from one place to another to carry out the production process. During production, tools and product parts are brought to the work center for the assembly according to the production plan. In order to carry out the assembly process efficiently workers need

to know about the position of tools and parts with accuracy and tools and parts should also be at the destined positions. In their work Huang et al. [HZJ07], [HZJ08] have presented a system that tags production tools and parts with RFID tags and provides real-time information about the location of the required tools on the shop floor.

Bauer et al [BJS04] conducted a study which showed that the development of the ratio of mobile resources to machines almost doubled over the last 20 years. This is a result of the high degree of individualization in capital goods manufacturing which is often coupled with a frequent changing of machinery settings and customized tools during production. Such a process quickly becomes non-transparent and hard to manage. Due to this reason, production planning based on current and exact information has become a central element of success for factories with high demands in productivity and flexibility. The central idea of the study is that capital goods manufacturers would benefit a great deal by incorporating RFID-based production tracking within their factories.

Similarly, Harun et al [HCW08] have presented a theoretical model and a generic RFID framework to build an RFID-enabled aerospace manufacturing environment. Their work is a step towards filling the gap between the physical flow of object in the aerospace manufacturing and their planning in the virtual world. According to the authors, an RFID based monitoring framework would enable airline manufacturers to better plan and control production, maintain product traceability, have inventory visibility and enhance labor productivity in addition with fulfilling a lot of regularity requirements.

Discussion: Baudin et al. [BR05a] presented a detailed discussion on applications of RFID in manufacturing. Some of the major applications that they discussed include warehouse management, manufacturing engineering, and mistake-proofing of mixed-flow assembly etc. Similarly, automobile manufacturers have also successfully deployed RFID based systems in their production facilities. Prominent examples include the use of RFID technology at Ford plant in Cuautitlan, Mexico [Joh02] and the use of RFID technology by BMW and Vauxhall [BL97], [ZGP04] to accurately customize the cars according to the customer specifications.

In addition to this, Tan et al [TWLW08] presented a solution that used RFID technology to perform automatic stacking and sorting in a Tobacco manufacturing plant. Fagui et al [LM06] developed an RFID based Activity Monitoring System (RAMS) for discrete manufacturing. While Latham et al [LWG08] developed an RFID based time tracking system for moisture sensitive devices for Universal Avionics.

All the applications discussed thus far, primarly deals with tracking or performing simple checks on products using RFID technology, where the primary goal is to improve productivity by eliminating manual labour and increase productivity by decreasing the number of errors that are made during production. However, none of the systems or deployments deal with the inherent unreliable nature of RFID technology and do

not present any solution to increase the accuracy of the RFID deployments in the production environments. The primary difference between our system and the ones discussed above is that we have developed a comprehensive probabilistic algorithm that assigns probabilities to each of the detected part and the sequences and hence provides a measure of accuracy for each detection.

In addition to the systems that have already been developed and deployed, we also took a look at some proposed solutions. As an example, Bauer et al [BJS04] suggested that mobile computing can be used in production environments to track product parts and tools and hence can provide great advantages in tracking men and materials. Harun et al [HCW08] have presented a theoretical model and a generic RFID framework to build an RFID-enabled aerospace manufacturing environment. Both of these works, do not present a concrete implementation and primarily focus on discussing the advantages of having RFID deployments in manufacturing environments.

5.5 Summary

In this chapter we have presented the Consistency stack, which divides the consistency issues into separate layers. These layers address inconsistencies that may arise due to the unreliability of RFID devices and issues in production environments due to variant production. We have also presented a real-time production monitoring framework that uses a probabilistic model to ensure reliable real-time production monitoring. A sequence detection algorithm is also presented that attaches probabilistic guarantees to the object sequences detected by the RFID readers.

Our evaluations show that our framework performs better when the *pr*s are uniformly distributed across the *vr*s. In addition to this the sequence detection algorithm is able to detect extended sequences with more than 90% probability even when the accuracy/reliability of the prs is only 70%.

Chapter 6

Self-Calibration of RFID Reader Probabilities in a Smart Variant Production Environment

Accuracy and precision is of utmost importance in manufacturing environments. One of the major stumbling blocks for RFID deployments in manufacturing industry is the inherent unreliability of the RFID technology. In the previous chapter, we proposed an RFID-based framework that uses multiple RFID readers deployed on the production lines to monitor product parts and product part sequences. The basic idea of the approach was to detect product parts reliably using unreliable RFID readers. Redundancies in the deployment of RFID readers were exploited to increase the confidence in the detected sequences.

A short coming of the initial approach is that we assume that RFID readers would have fixed probabilities of correct readings which are known a priori. This has nothing to do with reality, since the hard fact is that the reliability of an RFID reader is dynamic. In addition to this the actual reliability of the RFID reader can be different from the reliability guarantees provided by the reader manufacturer depending on the deployment setup. In order to overcome these issues, we propose an efficient and scalable self-calibration algorithm in this chapter that dynamically updates the probabilities of RFID readers. This ensures that the probabilities of RFID readers reflect the current conditions at all times, and hence the detections of a faulty reader has a minimalistic effect on the overall performance of the real-time monitoring of product parts. We have conducted simulations to evaluate the probability self-calibration algorithm under different settings. The evaluations show that our algorithm is able to estimate the probabilities of physical readers with an accuracy of more than 90% even for physical readers having reliability as low as 70%.

The rest of the chapter is structured as follows. An extension of our system model is discussed in Section 6.1. In Section 6.2 we formally discuss the problem that we want to solve. Section 6.3 presents the probability self-calibration algorithm. Section 6.4 discusses the evaluations, followed by a review of related work in Section 6.5. Finally,

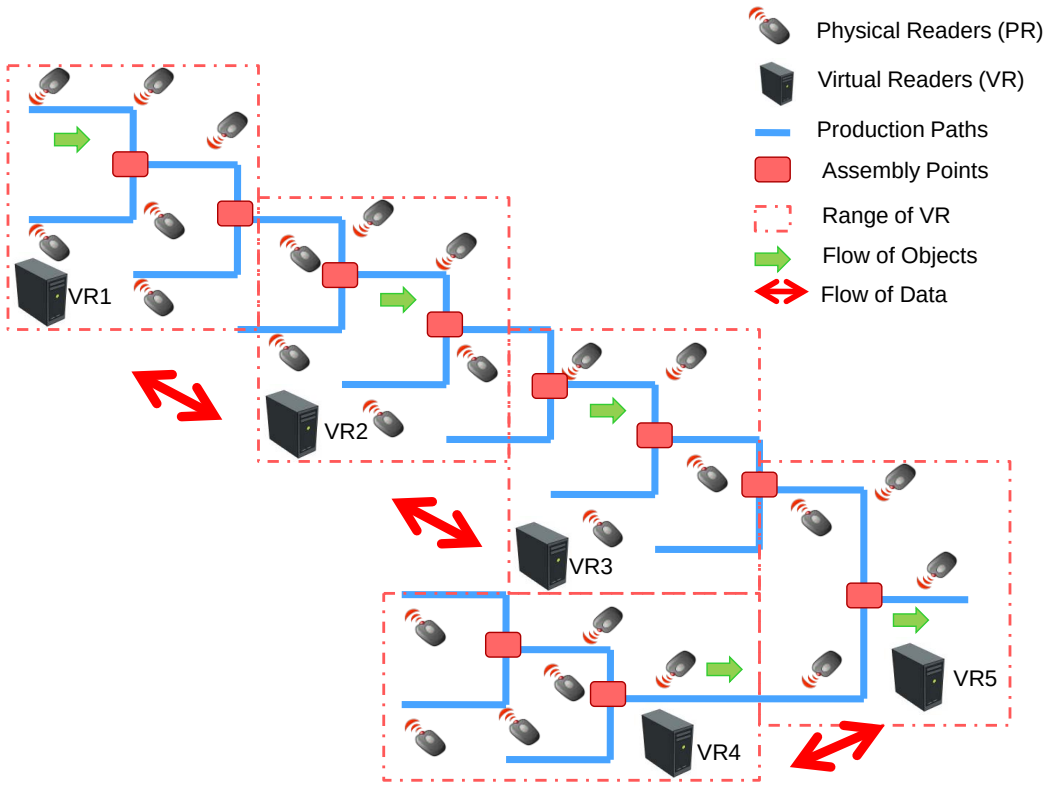


Figure 6.1: System Deployment

we conclude the chapter with a short summary in Section 6.6.

6.1 System Model Extension

The system model has already been described at length (cf. Chapter 3). However, the few additions that were necessary for the physical readers *pr*s to self-calibrate their sensing probabilities are discussed below:

Virtual Readers: Virtual readers, denoted as *vr*s, represent an abstraction of physical readers. Each *pr* is connected to exactly one *vr*. The one change from the system model discussed earlier (cf. Chapter 3) is that the communication between *vr*s is now bi-directional i.e. each *vr* both sends and receives information to/from its predecessor vr_{i-1} and successor vr_{i+1}. Figure 6.1 shows this modification. The *vr*s are responsible for detecting different failures which are detailed in the RFID reader errors (cf. Chapter 3.4). In addition to this, the *vr*s also compute and continuously re-configure the probabilities of RFID readers.

Each *vr* also has access to two global variables, planned product part order PPO_{pln} and actual product part order PPO_{act}. PPO_{pln} contains the list of all product parts and

the order in which they are suppose to move on the production line. This product part order is planned during production planning and has already been discussed extensively (cf. Chapter 4). In the beginning the PPO_{act} contains the same product part order as PPO_{pln}. However, over time PPO_{act} reflects the actual order of product parts moving on the production lines.

We have already designed a sequence detection algorithm (cf. Chapter 5.2) to detect the actual product part order from a set of RFID readings. Therefore, we would assume that PPO_{act} contains the exact order in which product parts are moving across the production lines at all times.

6.2 Problem Statement

The goal of our algorithm is to eliminate RFID based sensing errors in order to have an accurate model of the physical world. To achieve this goal, we use a probabilistic model which is discussed below:

Every *pr* has an associated probability $p(pr_{act})$, which reflects the supposed reliability of the *pr*, and an estimated probability $p(pr_{est})$, which contains the probability of the *pr* as estimated by our algorithm (cf. Section 6.3.3). The *probability of read event* $p(e)$ is the probability with which product part o was correctly detected by *pr*. For any read event e, $p(e) = p(pr_{est})$.

In reality, the probability of a *pr* depends on various *pr* properties and its deployed environment. For instance, $p(pr_{act})$ is influenced by the type of the reader, the orientation of the reader's antenna and the RFID tag that is being read, and so on. Therefore, each reader has an individual probability which is hard to know a-priori. Our goal is to develop a self-calibrating system that estimates and adapts $p(pr_{est})$ during production monitoring.

6.2.1 Physical Changes on Production Lines

Physical changes on the production lines are deviations from PPO_{pln} as a result of factors such as human intervention, and mechanical failures etc. These deviations result in contradictory readings, which have to be distinguished from erroneous reads/RFID reading errors.

Contradictory Readings: Contradictory readings are readings that are categorized as RFID errors (cf. Chapter 3.4) by the system when in fact they depict the true reality of the physical world. Contradictory readings occur due to physical changes on the production lines. Changes can occur due to product part removal such as $(o_i < o_j < o_k)$ getting changed to $(o_i < o_k)$, product part insertion such as $(o_i < o_j)$ getting changed to $(o_i < o_m < o_j)$ or change in product part order such as $(o_i < o_j)$

getting changed to $(o_j < o_i)$. The net effect of all three types of changes is a change in the partial sequence ps. So in case of change in product part order from $(o_i < o_j)$ to $(o_j < o_i)$, the production monitoring system might assume that the new reality $(o_j < o_i)$ is merely a false reading by some physical readers and hence would categorize such readings as false readings.

In variant production, product parts need to move through production lines in predefined order so that they can be correctly assembled with their corresponding parts. Changes in partial sequence ps would result in inconsistent final products and hence are not desirable. The goal of detecting physical changes is to detect inconsistencies between physical world and the production plan. Detection of RFID reader errors is required to accurately detect physical changes since an inaccurate model of the physical world would make it impossible to make correct assumptions about inconsistencies between the physical world and the production plan. It is worth noting that the effect of a physical change and an RFID reader error might at first look similar w.r.t the detected read event. For instance, an out of order read may initially look similar to a physical change of product part order. Therefore, we need algorithms to distinguish between RFID reader errors and contradictory readings.

6.3 Self-Calibration of RFID Reader Probabilities

In this section we present concepts for the self-calibration of physical reader probabilities. First we provide an overview of our algorithm (cf. Section 6.3.1). Then we discuss the detection of possible partial sequences (cf. Section 6.3.2). After that we describe the actual probability calibration algorithm (cf. Section 6.3.3) and in the end we explain the process of change detection (cf. Section 6.3.4).

6.3.1 Overview

In this section we will give a brief overview of our approach. The basic steps of our algorithm follows:

1. Detect possible partial sequences ps. The outcome of this step are multiple possible ps which are stored in a global partial sequence list PSL_g, which contains all the possible ps deduced by our system. Unlike the PSL introduced in the previous chapter, PSL_g is a global list and contains the partial sequences detected throughout the system i.e. by all virtual readers. Where as PSL contained partial sequences detected at a particular virtual reader.
2. For each ps, calculate $p(ps)$ which is a probability for ps being the correct sequence. $p(ps)$ is calculated from the probabilities of associated read events that participated in the detection of ps.

3. Determine the most probable partial sequence ps_{mp} amongst the different conflicting ps in PSL_g. If there are two conflicting ps, ps_i ($o_i < o_j$) and ps_j ($o_j < o_i$) with $p(ps_i)$ being 0.9 and $p(ps_j)$ being 0.81, ps_i will be considered as the most probable partial sequence ps_{mp}. If a partial sequence ps does not have any conflicting ps, it is automatically considered as the most probable partial sequence ps_{mp}. So in a nutshell, all unique partial sequences are considered as ps_{mp}, whereas in case of non-unique partial sequences, the sequence with the highest probability is considered as ps_{mp}. We assume that ps_{mp} reflects reality, which is a reasonable assumption if many redundant readings have been considered.
4. Calibrate the probability of each individual RFID reader by finding out how many correct and incorrect readings it has made. This is done by comparing the read events of a pr with all the ps_{mp} within PSL_g. Every reading that is not consistent with the most probable partial sequence ps_{mp} is considered an incorrect reading. In order to distinguish between correct and incorrect readings, we consider different possible RFID reader errors(false, out of order, and missed readings).

6.3.2 Probabilistic Partial Sequence Detection

The process of calibrating RFID reader probabilities involves three steps:

1. Detection of possible partial sequences.
2. Determination of most probable partial sequences
3. Self-Calibrating the probabilities of RFID readers.

Step 1: Detection of possible partial sequences was the topic of sequence detection algorithm and has already been discussed at length in the previous chapter (cf: Chapter 5.2). Step 2 is a very simplistic task in which all the unique partial sequences are tagged as ps_{mp}, whereas in case of non-unique partial sequences, the partial sequence with the highest probability is tagged as ps_{mp}. The third step, self-calibration of RFID reader probabilities is the most important part of this chapter and is described at length in the subsequent section (cf. Section 6.3.3).

6.3.3 Self-Calibration of RFID Reader Probabilities

The core idea of the probability self-calibration algorithm is to detect the RFID reader errors and physical changes and dynamically update the estimated probability $p(pr_{est})$ of a pr to reflect the reliability of that pr at a particular instance of time. The probability $p(e)$ of a read event e is dependent on the probability $p(pr)$ of the pr that detected the event. In other words $p(e) = p(pr)$. Therefore any inaccuracy in the probabilities of prs will induce errors in probabilities of the detected events. In order to have correct

and accurate probabilities of read events the probability of prs should be calibrated over time.

We have already mentioned that for the purpose of our algorithm we would assume the probability of the read event e to be equal to the estimated probability $p(pr_{est})$ of the physical reader pr i.e. $p(e) = p(pr_{est})$. The idea behind this is that since we would be continously calibrating the estimated probability $p(pr_{est})$, the probability of the read event $p(e)$ would at all times reflect the real probability of the physical reader and not some factory assigned value.

The PSL_g is a probabilistically ordered list, such that if there are two conflicting ps $(o_i < o_j)$ and $(o_i < o_k)$, the ps with the higher probability would be stored above the other one. Furthermore, the more probable partial sequence ps_{mp} would be assumed to be the actual partial sequence on the production line.

The estimated probability $p(pr_{est})$ of a pr is computed as:

$p(pr_{est}) = (productparts - errors)/productparts$;

where $productparts$ is the total number of product parts that passed through the physical reader pr on the production line, and $errors$ is the sum of all errors committed by the pr, i.e., $errors$ = n(missed reads) + n(false reads) + n(out of order reads) (cf. Algorithm 3 line 26-29). $p(pr_{est})$ is re-computed whenever a product part is detected or is categorized as a false, missed or out-of-order read.

We do not consider duplicate readings since a duplicate does not carry additional information. We filter out duplicate readings in a pre-processing step by aggregating directly succeeding read events of a product part o by a pr into one read event for the product part o. Furthermore, we do not have to distinguish between the different types of errors in this formula, as we have already explained that we can only make a statement about the order of two product parts if at least two read events are correct. An incorrect event cannot be used to determine the true order since any kind of error (false, out of order, missed reads) will invalidate the partial sequence ps.

False Reading Detection Whenever a read event $e_i = (o_i, pr_i, t_i)$ is detected by a pr_i, we compare o_i with PP_{pln}, which is a list that contains all the product parts that are planned to pass through this production line to determine if o_i was supposed to pass through this line. If o_i is not a part of PP_{pln}, we compare the partial sequence ps_i $(o_{i-1} < o_i)$ detected at pr_i with all the partial sequences in PSL_g. In case ps_i $(o_{i-1} < o_i)$ is not ps_{mp} in PSL_g i.e. there exists ps_j that contradict partial sequence ps_i and have a higher probability, o_i is categorized as a false read. ps_i $(o_{i-1} < o_i)$ is inserted into the FPL_{pr_i}, which is a list that contains all the falsely detected partial sequences by pr_i. After adding the ps to the FPL_{pr_i}, $p(pr_{est})$ of pr_i is recalculated (cf. Algorithm 3 line 14-23).

To explain this intuitively, if we have two partial sequences ps_i $(o_{i-1} < o_i)$, and ps_j $(o_{i-1} < o_j)$ with probability of ps_i being 0.8 and probability of ps_j being 0.9, then ps_i

Algorithm 3 Probability Self-Calibration Algorithm

1: Let $e_i = (o_i, pr_i, t_i)$ be a read event detected at vr_i

2: $DPL_{vr_i} = o_i$
3: $DPL_{pr_i} = o_i$

4: **if** $o_i \in PP_{pln}$ **then** {o_i not a false read}
5: **if** $((o_{i-1} < o_i) \in PSL_g \; AND (o_{i-1} < o_i).probability > (o_{i-1} < any) AND (o_{i-1} < o_i).probability > (any < pp_i))$ **then**
6: o_i not an out-of-order read
7: **for all** $pr_n \in vr_i$ where n=0 to i-1 **do**
8: **if** $\neg o_i \in DPL_{pr_n}$ **then** {pr_n missed to detect o_i}
9: $MPL_{pr_n} = o_i$
10: **trigger** estimateProbability(pr_n)
11: **end if**
12: **end for**
13: **else** {o_i is Out-of-Order Read}
14: $OOPL_{pr_i} = (o_{i-1} < o_i)$
15: **trigger** estimateProbability(pr_i)
16: **end if**
17: **else** { o_i not in PP_{pln} }
18: **if** ($(pp_{i-1} < pp_i)$ is most probable $ps \in PSL_g$) **then**
19: **for all** $pr_n \in vr_i$ where n=0 to i-1 **do**
20: **if** $(o_{i-1} < o_i) \in FPL_{pr_n}$ **then**
21: remove o_i from FPL_{pr_n}
22: **trigger** estimateProbability(pr_n)
23: **end if**
24: **end for**
25: **else** { o_i is a false read }
26: $FPL_{pr_i} = (o_{i-1} < o_i)$;
27: remove o_i from DPL_{vr_i};
28: **trigger** estimateProbability(pr_i)
29: **end if**
30: **end if**

31: **if** $((o_{i-1} < o_i).probability < ps_{conflicting}.probability)$ **then**
32: **trigger** ChangeEvent (PSL_g, vr_{i-1}, $ps_{conflicting}$)
33: **end if**

34: **for procedure** estimateProbability(pr_i) **do**
35: $productparts = n(DPL_{pr_i}) + n(MPL_{pr_i})$
36: $errors_{pr_i} = n(MPL_{pr_i}) + n(FPL_{pr_i}) + n(OOPL_{pr_i})$
37: $p(pr_i est) = \frac{productparts - errors_{pr_i}}{productparts}$
38: **end for**

can not be the correct partial sequence. This further tells us that product part o_i can not be behind product part o_{i-1}. Furthermore, since o_i is not in PP_{pln}, it was never supposed to be on this production line and hence is a false reading.

Out of Order Reading Detection Once we have determined that o_i is not a false read, we try to figure out if pr_i has detected o_i in the correct order. In order to determine this we compare ps_i $(o_{i-1} < o_i)$ with PSL_g. If ps_i $(o_{i-1} < o_i)$ is found to have a higher probability then any partial sequence having o_{i-1} as the first part in the sequence or o_i as the later part in the sequence, then $(o_{i-1} < o_i)$ is not an out-of-order read. However, if $(o_{i-1} < o_i)$ is a partial sequence with a lower probability as compared to its conflicting partial sequence we insert it into out of order parts list $OOPL_{pr_i}$, which contains all the partial sequences that were deduced as a result of out of order reads by pr_i. The $p(pr_{est})$ for pr_i is also recalculated at this time (cf. Algorithm 3 line 5-13).

In order to further elaborate this, lets assume that pr_i has detected a partial sequence ps_i $(o_{i-1} < o_i)$ and the probability of ps_i is 0.7. Lets further assume that we have another partial sequence ps_j $(o_{i-1} < o_j)$ in PSL_g such that the probability of ps_j is 0.9 and hence ps_j is currently the most probable partial sequence. Since, we have already eliminated the possibility of product part o_i being a false reading, the only possibility is that the physical reader pr_i has detected o_i out of order. In other words if there exists a conflicting partial sequence ps_j having a higher probability, then ps_i is an out of order partial sequence.

Missed Reading Detection Once we have eliminated the possibility of o_i to be a false positive (false read, out of order read), we try to determine if there exist some prs that missed out on detecting o_i. This is done by comparing o_i with DPL_{pr} of the prs that are deployed before pr_i to find out if they have also detected o_i. If o_i is not already present in a particular pr's detected parts list (say DPL_{pr_h}), it is placed in the missed parts list MPL_{pr_h}, which is a list that contains all the product parts that pr_h failed to detect. The $p(pr_{est})$ for pr_h is also recalculated (cf. Algorithm 3 line 8-10).

6.3.4 Detection of Physical Changes

The objective of the probability self-calibration algorithm (cf. Section 6.3.3) is to detect how many correct and incorrect readings a physical reader has made and then reflect this in the probability of the respective physical reader. However, we have already mentioned before (cf. Section 6.2.1) that the physical changes on the production lines are initially similar to incorrect readings. Due to this the probabilities of readers that detect physical changes are at first penalized incorrectly. In order to rectify this issue, it is important to detect physical changes. In this section, we will analyse different classes of physical changes to see which ones are actually critical, and how they can be distinguished from RFID reader errors.

Algorithm 4 Change Detection Event

```
1: for upon event ChangeEvent(PSL_g, vr_{i+1}, ps_conflicting) do
2:   for all pr_n ∈ vr_i do
3:     for all ps ∈ FPL_{pr_n} do
4:       if (ps == ps_conflicting) then
5:         remove ps_conflicting from FPL_{pr_n}
6:         trigger estimateProbability(pr_n)
7:       end if
8:     end for
9:     for all pso ∈ OOPL_{pr_n} do
10:      if (ps == ps_conflicting) then
11:        remove ps_conflicting from OOPL_{pr_n}
12:        trigger estimateProbability(pr_n)
13:      end if
14:    end for
15:    for all ps ∈ MPL_{pr_n} do
16:      if (ps == ps_conflicting) then
17:        remove ps_conflicting from MPL_{pr_n}
18:        trigger estimateProbability(pr_n)
19:      end if
20:    end for
21:  end for
22: end for
```

If a product part o_i is removed from the production line at a specific point, the *pr*s deployed prior to this point will not be able to report this change. However, the *pr*s deployed ahead of the point of change would be able to detect this change, which will be reflected by the non-detection of o_i. vr_i will not penalize the probability of any of the *pr*s for missing to detect this part. This is because vr_i only finds out if pr_i has missed out on detecting o_i if o_i is later detected by pr_j, where $(pr_i \sqsubset pr_j)$ (cf. Section 6.3.3).

If the partial sequence is changed as a result of either *sequence reversal* i.e. $(o_i < o_j)$ is changed to $(o_j < o_i)$ or *product part insertion* i.e. $(o_i < o_k)$ is changed to $(o_i < o_j < o_k)$, this change will soon be reflected in the PSL of vr_{i+1}. Initially, these two types of physical changes cannot be distinguished from RFID reader errors. Therefore, our strategy is to wait till the probability of the new partial sequence becomes greater then the probability of the older and conflicting partial sequence(s) before we make a definite statement about the actual product part order PPO_{act}. Moreover, old readings (readings before the physical change) will time out since we apply a time to live mechanism to readings. This increases the probability of the current sequence over time.

As an example, assume that the current sequence on the production line is $(o_1 < o_2 <$

o_3) with partial sequence ($o_1 < o_2$) having a probability of 0.9 and ($o_2 < o_3$) also has a probability of 0.9. However, if this sequence on the production line gets changed to ($o_1 < o_3$) as a result of removal of o_2 from the production line. Since prs would no longer be able to detect o_2, it would time out along with all the partial sequences in which it is participating. Whenever a product part o is detected by a pr its ttl is set to 1. This ttl is then decreased by a certain time unit every time a pr on the production line fails to detect it. However, if after being missed out by two prs, the third pr detects the product part o, its ttl is again refreshed to 1. Once a product part o is timed out, all the partial sequences in which product part o was participating are also timed out. Eventually the probability of ($o_1 < o_3$) would become greater than the probability of ($o_1 < o_2$) and ($o_2 < o_3$), since the two partial sequences in which o_2 was participating will be timed out. Whenever the probabilities of conflicting/non-unique partial sequences interchange, we trigger a change event which is also sent to the predecessor vr_{i-1} of this vr_i (cf. Algorithm 3 line 24-25).

Once a changed event is received/detected at a vr_i, the vr_i compares the conflicting partial sequences ($o_1 < o_2$) and ($o_2 < o_3$) with the FPL_{pr}, $OOPL_{pr}$ and MPL_{pr} of all of its prs to find and remove the conflicting partial sequences from these lists. The $p(pr_{est})$ of all prs that contained ($o_1 < o_2$) and ($o_2 < o_3$) in their lists is also recomputed (cf. Algorithm 4). If the probability of pr is decreased incorrectly for detecting the new reality, the recomputation of the probabilities of every pr that detected the conflicting partial sequence rectifies this error.

6.4 Evaluations

In this section, we discuss the performance of the probability self-calibration algorithm under simulated settings. We decided to evaluate our algorithm in a simulated setting because a simulation environment provides the possibility to evaluate a large scenario. Furthermore, we also wanted to test our algorithm in a controlled setting, whereby we could control/set the ground truths, such as the probabilities of prs. The simulations were performed using PeerSim [JMJV09]. All simulations were performed with 8,000 prs distributed across 1,000 nodes (vrs), except where otherwise specified.

A cycle in our simulations is the time taken by a product part o to move from one pr to the next one. So after every cycle a new product part is introduced on the production line, while the previous ones move ahead by one pr. We start calculating $p(pr_{est})$ after 50 cycles so that we have enough redundant readings to make a reasonable estimate. $p(pr_{est})$ is then continuously re-calculated every 5 cycles. To evaluate the performance of our algorithm we calculate the *accuracy* with which we are able to determine the estimated probability of each pr:

$$accuracy = (1 - |p(pr_{est}) - p(pr_{act})|/p(pr_{act})) * 100$$

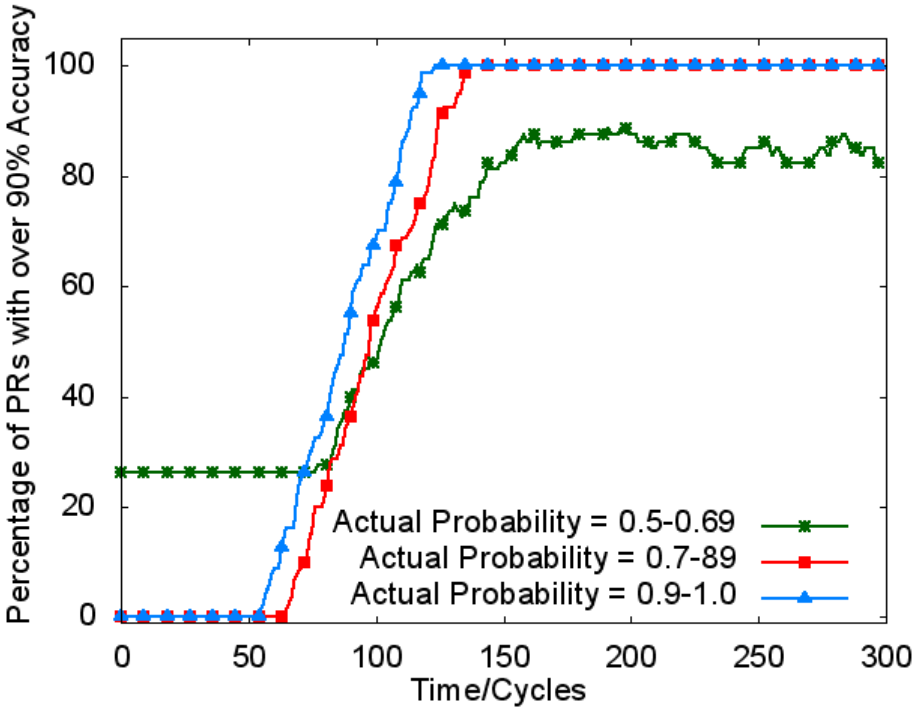


Figure 6.2: Calibration accuracy with random accuracy of PRs

The main performance metric is the time (in cycles) it takes for the algorithm to estimate the probabilities of all *pr*s with a high reliability.

6.4.1 Effect of Actual Probability of Physical Readers

In this scenario we evaluate the effect of different actual probabilities of *pr*s on the performance of the self-calibration algorithm. *pr*s are distributed uniformly among the *vr*s. The rate of induced physical changes is 50 cycle per change and the simulation is run for 300 cycles. The results (cf. Figure 6.2) show that the higher the actual probability of *pr*s, the less time it takes to calibrate the estimated probability with a relatively high accuracy. This is obvious since it is much harder to estimate the probabilities of *pr*s if there are only a few accurate readings.

6.4.2 Effect of Change in Actual Probability of Physical Readers

In this scenario we evaluated the time it takes for the algorithm to calibrate the estimated probabilities, when the actual probabilities of the *pr*s gets changed. Initially we assigned a random actual probability to each *pr* in the range of 0.5-0.9. Then the reliability of all *pr*s is abruptly changed. The new actual probabilities also lie within

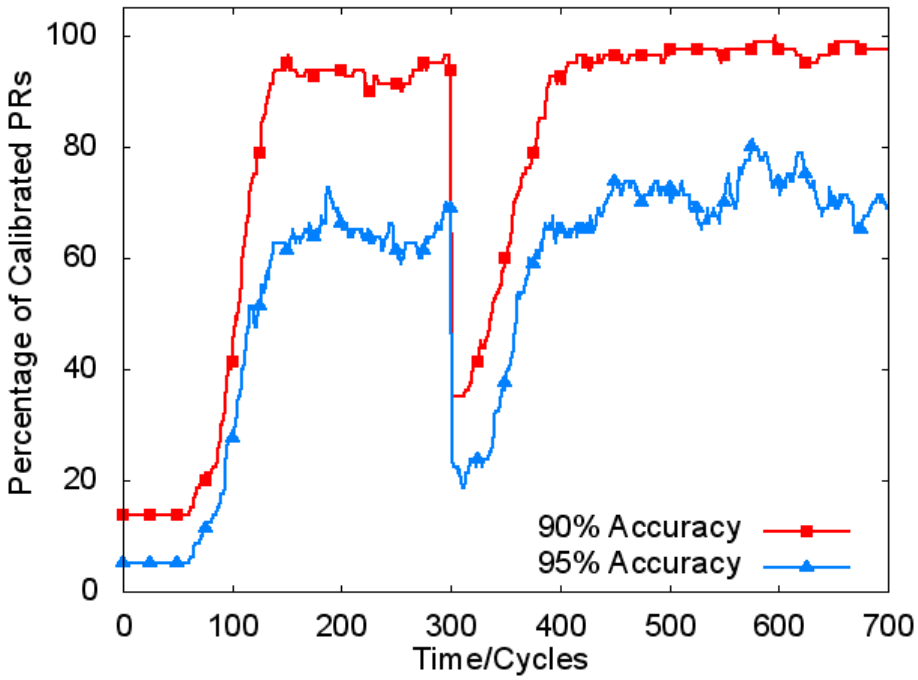


Figure 6.3: Calibration accuracy with change in actual probability

the range of 0.5-0.9. The evaluation runs for 700 cycles, and the changes were induced at the 300th cycle. Figure 6.3 shows the percentage of all *pr*s having a reliability of over 90% and 95% over the time. After the change is induced at cycle 300, the accuracy of estimated probability drops significantly. However the system becomes stable once again after approximately 100 cycles.

6.4.3 Effect of Rate of Physical Changes

In this evaluation we observed the effect on the performance of self-calibration algorithm under varying rates of physical changes. The actual probability of *pr*s is fixed to 0.7 and they are distributed uniformly among the *vr*s. The algorithm's performance was tested under three different change rates 50, 25, and 5 cycle/change. Figure 6.4 shows the percentage of *pr*s which have an estimation accuracy of over 90% over time. The results are almost same for 50 cycle/change and 25 cycle/change as can be seen in Figure 6.4, but for the scenario in which we were inducing a change every 5 cycles the calibration time increases along with a decrease in the accuracy of estimated probability. However, once the system reaches a stable state, the rate of change does not have much effect on accuracy of estimated probabilities.

6.4.4 Calibrated vs Uncalibrated System

In this scenario we observed the difference between a system in which *pr* probabilities are calibrated vs one which does not perform calibrations. We took three sets of readings by setting the actual probabilities of *pr*s to 0.5, 0.7, and 0.9. We then set the estimated probability of all physical readers to 0.5. Figure 6.5 shows how the estimated probability changes over time. In an uncalibrated system since the estimated probability does not change i.e. $p(pr_{est}) = p(pr_{act})$, the error between the actual probability and the estimated probability will never reduce. It is obvious from the results that the error between actual and estimated probabilities in this uncalibrated system could at best be 0 and at worst be 0.4. In normal scenarios, the error will depend on the difference between the configured probabilities for readers and their actual probabilities. However, the error in our system once the algorithm calibrates the estimated probabilities is never greater than 0.05.

6.4.5 Effect of Number of Virtual Readers

In this scenario we observed the effect of the number of *vr*s on the performance of the self-calibration algorithm. The actual probabilities of *pr*s were fixed to 0.7 for this experiment and *pr*s were distributed uniformly among all *vr*s. The number of *vr*s used were 1000, 2500 and 5000. The simulation was run for 300 cycles, with a physical change in the product part order being induced every 50 cycles. Figure 6.6 shows the percentage of *pr*s that attained a calibration accuracy of over 90% over time. It is clear from Figure 6.6 that calibration time is reduced with an increase in the number of *vr*s, but this also increases the error in the probability estimations. The calibration time is reduced because each *vr* has lesser number of *pr*s and hence a small data sample, which enables the *vr*s to estimate the probabilities quickly. But now since each *vr* has a smaller data sample, the estimation accuracy suffers.

6.4.6 Effect of Distribution of Physical Readers

In this scenario, we observed the effect of *pr* distribution on the performance of the self-calibration algorithm. The actual probability of *pr*s was set to 0.7 and the simulation was run for 300 cycles with changes induced at every 50^{th} cycle. The *pr*s were distributed amongst the *vr*s using uniform and zipfian distribution. In zipfian distribution we set $\alpha = 1.0$. This ensured that 80% of the *pr*s are distributed across half of the *vr*s, whereas the remaining 20% of the *pr*s are distributed across the remaining half of the *vr*s. Figure 6.7 shows the results of this evaluation. It is obvious from the results that the distribution of *pr*s has no effect on the accuracy of the calibration process.

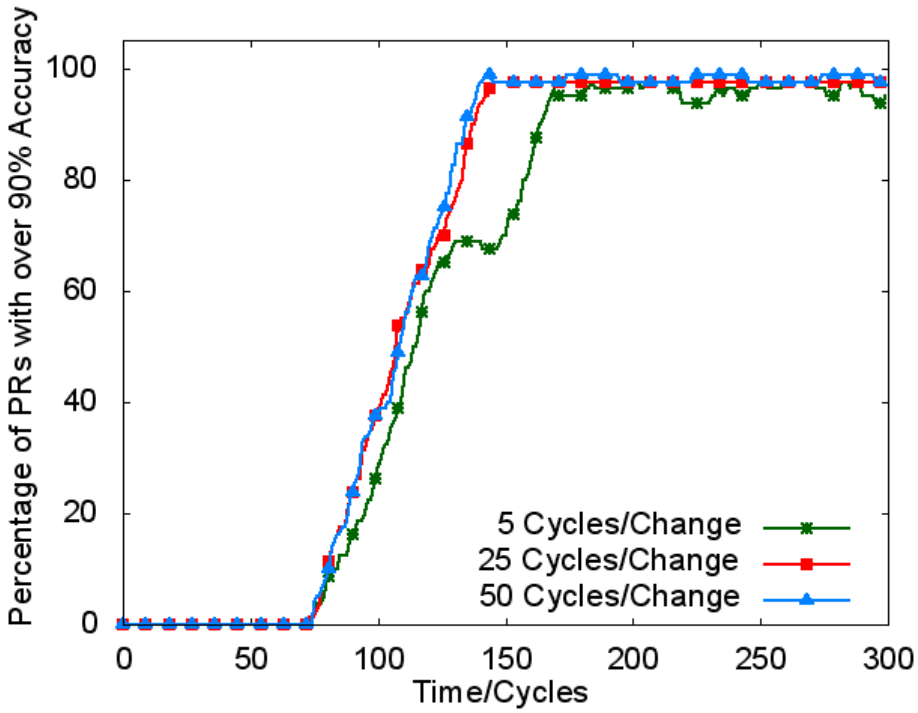


Figure 6.4: Calibration accuracy with different rate of change

6.5 Related Work

In this chapter, we presented a self-calibration algorithm to calibrate the probabilities of RFID readers so as to have a higher reliability for the objects detected by these RFID readers. RFID devices are inherently unreliable, therefore the reliability issue has been at the core of many research efforts undertaken in this area. In this section we will briefly discuss some of the research work that has been done to enhance reliability in RFID devices under different deployment scenarios.

Reliable Estimations of RFID Tags in Retail Industry: A lot of retailers have now deployed RFID technology to optimize their inventory and commodity flows [GSH07]. However, this deployment has lead to a separate set of issues due to the unreliable nature of RFID devices. The issue is that RFID devices do no read all the tags for a given assembly of items, as some of the tags are always at the blind spot of a reader. Due to this, the retailers can not have a reliable estimate of their inventory for replenishment planning and other applications.

RFID devices can not determine their tag-identification rates on their own. Similarly, unread tags can not be determined till the physical conditions that led to their non-detection are changed.

In order to solve this issue and have a reliable estimate of the total RFID tags within

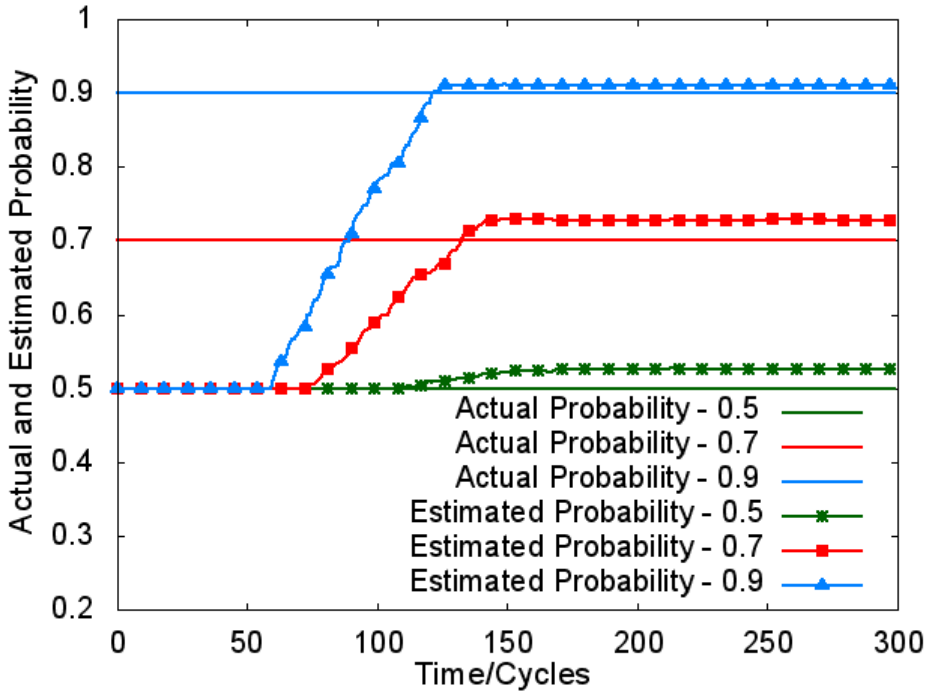


Figure 6.5: Calibrated vs Uncalibrated system

a certain area, Weiss et al [WFCBB08] have proposed a strategy which is similar to mark and recapture methods [Seb82], [Sch38] used by scientists to estimate an animal population's size. In mark and recapture methods, a portion of the population (whose size is being estimated) is captured, marked and then released. After a certain time, another portion of the population is captured and the number of species that were marked in the previous capture are counted. The theory is that the number of marked individuals in the second sample would be proportional to the number of marked individuals in the whole population of the species. So the total animal population is obtained by dividing the number of marked individuals by the proportion of marked individuals in the second sample. The formula for mark recapture method is given below:

$N = MC/N$; where

- N = estimate of the total animal population size
- M = total number of animals captured and marked in the first sample
- C = total number of animals captured in the second sample
- R = number of animals captured and marked in the first sample that were then present in the second sample.

The TagMark system proposed by Weiss et al [WFCBB08] works on the principle of

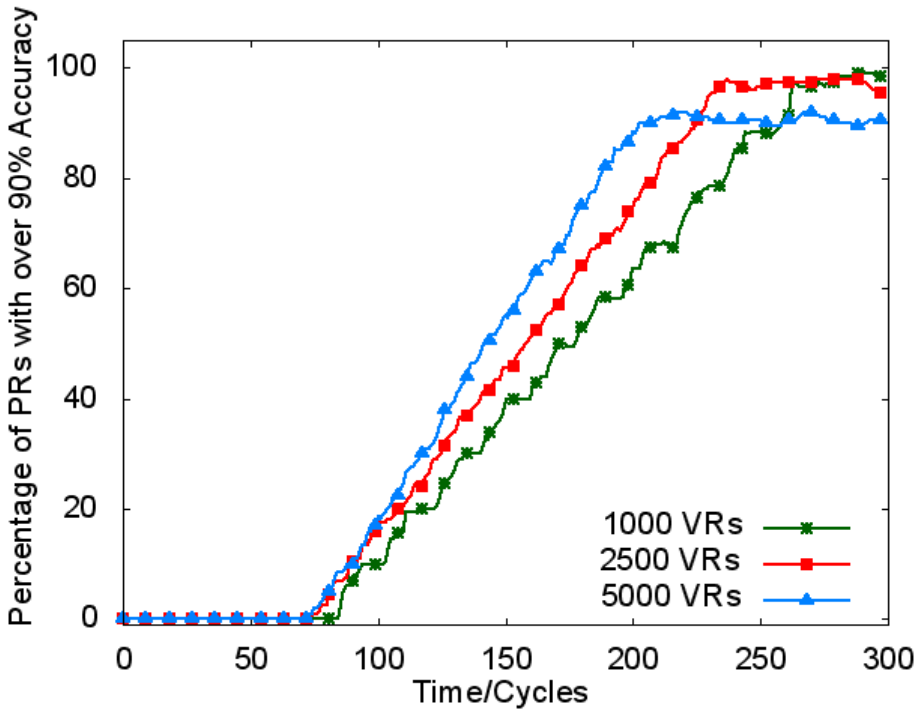


Figure 6.6: Calibration accuracy with different number of VRs

mark recapture method, except that the total population being estimated is that of RFID tags instead of animals. A portion of the inventory is captured and marked. After a certain time, the inventory is read again and the total number of tags/items that were marked previously in this second capture is used to determine the overall size of the inventory.

An assumption of the mark recapture method is that the study population is closed i.e. no individuals die, are born, or move into or out of the study area. This assumption is easier to control with retail inventories as compared to animal population in a given area.

Jacobsen et al [JNPL09] have also proposed a similar method to reliably estimate the total number of tags using multiple independent session readings. However, they have extended their work to provide estimates for the error probability and the probability that tags are missing as well.

Identifying RFID Tags Using Group Completeness Technique: Backes et al [BGK11] have employed a group completeness technique to address the missing tag problem. RFID tags of a certain group contain references to other tags within their group. This group of RFID tags are then placed on a certain consignment or set of pallets. The RFID reader maintains two sets 'X' and 'Y', which are initially empty. For each tag read, the reader stores the id of the tags in set X and the references

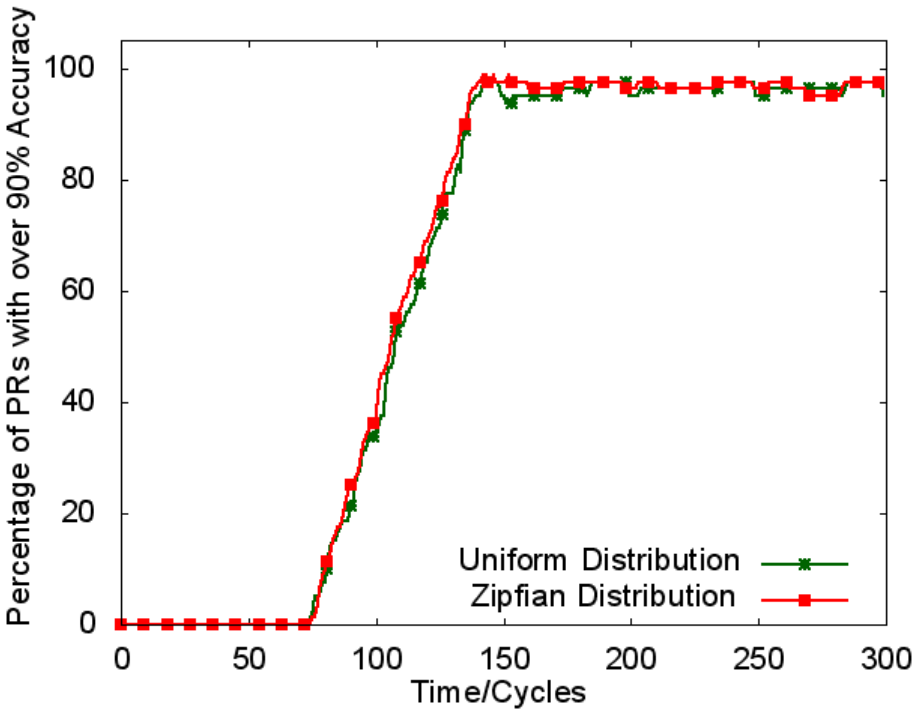


Figure 6.7: Calibration accuracy with different PR distributions

contained within this tag in set Y. The RFID reader repeats this process for all the items or pallets that move across the reader. Once the process is complete, the two sets are compared. If all the tags in set Y are not present in set X, the reader has missed out on reading certain tags. In this way, the system is able to find out with a high probability that the tags are missing.

ASSIST - Automated System for Surgical Instrument and Sponge Tracking: In [RMA+08] Rivera et al. developed an RFID based automated system (ASSIST) for tracking sponges that are used during surgeries. During medical surgeries, the problem of forgetting objects inside a patient's body might lead to his death. With all the caution taken by nurses and doctors, miscounting the sponges or other errors would bring the problem back. Reports estimate that forgetting sponges or other objects within the patients body happen once in every 1500 surgeries. ASSIST tries to solve this very problem. The system uses low frequency RFID since the readers in this case can read tags even if they are inside the patients body and covered with different body fluids or organs. The process of sponge tracking starts at a check-in station. It verifies that all sponges in a package are available and registers them in the database for future usage in the tracking process. When the system initiates, the number of available sponges are displayed on the GUI. All the sponges that are used are tagged with an RFID tag. After the doctor has used a sponge, he should discard it by putting it in a bucket that signifies a check-out station. The number of discarded and checked-

in sponges is updated when a sponge is discarded and is simultaneously displayed on the GUI. At the end of each operation the number of checked-in sponges should be equal to the number of checked-out sponges, else the system would generate an error message.

Once an error message is generated, the doctors use a patient scanner which is a blanket embedded with RFID antenna to find the missing sponges. The blanket is spread over the patients body to read any tag that might be present within the patient's body. X-rays can not be used for this purpose since they do not efficiently detect presence of a sponge if its near to bones. Rivera et al's experiments showed that the patient scanner can detect sponges tagged with RFID tags within a vivo porcine model in less than 5 seconds.

Reliability of RFID in Metal Environment: Arora et al. [AMK+07] studied the effects of metal on the performance of RFID at ultra-high frequency (UHF). In their experiments, they set up a metal sheet and attached an RFID tag to it. A robot holds the RFID reader and moves in front of the tags within a certain area. For each position of the RFID reader they recorded whether the reader was able to read the tag or not. The study found that for different metals the reading reliability is different.

Arora et al. also compared the reliability of RFID readers when different metals were used. The three different metals that they used in their study were: Brass, Aluminium, and mild steel. For reading reliability, mild steel settings gave the highest reading rate. Aluminium came second and Brass third. To improve the reading reliability the authors used three different techniques:

- using a spacer between the tag and metal,
- providing offset to the tag,
- and angling tags i.e. lifting them up from one end.

The first technique showed huge improvements in reliability without compromising the read rates. The second and third technique also improved the reliability, but offered less reading rates as compared to the first technique.

Reliability Factors in RFID Deployments: Rehmati et al. [RZHJ07] investigated several factors that affect the reliability of RFID system for tracking applications. These factors include:

- Distance between neighbouring tags.
- Distance between the tags and the RFID reader.
- Orientation of the tag when attached to an item.
- Location and the number of tags on an object.

In their work, the authors focused on passive tags as they have weak signal and a lower read reliability than active tags. They performed extensive experiments and found

that the reliability of readings decreases when the distance between the reader and the tags increase. Furthermore, reliability is increased when the distance between the tags is increased. The authors also tested six different orientation settings and found that the orientation of tags is a very significant factor in the reliability of RFID readers. Orienting tags perpendicularly against the reader reduces the reliability of reading to its minimum; however orienting tags such that they are facing the reader improves the reading reliability significantly.

The reliability of RFID readers can also be improved by simple and cost-effective redundancy techniques like redundancy at the reader level, the tag level and the antenna level. Different measurements taken during the study show that redundancy at tag level is the most effective form of redundancy. After tag level redundancy, antenna level redundancy has shown considerable improvements in system reliability. The reader level redundancy however, significantly reduced reliability as RFID readers did not support dense reader mode and the presence of several RFID readers caused reader to reader interference. These results can be helpful in deployment of RFID-based tracking applications which require a certain level of reliability.

Detecting Cloned Tags - Forced False Positives: Tag cloning is a technique in which you copy the data and values of a genuine tag and use that data in some other (counterfeit) tag. This activity is done to launch counterfeit products into the market. Tag cloning causes confusion in an RFID tracking and tracing system and a financial lose for businesses that depend on RFID technology to track their products. Traditional security techniques either rely on encryption [BGK+07], [Jue05], [WHC06], [PLHCTR09] or some form of authentication [Dim05], [TB06], [SM08] that is performed before a tag sends its data to a reader in order to avoid tag cloning. However, Lehtonen et al. [LMF09] have presented a non-cryptographic solution that relies on RFID traces to detect if a tag is genuine or counterfeit.

The basic idea of the approach is that a genuine tag has a normal predefined logistical route i.e. the product on which the tag is attached would move from location A to B and then to C. A cloned tag however, would not have such a logistical route and might have directly appeared at location C. Sensing a tag at location A, then sensing it at location B is called a transition from A to B. A probability is assigned to all transitions. Counterfeit tags either directly appear at a location or have transitioned through a low probability route and hence are easily weeded out.

Discussion: Weiss et al [WFCBB08] have tried to solve the issue of reliable estimation of total number of RFID tags within a certain area by employing mark and recapture methods [Seb82], [Sch38] used by scientists to estimate an animal population's size. Backes et al [BGK11] on the other hand have employed a group completeness technique to address the missing tag problem. RFID tags of a certain group contain references to other tags within their group. The references to all the tags in the group must be resolved in order for the system to assume that all the tags have been read.

Arora et al. [AMK+07] studied the effects of metal on the reliability of RFID technology. The study did not proposed a particular solution to the reliability issue, but nevertheless could be used by researchers trying to address the reliability problems in their systems research. Rehmati et al. [RZHJ07] investigated the impact of several factors on the reliability of RFID system for tracking applications. These factors include: distance between neighbouring tags, distance between the tags and the RFID reader, orientation of the tag when attached to an item, location and the number of tags on an object. Similar to Arora et al. [AMK+07], the work does not presents a specific solution to the reliability problem. But nevertheless is another important work towards benchmarking the core issues and factors that critically impact the reliability of RFID devices.

Tag cloning, is a method whereby you induce counterfeit tags into the system. The whole process could also be defined as a forced false positive problem - whereby an adversary is creating a situation to make your RFID system have false positives i.e. accept fake tags as real ones. Traditionally, the problem was addressed using encryption based solutions [BGK+07], [Jue05], [WHC06], [PLHCTR09] or using some form of authentication [Dim05], [TB06], [SM08]. However, Lehtonen et al. [LMF09] have presented a non-cryptographic solution that relies on RFID traces to detect if a tag is genuine or counterfeit. Our approach to detecting false positives etc. is similar to Lehtonen et al, in a sense that we also incorporate contextual knowledge of the plant layout and movment of production parts to weed out false positives.

What really sets us apart from the work done to address the unreliability of RFID devices to date is that the work done till now has either tried to some how estimate the total number of tags or avoid false tags using some filtering techniques. We on the other hand have tried to assign an reliability attribute to the RFID readers. This attribute or probability is than continuously calibrated to reflect the reliability with which the RFID device correctly detects product parts moving on the production lines. This reliability measure is then use to allocate a corresponding weightage to the readings of that RFID device.

6.6 Summary

In this chapter, we have presented concepts for the reliable monitoring of product parts in production with unreliable RFID sensors. Based on a probabilistic model, we have presented algorithms for self-calibration of RFID readers to reflect the probability of real errors. The basic idea of this approach is to exploit redundant readings to get an accurate model, which is then used for calibration.

The evaluations of our probability self-calibration algorithm shows that it reacts robustly to induced changes as it was able to calibrate itself to a stable state with an accuracy of greater than 90% even in the presence of changes induced after every 5

cycle. The evaluations further showed that the algorithm was more accurate if the *pr*s are clustered together within a small number of *vr*s, since that scenario provides each *vr* with a significantly larger RFID dataset to calibrate the probabilities. The evaluations also revealed the obvious fact that accuracy of probability self-calibration algorithm is dependent on *pr* reliability.

Chapter 7

RFID Based Complex Event Processing In A Smart Variant Production Environment

The focus of our work till now was to track and monitor production in real-time and to do so with a high degree of accuracy and precision. However, the low level RFID events, such as $part_1$ at $position_1$, are of little importance to the people running the factories. What plant managers want is access to high level information that allows them to see if everything is going according to plan or not. And in case, things are not going according to plan, they want to be informed about whats going wrong, where it is going wrong and what to do in order to remedy the situation.

This demand calls for a need to process low level RFID data and generate complex but meaningful manufacturing events that can be understood and acted upon by the people responsible for the production environments. In order to solve this issue, we have developed a probabilistic complex event processing framework that detects complex manufacturing events and assigns probabilities to these events. These probabilities are continuously updated and represent the confidence that the system has in the accuracy of a certain complex event at any instance of time. In particular our system detects the following complex manufacturing events: a) sequence errors, b) synchronization errors, c) delay errors, d) incorrect part position errors, and e) missing part errors.

Several RFID based CEP systems [ZZ08], [HYHZ08], [WDR06], [WLLB06] have been proposed by the research community. However, these systems merely provide an event language that can be used by the RFID community to generate general RFID-based complex events. The problem with the complex manufacturing events that we have listed above is that they can not be generated using a generic CEP event language. Due to this reason, we have had to design specific and novel algorithms to detect each of the complex manufacturing event that we have outlined above. The algorithms that we have designed can be thought of as operators in our CEP system that enable the manufacturing organizations to detect sequence, synchronization, delay, incorrect part position and missing part errors.

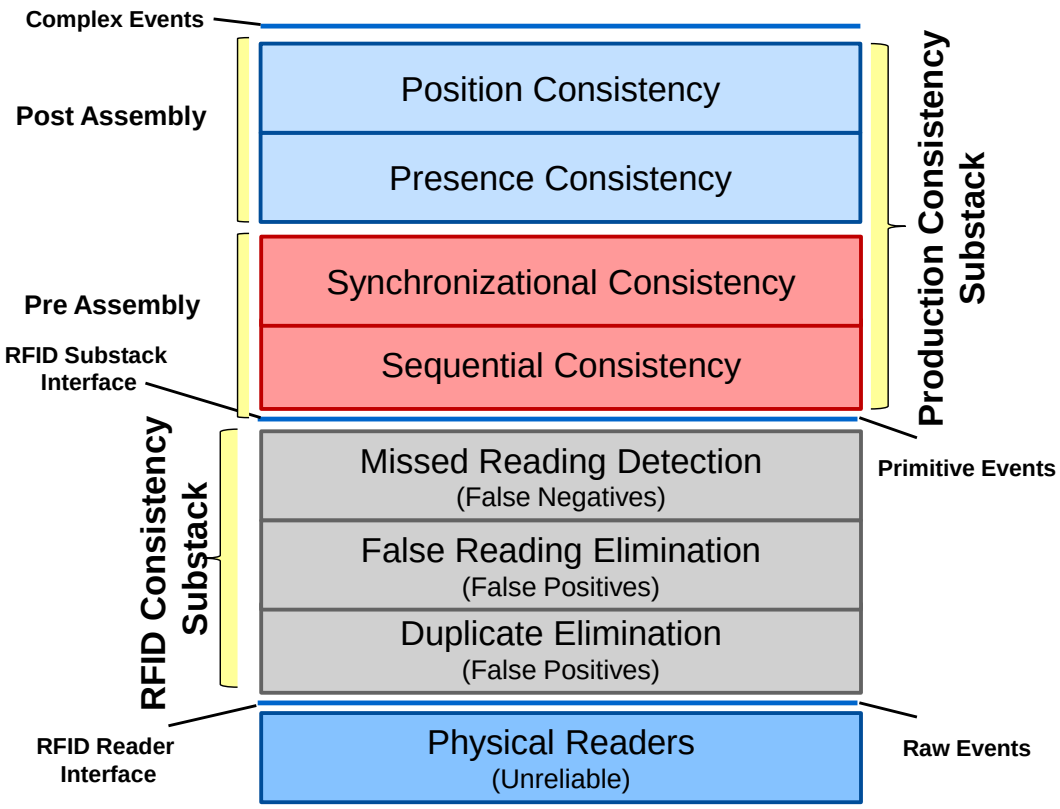


Figure 7.1: RFID Consistency Stack [Extended]

In addition to this, accuracy and reliability is of paramount importance in production environments. The reason for this being that plant managers can not work upon information unless it is highly accurate and precise. In order to fulfil this requirement, we have designed a probabilistic model that assigns probabilities to each and every complex event. These probabilities then serve as a measure of confidence about how reliable a certain complex event really is.

The main contributions presented in this chapter are as follows. First, we propose an extension to our previously introduced consistency stack (cf. Chapter 5.1), which is a conceptual model for describing different kinds of RFID reader and production errors. Second, we propose probabilistic models and algorithms for detecting different complex manufacturing errors/events. Third, we present simulation results to show that our approach reliably detects the complex manufacturing events that we set out to detect in the first place.

The rest of the chapter is structured as follows. The extension of the consistency stack is described in Section 7.1. In Section 7.2, we describe the different complex manufacturing events, where as the algorithms to detect and assign probabilities to the detected complex events are discussed in Section 7.3. Section 7.4 details the evaluations,

followed by a review of related work in Section 7.5. Finally, we conclude the chapter with a short summary in Section 7.6.

7.1 Consistency Stack Extension

We have categorized the different consistency issues that may arise during real-time production monitoring using RFID readers into a layered model called the consistency stack. The consistency stack is comprised of two distinct sub-stacks, the RFID consistency substack and the production consistency substack (cf. Figure 7.1). The RFID consistency substack deals with RFID reader errors that can cause problems in proper monitoring of the production processes, whereas the production consistency substack deals with inconsistencies between the production plan and the current state of product parts for e.g. the planned sequence vs. the actual sequence detected by the RFID consistency substack. The RFID consistency substack has been discussed previously (cf. Chapter 5.1), so for the purpose of brevity we will only discuss the different layers of the production consistency substack in the sub-sections below:

7.1.1 Production Consistency Substack

The production consistency substack ensures that the production process is consistent with the production plan and makes sure that errors are detected as quickly as possible. The production consistency stack is further sub-divided into pre-assembly consistency issues and post-assembly issues.

The pre-assembly issues (sequence and synchronization consistency) are issues that arise before the product parts have actually been assembled together. The detection of these issues results in avoidance of incorrect and undesirable products being assembled. The post-assembly issues are issues that are a result of incorrect and undesirable product assembly. The detection of the post-assembly issues would enable the factory to not ship out the undesirable products, or remove them from the production lines to avoid further processing.

The positioning of the pre-assembly and post-assembly issues within the production consistency stack is significant, because if the pre-assembly issues are not resolved in a timely manner, they invariably lead to post-assembly issues. As an example, consider that two product parts o_{n1} and o_{n2} are moving in sync with each other, where product part o_{n1} belongs to product $n1$ and product part o_{n2} belongs to product $n2$. This is a synchronization error, since only product parts belonging to the same product should move in sync with each other on the production lines. If both of these parts o_{n1} and o_{n2} would continue to move in sync with each other, they would reach an assembly point together and hence would be assembled with each other. This would lead to presence inconsistency, which is a post-assembly consistency issue.

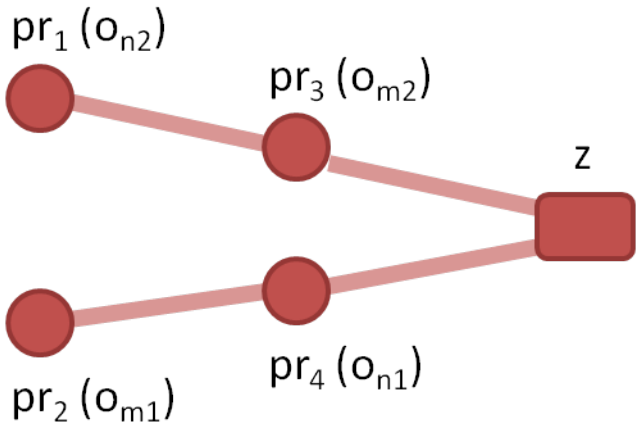


Figure 7.2: Sequence Inconsistency Leading to Synchronization Inconsistency

We have already discussed the pre-assembly consistency issues i.e. sequence consistency and synchronization consistency (cf. Chapter 5.1.2), so would only discuss the post-assembly consistency issues in the sections below.

Presence consistency: When two product parts are assembled together, it must be ensured that not only they are assembled correctly, but the right parts have been assembled together. The presence of incorrect parts on the product or absence of correct parts on the product is referred to as presence consistency.

As an example, lets consider the following scenario. We have an assembly point ap_1. At ap_1 two production lines meet and product parts from both the production lines are assembled together. If we have product part o_{n1} on one production line and o_{n2} on the other production line, they would be assembled together at ap_1. Here n denotes that both the product parts belong to product n. So in this scenario we will not have any presence inconsistency since firstly, both the product parts belong to the same product, and secondly both the product parts are present i.e. assembled into the final product.

If on the other hand we would have had product part o_{n1} on production line 1 and o_{m1} on production line 2, this would have lead to presence inconsistency, since the semi-assembled product that leaves the assembly point ap_1 should have had product parts o_{n1} and o_{n2}. The absence of the correct product part o_{n1} and the presence of the incorrect product part o_{m1} is a presence inconsistency.

Similarly, if we would have had product part o_{n1} on production line 1 and no product part on production line 2. The semi-assebled product would only have product part o_{n1} on it. This absence of product part o_{n2} is also a presence inconsistency.

Position consistency: Position consistency ensures that the right product part has been assembled at the right position. Position consistency is a superset of presence consistency, which merely demands that the right product part be assembled. Position

consistency goes a step further and requires that not only should the right product part be present on the assembled product, it should be assembled at the right position. However in situations were the position of parts is irrelevant, we can leave out position consistency checks.

In order to further understand position consistency lets take a concrete example from the lernfabrik (cf. Chapter 4). Within the Lernfabrik, we have three product parts that are assembled on the baseplate. The baseplate has three position holders for each of these three parts, which could be labeled as position 1, position 2, and position 3. Each of these three parts can be placed on any of the three positions. However, product part 3 (i.e. the thermometer or the hydrometer) is assembled by a robotic arm, which is programmed to place product part 3 on position 3. Before, assembling product part 3 on the baseplate, the robotic arm checks with the help of sensors and cameras if position 3 is empty. If it is, the robotic arm assembles product part 3 onto the product, if however, the position is not empty the robotic arm assumes that it has already assembled product part 3 and hence performs no task.

There can be a situation when a human worker by mistake places product part 2 on position 3 of a baseplate instead of position 2. In such a scenario, when this product would reach the robotic worker, it would not assemble product part 3 and would let the baseplate pass as is. This production error, resulting from incorrect positioning of product parts on the product within the lernfabrik can only be remedied by human intervention.

Discussion: Before moving on to the next section, it is pertinent to discuss the rationale behind the positioning of different production consistency issues within the production consistency substack. Within the production consistency stack, the pre-assembly consistency issues are below the post-assembly issues (cf. Figure 7.1). This is intuitive, since it is logical to first try and resolve the pre-assembly issues and then move on to resolve the post-assembly issues.

However, a second and perhaps the more important reason for this ordering is that each of the lower consistency issue, if left unresolved eventually leads to the higher level consistency issue. To understand this, lets consider a concrete example. According to a production plan, product parts o_{n1} and o_{m1} have to move on production line 1 such that $(o_{n1} < o_{m1})$ i.e. o_{n1} is ahead of o_{m1} and product parts o_{n2} and o_{m2} have to move on production line 2 such that $(o_{n2} < o_{m2})$ i.e. o_{n2} is ahead of o_{m2}.

o_{n1} and o_{m1} move according to plan on production line 1. However on production line 2, $(o_{m2} < o_{n2})$ (cf. Figure 7.2). This is a sequence inconsistency. In order to be assembled together correctly, product parts o_{n1} and o_{n2} should reach the assembly point in sync. with each other. However, since we have a sequence inconsistency on production line 2, product part o_{n1} would be moving on the production line in sync. with o_{m2}. So in this case, the sequence inconsistency has led to a synchronizational inconsistency as well. Now once, both o_{n1} and o_{m2} would reach the assembly point, they would be assembled

together. Since both product parts belong to different products, we now have presence inconsistency as well.

From the example presented above it is obvious that lower level issues such as sequence inconsistency if not addressed immediately and adequately would result in higher level issues such as position inconsistency. The opposite of this situation is not true i.e. a position consistency has no effect on sequence consistency.

7.2 Complex Manufacturing Events

Complex events are the result of processing primitive events using application rules/ operators. The type of a complex event depends on the application and the processing rules defined for it.

Since RFID devices are inherently unreliable, there is a possibility that physical readers *pr*s may detect objects moving on the production lines differently from each other. As an example there could be a situation whereby pr_1 make detect product part o_1 and o_3 and miss out on detecting product part o_2, where as pr_2 may detect product part o_1 and o_2 and miss out on detecting product part o_3. In such a scenario, the complex events detected from the read events of one pr would be different from the ones detected from the read events of the very next pr. To solve this issue, we assign probabilities to the derived complex events. These probabilities are then constantly updated as we gather more evidence regarding these complex events.

The *probability of a complex event* $p(ce)$ denotes the probability with which a deduced complex event matches reality.

In addition to the $p(ce)$ we also have complex event threshold ce_{thd}, which is a global variable. The purpose of this variable is to allow the system users to determine when to trigger the complex events. The complex events in the system are generated when the $p(ce)$ becomes greater than ce_{thd} i.e. $(p(ce) > ce_{thd})$. As an example suppose, the ce_{thd} is set to 0.8, as a result whenever a complex event would be detected it wont be triggered right away. Rather the system would wait till the probability of that complex event $p(ce)$ becomes greater than 0.8. Once the probability of the complex event $p(ce)$ becomes greater than 0.8, it would be generated/triggered.

In the sections below, we will discuss in detail the different complex events that are generated in our system alongwith the application rules/algorithms that we use to detect these events.

7.2.1 Sequence Error

In a variant production environment the product parts must reach the assembly points in certain defined sequences in order for the products to be assembled correctly. If

product parts are not moving on the assembly lines in defined sequences, we would have sequence errors.

As an example, lets assume that the planed product part order PPO_{pln} is $(o_1 < o_2 < o_3 < o_4)$. If due to some human error the actual order of product parts PPO_{act} on the production line is $(o_1 < o_2 < o_4 < o_3)$, we would have a sequence error. We have already discussed the process of partial sequence and extended sequence detections (cf. Chapter 5.2), however not every sequence detection leads to a sequence error. Sequence errors are merely those sequences that deviate from the planned product part order PPO_{pln}. The exact process of detecting and generating sequence error events is discussed in Section 7.3.1

7.2.2 Synchronization Error

The product parts should not only reach the assembly points in time, they should also reach the assembly points in certain defined time span in order to be correctly assembled with corresponding parts that would be arriving at the assembly points from other production lines. If a product part does not reach the assembly point in its defined time span, we would have a synchronization error. The synchronization error can be caused by both sequence issues and delays on the production line.

As an example, lets assume that product part o_{n1} is moving on production line 1 and product part o_{n2} is moving on product line 2. Now since, both of these product parts belong to product n, they should reach the assembly line at the same time in order to be assembled together. In order to reach the assembly point at the same time, both of these product parts should move on their respective production lines in sync with each other i.e. when o_{n1} is 2 minutes away from the assembly point, o_{n2} should also be only 2 minutes away from the assembly point. If o_{n1} is 2 minutes away from the assembly point but o_{n2} is 10 minutes away from the same assembly point, we would have a synchronization error event.

The synchronization error can be caused by both sequence issues and delays on the production line. The remedy to each of these specific case is different. A synchronization error caused by a delay on the production line can be rectified by stopping the product parts on the other production line as well. Syncrhonization errors caused as a result of sequence issues can not be resolved without resolving the sequence error.

The task of detecting whether product parts are in sync with each other or not and generating synchronization error events is explained in Section 7.3.2

7.2.3 Delay Error

Production lines could break or stop moving due to mechanical faults. Such a scenario would lead to delay errors.

As an example, lets assume that we have two production lines and 10 product parts moving on each of these production lines. The product parts moving on production line 1 are ($o_{a1} < o_{b1} < o_{c1} < o_{d1} < o_{e1} < o_{f1} < o_{g1} < o_{h1} < o_{i1} < o_{j1}$), where as the product parts moving on production line 2 are ($o_{a2} < o_{b2} < o_{c2} < o_{d2} < o_{e2} < o_{f2} < o_{g2} < o_{h2} < o_{i2} < o_{j2}$). If the production line 2 experiences a mechanical failure and breaks down after product part o_{c2} has passed, the system would initially believe that the physical readers deployed on the production line are missing out on detecting the product parts behind o_{c2}. However, the delay error detection algorithm (cf. Section 7.3.3) would soon realize that the *pr*s are not missing out on detecting the product parts, but rather its a delay on the production line.

Since the physical readers deloyed ahead of the fault would no longer be able to detect the product parts, the delay error would also lead to synchronization errors. The task of detecting delay errors on the production line is explained at length in Section 7.3.3.

7.2.4 Missing Part Error

Since a large part of manufacturing still requires human intervention which is an error prone endeavour, we frequently have situations whereby a worker misses out on assembling a certain part. Such a situation leads to missing part errors.

As an example, lets assumed that a human worker is standing at an assembly point and product part o_{n1} arrives on production line 1 and product part o_{n2} arrives on production line 2. The worker misses out on assembling product part o_{n2} with o_{n1} and lets product part o_{n1} pass through the assembly point. This is an error since all products leaving this assembly point should have both product parts.

It is highly likely that this worker would then assemble product part o_{n2} onto a different product and thus create yet another production error. Whenever, a product leaves an assembly point, we check if all the required parts are assembled or if the product is missing some part. If some part has not been assembled, we generate a missing part error. The actual task of detecting missing product parts is discussed in Section 7.3.4.

Although, this might seem to be a very trivial and straigh forward task, yet missing product parts are one of the most common production errors. We have already discussed in the introduction of this dissertation (cf. Chapter 1) about how Porsche has a lighting system to find out if a missing part error has occured. The way Porsche accomplishes this is by placing product parts in a cart at an assembly point. Once a car arrives at that assembly point, all parts within the cart must be assembled with the car. If the cart is not empty within five minutes, lights on the cart would start blinking to indicate that the worker(s) at this assembly point have not assembled all the parts onto one of the cars that left this assembly point within the last five minutes.

7.2.5 Incorrect Part Position Error

In some products, there is a possibility that two or more product parts could be assembled interchangeably. Each of these parts should be assembled at its correct location. However, since the design allows for one part to be assembled at the place of the other one, there is a possibility that a part is assembled at the wrong position. Such scenarios result in incorrect part position errors.

From a production consistency stack standpoint, incorrect part position error is actually a position consistency issue (cf. Section 7.1.1). The detection of incorrect part position error is discussed in detail in Section 7.3.5.

7.3 Probabilistic Complex Event Detection

The goal of probabilistic complex event detection is to detect complex manufacturing events reliably. Each complex event has an associated probability, which enables applications to query the real-time production monitoring framework for the most probable complex events that have been detected within the factory.

In this section we discuss the different complex event and explain how each of these events is detected. In addition to this, we also present the algorithms that are used to assign probabilities to these complex events so that we could have a confidence measure for each detected complex event.

7.3.1 Sequence Error Detection

In this section we present the core concepts for detecting sequence errors on the production lines. First we would provide an **overview** of our approach. Then we describe the actual **probabilistic sequence error detection algorithm** and explain how sequence error events are generated in our system.

Overview:

The main steps involved in detecting a sequence error are:

1. Detect all possible product part sequences ps. The outcome of this step are multiple product part sequences, each of which are stored in a global partial sequence list PSL_g, which contains all possible ps deduced by our system.
2. For every detected ps, we try to calculate $p(ps)$ which is a probability for ps being the correct sequence. $p(ps)$ is calculated from the probabilities of associated read events that participated in the detection of ps.
3. Whenever, the probability of the partial sequence $p(ps)$ is modified, we try to compare it with complex event threshold ce_{thd}. If the probability of the partial sequence $p(ps)$ is less than ce_{thd}, we do nothing.

Algorithm 5 Sequence Error Detection Algorithm

```
1: Let e_i = (o_i, pr_i, t_i) be a read event detected at vr_i
2: PSL_g;
3: ce_thd;
4: while pr_i detects o_i do
5:    ps = PartialSequenceDetection(o_i);
6:    if ps.p(ps) > ce_thd then
7:       if ps does not belongs to PPO_pln then
8:          ce_seq ← ps;
9:          trigger ce_seq;
10:      end if
11:   end if
12: end while
```

4. If however, the probability of the partial sequence $p(ps)$ is greater than ce_{thd}, we try to determine if it is a sequence error ce_{seq} or not.

Sequence Error Detection Algorithm:

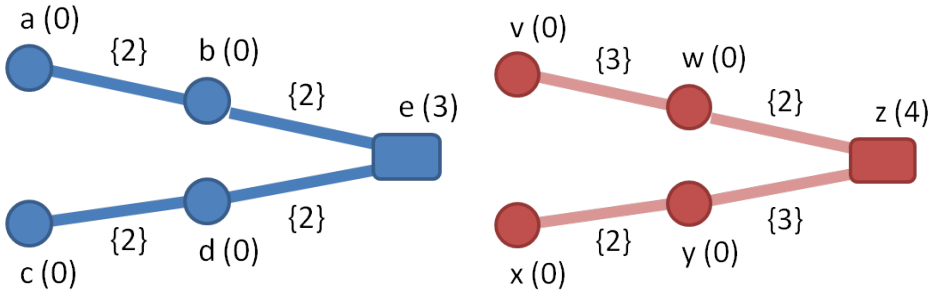
The process of sequence error detection involves two major steps:

- Partial Sequence Detection
- Sequence Error Generation

Partial Sequence Detection: The process of detecting partial sequences and assigning probabilities to these sequences (i.e. steps 1 and 2 of Overview) have already been discussed at length previously (cf: Chapter 5.2). So for the purpose of brevity we would assume that the partial sequences are detected using the sequence detection algorithm. Furthermore, each of these partial sequences have associated probabilities which are continuously updated to reflect the confidence with which these partial sequences match reality on the production lines.

Sequence Error Generation: Whenever a new sequence is detected or we have a change in the probability of the existing sequence (ps), we compare $p(ps)$ with complex event threshold ce_{thd}. If the probability of the sequence $p(ps)$ is less than ce_{thd}, we do nothing (cf: Algorithm 5 lines 4-6).

If however, the probability of the sequence $p(ps)$ is greater than ce_{thd}, we compare the sequence ps with the planned product part order PPO_{pln}, which is the sequence/order in which the product parts are supposed to pass on the production line. If the sequence ps belongs to the planned product part order, it means that everything is going according to plan. If on the other hand, the sequence ps is not present in the planned product part order PPO_{pln}, it is a sequence error and hence we generate the sequence error event ce_{seq} (cf: Algorithm 5 lines 6-9).



(a) (b)

Figure 7.3: Edge Distances and Synchronous Edges

The value of ce_{thd} is provided by the applications/end user and hence can be reconfigured to make the system report highly probable or less probable complex events according to the application requirements.

7.3.2 Synchronization Error Detection

In this section we present the concepts for probabilistic synchronization detection between product parts moving on the different production lines. In the beginning we would present the **probabilistic synchronization model**, that is used to detect the product part synchronizations. After that we provide an **overview** of our approach. Then we describe the actual **probabilistic synchronization error detection algorithm** and explain how synchronization error events are generated in our system.

Probabilistic Synchronization Model:

In order for the products to be assembled correctly, the different product parts should arrive at the assembly points within certain defined time spans i.e. the parts of a specific product should reach the assembly point in sync with each other. A delay or break up of this clockwork precision may lead to incorrect assembly of the product.

Before diving into the details of the probabilistic synchronization model, it is worthwhile to revisit some of the concepts discussed in the system model chapter (cf: Chapter 3) that are of relevance with the synchronization discussion. We know from the system model (cf: Chapter 3.2) that the production environment is modeled as a polytree with assembly points and physical readers denoted as vertices of the tree and production lines modeled as edges of the tree. Furthermore, both the edges and vertices in the polytree have weights which denote the amount of time it takes for that specific edge to perform its task. In case of the vertice, the weight signifies the amount of time the assembly point takes to assemble the product parts arriving at that assembly point.

Where as in case of the edge, the edge weight shows the amount of time it takes for product parts to move across the production line denoted by that particular edge.

From the edge and vertex weights we can compute the distance between any two edges or the distance between any physical reader and an assembly point. In order to further explain this lets take a concrete example with two sets of sub-trees shown in figure 7.3(a) and figure 7.3(b).

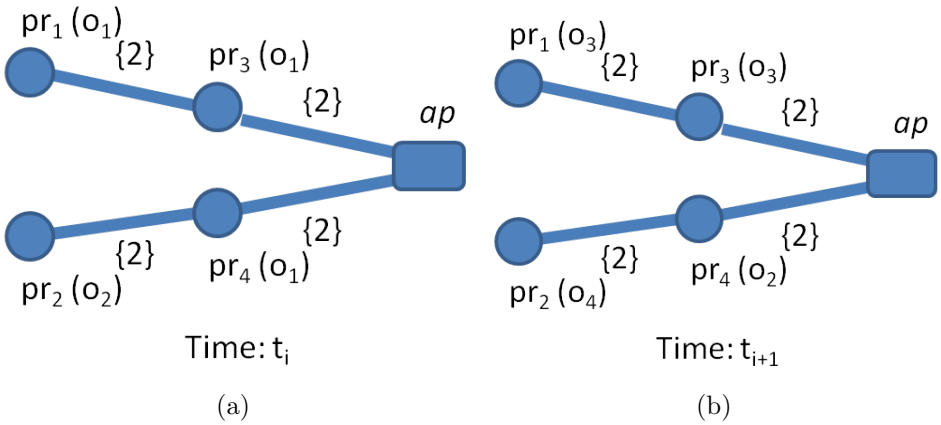
In figure 7.3(a) we have 5 vertices (a, b, c, d, e) and four edges (ab, be, cd, de). From the system model (cf: Chapter 3.2), we know that both physical readers *pr*s and assembly points are modeled as vertices. The difference between these two is that vertices representing physical readers *pr*s have null weights, where as vertices representing assembly point have integral weights. So, we can easily deduce that vertices (a, b, c, d) represent physical readers *pr*s, where as vertex e represents an assembly point. From the edge and vertex weights, we can easily compute the distance between any two edges. As an example, the distances between the different vertices of the sub-tree in figure 7.3(a) are: $d_G(a, e) = 4$, $d_G(b, e) = 2$, $d_G(c, e) = 4$, $d_G(d, e) = 2$. From this we can easily see that (a and c) and (b and d) have the same distance to vertex e i.e. ($d_G(a, e) = d_G(c, e) = 4$ and $d_G(b, e) = d_G(d, e) = 2$).

Similarly, in figure 7.3(b) we have 5 vertices (v, w, x, y, z) and four edges (vw, wz, xy, yz). In this scenario only two vertices (i.e. v and x) have the same distance from vertex z (i.e. $d_G(v, z) = d_G(x, z) = 5$)

A *synchronization error event* ce_{syn} (o_i *syn* o_j) implies that product part o_i belonging to product p_i is on production line pp_i at position a and product part o_j belonging to product p_j is on production line pp_j at position x, such that the distance from position a to the next assembly point ap is equal to the distance from position x to assembly point ap (i.e. $d_G(a, ap) = d_G(x, ap)$). This is an error because product parts o_i and o_j belong to different products, but would reach the assembly point ap at the same time and hence would be assembled with each other.

In order to detect synchronization errors, we map physical readers *pr*s on one production line with physical readers *pr*s on the other production line(s) such that both these mapped physical readers *pr*s have the same distance from the next assembly point. This mapping is defined as ($pr_{i_{pp_m}}$ *map* $pr_{j_{pp_n}}$), where pr_i on production line pp_m is mapped with pr_j on production path pp_n. In other words, if both pr_i and pr_j would detect product part o_i and o_j at time t_i. Then both of these product parts o_i and o_j would reach the next assembly point together and hence would be assembled with one another.

Due to the inherent unreliability of RFID readers we can also have *non-unique synchronizations* (o_i *syn* o_j) and (o_i *syn* o_k). This implies that o_i is in sync with both o_j and o_k. Since there can only be one object at a certain position at one instance of time, non-unique synchronizations cannot exist in reality.



(a) (b)

Figure 7.4: Non-Unique Syncrhonizations

In order to understand how we can have *non-unique synchronizations*, lets consider the following example. Figure 7.4 shows a scenario in which physical reader pr_1 and physical reader pr_2 have the same temporal distance from the assembly point ap and hence are mapped together. Similarly physical reader pr_3 and physical reader pr_4 are also mapped together. In Figure (a) physical reader pr_1 detects product part o_1 and physical reader pr_2 detects product part o_2 at the same time. Since both of these physical readers are mapped together, we would have a synchronization (o_1 *syn* o_2). These product parts then move forward on the production line and ultimately reach physical reader pr_3 and physical reader pr_4 respectively. However, at this stage physical reader pr_3 overshoots and detects product part o_3, while physical reader pr_4 detects product part o_2. Since physical reader pr_3 and physical reader pr_4 are also mapped together, we would have a synchronization (o_3 *syn* o_2).

From the example above, we have two product part synchronizations (o_1 *syn* o_2) and (o_3 *syn* o_2). This is not possible in reality, since such a situation would mean that product part o_1 and product part o_3 are at the same location at the same time. Such synchronizations are called non-unique synchronizations and arise as a result of inherent unreliability of RFID readers.

The probability of a synchronization error event $p(ce_{syn})$ denotes the probability with which a deduced synchronization error matches reality. The goal of the synchronization error detection algorithm is to determine if the product parts moving on different production lines are in sync with each other or not and if they are in sync. whether they belong to the same product or not.

Overview:

In this section we will provide a brief overview of our approach. The basic steps of the probabilistic synchronization error detection algorithm are as follows:

1. Detect all possible production part synchronizations. The outcome of this step

are multiple possible synchronizations *syn*s, which are stored in a *SynList*, which contains all possible synchronizations detected within the system. If we have n physical reader mappings ($pr_{i_{pp_m}}$ *map* $pr_{j_{pp_n}}$), then on every cycle (for a description of cycle see Chapter 5.3.1) we would have n synchronizations being detected.

2. Assign probabilities to the detection synchronizations. We have already explained that non-unique synchronizations can exist in reality. Probability assignment to synchronizations work in a way such that the probabilities of synchronizations that exist in reality increase over time, where as that of false synchronizations decrease over time.
3. Whenever, the probability of a synchronization *syn* is changed, we compare it with the complex event threshold ce_{thd}. If the probability of the synchronization *syn* is less than ce_{thd}, we do nothing.
4. If however, the probability of the synchronization *syn* is greater than ce_{thd}, we try to determine if a synchronization error ce_{syn} has occured or not.

Synchronization Error Detection Algorithm:

Before explaining the synchronization error detection algorithm it is pertinent to enlist the data structures that are used in the algorithm.

- event $e_i = (o_i, pr_i, t_i)$: Read event e_i by pr_i.
- $events_{pr_i} = (e_1, e_2, ... e_n)$: Set of read events of pr_i.
- $events_{vr_i} = (e_1, e_2, ... e_m)$: Set of read events at vr_i.
- *productpart* $o_i = (o_i, ttl)$: Product part o_i is associated with its id and a time to live *ttl* variable.
- Product part list $PPL = (o_1, o_2, ... o_n)$: Product part list contains all the product parts detected at the vr_i.
- Synchronization $syn_i = ((o_i\ syn\ o_j), events, p(syn_i), ttl)$: Each synchronization syn_i contains a product part synchronization of the form $(o_i\ syn\ o_j)$, the events that led to the detection of the product part synchronization, the probability with which the product part synchronization was detected and a time to live *ttl* variable that shows how recently the synchronization was detected.
- Synchronization list $SynList = (syn_1, syn_2, ... syn_n)$: Synchronization list *SynList* contains all the product part synchronizations *syn* detected at the vr_i.

A ce_{syn} occurs if $(o_i\ syn\ o_j)$, where product part o_i belonging to product p_i is in synchronization with product part o_j belonging to product p_j. Synchronization *syn* between two product parts is detected using the following rule:

If pr_i detects a product part o_i at time t_i on production path pp_m and pr_j detects a product part o_j at time t_i on production path pp_n, then $(o_i\ syn\ o_j)$ if and only if there exists a mapping between pr_i and pr_j i.e. ($pr_{i_{pp_m}}$ *map* $pr_{j_{pp_n}}$).

Algorithm 6 Synchronization Error Detection Algorithm

Let $e_i = (o_i, pr_i, t_i)$ be a read event detected at vr_i
PPL;
$SynList$;
ce_{thd};
while pr_i detects o_i **do**
  if o_i != new productpart **then**
    $PPL.o_i.ttl == 1$;
  else {o_i == new productpart}
    $o_i.ttl == 1$;
    $PPL \leftarrow o_i$;
    Detect synchronization syn using synchronization detection rules
    if synchronization syn_i is deduced **then**
      if synchronization syn_i == new synchronization **then**
        $syn_i.ttl == 1$;
        $syn_i.p(syn_i) \leftarrow$ **ComputeProbability(**syn_i**)**;
        $SynList \leftarrow syn_i$;
      else {synchronization syn_i != new synchronization}
        there exists a $SynList.syn_h$ such that $SynList.syn_h == syn_i$;
        $SynList.syn_h.ttl == 1$;
        $SynList.syn_h.o_1.addevents(syn_i.o_1.readevent)$;
        $SynList.syn_h.o_2.addevents(syn_i.o_2.readevent)$;
        $SynList.syn_h.p(syn) \leftarrow$ **ComputeProbability(**syn_h**)**;
      end if
    end if
  end if
  if $(p(syn_i) > ce_{thd})$ **then**
    if ($syn_i.o_1$ AND $syn_i.o_2$) does not belong to same product **then**
      trigger synchronization error ce_{syn};
    end if
  end if
end while
for function $p(syn)$ ComputeProbability(syn_i) **do**
  $p(syn_i) = (1 - [(1-p)^{syn_i.ne_{o_1}} + (1-p)^{syn_i.ne_{o_2}} - (1-p)^{syn_i.ne_{o_1} + syn_i.ne_{o_2}}])$;
  return $p(syn_i)$;
end for

Whenever a new product part o_i is detected, we place it in the product part list PPL and try to deduce the synchronization syn of the product part with the product parts detected on other production paths using the rule mentioned above. Once a

synchronization syn is deduced, it is added to the synchronization list $SynList$ (cf: Algorithm 6 lines 5-25).

The ttl of both the product part o_i and the deduced synchronization syn is initially set to 1 and is decreased by a certain amount on every time unit, which is 1 cycle in our simulations. The ttl of the product part o_i or the syn is refreshed to 1, if the product part o_i or the deduced synchronization syn is detected again (cf: Algorithm 6 lines 12-19). If the ttl of the product part o_i becomes 0, it is removed from the PPL. The removal of the product part o_i from the PPL indicates that product part o_i is no longer on any of the production lines. Once a product part o_i is removed from the PPL, all the synchronizations in which product part o_i was participating are removed from the $SynList$. In addition to this, a synchronization syn is also removed from the $SynList$, if the ttl of the synchronization syn becomes 0.

If the synchronization (o_i syn o_j) occurs in reality, its probability will increase over time. Once the probability of the synchronization $p(syn)$ exceeds the complex event threshold ce_{thd}, we check if both the product parts o_i and o_j belong to the same product. If both the product parts do not belong to the same product, a synchronization error event ce_{syn} is triggered (cf: Algorithm 6 lines 26-30).

The probability of a synchronization (o_1 syn o_2) is computed as:

$$p(ce_{syn}) = (1 - [(1-p)^{ne_{o_1}} + (1-p)^{ne_{o_2}} - (1-p)^{ne_{o_1}+ne_{o_2}}]$$

where ne_{o_1} is the number of read events for o_1 and ne_{o_2} is the number of read events for o_2. As long as at least one read event for o_1 and one read event for o_2 is correct, we can make a statement about the synchronization (o_1 syn o_2). Only if, either all read events for o_1 are incorrect $(1-p)^{ne_{o_1}}$ or all read events for o_2 are incorrect $(1-p)^{ne_{o_2}}$, we cannot derive a synchronization. In order to not count the incorrect read events twice, we subtract $(1-p)^{ne_{o_1}+ne_{o_2}}$. This gives us the probability that of no correct reading for ($o_1 < o_2$). Subtracting this probability from 1 gives us the probability of all the cases where at least one correct reading for o_1 and at least one correct reading for o_2 is included in the set of readings.

The formula for computing the probability of a synchronization is similar to probability computation for sequence errors (cf. Chapter 5.2.3). This is because both of these complex errors are generated as a result of event detections by two physical readers, each of which is independent but related to the other. Both the physical readers in question are independent because the performance and reliability of any one of the physical reader is not affected by the other reader or the detection of a product part by one reader is not affected by the detection or non-detection of that product part by the other physical reader. The physical readers in case of sequence errors are related in a sense that both are deployed on the same production line, where as the physical readers in case of synchronization errors have a special mapping relationship between one another.

7.3.3 Delay Error Detection

In this section we would present the concepts for probabilistic detection of delay on the production lines. First we present the **probabilistic delay model**, that is used to detect the delay on the production lines. Then we provide an **overview** of our approach. Finally, we describe the actual **probabilistic delay error detection algorithm** and explain how delay error events are triggered within our system.

Probabilistic Delay Model:

Time and precision is of critical importance in production. Any delay on a production path would cause product parts moving on this path to either arrive late at the assembly point and hence would either result in incorrect assembly of products or in an overall delay for the entire production. Delays are typically caused due to mechanical or human failures.

A delay error ce_{del} is different from missed reading, which is an RFID reader failure (cf: missed reads Chapter 3.4.4). In a missed read, an RFID reader fails to read a product part that is on the production line. However, in case of a delay, an RFID reader does not read a product part that is not on the production line. Both of these situations are fundamentally different i.e. in the case of missed read the RFID reader is failing to detect a product part that exists in reality, where as in the case of delay an RFID reader is detecting the non-existence of a product part on the production line. But from the system perspective, we have no way of distinguishing one from the other.

The solution to this problem relies on a key difference between delay error and missed reading. We hold an assumption, that in case of a missed reading, there will eventually be a physical reader that would be able to read the product part which was missed out by a prior physical reader, and hence we would find out that the product part in question still is still moving on the production line and that the prior reader actually missed out on detecting it. For an elaboration on how missed readings are detected, see how we detect missed readings during self calibration of RFID reader probabilities (cf: Chapter 6.3).

In case of delay error, since it would be a break down of the production line, none of the physical readers deployed on the production line would be able to detect the product part(s). Since, we assign probabilities to each and every complex event, the probability of delay error $p(ce_{del})$ would eventually increase above the complex event threshold ce_{thd} and hence we would have reasonable confidence in the fact that we have a delay on our production line.

A delay read event is a read event $e_{delay} = (d_i, pr_i, t_i)$; where t_i represents the time at which product part d_i was detected by physical reader pr_i. The product part d_i is not a real product part. We trigger the delay read event e_{delay} whenever a physical reader misses out on detecting a real product part. The detection of product part d_i signifies that the physical reader pr_i has not detected any real product part.

A *delay sequence ds* is a sequence of the form $(o_i < d)$ which implies that product part o_i is directly ahead of a special product part d on the production line or $(d < o_i)$ which implies that a special product part d is directly ahead of product part o_i on the production line or $(d < d)$ which implies that a virtual product part d is ahead of another virtual product part d.

Overview:

In this section we provide a brief overview of our approach. The core steps of the probabilistic delay detection algorithm are as follows:

1. Detect all possible delay sequences ds. The result of this step are multiple possible ds, which are stored in a delay list $DelList$, which contains all the possible delay sequences ds detected at a particular virtual reader. Not all of these delay sequences ds would be actual delays on the production line, as some or most of these delay sequences might be a result of missed reads.
2. Assign probabilities to the detected delay sequences. The probability assignment in our system works in a way such that the probability of the actual delay sequences ds would increase over time, where as the probability of the delay sequences created as a result of missed reads would decrease over time.
3. Whenever the probability of a delay sequence is modified as a result of new readings, we compare the probability of the delay sequence with the complex event threshold ce_{thd}. If the probability of the delay sequence is less then ce_{thd}, we do nothing.
4. If however, the probability of the delay sequence ds becomes greater than the complex event threshold ce_{thd}, the system no longer treats it as a possible delay and becomes sure that it really is a delay and triggers the delay error event ce_{del}.

Delay Error Detection Algorithm:

Before explaining the synchronization error detection algorithm it is pertinent to enlist the data structures that are used in the algorithm.

- event $e_i = (o_i, pr_i, t_i)$: Read event e_i by pr_i.
- $events_{pr_i} = (e_1, e_2, \ldots e_n)$: Set of read events of pr_i.
- $events_{vr_i} = (e_1, e_2, \ldots e_m)$: Set of read events at vr_i.
- *productpart* $o_i = (o_i, ttl)$: Product part o_i is associated with its id and a time to live ttl variable.
- Product part list $PPL = (o_1, o_2, \ldots o_n)$: Product part list contains all the product parts detected at the vr_i.
- Delay sequence $ds_i = ((o_i < d)$ or $(d < o_j)$, $events$, $p(ds_i)$, $ttl)$: Each delay sequence ds_i contains a delay sequence of the form $(o_i < d)$, the events that led

to the detection of the delay sequence ds_i, the probability with which the delay sequence was detected and a time to live ttl variable that shows how recently the delay sequence was detected.

- Delay Sequence list $DelList = (ds_1,\ ds_2,\ ...\ ds_n)$: Delay Sequence list $delList$ contains all the delay sequences ds detected at the vr_i.

Detection of a delay sequence ds is similar to detecting a partial sequence ps. A ds is detected using the following two rules:

1. If a physical reader pr_i detects a product part o_i at time t_i and then fails to detect any product part at time t_{i+1}, we trigger a delay event $e_{delay} = (d, pr_i, t_{i+1})$. This implies that pr_i has detected a non-object/non-product part d at time t_{i+1}. This in turn implies that o_i is ahead of d on the production path, i.e. $(o_i < d)$.
2. If pr_i is deployed directly before pr_j on the production path i.e. $(pr_i \sqsubset pr_j)$, then a product part o_i detected by pr_j at time t_i is ahead of the non-product part d detected by pr_i at the same time t_i, i.e. $(o_i < d)$.

Whenever a product part o_i or a virtual product part d is detected, it is added to the product part list PPL. Whenever a virtual product part d is detected, its ttl is set to 1 (cf: Algorithm 7 lines 9-10). This ttl is then decreased over time. If the virtual product part d is not detected before its ttl becomes 0, it is removed from the PPL. If the virtual product part is detected before its ttl becomes 0, its ttl is refreshed to '1' once again.

As soon as we add the virtual product part d to the product part list PPL, we try to deduce a delay sequence ds using the rules mentioned above (cf: Algorithm 7 lines 13-14). If a delay sequence ds is deduced, it is added to the delay list $DelList$, which contains all the delay sequences detected at the virtual reader vr. Every delay sequence in the $DelList$ is associated with three variables - a probability of the delay sequences, an event list containing all the read events e and delay events e_{delay} that led to the detection of the delay sequence ds, and a ttl of the delay sequence ds, which shows how recently the delay sequence ds was deduced (cf: Algorithm 7 lines 8-20).

If the newly deduced ds is already present in the $DelList$, its ttl is refreshed to 1 (cf: Algorithm 7 lines 21-23). Similar to the ttl of the virtual product part d, the ttl of the delay sequence ds is decreased over time. If the delay sequence ds is not deduced again before its ttl becomes 0, it is removed from the $DelList$. If however, the delay sequence ds is deduced again before its ttl becomes 0, its ttl is refreshed to '1' once again. Furthermore, if the ttl of a virtual product part d becomes 0, all the delay sequences ds in which it is participating are removed from the $DelList$.

The probability of a delay sequence ds is re-computed whenever we have new read events that lead to the deduction of the delay sequence ds (cf: Algorithm 7 lines 18 and 26). With any delay sequence ds there are only two possibilities. Either more read

Algorithm 7 Delay Error Detection Algorithm

Let $e_{delay} = (d_i, pr_i, t_i)$ be a delay read event detected at vr_i
PPL;
$DelList$;
ce_{thd};
while pr_i detects d_i **do**
  ds = DelaySequenceDetection(d_i);
end while
for function ds DelaySequenceDetection(d_i) **do**
  if d_i != new virtualproductpart **then**
    $PPL.d_i.ttl == 1$;
  else {d_i == new virtualproductpart}
    $d_i.ttl == 1$;
    $PPL \leftarrow d_i$;
    Detect delay sequence ds using delay sequence detection rules
    if delay sequence ds_i is deduced **then**
      if ds_i == new delaysequence **then**
        $ds_i.ttl == 1$;
        $ds_i.p(ds) \leftarrow$ **ComputeProbability(**ds_i**)**;
        $DelList \leftarrow ds_i$;
        return ds_i;
      else {delay sequence ds_i != new delaysequence}
        there exists a $DelList.ds_h$ such that $DelList.ds_h == ds_i$;
        $DelList.ds_h.ttl == 1$;
        $DelList.ds_h.o_1.addevents(ds_i.o_1.readevent)$;
        $DelList.ds_h.o_2.addevents(ds_i.o_2.readevent)$;
        $DelList.ds_h.p(ds) \leftarrow$ **ComputeProbability(**ds_h**)**;
        return $DelList.ds_h$;
      end if
    end if
  end if
  if $(p(ds) > ce_{thd})$ **then**
    trigger delay error ce_{del};
  end if
end for
for function $p(ds)$ ComputeProbability(ds) **do**
  $p(ds) = (1 - [(1-p)^{ds.ne_{o_1}} + (1-p)^{ds.ne_d} - (1-p)^{ds.ne_{o_1} + ds.ne_d}])$;
  return $p(ds)$;
end for

events would lead to the deduction of this delay sequence ds and hence its probability would increase or no read event would lead to the deduction of this delay sequence ds in which case its ttl would become 0 and it will be removed from the $DelList$.

If the delay sequence ds is created as a result of a missed read by a physical reader. The other physical readers prs deployed on the production line would/should successfully read the product part o_i and hence the delay sequence ds would eventually time out.

Once the probability of a delay sequence ds exceeds the ce_{thd}, we assume that this delay sequence is not a result of physical readers prs missing out on detecting a part and hence trigger the delay complex event ce_{del}.

There can be multiple delays on a particular production path such as $(o_i < d_i)$ and $(o_j < d_j)$. Our method of detecting delays allow us to detect and report these different delays separately. Additionally, each delay error event ce_{del} is of the form $(o_i < d_i)$. From this event, we can easily deduce the product part after which the delay has occurred. Furthermore, since each read event contains the time at which the event was detected and the physical reader that detected the event, we can easily identify the location of origin of each particular delay within the smart variant production environment.

The probability of a delay sequence $(o_1 < d_1)$ is computed as:

$$p(ds) = (1 - [(1-p)^{ne_{o_1}} + (1-p)^{ne_{d_1}} - (1-p)^{ne_{o_1}+ne_{d_1}}]$$

where ne_{o_1} is the number of read events for o_1 and ne_{d_1} is the number of read events for d_1. As long as at least one read event for o_1 and one read event for d_1 is correct, we can make a statement about the delay sequence $(o_1 < d_1)$. Only if, either all read events for o_1 are incorrect $(1-p)^{ne_{o_1}}$ or all read events for d_1 are incorrect $(1-p)^{ne_{d_1}}$, we cannot derive a sequence. In order to not count the incorrect read events twice, we subtract $(1-p)^{ne_{o_1}+ne_{d_1}}$. This gives us the probability that of no correct reading for $(o_1 < d_1)$. Subtracting this probability from 1 gives us the probability of all the cases where at least one correct reading for o_1 and at least one correct reading for d_1 is included in the set of readings.

The probability of a delay sequence is computed in exactly the same way as we compute the probability of a partial sequence (cf: Chapter 5.2.3), since essentially both are a sequence of product parts moving on the production line.

7.3.4 Missing Part Error Detection

Missed assembly is a frequent issue in production environments. Due to mental or physical stress a worker may miss out on assembling a certain product part on to a product. Whenever, two product parts o_1 and o_2 are assembled together at an assembly point, the physical readers deployed ahead of this assembly point would detect both the product parts at the same time. Whenever a physical reader detects two product parts

at the same time, we perform data aggregation and create a new product part. In case, the physical reader detects product parts o_1 and o_2, we would create an aggregated product part $(o_1 : o_2)$ and store this newly aggregated/assembled product part in the assembled parts list $AsmList$ which contains a list of all the product parts that have been assembled together, alongwith the events that led to this aggregation/assembly and a probability assigned to each of these aggregated parts. The probability of the aggregated parts depicts the confidence with which we know that both the aggregated parts exist in reality.

In case a worker fails to assemble a product part with another product part, the physical readers deployed ahead of the assembly point would be able to detect only one product part, which would ultimately result in a missing part error event. In order to explain this further, lets consider a concrete example. Figure shows a situation in which product part o_1 and o_2 reaches the assembly point at the same time. However, the worker at the assembly point picks up product part o_2 and forgets to assemble it with o_1 and puts it on some shelf. This results in product part o_1 passing through the assembly appoint without being assembled with product part o_2. After the assembly point, each physical reader should detect two product parts. However, in this case the physical reader deployed ahead of the assembly point would only detect product part o_1. In this situation, we would assume that the physical reader has detected product part o_1 and has failed to detect the second product part, which is a similar situation to detecting a virtual product part d. So we would create an aggregated product part $(o_1 : d)$ which would signify that the physical reader detected a product part o_1 and failed to detect a product part (hence the detection of virtual product part d). Since product part o_1 is not assembled together with anyother product part, the situation would be repeated at every subsequent physical reader and the read events of each physical reader would be added to the event list of of the aggregated product part $(o_1 : d)$ in the $AsmList$. With each new detection of the aggregated product part, we continue to update the probability of the aggregated product part.

Once the probability of the aggregated product part becomes greater then the complex event threshold ce_{thd}, we compare the aggregated product part, with the planned product part order PPO_{pln} to see if both the assembled product parts belong to the same product. If they do not belong to the same product, which would obviously be the case in the example explained above, we trigger a missing part error event ce_{mpp}

The probability of the aggregated product parts $(o_1 : d_1)$ is computed as:

$$p(parts_{agg}) = (1 - [(1-p)^{ne_{o_1}} + (1-p)^{ne_{d_1}} - (1-p)^{ne_{o_1}+ne_{d_1}}]$$

where ne_{o_1} is the number of read events for o_1 and ne_{d_1} is the number of read events for d_1. As long as at least one read event for o_1 and one read event for d_1 is correct, we can make a statement about the aggregated product parts $(o_1 : d_1)$. Only if, either all read events for o_1 are incorrect $(1-p)^{ne_{o_1}}$ or all read events for d_1 are incorrect $(1-p)^{ne_{d_1}}$, we cannot derive a conclusion regarding the aggregated product parts. In

Algorithm 8 Missing Part Error Detection Algorithm

1: Let $e_i = (o_i, pr_i, t_i)$ and $e_j = (d_j, pr_i, t_i)$ be read events detected simultaneously by pr_i
2: o_1 = null;
3: o_2 = null;
4: PPL;
5: $AsmList$;
6: ce_{thd};
7: **while** pr_i detects multiple parts **do**
8: $o_1 \leftarrow e_i.o_i$;
9: $o_2 \leftarrow e_j.d_j$;
10: $parts_{agg}.o_1 \leftarrow o_1$;
11: $parts_{agg}.o_2 \leftarrow o_2$;
12: **if** $parts_{agg}$ == new aggregared part **then**
13: $parts_{agg}.ttl \leftarrow 1$;
14: $parts_{agg}.p(parts_{agg} \leftarrow$ ComputeProbability(o_1, o_2);
15: $AsmList \leftarrow parts_{agg}$;
16: **else** {aggregated parts (o_1:o_2) != new aggregated parts}
17: $\exists\ AsmList.parts_{agg_i}$ such that $parts_{agg_i}.o_1 == o_1$ AND $parts_{agg_i}.o_2 == o_2$;
18: $AsmList.parts_{agg_i}.ttl == 1$;
19: $AsmList.parts_{agg_i}.o_1.addevents(o_1.readevent)$;
20: $AsmList.parts_{agg_i}.o_2.addevents(o_2.readevent)$;
21: $AsmList.parts_{agg_i}.p(parts_{agg} \leftarrow$ **ComputeProbability(**$parts_{agg_i}.o_1$**,** $parts_{agg_i}.o_2$);
22: **end if**
23: **if** ($p(parts_{agg}) > ce_{thd}$) **then**
24: **if** $parts_{agg_i}.o_1$ AND $parts_{agg_i}.o_2$ does not belong to same product **then**
25: **trigger** missing part error ce_{mpp};
26: **end if**
27: **end if**
28: **end while**
29: **for function** $p(parts_{agg})$ ComputeProbability(o_1, o_2) **do**
30: $p(parts_{agg}) = (1 - [(1-p)^{ne_{o_1}} + (1-p)^{ne_{o_2}} - (1-p)^{ne_{o_1}+ne_{o_2}}])$;
31: **return** $p(parts_{agg})$;
32: **end for**

order to not count the incorrect read events twice, we subtract $(1-p)^{ne_{o_1}+ne_{d_1}}$. This gives us the probability of no correct reading for $(o_1 : d_1)$. Subtracting this probability from 1 gives us the probability of all the cases where at least one correct reading for o_1 and at least one correct reading for d_1 is included in the set of readings.

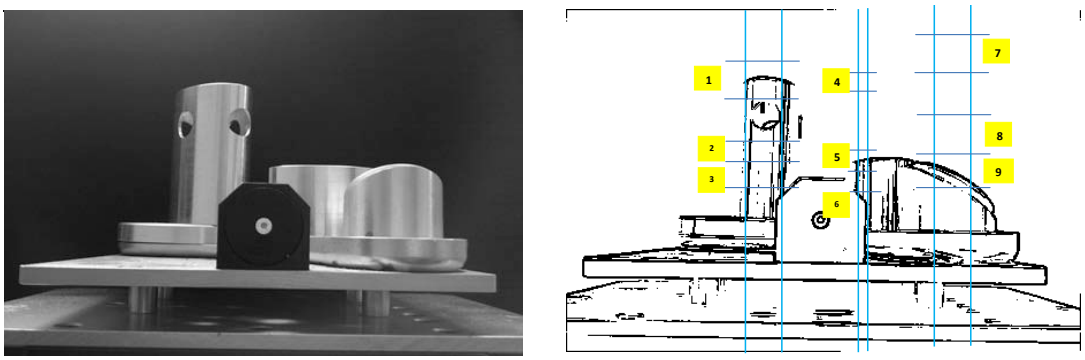


Figure 7.5: Grey Scale Image (Left), Edge Detection Applied (Right)

Missing Part Error Variants:

The aggregated product parts can be of two forms:

1. Missing part $(o_1 : d_1)$: This scenario has already been explained above. In this situation a product part has not been assembled with product part o_1 and d_1 signifies that a product part is missing from this assembly.
2. Incorrect part $(o_{1_n} : o_{2_m})$: In this scenario, the aggregated part shows that product part o_{1_n} which belongs to product n has been assembled together with product part o_{2_m} which belongs to product m. This is an error, since the a product part should only be assembled together with product parts that belong to the same product. In other words product part o_{1_n} should have been assembled with product part o_{2_n}, both of which belongs to product n. The aggregated product part $(o_{1_n} : o_{2_m})$, signifies that the correct product part o_{2_n} is missing from the assembly. This special case of missing part error can occur as a result of sequence errors that are not resolved before the product parts reach the assembly points.

7.3.5 Incorrect Part Position Error Detection

In production, the right product part should be assembled at the right position. As an example, in the Lernfabrik [Ben09], all the three product parts can be placed at any of the three positions of the base plate. However, we also have a robotic arm in the factory that only places product parts at a specific pre-programmed position on the base plate. If a worker places a product part at the position on which the robotic arm is supposed to place its product part, the robotic arm will never be able to assemble its product part and hence the assembly of this specific product will never be completed.

Detection of such an error is not possible using RFID technology, because with RFID technology we can only detect the presence or absence of a product part and not the position at which the part was placed or assembled. To detect such errors we are using

Algorithm 9 Part Position Detection

```
1: Grey Scale Transform ( )
2: Sobel Edge Detection Algorithm ( Threshold = 128 )
3: position = 1
4: while position < 4 do
5:   if position.count(square1) > Threshold then
6:     position ← large cup
7:   else
8:     if position.count(square2) > Threshold then
9:       position ← small cup
10:    end if
11:  else
12:    if position.count(square3) > Threshold then
13:      position ← watch
14:    end if
15:  else
16:    position ← empty
17:  end if
18:  position + +
19: end while
```

an image processing based technique. We have two cameras installed immediately after the assembly points in the Lernfabrik. These cameras capture the images of the base plates with product parts on it, as these plates move across the production lines.

In our product, since all the parts are of different size and shape, we are able to find out the location of the assembled part using basic image detection techniques.

Once an image is captured, we apply 256 degree grey scale transformation and then apply sobel edge detection algorithm with a threshold of 128. Figure 7.5 shows the grey scale image and an image with edge detection applied.

The large cup occupies all the three squares at a position, whereas the small cup occupies only two squares and the watch just one featuring square. If at any position all the three squares are occupied, then the large cup is located at that position, if two squares of the image are occupied then the small cup is located there, if just one square is occupied then the watch is located at that position. The algorithm for detecting part position is listed in Algorithm 9.

The incorrect position detection algorithm provides us with the type of object on each position in the plate. This result is compared with the planned positioning of parts. In case of any discrepancies, an incorrect part position error ce_{ipp} is triggered.

7.4 Evaluations

In this section, we discuss the performance of the probabilistic complex event processing system under simulated settings. We decided to evaluate our system in a simulated setting because a simulation environment provides the possibility to evaluate a large scenario. Furthermore, we also wanted to test our system in a controlled setting, whereby we could set the ground truths, such as the probabilities of prs. The simulations were performed using PeerSim [JMJV09], a large scale distributed P2P discrete event simulator. All simulations were performed with 8,000 physical readers prs distributed across 1000 nodes/virtual readers vrs.

A cycle in our simulations is the time taken by a product part to move from one pr to the next. So after every cycle a new product part is introduced on the production line, while the previous ones move ahead by one pr.

7.4.1 Accuracy and Precision

Before moving on to discuss the evaluations, it is pertinent to explain accuracy and precision since both of these terms would be used extensively in the evaluations. We have evaluated our system in terms of its accuracy and precision.

Accuracy of a measurement system is defined as the degree of closeness of measurements of a quantity with respect to the quantity's actual or true value. In our scenario, complex events triggered by our system would be accurate if they represent the fact i.e. a delay error event ce_{del} would be accurate if there would be a delay on the production lines.

Precision of a measurement system is defined as the degree with which the results could be repeated or reproduced. In our scenario, the complex events triggered would be precise if a large number of preceding and succeeding complex events would also report the same fact i.e. a delay error event would be precise if preceding and/or succeeding events also report a delay on the production line. In other words the complex event processing system would not be very precise, if one physical reader reports an error on the production line, where as the next physical reader does not detect the said error.

For the purpose of measurements, whenever an event was triggered in our system, we categorized it as either true positive, false positive, true negative or false negative. Accuracy and precision was then calculated as:

accuracy = (no of true positives + no of true negatives) / (no of true positives + no of false positives + no of false negatives + no of true negatives)

precision = no of true positives / (no of true positives + no of false positives)

7.4.2 Comparison of False, Missed, Out of Order and Uniformly Distributed Raw RFID Errors

In this scenario, we observed the effect of false, missed, out of order and uniformly distributed false, missed and out of order RFID errors on the accuracy and precision of complex events. For the purpose of these evaluations we set the actual probability of physical reader *pr*s to 0.7 and also fixed the CE Threshold to 0.7. The value of CE Threshold is used by our system to determine when to trigger a complex event. If the value of CE Threshold is set to 0.7, our CEP system would trigger a complex event once the probability of the complex event is greater than 0.7.

Accuracy: Figure 7.6(a) shows the results for accuracy measurements of false, missed, out of order and uniformly distributed raw RFID Errors. The results show that initially accuracy of the detected complex events shot up to 80% (0.8). However, this was because in the beginning there were only a small no of complex events in the system. When the number of complex events that were being detected increased the accuracy started to come down. However, the accuracy of the system never dropped below 50% even with a *pr* probability and CE Threshold of 0.7.

From the simulation results its obvious that the accuracy of the system is not affected by the type of raw RFID errors that caused the complex event.

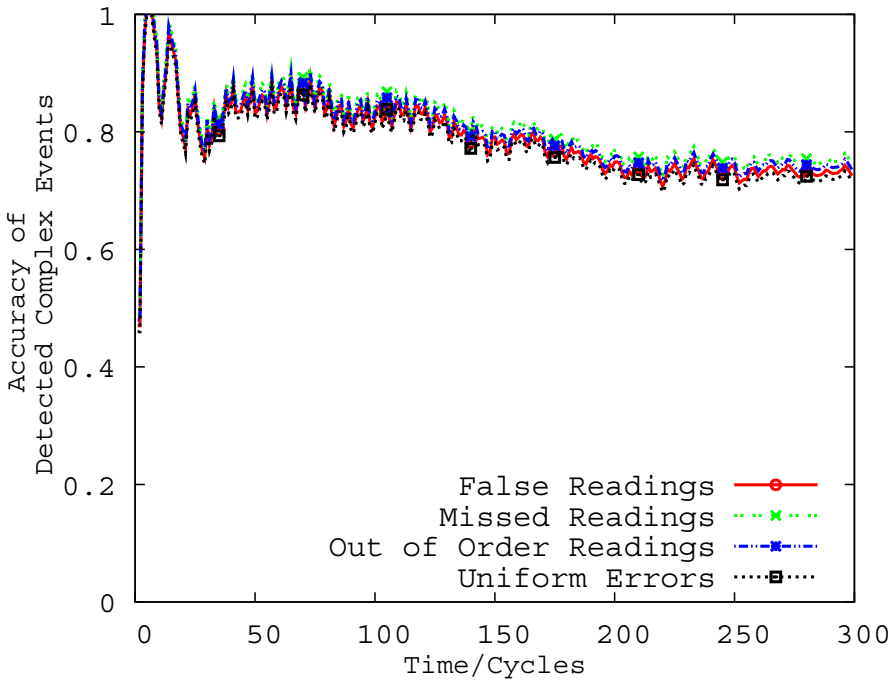
Precision: Figure 7.6(b) shows the results for precision measurements of false, missed, out of order and uniformly distributed raw RFID Errors. From the results, it is evident that the system was able to detect complex events with a very high precision. In the beginning the precision of the system reached as high as 100%. However, as more complex events were detected, the precision of detecting complex events started decreasing and eventually stabilised at around 70%.

The simulation results also reveal that the precision of the system is not affected by the type of raw RFID errors that triggered the complex events.

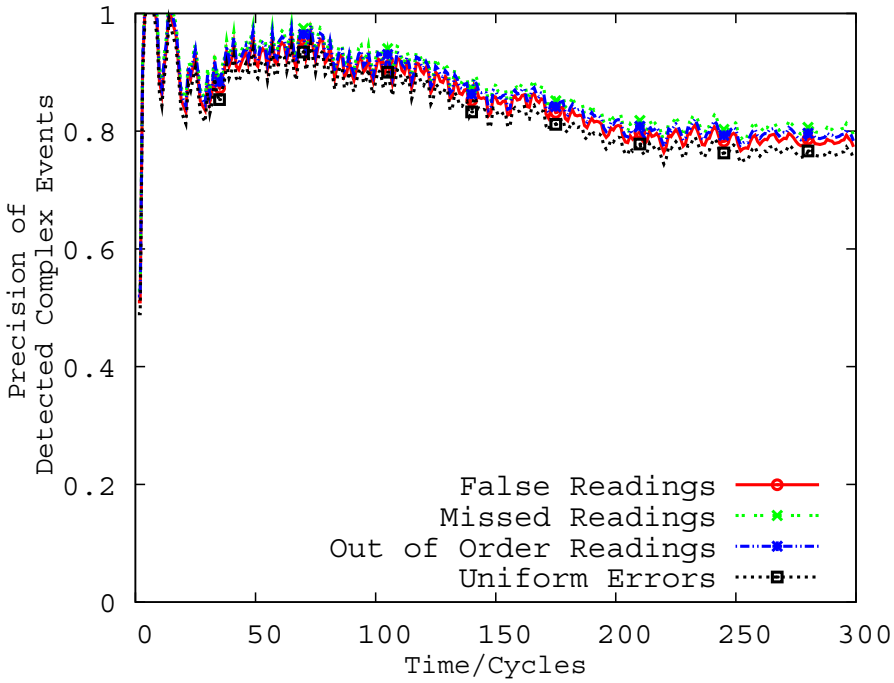
7.4.3 Comparison of Complex Errors

In this scenario, we compared the different complex events. The objective was to find out if the complex event processing framework detects different complex events with the same accuracy and precision or not. For the purpose of these evaluations the actual probability of *pr*s was set to 0.7 and the CE Threshold was set to 0.7.

Accuracy: Figure 7.7(a) shows the results for accuracy measurements of different types of complex events. The simulation results show that initially the accuracy of the detected complex events rose to around 80%, but then stabilized after around 50 cycles at 50%. The initial increase in accuracy could be attributed to a low number of complex events within the system. As the number of complex events increased the system started to move towards its equilibrium mark. The 50% accuracy in detecting

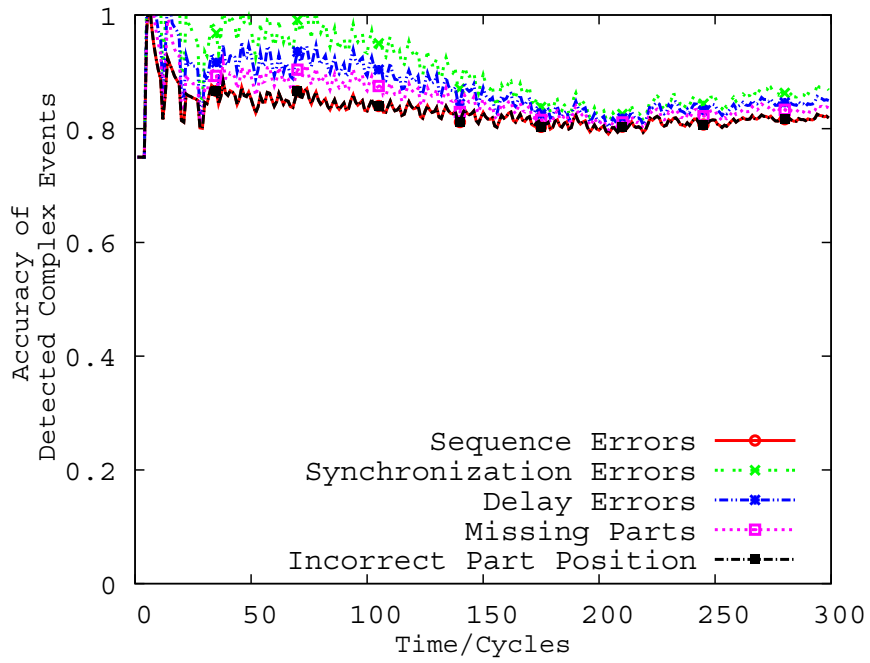


(a) Accuracy

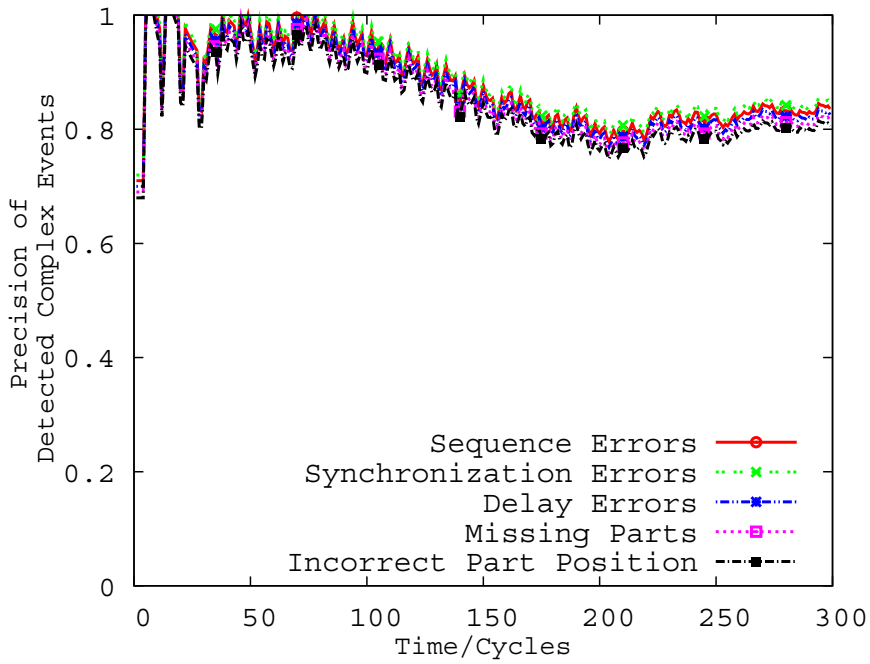


(b) Precision

Figure 7.6: Accuracy and Precision Comparison of False, Missed, OO & Uniform Errors



(a) Accuracy



(b) Precision

Figure 7.7: Accuracy and Precision Comparison of Sequence, Synchronization, Delay, Missing Part & Incorrect Part Position Error

different complex events is certainly a very good performance specially given the fact that the underlying physical readers were only 70% accurate and the CE Threshold was set 0.7.

From the simulation results its obvious that the accuracy of the system is not affected by the type of complex events.

Precision: Figure 7.7(b) shows the results for precision measurements of different types of complex events. The results show that precision of our system is higher than its accuracy. The probabilistic complex event processing system was able to detect different types of complex events with more than 70% precision, even when the underlying physical readers were only 70% accurate.

Another observation that needs to be mentioned here is that the system is not affected by the type of complex events.

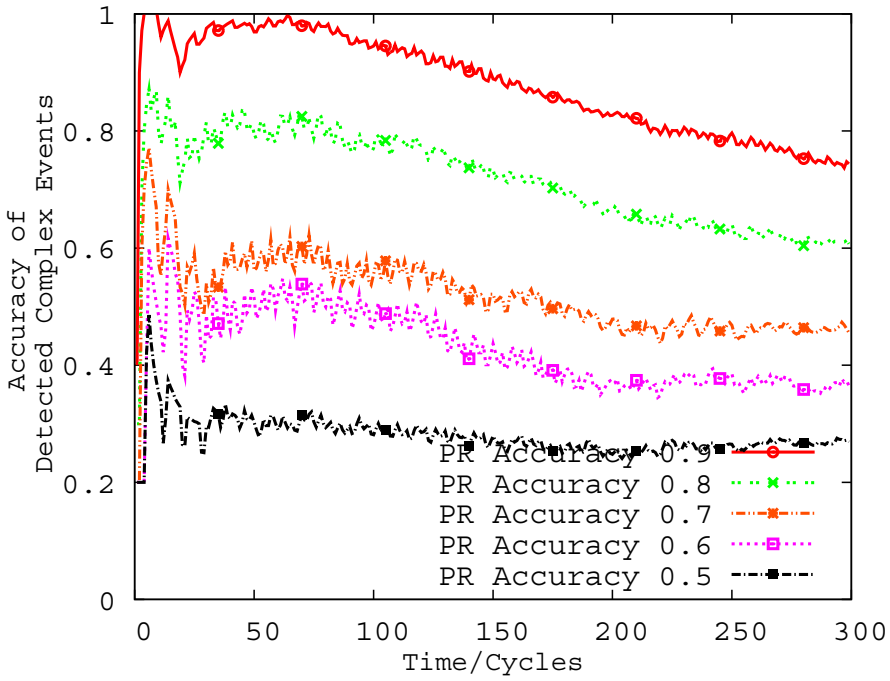
7.4.4 Comparison of Reliability of Physical Readers

In this scenario, we evaluated the performance of the CEP system by varying the accuracy of the physical readers. We took five set of readings by setting the actual probability of *pr*s to 0.5, 0.6, 0.7, 0.8, and 0.9. The objective was to observe how the complex event processing system performs in the presence of both reliable and unreliable raw RFID data. The CE Threshold for the system was set to 0.7.

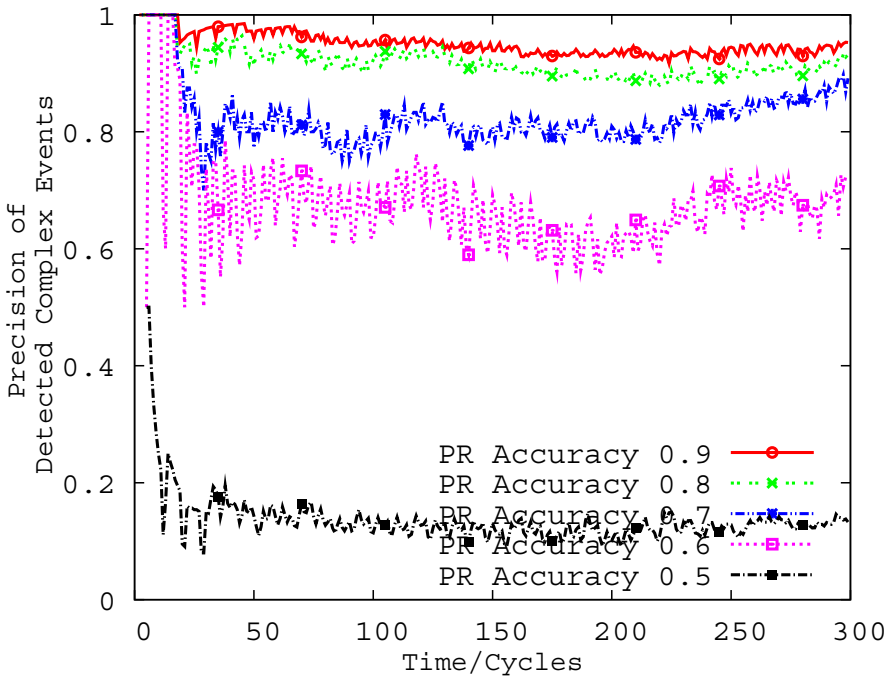
Accuracy: Figure 7.8(a) shows the results for accuracy measurements. The simulation results show that initially the accuracy of detected complex events rose higher but then stabilized a little lower. The accuracy of complex events when the accuracy of underlying physical readers was 90% rose to 80% but then stabilized around 60% level. Similarly, when the accuracy of underlying physical readers was 50%, the accuracy of complex events detected remained stable at 30% level. From the results, it is obvious that complex events are detected with higher accuracy when the underlying physical readers have a higher accuracy.

Precision: Figure 7.8(b) shows the results for precision measurements. The simulation results show that the precision of detected complex events is more than 90% when the accuracy of physical readers is 90%. The precision of detected complex events falls with a decrease in the accuracy of physical readers. The precision of complex events with physical reader accuracy of 60% remained around 70%.

One striking observation is that the precision of complex events decreased symmetrically with the decrease in physical reader accuracy as long as the physical reader accuracy remained above 60%. Once the physical reader accuracy is decreased to 50%, the precision of complex events falls off the grid and the detected complex events were only around 10% precise.



(a) Accuracy



(b) Precision

Figure 7.8: Accuracy and Precision Comparison of PR 50, 60, 70, 80, & 90

7.4.5 Comparison of Complex Event Thresholds

In this scenario, we evaluated the performance of the CEP system by varying the CE Threshold. The system was observed under five different CE Threshold levels i.e. CET 50, 60, 70, 80, and 90. The actual probability of *pr*s was set to 0.7 for these evaluations.

Accuracy: Figure 7.9(a) shows the results for accuracy measurements. The results show that the accuracy of detected complex events shot up to as high as 80% however with the passage of time and as more complex events were detected the overall accuracy of the detected complex events stabilized between 60 and 50 percentile. In general the accuracy of detected complex events increased with the increase in the CE Threshold, however, the increase in accuracy of detected complex events is not remarkable.

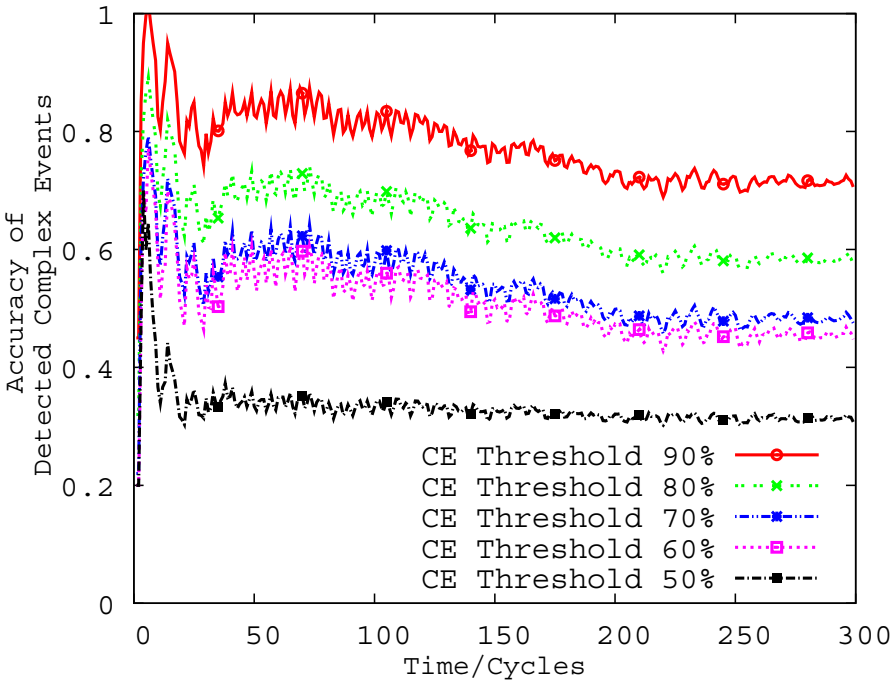
The increase in the accuracy of detected complex events remain symmetrical till the CE Threshold remain above 60%. However, when the CE Threshold is fixed at 50%, the accuracy of detected complex events fell to as low as 30%. This means that our system can not detect complex events with a reasonable reliability if the CE Threshold is lowered to 50%.

Precision: Figure 7.9(b) shows the results for precision measurements. The results show that the precision of detected complex events rose to as high as 100%. However once we have had a significant number of complex events within the system, the overall precision of the detected complex events stabilized between 90 and 70 percentile. In general the precision of detected complex events increased with the increase in the CE Threshold. But the increase in the precision of detected complex events is not remarkable as long as the CE Threshold remained above 60%.

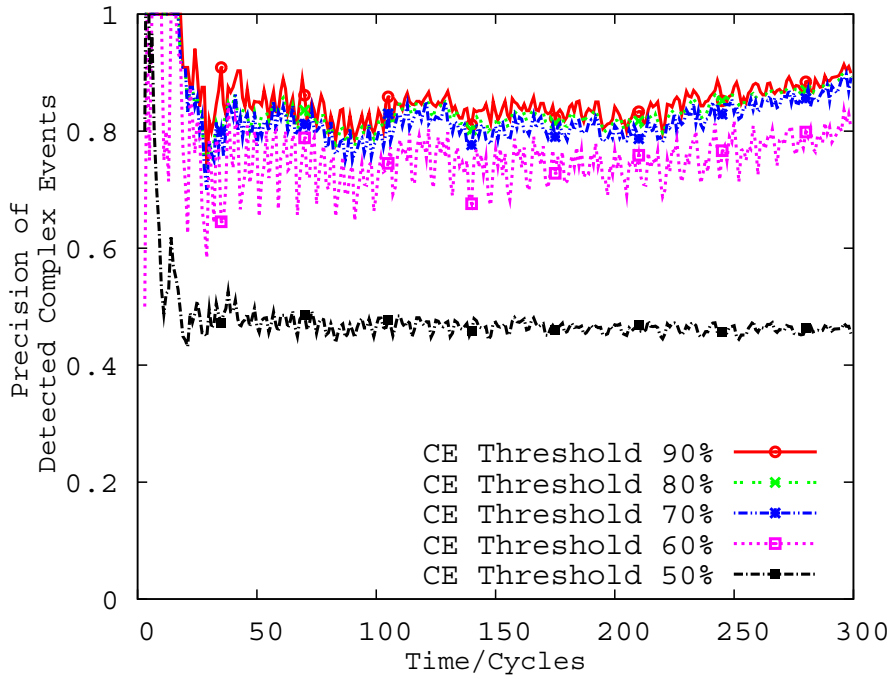
The increase in the precision of detected complex events remain symmetrical till the CE Threshold remain above 60%. However, when the CE Threshold is fixed at 50%, the precision of detected complex events fell to around 40%. This means that our system can not detect complex events with a reasonable precision if the CE Threshold is lowered to 50%.

7.4.6 Probabilistic Complex Event Processing System vs Non Probabilistic Complex Even Processing System

In this section we have compared our probabilistic complex event processing system with a non probabilistic complex event processing system. The non probabilistic complex event processing system does not assign probabilities to complex events, so there is no way of finding out how accurate a certain complex event really is. In addition to this the non probabilistic system triggers a complex event as soon as it is detected and unlike the probabilistic system does not wait for the probability of the complex event to reach a certain threshold. This is because in the non probabilistic scenario complex events do not have associated probabilities.



(a) Accuracy



(b) Precision

Figure 7.9: Accuracy and Precision Comparison of CET 50, 60, 70, 80, 90

In our evaluations we have mimicked the behaviour of a non probabilistic complex event processing system by fixing the CE Threshold value to 0. This would ensure that the system would trigger a complex event as soon as it is detected, which is exactly what a non probabilistic system is supposed to do.

Effect of Complex Event Thresholds

In this scenario we compared the non-probabilistic complex event processing system (i.e. a system having CE Threshold = 0) with a probabilistic complex event processing system having CE Thresholds of 50, and 90. The only variable in these experiments was the accuracy of the physical readers.

PR Probability 50%:

Accuracy: Figure 7.10(a) shows the results for accuracy measurements. In this experiment we fixed the *pr* accuracy at 50%. The results of this experiment are quiet predictable and show that the higher the CE Threshold value the more accurately our system detects complex events. As a matter of fact, for *pr* accuracy of 0.5 (50%), the CE Threshold of 90% is almost 400% more accurate than a non probabilistic complex event detection system.

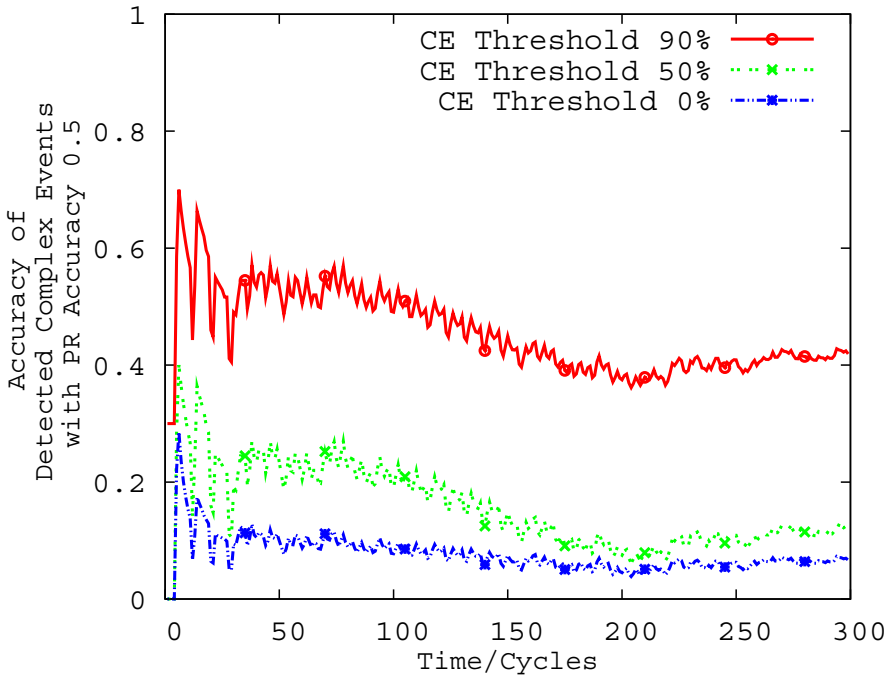
The accuracy of detecting complex events with CE Threshold 90 was initially around 60% but then stabilized and remained greater than 40% at all times. However, there was no marked difference between the accuracy of detecting complex events in a non probabilistic system and a probabilistic system having a *pr* accuracy of 50% and CE Threshold of 50%.

Precision: Figure 7.10(b) shows the results for precision measurements. In this experiment we fixed the *pr* accuracy at 50%. The results show that the precision of detected complex events increased symmetrically when the CE Threshold was increased from 50 to 90. The precision of detected complex events for CE Threshold was initially around 80% level, but then continued to swing wildly between 70% and 40% mark. Similarly the precision of detected complex events for CE Threshold was initially around 60% level, but then continued to swing between 50% and 20%. This behaviour is indicative of the fact that we can not have a high and predictable precision measure with low physical reader *pr* accuracy.

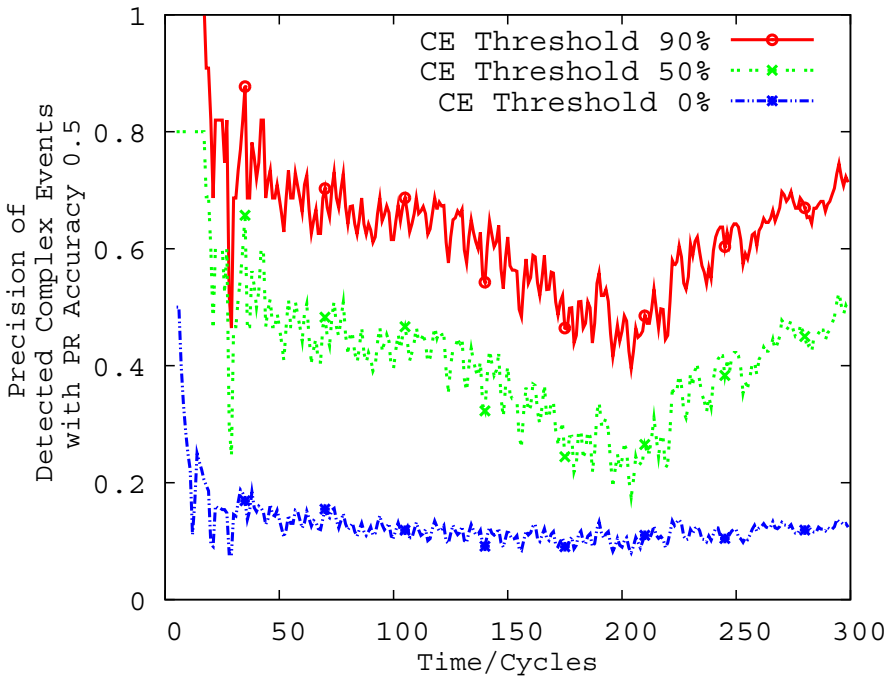
The precision of CE Threshold 0 (i.e. non probabilistic complex event detection system), remained constant around 10% level. This shows that even with extremely low *pr* accuracy, the probabilistic complex event detection system is almost 400% more precise than the non-probabilistic complex event processing system.

PR Probability 70%:

Accuracy: Figure 7.11(a) shows the results for accuracy measurements. In this experiment we fixed the *pr* accuracy at 70%. Similar to Figure 7.10(a), this experiment also reveals that the probabilistic complex event detection system detects events with higher accuracy when the CE threshold is higher. If we compare the performance



(a) Accuracy



(b) Precision

Figure 7.10: Accuracy and Precision Comparison of CET 90, 50 and 0 with PR 50

of CE Threshold 90 with non probabilistic complex event detection system (i.e. CE Threshold 0), we can observe that that CE Threshold 90 is almost 600% more accurate than the non-probabilistic complex event detection system when the *pr* accuracy is fixed at 0.7.

The accuracy of detecting complex events with CE Threshold 90 was initially around 80% but then stabilized at a lower level. However, the accuracy of CE Threshold 90 never went below 60%. Similarly, the accuracy of CE Threshold 50 initially spiked to 60% but then stabilized around 30%, whereas the accuracy of CE Threshold 0 (i.e. non probabilistic complex event detection system) remained around 10%.

Precision: Figure 7.11(b) shows the results for precision measurements. In this experiment we fixed the *pr* accuracy at 70%. The probabilistic complex event processing system was able to detect complex events with a precision of 80%. If we compare, the precision of CE Threshold 90 with the precision of CE Threshold 90 in Figure 7.10(b), then we can easily see that the precision of detecting complex events increased by around 300%-400% for CE Threshold 90. If we compare the performance of CE Threshold 90 with non probabilistic complex event detection system (i.e. CE Threshold 0), we can see that CE Threshold 90 is almost 800% more precise than non-probabilistic complex event detection system when the *pr* accuracy is fixed at 0.7.

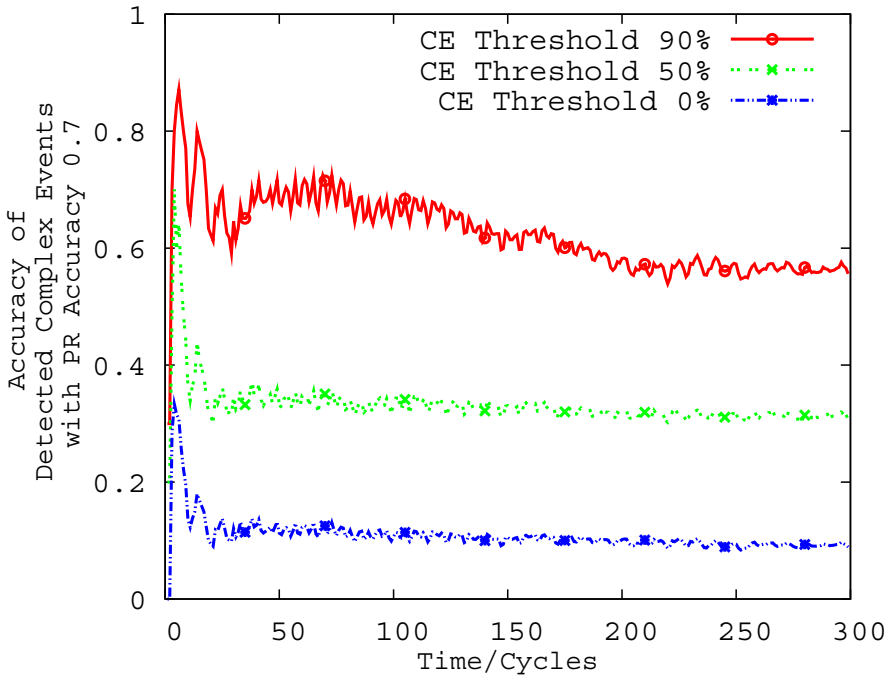
The precision of non probabilistic system (i.e. CE Threshold 0) remained constant around 10%. The precision of CE Threshold 50 remained around 50%, which translates to an increase of 400% over non probabilistic system (i.e. CE Threshold 0). If we compare the precision of CE Threshold 0 with the precision of CE Threshold 0 in Figure 7.10(b), we can easily conclude that the change in the accuracy of *pr*s has no effect on the precision of non probabilistic system.

PR Probability 90%:

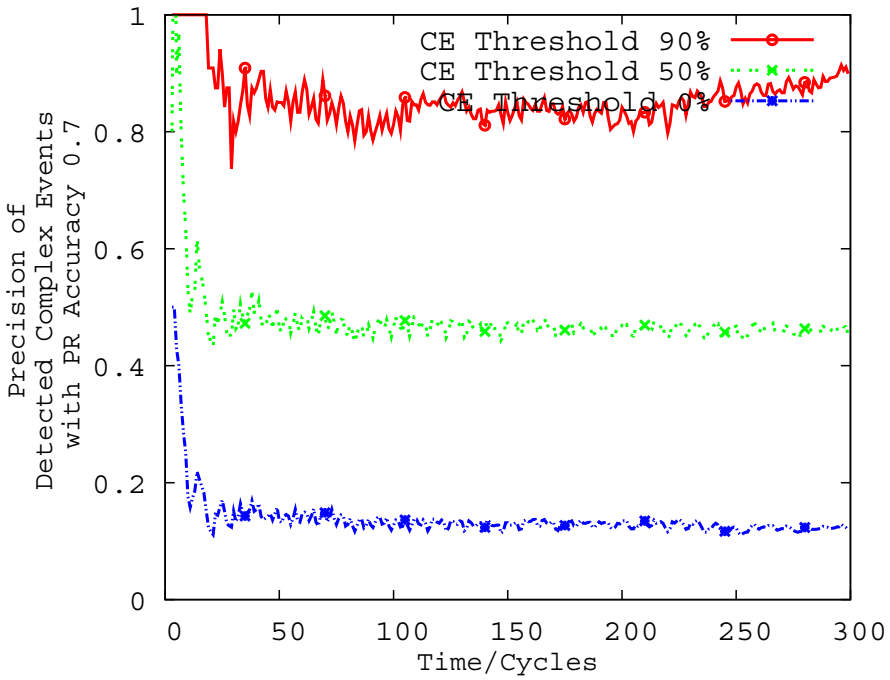
Accuracy: Figure 7.12(a) shows the results for accuracy measurements. In this experiment we fixed the *pr* accuracy at 90%. As has been observed in Figure 7.10(a) and Figure 7.11(a), we can see that higher the CE Threshold, the more accurately we are able to detect complex events. When we compare the performance of CE Threshold 90 with non probabilistic complex event detection system (i.e. CE Threshold 0), we can observe that that CE Threshold 90 is almost 800% more accurate than the non-probabilistic complex event detection system when the *pr* accuracy is fixed at 0.9. From Figure 7.10(a),7.11(a),7.12(a) we can conclude that we are able to achieve a 100% increase in accuracy in detecting the complex events for every 0.1 increase in the accuracy of *pr*s for CE Threshold of 90.

The accuracy of detecting complex events with CE Threshold 90 was initially around 90% but then stabilized at a lower level. However, the accuracy of CE Threshold 90 never went below 70%.

Another important observation is that unlike Figure 7.6(a) and Figure 7.11(a), CE Threshold 50 was remarkably more accurate than CE Threshold 0. CE Threshold 50



(a) Accuracy



(b) Precision

Figure 7.11: Accuracy and Precision Comparison of CET 90, 50 and 0 with PR 70

initially started at 60% accuracy but then eased off a bit, however, it always remained above 40% accuracy level. In terms of percentages, CE Threshold 50 was almost 300% more accurate than a non probabilistic system with a *pr* accuracy of 90%.

If we compare the results of Figure 7.10(a) with Figure 7.12(a), we can observe that the accuracy of CE Threshold 90 in Figure 7.10(a) is comparable to the accuracy of CE Threshold 50 in Figure 7.12(a).

Precision: Figure 7.12(b) shows the results for precision measurements. In this experiment we fixed the *pr* accuracy at 90%. The precision of CE Threshold 90 started off at around 100% level and remained fairly stable and never went below 90%. Similarly there was no significant difference between the precision of CE Threshold 90 and CE Threshold 50. However, precision of detected complex events for CE Threshold 0 (i.e. non probabilistic system) was very poor and remained constant around 10% level. When we compare the performance of CE Threshold 90 and 50 with non probabilistic complex event detection system (i.e. CE Threshold 0), we can observe that CE Threshold 90 and 50 are almost 800% more precise than the non-probabilistic complex event detection system when the *pr* accuracy is fixed at 0.9.

A comparison of CE Threshold 90 with CE Threshold 90 in Figure 7.10(b) and Figure 7.11(b) shows that precision of detected complex events increase with an increase in the accuracy of *pr*s. However, a comparison of CE Threshold 50 with CE Threshold 50 in Figure 7.10(b) and Figure 7.11(b) shows no such pattern and instead reveals that precision of detected complex events is extremely low and undeterministic as long as accuracy of *pr*s remain below 90%. But when the accuracy of *pr*s is 90%, the precision of detected complex events with CE Threshold 50 is almost equal to that of CE Threshold 90. This leads us to conclude that the precision of complex events is dependent upon the accuracy of *pr*s and the variation in CE Thresholds between 50-90 have very little effect on it.

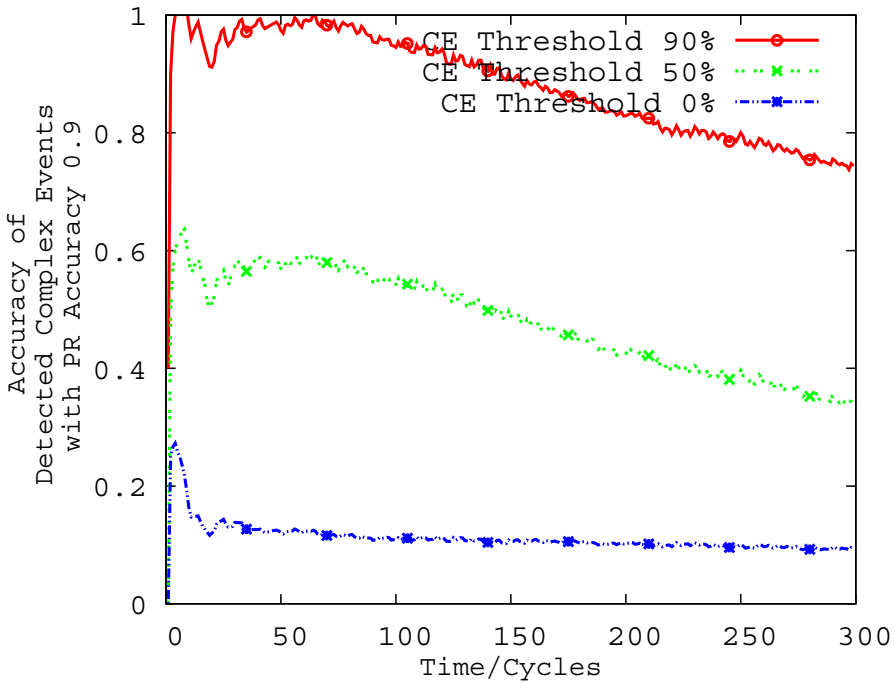
If we compare the precision of CE Threshold 0 with the precision of CE Threshold 0 in Figure 7.10(b) and Figure 7.11(b), we can easily conclude that if the CE Threshold is 0, then the change in the accuracy of physical readers *pr*s has no effect on the precision.

Effect of Physical Reader Accuracy

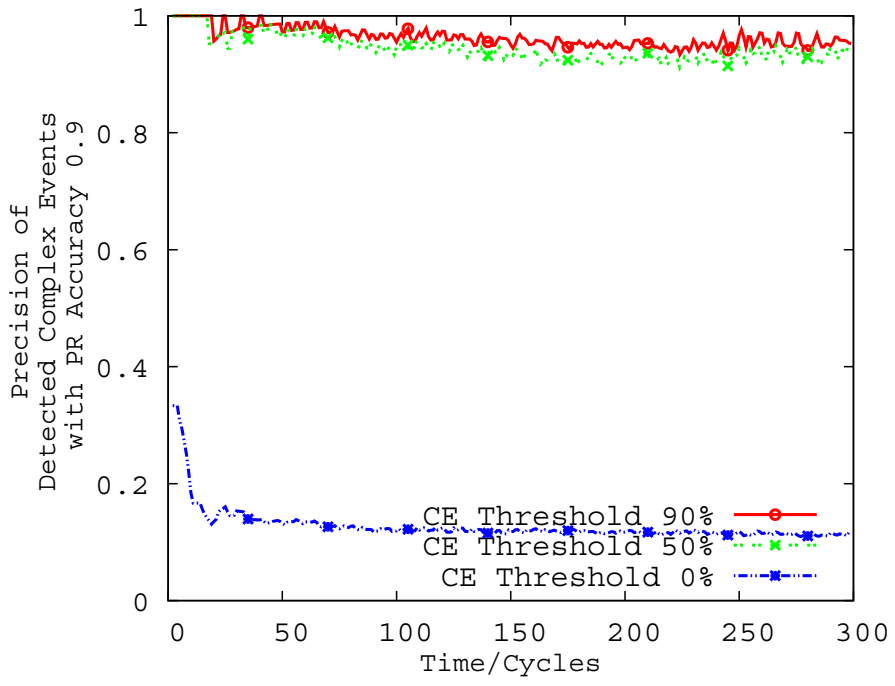
In this scenario we compared the probabilistic and non-probabilistic complex event processing system by varying the physical reader accuracies. The different physical reader probabilities that we used in our evaluations were pr 90, 70, and 50.

CE Threshold 90%:

Accuracy: Figure 7.13(a) shows the results for accuracy measurements. In this experiment we fixed the CE Threshold at 90%. The experiment further reveals what we have already observed before that if the CE Threshold is kept constant the accuracy of detecting complex events increases with an increase in the accuracy of underlying physical readers. The behaviour however, can not be generalized and is a characteristic

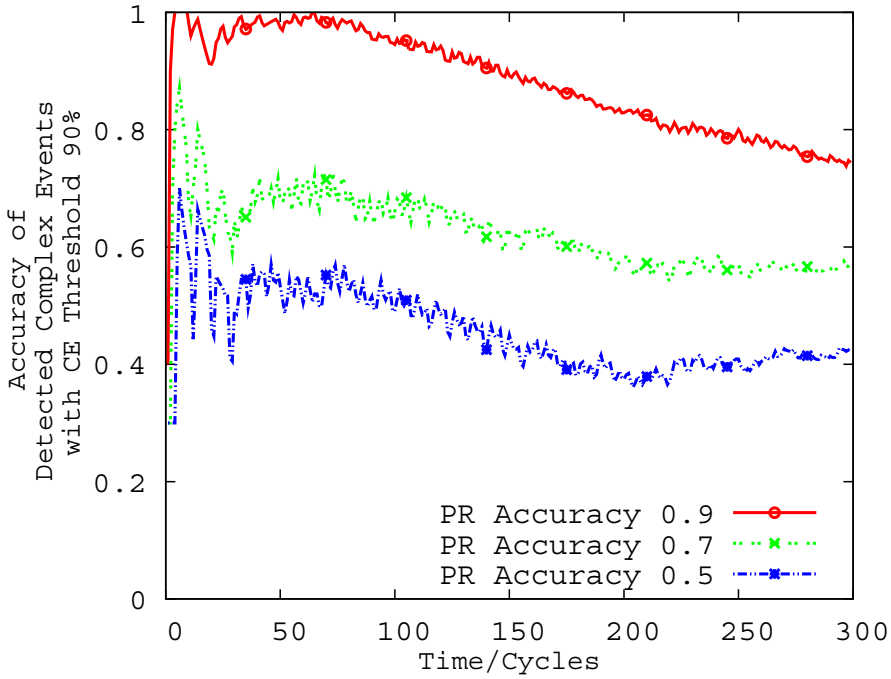


(a) Accuracy

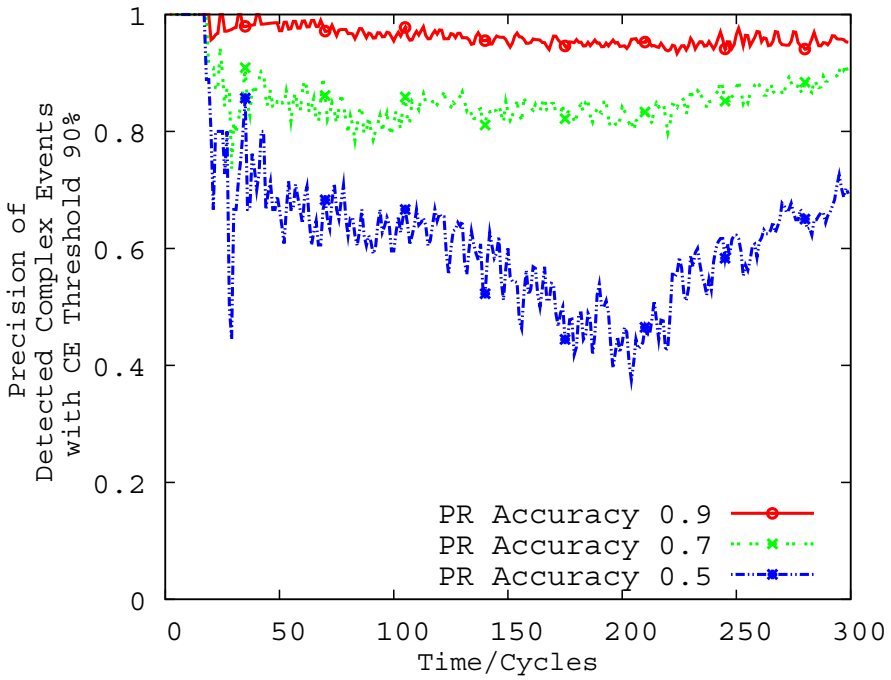


(b) Precision

Figure 7.12: Accuracy and Precision Comparison of CET 90, 50 and 0 with PR 90



(a) Accuracy



(b) Precision

Figure 7.13: Accuracy and Precision Comparison of CET 90 with PR 90,70, 50

of a relatively high CE Threshold. We would see later on that CE Threshold 50 and non probabilistic complex event detection system does not exhibit the same behaviour.

The accuracy of complex event detection with PR 90 remained above 60%, PR 70 remained above 50% and PR 50 remained above 40% with a CE Threshold of 90. This means that for CE Threshold 90, every 20% increase in the *pr* accuracy led to a 20% increase in the accuracy of detected complex events.

Precision: Figure 7.13(b) shows the results for precision measurements. In this experiment we fixed the CE Threshold at 90%. The general observation of this experiment was that the precision of complex event detection increased with the increase in the accuracy of underlying physical readers. The precision of complex events detected remained above 90% when the physical reader *pr* accuracy was 90%. The precision of detected complex events remained greater than 80% with *pr* accuracy of 70% and remained above 40% with *pr* accuracy of 50%.

The precision of detected complex events swung wildly when the *pr* accuracy was fixed at 50%. This shows that the precision can not be determined precisely when the *pr* accuracy is this low.

CE Threshold 50%:

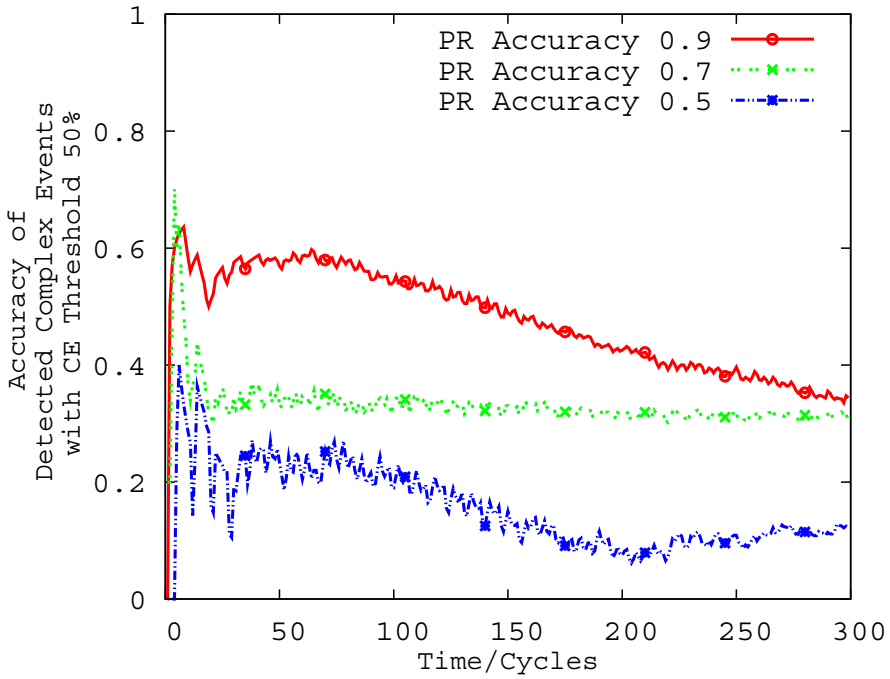
Accuracy: Figure 7.14(a) shows the results for accuracy measurements. In this experiment we fixed the CE Threshold at 50%. The accuracy of detected complex events for *pr* accuracy of 90% remained above 40%.

However, the accuracy of physical reader *pr* 70 and *pr* 50 show an interesting picture. The accuracy of detecting complex events when the accuracy of *pr* was 70 remained fairly constant at around 30% level. However, the accuracy of *pr* 50 initially swung wildly and then continued to drop before stabilizing at around 10% level. From this experiment it is obvious that we can not have a reliable accuracy measure of complex events when the CE Threshold is lowered to 50%.

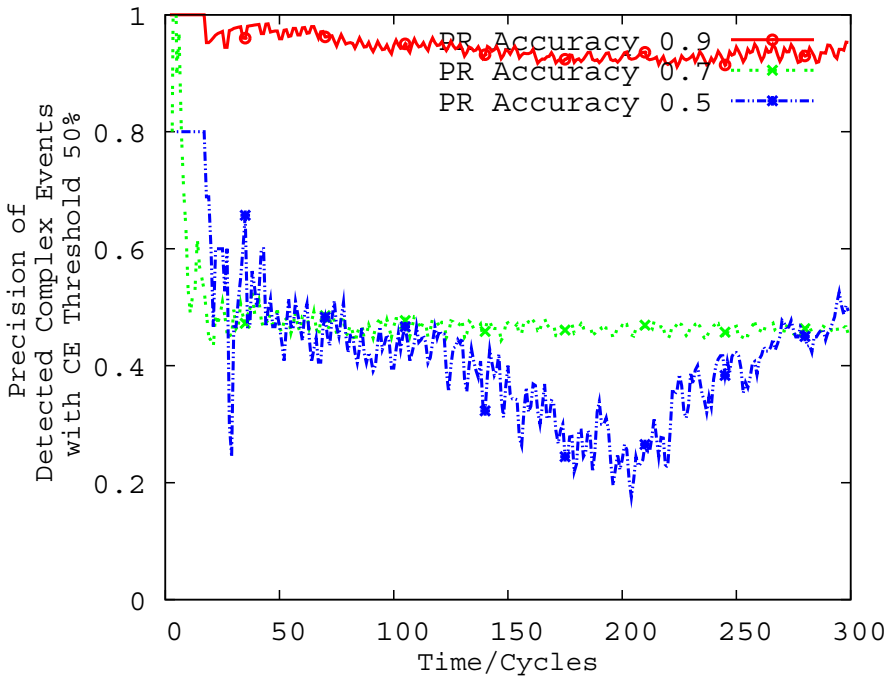
Precision: Figure 7.14(b) shows the results for precision measurements. In this experiment we fixed the CE Threshold at 50%. The precision of detected complex events remained extremely high for *pr* 90, and never went below 90%. This clearly shows that precision is highly dependent on accurate underlying physical readers.

The precision of *pr* 70 remained constant around 50%. This shows that by reducing the *pr* accuracy by 20%, the precision of detected complex events decreased by almost 50%. The precision of *pr* 50 presented an interesting picture. Precision of *pr* 50 initially started at around 60% but then continued to swing wildly. It went to as low as 20% before increasing again. However, the precision for *pr* 50 never stabilized above 60%. This shows that the precision with which complex events are detected is very hard to ascertain, if the underlying physical readers are only 50% accurate.

CE Threshold 0%:



(a) Accuracy



(b) Precision

Figure 7.14: Accuracy and Precision Comparison of CET 50 with PR 90,70, 50

Accuracy: Figure 7.15(a) shows the results for accuracy measurements. In this experiment we fixed the CE Threshold at 0%. In simpler words, the experiment shows the accuracy of detected complex events in a non probabilistic complex event detection system under varyingly accurate physical readers. Figure 7.15(a) shows that the accuracy of detected complex events remained around 10% for *pr* accuracy of 90%. Furthermore, the accuracy of complex events for *pr* probability of 70 was exactly the same as that of events detected with *pr* accuracy of 90%. Similarly the accuracy of detected complex events with *pr* accuracy of 50% wasn't much different either.

This experiment makes it abundantly clear that irrespective of how accurate or inaccurate the underlying physical readers are, the detected complex events can not have any meaningful accuracy, if we do not have a probabilistic complex event detection system. Triggering each and every complex event as soon as it is detected would lead to an extremely low accuracy for the detected complex events.

If we compare the results of this experiment with Figure 7.13(a), we would see that the accuracy of detected complex events increased by 600% (CE Threshold 90 and PR 90) in the best case and by 300% (CE Threshold 90 and PR 50) in the worst case.

Precision: Figure 7.15(b) shows the results for precision measurements. In this experiment we fixed the CE Threshold at 0%. The results show that the precision of detected complex events remained around 10% for physical readers having an accuracy of 90%, 70%, and 50%.

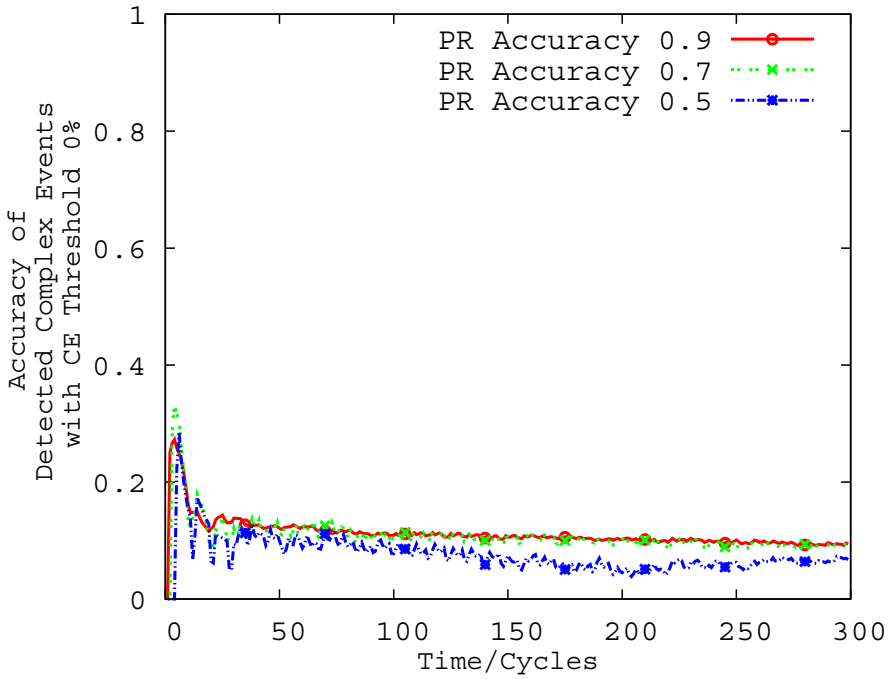
The take away point of this experiment is that the precision of the detected complex events is not effected by an increase or decrease in the accuracy of underlying physical readers, if these complex events does not have associated probabilities and are triggered as soon as they are detected. Figure 7.15 serves as a every emphatic motivation for using probabilistic complex event detection in a smart variant production environment.

If we compare the results of this experiment with Figure 7.13(b) and Figure 7.14(b), we would see that the precision of detected complex events increased by 800% (CE Threshold 90 and PR 90) in the best case and by 300% (CE Threshold 50 and PR 50) in the worst case.

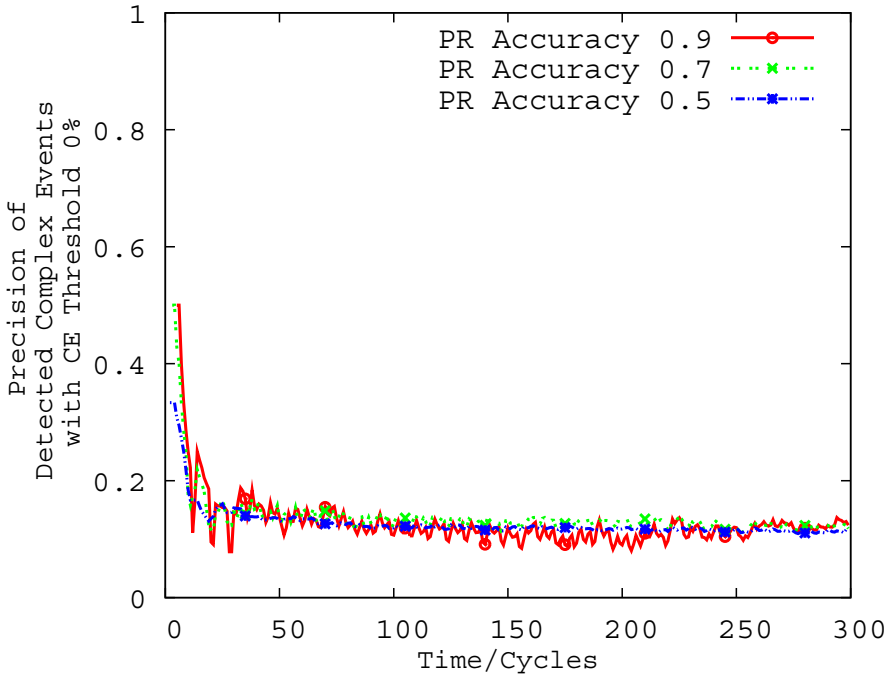
7.5 Related Work

Over the years, much research effort has been put into developing complex event processing systems. In the sections below, we will take a brief look at some of the important systems that have been designed and developed.

DistCED: distributed complex event detection DistCED [PSB03], which was developed in Java, is perhaps one of the earliest distributed complex event processing systems. The framework uses extended, non-deterministic finite state automata to detect events. The system defines six operators which are concatenation. sequence,



(a) Accuracy



(b) Precision

Figure 7.15: Accuracy and Precision Comparison of CET 0 with PR 90,70, 50

iteration, alternation, timing and parallelisation. These operators are specified using an expressive CE language. In order to overcome the issue of incorrect event detection DistCED uses a delayed detection policy that makes detectors wait till events are stable enough for detection. This ensure that late arrival of events due to network and node issues do not cause problems. Although the authors proposed automata distribution, they did not presented any algorithms or cost models for this distribution.

SASE: CEP over streams: Wu et al [WDR06], [GWC+06] carried out another pioneering work in the field. The authors developed a CEP system called SASE which can be used to execute complex event queries over real-time streams of RFID readings. For the purpose of their work, the authors assumed an infinite sequence of events to be an event stream, since RFID event streams tend to vary from video and audio streams in a way that RFID streams consist of discrete and distinct events which could be intermittently dispersed over time.

The basic structure of the language used to define complex events is:

```
EVENT <event pattern>
[WHERE <qualification>]
[WITHIN <window>]
```

where event pattern describes the type of events which in our scenario would have been sequence, synchronization, delay etc. Qualification describes the conditions that needs to be fulfilled in order for the raw events to be correlated and composed into complex events. In general the WHERE clause is a boolean combination of predicates that use one of the six comparison operators ($=$, $\neq$, $>$, $<$, $\leq$, $\geq$). The Window describes the time span within which the events should occur to be correlated together.

For a given sequence of events as input, the output is also a sequence of events which are composed together. The complex events within the SASE system are generated by concatenating all the attributes of the matching input events, and hence the resulting events can provide all necessary information to monitoring applications. One of the short comings of the SASE language is that it only allows for transformation of primitive events to complex events and not from complex events to (even more) complex events. Another issue is that the SASE language assumes total ordering of events.

Bridging physical and virtual worlds: CEP for RFID data streams: Wang et al [WLLB06] have also proposed an RFID based complex event processing system. The system uses ECA (Event Condition Action) based rules to trigger complex events based on raw RFID events. Wang et al divided the complex event constructors/operators into two categories: non-temporal event constructors and temporal event constructors. The non-temporal constructors include OR, AND and NOT operators whereas the temporal constructors comprise of SEQ (sequence), TSEQ (distance constrained sequence), and WITHIN constructors.

The authors have also presented constructs to create rules, which can be thought of as macros. The general construct of a rule is presented below:

```
CREATE RULE rule id, rule name
ON event
IF condition
DO action 1; action 2; ...; action n
```

These rules can be used to perform low level tasks such as data filtering to eliminate duplicates or high level semantic tasks such as defining rules to create location transformation or to create containment relationships.

Cayuga: a high-performance event processing engine

Cayuga [DGH+06], [DGP+07] is a single server based CEP system developed at Cornell University. Event streams are infinite sequences of relational tuples with interval-based timestamps. The Cayuga query language is derived from an event algebra [DGH+06]; and is basically a simple mapping of the SQL operators to algebra operators having an SQLish syntax. The query language is quiet similar to the SASE event query language [WDR06] discussed in the previous section. Each query has the following form:

```
SELECT {attributes}
FROM {algebra_expression}
PUBLISH {output_stream}
```

The SELECT clause specifies the attributes in the output stream schema, the FROM clause specifies a Cayuga event pattern, and the PUBLISH clause gives the output stream a name. Cayuga's event algebra has six operators: projection, selection, renaming, union, conditional sequence and iteration. Event algebra expressions are detected by nondeterministic finite automata, which can detect unbounded sequences. This is again something that is quiet similar to both SASE and DistCED. To achieve high performance, Cayuga uses custom heap management, indexing of operator predicates and reuse of shared automata instances. However, Cayuga does not support automated query rewriting and distributed detection. The task of distributing Cayuga automata is made quiet difficult by the fact that the system merges event algebra expressions into a single automaton, which would have to be partitioned across nodes for distribution.

Distributed CEP with query rewriting: NEXT is a CEP system developed by Schultz et al [SMMP09], that tries to perform distributed event detection along with performing query re-writing. The system has been designed and developed by the same team that was behind the DistCED system. So similar to DistCED, Next also uses an expressive automata-based approach for event detection. The events are defined by

the users using a high level SQL-like language that has six basic operators. These six operators are: the filter operator, the union operator, the next operator, the iteration operator, the exception operator, and the time operator.

The system can be considered an upgrade to the DistCED system in terms of the distributed capabilities. In this system the authors have tried to develop a system that is capable of deploying operators on distributed nodes. In order to do this, the event language facilitates the rewriting of expressions into equivalent ones and the automata model provides facilities to deploy detection operators across multiple machines. In addition to this, the system is capable of making optimisation and placement decisions according to cost functions derived from the resource consumption of event automata. This cost model is further used to rewrite queries with the express purpose of reducing the CPU consumption of these queries.

CEP in RFID middleware: a three layer perspective: Hu et al. [HYHZ08] also proposed an RFID based complex event processing middleware. The CEP middleware proposed by Hu et al. divides the tasks of detecting complex events into three different layers: logic structure, temporal constraint, and event detection. In the logic structure layer, the logical relations between the different event types are defined. This step could also be termed as the event definition phase. In the temporal constraint layer, the temporal conditions are described over the events defined in the previous layer. Whereas, the event detection layer deals with the actual detection of complex events. Some of the event operators defined by Hu et al. include: aggregation, disjunction, conjunction, negation, sequence and within. For the purpose of event detection, the authors have used petri-nets, which are directed bi-partite graphs consisting of transitions (i.e. events that may occur) and states (i.e. states of complex events).

Discussion: Several RFID based CEP systems [ZZ08], [HYHZ08], [WDR06], [WLLB06] have been proposed by the research community. These systems differ from our endeavour in two important ways. Firstly, the prime objective of these systems was to develop a CEP based system for processing RFID streams. None of these systems didnt come up with a new algorithm or method to detect complex events. However, in our case we have designed specific algorithms to detect specific type of complex manufacturing events that are relevant for our scenario.

Secondly, since we intended to use our system in production environments where reliability and accuracy is of paramount importance, we developed a comprehensive probabilistic complex event detection framework that assigns probabilities to each detected complex event. This allows the applications to know with certainity how reliable or unreliable a certain complex event is. None of the CEP systems discussed above have any accuracy measure for the complex events that they generate.

7.6 Summary

In this chapter, we have presented concepts and algorithms to detect complex manufacturing events. We further designed mechanisms to assign probabilities to these complex manufacturing events so as to have a measure of how accurate a certain event really is. In addition to this we also extended the RFID consistency stack that was initially presented in Chapter 5.1. The stack now includes two additional production issues i.e. presence and position consistency. Finally we tested out system in a simulated setting.

The simulations that we carried out revealed that for a given CE Threshold, there was little to no difference in accuracy or precision of the system when it came to detecting manufacturing errors that were solely caused by false readings, missed readings, out of order readings or were a combination of all of these basic errors. The reason for this is that when it comes to the complex event level, all RFID related errors behave the same way as long as the CE Threshold value is kept constant for all these different errors.

Our evaluations for complex manufacturing errors revealed pretty much the same results as experienced with raw RFID errors i.e. for a given CE Threshold, the accuracy and precision with which different complex manufacturing errors were detected remained fairly similar. The accuracy of detecting complex events for a CE Threshold of 0.7 remained around 60%, whereas the precision remained around 80%.

Furthermore, we studied the effect of reliability of physical readers on the overall accuracy and precision of the entire CEP system. Our studies showed that complex events were detected with a higher accuracy and precision with an increase in the accuracy of underlying physical readers. The system was able to detect complex events with 60% accuracy and 95% precision when the accuracy of the physical readers was around 90%. When the accuracy of physical readers was reduced to 50%, the accuracy of complex events decreased to 30% and the precision decreased to 10%.

The studies conducted to find the effect of CE Thresholds on the overall system accuracy and precision revealed that the accuracy and precision of complex manufacturing events was unpredictable for a CE Threshold of 50% or lower. However, when the CE Threshold was higher than 60% the accuracy and precision of detecting complex events increased with an increase in the CE Threshold.

In addition to the studies discussed above, we also conducted comparative studies to find out how the probabilistic CEP system would fare against a non-probabilistic CEP system. We compared the two systems firstly by varying the CE Thresholds and then by varying the physical reader accuracies. The results revealed that:

Effect of CE Thresholds:

- PR accuracy of 50%: The probabilistic CEP system was 400% more accurate and around 600% more precise than the non-probabilistic system when CE Threshold was 90%.

- PR accuracy of 70%: The probabilistic CEP system was 600% more accurate and around 400% more precise than non-probabilistic system when the CE Threshold was 90%.
- PR accuracy of 90%: The probabilistic CEP system was almost 800% more accurate and precise than non-probabilistic system when the CE threshold was 90%.

Effect of PR Accuracy:

- CE Threshold of 90%: With a CE Threshold of 90%, the accuracy of complex event detections increased by roughly 20% for every 20% increase in the accuracy of underlying physical readers.
- CE Threshold of 50%: The results show that the accuracy of detecting complex events when the accuracy of physical readers *pr* was 70 remained fairly constant at around 30% level. However, the accuracy of complex events with pr 50 initially swung wildly and then continued to drop before stabilizing at around 10% level. The experiments revealed that 50% is an extremely low threshold level to achieve predictable results.
- CE Threshold of 0%: The accuracy of detected complex events remained around 10% for physical reader *pr* accuracy of 90, 70 and 50. The results show that if we do not assign probabilities to complex events and trigger them as soon as they are detected, the complex events can not be reported with any meaningful accuracy irrespective of how accurate the underlying physical readers are.

Chapter 8

Summary and Future Work

In this chapter, we summarize the main contributions of this thesis and present a brief discussion on possible future work.

8.1 Summary

We have divided the summary into 5 sub-sections, each corresponding to a chapter in this work. Section 8.1.1 provides the motivation for our work. Section 8.1.2 discusses the system model which forms the basis for the entire work carried out in the subsequent chapters. Section 8.1.4 briefs the real-time production monitoring and sequence detection algorithm. Section 8.1.5 summarizes the self-calibration algorithm that was designed to continuously calibrate the probabilities of RFID readers. Section 8.1.6 presents the essence of the complex manufacturing events and the algorithms used to assign probabilities to these events.

8.1.1 Case Studies

Service offerings are now either replacing or becoming as important and significant as product offerings in terms of revenues for companies. Due to this reason companies are now increasingly bundling services with products. In this chapter, we surveyed some of the successful services offered by both military and commercial organizations. In addition to this, we presented the notion of the smart variant production environment and discussed the different components or building blocks that are needed for such an environment. The design of some of these components (such as the real-time production monitoring and complex event detection within the production environments) forms the core of our work. In addition, we also presented some of the services that such a production facility could offer to the company and to its customers.

In short a smart variant production environment would lead to: improved lead time, increased productivity, improved inventory management, improved customer relationships, reduced human errors, reduce manpower and manual data recording, reduction of product part and mobile tool losses.

Each of the benefit listed above has the potential of saving a production organization millions in revenues each year.

8.1.2 System Model

Within the system model chapter (cf: Chapter 3) we formally discussed the model on which our system is built. In specific we formulated the notion of physical readers, virtual readers, network topology, and the system events. Further we also discussed the event model and detailed the events that are generated in our system. In addition to this we also formalized the different RFID reader failures such as duplicate readings, false readings, out-of-order readings, and missed readings that should be considered by any reliable RFID framework. Furthermore, we also formalized the failure model and the network and node failure assumptions.

8.1.3 Lernfabrik

Within the Lernfabrik chapter (cf: Chapter 4) we presented the Lernfabrik at length, including the different components or modules that are present within the factory. Furthermore, we performed comprehensive evaluations regarding the reliability of RFID readers. This data from these evaluations was later used to fine tune the simulations that were done to validate the various algorithms designed in this work.

Within the Lernfabrik we conducted evaluations to study the effects of tag placement, multiple tag reads, tag orientation, RFID reader power, and reader interference on the accuracy of RFID readers. The findings of our study are summarized below:

- Tag Placement: RFID tags were placed on the metallic product parts in two different ways. Firstly they were placed on the parts directly and in the second setting they were placed with a spacer between the tag and the part. The evaluations showed that tags placed with a spacer have had far higher read accuracy then the ones placed directly on the metal objects.
- Differences among Readers: Three different RFID readers were tested under exactly the same physical conditions. The results showed that even RFID readers of the same model and from the same manufacturer have different accuracy even when they are operating under the same conditions.
- Multiple Object Reads: The false negatives for RFID readers was slightly higher for 3 simultaneous tag reads, as compared to a setting in which the readers had

to read just 1 RFID tag at a time. Experiments conducted by other researchers [PSR+06], [Vio05], [HC06] show that the accuracy of RFID readers fall sharply when they have to read more than five tags simultaneously. We only conducted evaluations for 3 tags because that was the maximun number of product parts in our product on the Lernfabrik.

- Tag Orientation: We evaluated two different tag orientation settings. In the first setting the tags were placed on the product parts in such a manner that when they pass across the readers, they would be directly facing the readers. In the second setting the tags were placed randomly on the product parts. The evaluations showed that the RFID readers have far higher accuracy when they have to read tags which are directly facing them.
- Reader Power: The reader power evaluations revealed that reducing the reader power reduces the out-of-order readings. This is quiet intuitive, since with higher power, the read range of an RFID reader increases which results in the reader reading tags out of order.
- Reader Interference: The experiments for reader interference revealed that if two readers are deployed within the read range of each other. The out-of-order readings of the later reader is less than that of the one deployed before it. This is because the signals of the initial reader interferes with the second reader and decreases the read range of the later reader.

The results of RFID reader evaluations provides further motivation for a real-time monitoring framework that not just monitors the production environment but also provide guarantees regarding the reliability of RFID devices.

8.1.4 Probabilistic Sequence Detection of Product Parts

In our work we presented the Consistency stack (cf. Chapter 5). The stack is conceptual in nature and divides the consistency issues into separate layers. These layers address inconsistencies that may arise due to the unreliability of RFID devices and issues in production environments due to variant production. The consistency stack is the first effort of its kind to formalize the consistency issues (such as duplicate readings, missed readings, and false readings) that may arise as a result of using RFID devices for real-time production monitoring.

Furthermore, we have also designed a sequence detection algorithm that attaches probabilistic guarantees to the product part sequences detected by the RFID readers deployed along the production paths.

Our evaluations show that our framework performs better when the physical readers prs are uniformly distributed across the vrs. Partial sequences were being detected with greater than 90% probability when the physical readers were distributed uniformly as

against roughly 80% probability which was achieved when the readers were distributed in zipfian manner. The improvement in probability was more evident with partial sequences, as uniformly distributed physical readers were able to detect these partial sequences with around 40% probability as compared to the roughly 10% probability of zipfian distribution.

The evaluations for physical reader reliability showed that the sequence detection algorithm was able to detect 90% of all partial sequences with greater than 90% probability even when the physical reader accuracy was 0.7. An even greater number of partial sequences were detected with greater than 90% probability with a *pr* accuracy of 0.9. Even with a sequence length of 16, more than 80% of all extended sequences were detected with greater than 90% probability when the accuracy of the *pr*s was only 70%. The results clearly show that the sequence detection algorithm does a very good job of detecting sequence with extremely high probability even when the physical readers are only 70% accurate.

8.1.5 Probabilistic Self Re-calibration of RFID Reader Probabilities

In this chapter (cf. Chapter 6), we discussed that RFID readers are inherently unreliable and this unreliability can increase or decrease (mostly increase) over time. Therefore, in order to have accurate monitoring of product parts using RFID readers, it is important to constantly calibrate the physical reader *pr* accuracy. To achieve this objective, we presented an algorithm that allows RFID readers to self-calibrate their accuracy to reflect the probability with which each individual physical reader is detecting product parts moving on the production lines.

The basic idea of the self-calibration algorithm is to find out how many correct and incorrect readings an RFID reader has made. This is done by comparing the readings of an RFID reader with the most probable readings. This gives us the accuracy with which the reader is detecting the product parts.

Our evaluations for the effect of actual probability of physical readers showed that the higher the actual probability of physical readers *pr*s, the less time the system needed to calibrate the estimated probability of these physical readers with a relatively high accuracy. This should not be a surprise for anyone, because if we consider the converse of this situation it would become evident that we can not estimate the actual probabilities of physical readers with little or no accurate readings being performed by the physical readers.

In addition to the observation made above, our evaluations showed that the self-calibration algorithm reacted robustly to induced physical changes. The algorithm was able to calibrate itself to an accuracy of greater than 90% even in the presence of changes induced after every 5 cycle. The only difference that the higher rate of change made was that the algorithm needed a little more time to reach a stable state as compared with a lower rate of change scenario.

Our evaluations further revealed that the time to calibrate the estimated probabilities of physical readers is reduced if a large number of physical readers are clustered together in one virtual reader. The reason for this is that, if we spread physical readers sparsely across a large number of virtual readers, each reader would have a smaller data sample to perform its computations and as a result the estimated accuracy would suffer.

We also compared the performance of our calibrated system with a system in which reader probabilities are not calibrated. In an uncalibrated system, the difference between the estimated probability of the physical reader and its actual probability will be induced at system initialization time or if the reader breaks down. The difference between estimated and actual probabilities in such a system would never be reduced and would continue to adversely impact the overall reliability of the entire infrastructure. Theoretically the difference between the estimated and actual probability of physical reader in an uncalibrated system could be as high as 0.99, since one can fix the probability of the reader at 0.99 and the reader can then break down and stop performing and hence have an actual accuracy of 0.0. However, in a calibrated system the difference between the estimated probability and the actual probability of physical readers is never greater than 0.05.

8.1.6 Probabilistic Complex Manufacturing Event Detection

In this chapter (cf. Chapter 7), we have presented concepts for the detection of complex manufacturing events using unreliable RFID readings. Based on a probabilistic model we presented the algorithm to assign probabilities to each of the detected complex manufacturing events. The basic idea of our approach is to exploit redundant readings to get an accurate picture of the real world. In addition to detecting complex manufacturing events with a high degree of accuracy, we also extended the RFID consistency stack from Chaper 5.1 to further include important production issues. Furthermore, we performed comprehensive evaluations in a simulated environment to test our algorithms.

The simulations that we carried out revealed that for a given CE Threshold, there was little to no difference in accuracy or precision of the system when it came to detecting manufacturing errors that were solely caused by false readings, missed readings, out of order readings or were a combination of all of these basic errors. The reason for this is that all RFID related errors behave the same way with respect to the complex events as long as the CE Threshold value is kept constant for all these different errors.

Our evaluations for complex manufacturing errors revealed pretty much the same results as experienced with raw RFID errors i.e. for a given CE Threshold, the accuracy and precision with which different complex manufacturing errors were detected remained fairly similar. The accuracy of detecting complex events for a CE Threshold of 0.7 remained around 60%, whereas the precision remained around 80%.

Furthermore, we studied the effect of reliability of physical readers on the overall accuracy and precision of the entire CEP system. Our studies showed that complex events were detected with a higher accuracy and precision with an increase in the accuracy of underlying physical readers. The system was able to detect complex events with 60% accuracy and 95% precision when the accuracy of the physical readers was around 90%. When the accuracy of physical readers was reduced to 50%, the accuracy of complex events decreased to 30% and the precision decreased to 10%.

The studies conducted to find the effect of CE Thresholds on the overall system accuracy and precision revealed that the accuracy and precision of complex manufacturing events was unpredictable for a CE Threshold of 50% or lower. However, when the CE Threshold was higher than 60% the accuracy and precision of detecting complex events increased with an increase in the CE Threshold.

In addition to the studies discussed above, we also conducted comparative studies to find out how the probabilistic CEP system would fare against a non-probabilistic CEP system. We compared the two systems firstly by varying the CE Thresholds and then by varying the physical reader accuracies. The results revealed that:

Effect of CE Thresholds:

- PR accuracy of 50%: The probabilistic CEP system was 400% more accurate and around 600% more precise than the non-probabilistic system when CE Threshold was 90%.
- PR accuracy of 70%: The probabilistic CEP system was 600% more accurate and around 400% more precise than non-probabilistic system when the CE Threshold was 90%
- PR accuracy of 90%: The probabilistic CEP system was almost 800% more accurate and precise than non-probabilistic system when the CE threshold was 90%.

Effect of PR Accuracy:

- CE Threshold of 90%: With a CE Threshold of 90%, the accuracy of complex event detections increased by roughly 20% for every 20% increase in the accuracy of underlying physical readers.
- CE Threshold of 50%: The results show that the accuracy of detecting complex events with pr 70 remained fairly constant at around 30% level. However, the accuracy of pr 50 initially swung wildly and then continued to drop before stabilizing at around 10% level. The experiments revealed that 50% is an extremely low threshold level to achieve predictable results.
- CE Threshold of 0%: The accuracy of detected complex events remained around 10% for physical reader accuracy of 90, 70 and 50. The results show that if we do not assign probabilities to complex events and trigger them once they have crossed

a certain accuracy threshold, the complex events can not be detected with any meaningful accuracy irrespective of how accurate the underlying physical readers are.

8.2 Future Work

There exists many ways to extend the work in this thesis or in general to perform future research in the area of real-time production monitoring. However, in the following sections, we would discuss three of the most pressing problems that still need to be solved. Section 8.2.1 discusses the issues with efficient dissemination of complex manufacturing events, Section 8.2.2 presents the problem of quality of service in a manufacturing environment that has deployed an event dissemination system and finally Section 8.2.3 presents the issue of production line fragmentation.

8.2.1 Efficient Dissemination of Complex Manufacturing Events

We undertook the task of detecting complex manufacturing events and assigning them probabilities, so as to know how reliable each and every event is (cf. Chapter 7). The events that we have detected so far include sequence, synchronization, delay, missing part and incorrect part position error. Currently all of these manufacturing events are generated during the production process at the shop floor. In the future however, we expect that a smart variant production environment would generate complex events across both operational domains (such as inventory, logistics etc) and business domains (such as HR, finance, etc).

In order to elaborate these future events lets discuss some concrete examples. Imagine an employee working on the production line and making repetitive errors. The production environment can define some rules to generate events which would inform the company about how good or bad a worker performed on the production line during the past hour, day, week or month. In case of excessive mistakes the worker could be pulled out of the production line immediately. The events generated for such mistakes would be similar to the yellow and red cards shown to footballers who make fouls or improper tackles on the football field. Such events would fall under the HR category and should be reported to HR department for preventive/corrective actions.

Another set of events that would potentially be generated include events that would inform the company that the inventory of a certain part is now at critical levels and new shipments of the parts must be ordered for smooth and uninterrupted continuation of production. Such events would belong to the inventory domain.

From the two examples discussed above, it is obvious that in the future a smart real-time production monitoring framework would generate complex events that would belong to specific domains i.e. events should have specific recipients and hence should

not be flooded to the entire system. In order to achieve this task, the smart real-time production monitoring framework must have a proper and efficient event dissemination mechanism. Publish/Subscribe is the paradigm of choice for efficient and reliable event/message dissemination. In a publish/subscribe system, the sender (publisher) of the event does not send the event directly to the receiver (subscriber). The generated events are categorized into classes, with the publisher having no knowledge about the number or type of subscribers that would receive these events. Similarly, the subscribers register to receive events belonging to a certain class, without having any knowledge of what and who generated those events.

Jin et al. [JZL+09] have already proposed using publish/subscribe to disseminate raw RFID events. However, the issue with their work is that it does not cater to production environments and hence does not provide for the fact that since RFID readings could be erroneous, a lot of incorrect events would be generated. So, although publish/subscribe mechanism is used to avoid flooding events to all applications Jin et al's system still floods applications with incorrect events.

We propose that complex manufacturing events should first be assigned probabilities in order to filter out the incorrect events and then be disseminated using a publish/subscribe system in order to ensure that events are only sent to the applications that want and are supposed to get those events.

8.2.2 QoS Requirements for Complex Manufacturing Events

From the discussion in the previous section it is obvious that we would need a publish/subscribe based system to efficiently disseminate complex manufacturing events across different operational and administrative domains. In a manufacturing environment information should not just be generated, but should be generated as quickly as possible and as reliably as possible so as to rectify the problems before they could become catastrophic issues. This time critical and reliability centric nature of manufacturing environments would pose certain quality of service requirements on the publish/subscribe system that would be deployed to address event dissemination.

The issue with publish/subscribe systems is that they are inherently decoupled - i.e. producers and receivers are not directly connected with each other. This poses serious research questions for any system that uses publish/subscribe as its core infrastructure and yet is tasked to disseminate events in a timely manner. Due to this reason, we believe that there is a serious need to develop algorithms to address QoS requirements - with special emphasis towards QoS requirements (such as end-to-end delay and security and confidentiality of events being disseminated) that would be relevant to manufacturing organizations.

8.2.3 Fragmentation on the Production Lines

It is very typical to have errors during production. Some of the common errors that have already been discussed in this work include: sequence, synchronization, delay, missing parts and incorrect part position errors. The only way to resolve these errors is to remove the products that have experienced errors from the production line in order to avoid completion of incorrect final products. However, removal of the faulty products from the production line is only a partial solution as the product that has been removed still needs to be manufactured and delivered to the customer. Due to this reason, once the error with the product is resolved, it should be re-scheduled and re-inducted onto the production line.

A very simplistic solution is to re-schedule these products for the next day/batch of production. This is the current state of the art in manufacturing, as companies re-schedule the faulty products for the next round of production. However, once a product is removed from the production line, a gap is created on the production line. This is a real physical gap and is both temporal and spatial in nature. In order to explain this, lets imagine a scenario in which three product parts were moving on the production line (o_1, o_2, o_3). After sometime, the smart real-time production monitoring framework realizes that o_2 is faulty and hence removes it from the production line. Due to this reason, o_2 is pulled off from the line and all its corresponding parts moving on any other lines would also be pulled off. Now the state of the production line would be $(o_1, , o_3)$ i.e. the production line would have product part o_1, a gap in place of where o_2 would have been and o_3. After the workers at a certain assembly point have worked on o_1, they would wait for o_3 to reach the assembly point in order to work on it. This idle time would have previously been spent while working on o_2.

From the example presented above it is obvious that each product that is pulled off the assembly line only saves the factory the hassle of building that product from scratch. The factory still losses out on precious time that could have been spent building that product. Each product pulled from the production line is equal to a product that the factory would never be able to manufacture. This is clearly a waste of precious time and resources for production environments. In our discussions with the manufacturing people, they expressed extreme desire for a solution to this problem i.e. for some sort of a mechanism to have re-induction of faulty product parts onto the production lines during the normal course of production.

One way to address this issue is to frame it as a fragmentation and de-fragmentation problem of the file system domain. In file-systems, the term fragmentation is used to describe the inability of the file system to lay out related data sequentially. Due to fragmentation, the disk head movement or seeks would increase which would result in a decreased throughput. Many methods have been proposed to solve fragmentation in file systems [JIQ96], [AM04], [Dav95], [Dav98] , however these methods are inapplicable for fragmentation on the production lines due to the simple fact that fragments on the

production lines are temporal and spatial in nature i.e. they move over time across the production lines. So although the problem that we face in production environments can be framed as a fragmentation problem, we would need new and innovative solutions to solve these fragmentation issues on production lines.

Bibliography

[AEMGG+05] Michael Abd-El-Malek, Gregory R. Ganger, Garth R. Goodson, Michael K. Reiter, and Jay J. Wylie. Fault-scalable byzantine fault-tolerant services. *SIGOPS Oper. Syst. Rev.*, 39(5):59–74, October 2005.

[AKFR07] Nova Ahmed, Rajnish Kumar, Robert Steven French, and Umakishore Ramachandran. Rf2id: A reliable middleware framework for rfid deployment. In *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, pages 1–10. IEEE, 2007.

[AM04] Yutaka Arakawa and Seiji Mizukoshi. Apparatus and method for defragmentation in disk storage system, August 17 2004. US Patent 6,779,081.

[AMK+07] Kanik Arora, Hugo Mallinson, Anand Kulkarni, James Brusey, and Duncan McFarlane. The practical feasibility of using rfid in a metal environment. In *Wireless Communications and Networking Conference, 2007. WCNC 2007. IEEE*, pages 1679–1683. IEEE, 2007.

[AMQ13] Pierre-Louis Aublin, Sonia Ben Mokhtar, and Vivien Quéma. Rbft: Redundant byzantine fault tolerance. In *Distributed Computing Systems (ICDCS), 2013 IEEE 33rd International Conference on*, pages 297–306. IEEE, 2013.

[BBQ+07] Roberto Baldoni, Roberto Beraldi, Vivien Quema, Leonardo Querzoni, and Sara Tucci-Piergiovanni. Tera: topic-based event routing for peer-to-peer architectures. In *Proceedings of the 2007 inaugural international conference on Distributed event-based systems*, pages 2–13. ACM, 2007.

[Ben09] BenjaminGau. itrame lernfabrik aie, 20009.

[BGK+07] Lejla Batina, Jorge Guajardo, Tim Kerins, Nele Mentens, Pim Tuyls, and Ingrid Verbauwhede. Public-key cryptography for rfid-tags. In *Pervasive Computing and Communications Workshops, 2007. PerCom Workshops' 07. Fifth Annual IEEE International Conference on*, pages 217–222. IEEE, 2007.

[BGK11] Michael Backes, Thomas R Gross, and Guenter Karjoth. Tag identification system, August 30 2011. US Patent 8,009,016.

[BJS04] Martin Bauer, Lamine Jendoubi, and Oliver Siemoneit. Smart factory–mobile computing in production environments. In *Proceedings of the MobiSys 2004 Workshop on Applications of Mobile Embedded Systems (WAMES 2004)*. Citeseer, 2004.

[BKS05] ZHEN Bin, Mamoru Kobayashi, and Masashi Shimizu. Framed aloha for multiple rfid objects identification. *IEICE Transactions on Communications*, 88(3):991–999, 2005.

[BL97] Alexander Brewer and Thomas L Landers. *Radio frequency identification: A survey and assessment of the technology*. University of Arkansas, 1997.

[BLE+07] TS Baines, Howard W Lightfoot, Steve Evans, Andy Neely, Richard Greenough, Joe Peppard, R Roy, Essam Shehab, A Braganza, Ashutosh Tiwari, et al. State-of-the-art in product-service systems. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 221(10):1543–1552, 2007.

[BM05] James Brusey and Duncan McFarlane. Towards robust part tracking in an automated manufacturing process using rfid. In *World Congress*, volume 16, pages 1481–1481, 2005.

[BMKL01] David L Brock, Timothy P Milne, Yun Y Kang, and Brendon Lewis. The physical markup language. *Auto-ID Center White Paper MIT-AUTOID-WH-003*, 2001.

[BP00] Paramvir Bahl and Venkata N Padmanabhan. Radar: An in-building rf-based user location and tracking system. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 775–784. Ieee, 2000.

[BR05a] Michel Baudin and Arun Rao. Rfid applications in manufacturing. *USA, Palo Alto: MMTI*, pages 1–12, 2005.

[BR05b] Leonid Bolotnyy and Gabriel Robins. Randomized pseudo-random function tree walking algorithm for secure radio-frequency identification. In *Automatic Identification Advanced Technologies, 2005. Fourth IEEE Workshop on*, pages 43–48. IEEE, 2005.

[Bro01] David L Brock. The electronic product code â a naming scheme for physical objects. *Auto-ID Center White Paper MIT-AUTOID-WH-002*, 2001.

[BWL06] Yijian Bai, Fusheng Wang, and Peiya Liu. Efficiently filtering rfid data streams. In *CleanDB*. Citeseer, 2006.

[BWL+07] Yijian Bai, Fusheng Wang, Peiya Liu, Carlo Zaniolo, and Shaorong Liu. Rfid data processing with a data stream query language. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pages 1184–1193. IEEE, 2007.

[Car12] Car2Go. Car2go: Overview for the city council of seattle, 2012.

[CCK04] Ho-Seung Choi, Jae-Ryon Cha, and Jae-Hyun Kim. Fast wireless anti-collision algorithm in ubiquitous id system. In *Vehicular Technology Conference, 2004. VTC2004-Fall. 2004 IEEE 60th*, volume 6, pages 4589–4592. IEEE, 2004.

[CI05] Vinton G Cerf and Robert E Icahn. A protocol for packet network intercommunication. *ACM SIGCOMM Computer Communication Review*, 35(2):71–82, 2005.

[CJV10] Chen Chen, Hans-Arno Jacobsen, and Roman Vitenberg. Divide and conquer algorithms for publish/subscribe overlay design. In *Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on*, pages 622–633. IEEE, 2010.

[CK05] Jae-Ryong Cha and Jae-Hyun Kim. Novel anti-collision algorithms for fast object identification in rfid system. In *Parallel and Distributed Systems, 2005. Proceedings. 11th International Conference on*, volume 2, pages 63–67. IEEE, 2005.

[CML+06] James Cowling, Daniel Myers, Barbara Liskov, Rodrigo Rodrigues, and Liuba Shrira. Hq replication: A hybrid quorum protocol for byzantine fault tolerance. In *Proceedings of the 7th Symposium on Operating Systems Design and Implementation*, OSDI '06, pages 177–190, Berkeley, CA, USA, 2006. USENIX Association.

[CMTV07] Gregory Chockler, Roie Melamed, Yoav Tock, and Roman Vitenberg. Spidercast: a scalable interest-aware overlay for topic-based pub/sub communication. In *Proceedings of the 2007 inaugural international conference on Distributed event-based systems*, pages 14–25. ACM, 2007.

[Con08] Daimler Mobility Concepts. Car2go, 2008.

[CRW01] Antonio Carzaniga, David S Rosenblum, and Alexander L Wolf. Design and evaluation of a wide-area event notification service. *ACM Transactions on Computer Systems (TOCS)*, 19(3):332–383, 2001.

[CS88] Israel Cidon and Moshe Sidi. Conflict multiplicity estimation and batch resolution algorithms. *Information Theory, IEEE Transactions on*, 34(1):101–110, 1988.

[CTCC03] Sean Clark, Ken Traub, Dipan Anarkat Uniform Code Council, and Ted Osinski Uniform Code Council. Auto-id savant specification 1.0 2. *Technical Report, Auto-Id Center*, 2003.

[CWA+09] Allen Clement, Edmund L Wong, Lorenzo Alvisi, Michael Dahlin, and Mirco Marchetti. Making byzantine fault tolerant systems tolerate byzantine faults. In *NSDI*, volume 9, pages 153–168, 2009.

[Dav95] William Davy. Method for eliminating file fragmentation and reducing average seek times in a magnetic disk media environment, March 14 1995. US Patent 5,398,142.

[Dav98] William Davy. Method for eliminating file fragmentation and reducing average seek times in a magnetic disk media environment, September 15 1998. US Patent 5,808,821.

[DBD04] Pradip De, Kalyan Basu, and Sajal K Das. An ubiquitous architectural framework and protocol for object tracking using rfid tags. In *Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004. The First Annual International Conference on*, pages 174–182. IEEE, 2004.

[DGH+06] Alan Demers, Johannes Gehrke, Mingsheng Hong, Mirek Riedewald, and Walker White. Towards expressive publish/subscribe systems. In *Advances in Database Technology-EDBT 2006*, pages 627–644. Springer, 2006.

[DGP+07] Alan J Demers, Johannes Gehrke, Biswanath Panda, Mirek Riedewald, Varun Sharma, Walker M White, et al. Cayuga: A general purpose event monitoring system. In *CIDR*, volume 7, pages 412–422, 2007.

[Dim05] Tassos Dimitriou. A lightweight rfid protocol to protect against traceability and cloning attacks. In *Security and Privacy for Emerging Areas in Communications Networks, 2005. SecureComm 2005. First International Conference on*, pages 59–66. IEEE, 2005.

[DK03] Michael J Dennis and Ajit Kambil. Service management: Building profits after the sale. *Supply Chain Management Review, V. 7, NO. 3 (JAN./FEB. 2003), P. 42-48: ILL*, 2003.

[EPC04] EPC EPCglobal. Radio-frequency identity protocols class-1 generation-2 uhf rfid protocol for communications at 860 mhz–960 mhz version 1.0. 9. *K. Chiew et al./On False Authenticationsfor C1G2 Passive RFID Tags*, 65, 2004.

[FJK+05] Michael J Franklin, Shawn R Jeffery, Sailesh Krishnamurthy, Frederick Reiss, Shariq Rizvi, Eugene Wu, Owen Cooper, Anil Edakkunni, and Wei Hong. Design considerations for high fan-in systems: The hifi approach. In *CIDR*, pages 290–304, 2005.

[FL05] Christian Floerkemeier and Matthias Lampe. Rfid middleware design: addressing application requirements and rfid constraints. In *Proceedings*

of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies, pages 219–224. ACM, 2005.

[Flo06] Christian Floerkemeier. Transmission control scheme for fast rfid object identification. In *Pervasive Computing and Communications Workshops, 2006. PerCom Workshops 2006. Fourth Annual IEEE International Conference on*, pages 6–pp. IEEE, 2006.

[For59] Ford. Ford motor credit company, 1959.

[For07] Henry Ford. *My life and work*. Cosimo, Inc., 2007.

[FW06] Christian Floerkemeier and Matthias Wille. Comparison of transmission schemes for framed aloha based rfid protocols. In *Applications and the Internet Workshops, 2006. SAINT Workshops 2006. International Symposium on*, pages 4–pp. IEEE, 2006.

[GFL87] Albert G Greenberg, Philippe Flajolet, and Richard E Ladner. Estimating the multiplicities of conflicts to speed their resolution in multiple access channels. *Journal of the ACM (JACM)*, 34(2):289–325, 1987.

[GKQV10] Rachid Guerraoui, Nikola Knežević, Vivien Quéma, and Marko Vukolić. The next 700 bft protocols. In *Proceedings of the 5th European conference on Computer systems*, pages 363–376. ACM, 2010.

[GM92] GM. General motors financial company, 1992.

[GSH07] Gary M Gaukler, Ralf W Seifert, and Warren H Hausman. Item-level rfid in the retail supply chain. *Production and Operations Management*, 16(1):65–76, 2007.

[GWC+06] Daniel Gyllstrom, Eugene Wu, Hee-Jin Chae, Yanlei Diao, Patrick Stahlberg, and Gordon Anderson. Sase: Complex event processing over streams. *CoRR*, abs/cs/0612128, 2006.

[HC06] Taimur Hassan and Samir Chatterjee. A taxonomy for rfid. In *System Sciences, 2006. HICSS'06. Proceedings of the 39th Annual Hawaii International Conference on*, volume 8, pages 184b–184b. IEEE, 2006.

[HCW08] K Harun, K Cheng, and M Wibbelmann. Rfid-enabled aerospace manufacturing: Theoretical models, simulation and implementation issues. In *Industrial Engineering and Engineering Management, 2008. IEEM 2008. IEEE International Conference on*, pages 1824–1829. IEEE, 2008.

[HKDR10] Bilal Hameed, Imran Khan, Frank Durr, and Kurt Rothermel. An rfid based consistency management framework for production monitoring in a smart real-time factory. In *Internet of Things (IOT), 2010*, pages 1–8. IEEE, 2010.

[HMW+11] Bilal Hameed, Jörge Minguez, Michael Wörner, Philip Hollstein, Sema Zor, Stefan Silcher, Frank Dürr, and Kurt Rothermel. The smart real-time factory as a product service system. In *Functional Thinking for Value Creation*, pages 326–331. Springer, 2011.

[HRDR12] Bilal Hameed, Farhan Rashid, Frank Dürr, and Kurt Rothermel. Self-calibration of rfid reader probabilities in a smart real-time factory. In *Pervasive Computing*, pages 253–270. Springer, 2012.

[HVBW01] Jeffrey Hightower, Chris Vakili, Gaetano Borriello, and Roy Want. Design and calibration of the spoton ad-hoc location sensing system. *unpublished, August*, 2001.

[HW98] Don R Hush and Cliff Wood. Analysis of tree algorithms for rfid arbitration. In *Information Theory, 1998. Proceedings. 1998 IEEE International Symposium on*, page 107. IEEE, 1998.

[HYHZ08] Wenhui Hu, Wei Ye, Yu Huang, and Shikun Zhang. Complex event processing in rfid middleware: A three layer perspective. In *Convergence and Hybrid Information Technology, 2008. ICCIT'08. Third International Conference on*, volume 1, pages 1121–1125. IEEE, 2008.

[HZJ07] George Q Huang, YF Zhang, and PY Jiang. Rfid-based wireless manufacturing for walking-worker assembly islands with fixed-position layouts. *Robotics and Computer-Integrated Manufacturing*, 23(4):469–477, 2007.

[HZJ08] George Q Huang, YF Zhang, and PY Jiang. Rfid-based wireless manufacturing for real-time management of job shop wip inventories. *The International Journal of Advanced Manufacturing Technology*, 36(7-8):752–764, 2008.

[IAM09] Alexander Ilic, Thomas Andersen, and Florian Michahelles. Increasing supply-chain visibility with rule-based rfid data analysis. *Internet Computing, IEEE*, 13(1):31–38, 2009.

[Ina09] Tatsuya Inaba. Inference of product quality by using rfid-enabled traceability information a study on the us pharmaceutical supply chain. In *RFID, 2009 IEEE International Conference on*, pages 298–305. IEEE, 2009.

[Inc11] MarketsandMarkets Research Inc. Chipless rfid market (2011 - 2016) - global forecasts by applications (retail, supply chain, aviation, healthcare, smart card, public transit & others). November 2011.

[JCL+10] Hans-Arno Jacobsen, Alex King Yeung Cheung, Guoli Li, Balasubramaneyam Maniymaran, Vinod Muthusamy, and Reza Sherafat Kazemzadeh. The padres publish/subscribe system., 2010.

[JIQ96] Richard P Jernigan IV and Scott D Quinn. Two-pass defragmentation of compressed hard disk data with a single data rewrite, November 12 1996. US Patent 5,574,907.

[JMJV09] Márk Jelasity, Alberto Montresor, Gian Paolo Jesi, and Spyros Voulgaris. Peersim: A peer-to-peer simulator. *URL: http://peersim. sourceforge. net*, 2009.

[JNPL09] Rasmus Jacobsen, Karsten Fyhn Nielsen, Petar Popovski, and Torben Larsen. Reliable identification of rfid tags using multiple independent reader sessions. In *RFID, 2009 IEEE International Conference on*, pages 64–71. IEEE, 2009.

[Joh02] Dick Johnson. Rfid tags improve tracking, quality on ford line in mexico. *Control Engineering*, 49(11):16, 2002.

[Jon06] Peter Jones. Networked rfid for use in the food chain. In *Emerging Technologies and Factory Automation, 2006. ETFA'06. IEEE Conference on*, pages 1119–1124. IEEE, 2006.

[Jue05] Ari Juels. Minimalist cryptography for low-cost rfid tags. In *Security in Communication Networks*, pages 149–164. Springer, 2005.

[JZL+09] Beihong Jin, Xinchao Zhao, Zhenyue Long, Fengliang Qi, and Shuang Yu. Effective and efficient event dissemination for rfid applications. *Comput. J.*, 52(8):988–1005, November 2009.

[KAD+10] Ramakrishna Kotla, Lorenzo Alvisi, Mike Dahlin, Allen Clement, and Edmund Wong. Zyzzyva: Speculative byzantine fault tolerance. *ACM Trans. Comput. Syst.*, 27(4):7:1–7:39, January 2010.

[KAE00] Satish Kumar, C Alaettinglu, and Deborah Estrin. Scalable object-tracking through unattended techniques (scout). In *Network Protocols, 2000. Proceedings. 2000 International Conference on*, pages 253–262. IEEE, 2000.

[KCN07] Sang-Hyun Kim, Morris A Cohen, and Serguei Netessine. Performance contracting in after-sales service supply chains. *Management Science*, 53(12):1843–1858, 2007.

[Kho11] D Khodawandi. Principal of porsche consulting, 2011.

[KSCB03] Robin Koh, E Schuster, Indy Chackrabarti, and Attilio Bellman. Securing the pharmaceutical supply chain. *White Paper, Auto-ID Labs, Massachusetts Institute of Technology*, 2003.

[LCW08] Dominik Lucke, Carmen Constantinescu, and Engelbert Westkämper. Smart factory-a step towards the next generation of manufacturing. In *Manufacturing Systems and Technologies for the New Frontier*, pages 115–118. Springer, 2008.

[LEWM05] K Lee, John C Eidson, Hans Weibel, and Dirk Mohl. Ieee 1588-standard for a precision clock synchronization protocol for networked measurement and control systems. In *Conference on IEEE*, volume 1588, 2005.

[LJL05] Su-Ryun Lee, Sung-Don Joo, and Chae-Woo Lee. An enhanced dynamic framed slotted aloha algorithm for rfid tag identification. In *Mobile and Ubiquitous Systems: Networking and Services, 2005. MobiQuitous 2005. The Second Annual International Conference on*, pages 166–172. IEEE, 2005.

[LLGF08] Zhekun Li, Fuyu Li, Lei Gao, and Yujing Fan. Concurrent intelligent manufacturing based on rfid. In *Advanced Design and Manufacture to Gain a Competitive Edge*, pages 521–530. Springer, 2008.

[LLS00] Ching Law, Kayi Lee, and Kai-Yeung Siu. Efficient memoryless protocol for tag identification. In *Proceedings of the 4th international workshop on Discrete algorithms and methods for mobile computing and communications*, pages 75–84. ACM, 2000.

[LM06] Fagui Liu and Zhaowei Miao. The application of rfid technology in production control in the discrete manufacturing industry. In *Video and Signal Based Surveillance, 2006. AVSS'06. IEEE International Conference on*, pages 68–68. IEEE, 2006.

[LMF09] Mikko Lehtonen, Florian Michahelles, and Elgar Fleisch. How to detect cloned tags in a reliable way from incomplete rfid traces. In *RFID, 2009 IEEE International Conference on*, pages 257–264. IEEE, 2009.

[LW09] Tianbao Li and Dong Wang. Experimental studying measurement metrics of rfid system performance. In *Anti-counterfeiting, Security, and Identification in Communication, 2009. ASID 2009. 3rd International Conference on*, pages 233–237. IEEE, 2009.

[LWG08] Bonnie Latham, Jeremy Wright, and Jesse Green. Rfid solution to manufacturing process. In *Systems and Information Engineering Design Symposium, 2008. SIEDS 2008. IEEE*, pages 340–343. IEEE, 2008.

[MB76] Robert M Metcalfe and David R Boggs. Ethernet: distributed packet switching for local computer networks. *Communications of the ACM*, 19(7):395–404, 1976.

[Mil91] David L Mills. Internet time synchronization: the network time protocol. *Communications, IEEE Transactions on*, 39(10):1482–1493, 1991.

[ML05] Jihoon Myung and Wonjun Lee. An adaptive memoryless tag anticollision protocol for rfid networks. In *IEEE ICC*, 2005.

[Moc83] Paul V Mockapetris. Domain names: Implementation specification. 1983.

[MZV07] Tova Milo, Tal Zur, and Elad Verbin. Boosting topic-based publish-subscribe systems with dynamic clustering. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 749–760. ACM, 2007.

[NC05] Mamatha Nanjundaiah and Vipin Chaudhary. Improvement to the anticollision protocol specification for 900mhz class 0 radio frequency identification tag. In *Advanced Information Networking and Applications, 2005. AINA 2005. 19th International Conference on*, volume 2, pages 616–620. IEEE, 2005.

[NCL+07] EWT Ngai, TCE Cheng, Kee-hung Lai, PYF Chai, YS Choi, and RKY Sin. Development of an rfid-based traceability system: Experiences and lessons learned from an aircraft engineering company. *Production and Operations Management*, 16(5):554–568, 2007.

[NLLP04] Lionel M Ni, Yunhao Liu, Yiu Cho Lau, and Abhishek P Patil. Landmarc: indoor location sensing using active rfid. *Wireless networks*, 10(6):701–710, 2004.

[PFP04] Petar Popovski, Frank HP Fitzek, and Ramjee Prasad. Batch conflict resolution algorithm with progressively accurate multiplicity estimation. In *Proceedings of the 2004 joint workshop on Foundations of mobile computing*, pages 31–40. ACM, 2004.

[PLHCTR09] Pedro Peris-Lopez, Julio Cesar Hernandez-Castro, Juan ME Tapiador, and Arturo Ribagorda. Advances in ultralightweight cryptography for low-cost rfid tags: Gossamer protocol. In *Information Security Applications*, pages 56–68. Springer, 2009.

[Pro03] A Prodromou. Tesco deploys class 1 epc tags. *RFID Journal, http://www.rfidjournal.com/article/view/587*, 2003.

[PSB03] Peter R. Pietzuch, Brian Shand, and Jean Bacon. A framework for event composition in distributed systems. In *Proceedings of the ACM/IFIP/USENIX 2003 International Conference on Middleware*, Middleware '03, pages 62–82, New York, NY, USA, 2003. Springer-Verlag New York, Inc.

[PSR+06] BS Prabhu, Xiaoyong Su, Harish Ramamurthy, Chi-Cheng Chu, and Rajit Gadh. Winrfid: a middleware for the enablement of radiofrequency identification (rfid)-based applications. *Mobile, Wireless, and Sensor Networks, Wiley-Interscience*, pages 313–336, 2006.

[PTMS07] Béla Pátkai, Lila Theodorou, Duncan McFarlane, and Kyle Schmidt. Requirements for rfid-based sensor integration in landing gear monitoring–a case study. *Power*, 5:4, 2007.

[Rel12] Press Release. Rolls-royce celebrates 50th anniversary of power-by-the-hour, 2012.

[Rep12] Tech Report. What that car really costs to own, 2012.

[Rep13] Web Report. Car2go car club expands london service, May 2013.

[Rif13] Philipp Riffelmacher. Konzeption einer lernfabrik für die variantenreiche montage. 2013.

[RMA+08] Nilo Rivera, Rosemary Mountain, Lia Assumpcao, Allen A Williams, AB Cooper, Douglas L Lewis, Richard C Benson, Joseph A Miragliotta, Mike Marohn, and Russell H Taylor. Assist-automated system for surgical instrument and sponge tracking. In *RFID, 2008 IEEE International Conference on*, pages 297–302. IEEE, 2008.

[Rob05] Mark Roberti. Epc reduces out-of-stocks at wal-mart. *RFID Journal. Retrieved March*, 19:2009, 2005.

[RR02] Zhigang Rong and Theodore S Rappaport. Wireless communications: Principles and practice. *Prentice Hall*, 2002.

[RZHJ07] Ahmad Rahmati, Lin Zhong, Matti Hiltunen, and Rittwik Jana. Reliability techniques for rfid-based object tracking applications. In *Dependable Systems and Networks, 2007. DSN'07. 37th Annual IEEE/IFIP International Conference on*, pages 113–118. IEEE, 2007.

[SAB+00] Bill Segall, David Arnold, Julian Boot, Michael Henderson, and Ted Phelps. Content based routing with elvin4. In *Proceedings of AUUG2K*, 2000.

[Sch38] Zoe Emily Schnabel. The estimation of total fish population of a lake. *The American Mathematical Monthly*, 45(6):348–352, 1938.

[Sch83] Frits Schoute. Dynamic frame length aloha. *Communications, IEEE Transactions on*, 31(4):565–568, 1983.

[SD03] Dieter Spath and Lutz Demuß. Entwicklung hybrider produkte gestaltung materieller und immaterieller leistungsbundel. In *Service Engineering*, pages 467–506. Springer, 2003.

[Seb82] George Arthur Frederick Seber. *The estimation of animal abundance*. Griffin, 1982.

[SM08] Boyeon Song and Chris J Mitchell. Rfid authentication protocol for low-cost tags. In *Proceedings of the first ACM conference on Wireless network security*, pages 140–147. ACM, 2008.

[SMMP09] Nicholas Poul Schultz-Møller, Matteo Migliavacca, and Peter Pietzuch. Distributed complex event processing with query rewriting. In *Proceedings of the Third ACM International Conference on Distributed Event-Based Systems*, page 4. ACM, 2009.

[SRSMN06] David Simplot-Ryl, Ivan Stojmenovic, Aleksandar Micic, and Amiya Nayak. A hybrid randomized protocol for rfid tag identification. *Sensor Review*, 26(2):147–154, 2006.

[TB06] Pim Tuyls and Lejla Batina. Rfid-tags for anti-counterfeiting. In *Topics in Cryptology–CT-RSA 2006*, pages 115–131. Springer, 2006.

[TKK+11] Muhammad Adnan Tariq, Boris Koldehofe, Gerald G Koch, Imran Khan, and Kurt Rothermel. Meeting subscriber-defined qos constraints in publish/subscribe systems. *Concurrency and Computation: Practice and Experience*, 23(17):2140–2153, 2011.

[TKKR12] Muhammad Adnan Tariq, Boris Koldehofe, Gerald G Koch, and Kurt Rothermel. Distributed spectral cluster management: A method for building dynamic publish/subscribe systems. In *Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems*, pages 213–224. ACM, 2012.

[Toy00] Toyota. Toyota financial services corporation, 2000.

[Tsu88] Paul F Tsuchiya. The landmark hierarchy: A new hierarchy for routing in very large networks. In *ACM SIGCOMM Computer Communication Review*, volume 18, pages 35–42. ACM, 1988.

[TWLW08] Jie Tan, Hongwei Wang, Dan Li, and Qigang Wang. A rfid architecture built in production and manufacturing fields. In *Convergence and Hybrid Information Technology, 2008. ICCIT'08. Third International Conference on*, volume 1, pages 1118–1120. IEEE, 2008.

[Vio05] Bob Violino. The basics of rfid technology. *RFID Journal, http://www.rfidjournal.com/article/articleview/1337/1/129/*, 2005.

[Vog02] Harald Vogt. Efficient object identification with passive rfid tags. In *Pervasive Computing*, pages 98–113. Springer, 2002.

[Vol94] VolksWagon. Volkswagen financial services ag, 1994.

[Vol13] VolareInc. Volare rfid readers, May 2013.

[VPR09] John K Visich, John T Powers, and Christopher J Roethlein. Empirical applications of rfid in the manufacturing environment. *International Journal of Radio Frequency Identification Technology and Applications*, 2(3):115–132, 2009.

[WDR06] Eugene Wu, Yanlei Diao, and Shariq Rizvi. High-performance complex event processing over streams. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, SIGMOD '06, pages 407–418, New York, NY, USA, 2006. ACM.

[WFCBB08] Leonardo Weiss Ferreira Chaves, Erik Buchmann, and Klemens Böhm. Tagmark: Reliable estimations of rfid tags for business processes. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 999–1007. ACM, 2008.

[WHC06] Kirk HM Wong, Patrick CL Hui, and Allan CK Chan. Cryptography and authentication on rfid passive tags for apparel products. *Computers in Industry*, 57(4):342–349, 2006.

[WLLB06] Fusheng Wang, Shaorong Liu, Peiya Liu, and Yijian Bai. Bridging physical and virtual worlds: complex event processing for rfid data streams. In *Advances in Database Technology-EDBT 2006*, pages 588–607. Springer, 2006.

[WLWT07] Jiahao Wang, Zongwei Luo, Edward C Wong, and CJ Tan. Rfid assisted object tracking for automating manufacturing assembly lines. In *e-Business Engineering, 2007. ICEBE 2007. IEEE International Conference on*, pages 48–53. IEEE, 2007.

[YSM05] Kok-Kiong Yap, Vikram Srinivasan, and Mehul Motani. Max: human-centric search of the physical world. In *Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 166–179. ACM, 2005.

[ZCJ+04] Feng Zhou, Chunhong Chen, Dawei Jin, Chenling Huang, and Hao Min. Evaluating and optimizing power consumption of anti-collision protocols for applications in rfid systems. In *Proceedings of the 2004 international symposium on Low power electronics and design*, pages 357–362. ACM, 2004.

[ZGP04] Li Zhekun, Rajit Gadh, and BS Prabhu. Applications of rfid technology and smart parts in manufacturing. In *Proceedings of DETC*, volume 4, pages 1–7. Citeseer, 2004.

[Zim80] Hubert Zimmermann. Osi reference model–the iso model of architecture for open systems interconnection. *Communications, IEEE Transactions on*, 28(4):425–432, 1980.

[ZKS04] Bin Zhen, Mamoru Kobayashi, and Masashi Shimizu. To read transmitter-only rfid tags with confidence. In *Personal, Indoor and Mobile Radio Communications, 2004. PIMRC 2004. 15th IEEE International Symposium on*, volume 1, pages 396–400. IEEE, 2004.

[ZZ08] Guangqian Zhang and Li Zhang. Study of cep-based rfid data processing model. In *Intelligent Information Technology Application, 2008. IITA'08. Second International Symposium on*, volume 3, pages 254–258. IEEE, 2008.