Expressive Content-Based Routing in Software-Defined Networks

Sukanya Bhowmik, Muhammad Adnan Tariq, Jonas Grunert, Deepak Srinivasan, and Kurt Rothermel

University of Stuttgart, {first name.last name}@ipvs.uni-stuttgart.de

***Abstract*—With the vision of Internet of Things gaining popularity at a global level, efficient publish/subscribe middleware for communication within and across data centers is extremely desirable. In this respect, the very popular Software-Defined Networking, which enables publish/subscribe middleware to perform line-rate filtering of events directly on hardware, can prove to be very useful. While deploying content filters directly on switches of a software-defined network allows optimized paths, high throughput rates, and low end-to-end latency, it suffers from certain inherent limitations with respect to number of bits available on hardware switches to represent these filters. Such a limitation affects expressiveness of filters, resulting in unnecessary traffic in the network.**

In this paper, we explore various complementary techniques to represent content filters expressively while being limited by hardware. We implement and evaluate techniques that i) use workload, in terms of events and subscriptions, to represent content, and ii) efficiently select attributes to reduce redundancy in content. Our detailed performance evaluations show the potential of these techniques in reducing unnecessary traffic when subjected to different workloads. Furthermore, the techniques proposed in this paper require significant updates to the network, i.e., the data plane, which must be performed in a consistent manner to ensure desired system behavior. As a result, in this paper, we, also, design and evaluate a light-weight approach that ensures data plane consistency in the presence of dynamic network updates.

***Index Terms*—Content-based Routing, Software-Defined Networking, bandwidth efficiency, hardware limitations**

I. Introduction

The Internet of Things (IoT) has brought with it a global wave that envisions a future which can seamlessly connect digital and physical objects with the use of suitable technologies. This vision is being aptly complemented with the fast progress in sensors, actuators, cloud computing, and technologies for efficient and transparent communication. For suitable communication in IoT, global cloud providers already offer the very popular publish/subscribe (pub/sub) communication pattern. Pub/sub, especially content-based pub/sub, allows loosely coupled producers of content (i.e., publishers) and consumers of published content (i.e., subscribers) to interact transparently in a bandwidth-efficient manner. Subscribers express specific interests which are then used to install filters on content-based routers between publishers and subscribers, ensuring the dissemination of only relevant content to each subscriber. Pub/sub forms the backbone of data centers powering the IoT vision ahead. For example, Google uses Cloud Pub/Sub to 'connect anything to everything' in an IoT environment. Also, Microsoft uses Azure Event Hubs, a highly scalable pub/sub service to connect devices and applications across IoT platforms.

In recent times, the foundation of cloud computing has been influenced by Software-Defined Networking (SDN). In fact, for almost a decade, Google has been exploiting the benefits of SDN to power Google's data center (DC) WAN, B4 [19]. Microsoft, too, has been using SDN to flexibly and reliably operate Microsoft Azure [3]. The advantage of a network architecture like SDN is that it enables software to flexibly configure the network. SDN allows the extraction of all control logic from hardware switches and hosts them on a logically centralized controller, thus establishing a clear separation between the control plane and the data (forwarding) plane. The controller has an integrated view of the network and can flexibly configure it in a resource-efficient manner with the help of popular standards like Openflow [14]. While SDN has been extensively considered for dynamic resource sharing, WAN VPN, etc., across data centers, the potential of SDN to realize content-based pub/sub, the backbone of data center communication, has also recently been explored in literature [9], [35]. Traditionally, content-based pub/sub systems are implemented as an overlay network of software brokers (e.g., [32], [12], [18]). Implemented in the application layer, their performance is still far behind the performance of communication protocols implemented on the network layer with respect to throughput, end-to-end latency, etc. This is because these middleware implementations are unable to exploit the performance benefits of standard multilayer switches or hardware routers capable of forwarding packets at line-rate and achieving data rates of 10 Gbps and more using dedicated hardware such as Ternary Content Addressable Memory (TCAM). While IP multicast trees might be enough to cater to the needs of topic-based pub/sub systems, expressive content-based pub/sub systems have far more demanding requirements [34], [36]. As a result, recently, the capabilities of SDN have been used to realize the PLEROMA [9], [35] middleware that enables in-network content-based filtering of published events directly on SDN-compliant switches. Since the logically centralized controller has a global view of the underlying topology, it is the controller that establishes optimized paths between publishers and relevant subscribers by installing content filters directly on the switches along these paths, thus enabling line-rate filtering and forwarding of events.

To show the performance gains, in terms of end-to-end latency of events, of PLEROMA as compared to a state-of-the-art overlay-based middleware solution involving software filtering of events, we refer to Figure 1. Please note that the soft-

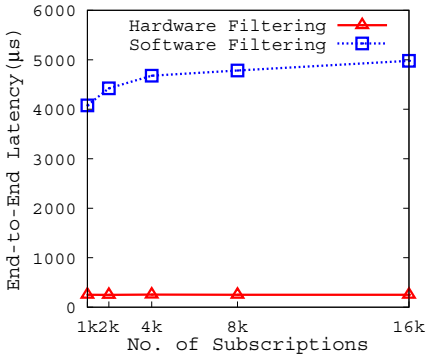


Fig. 1: Comparing PLEROMA with state-of-the-art [8]

ware filtering is implemented as a very efficient parallelized matching pub/sub service that enables one-hop forwarding of events similar to a state-of-the-art pub/sub system called Bluedove [26]. Figure 1 clearly shows that hardware filtering in PLEROMA is significantly faster than software filtering in a state-of-the-art middleware implementation (cf. [8], Section 7). Note that end-to-end latency of events in other state-of-the-art pub/sub systems, that perform filtering in software, is similar to that of Bluedove and is in the order of several milliseconds to seconds [5], [28].

Even though the dissemination of content in a software-defined network occurs at line-rate, nevertheless, content-based pub/sub using SDN suffers from certain inherent limitations that result in bandwidth wastage. It should be noted that the effectiveness of content-based routing relies heavily on the expressiveness of content filters which are responsible for filtering out unnecessary traffic to ensure bandwidth-efficient communication. In an SDN-based pub/sub, these content filters are represented by the match fields of flows in TCAM of switches. This implies that content filters are limited by the bits available for filter representation at the selected match field (e.g., IPv6 address, VLAN tag). For instance, the choice of the destination IPv6 address to represent content filters allows a maximum of 128 bits which in reality would further reduce as the entire range of IP addresses may be shared among multiple applications. Moreover, IPv6 is not widely deployed and the use of IPv4 addresses instead can further impede the expressiveness of filters. Jokela et al., in LIPSIN [22], also target filtering on hardware in the context of topic-based pub/sub by encoding forwarding paths in packet headers. However, for a considerably small topology, even the use of a staggering 248 bits in the packet header does not suffice to prevent unnecessary traffic in the system ($\sim$10%).

The above limitations may significantly impact bandwidth usage—something that is truly critical in a cloud environment, where the network can pose to be a significant bottleneck [24]. As a result, in this paper, we significantly extend a previous publication [7] to provide a comprehensive exploration of techniques that address concerns with bandwidth efficiency in the context of content-based filtering on hardware switches. First, we propose two techniques—selective indexing and adaptive spatial indexing—that consider workload in the system in terms of events and subscriptions to expressively map content to match fields of flows on hardware switches. Then, we present algorithms with varying complexities to efficiently identify and neglect redundant attributes or dimensions in the content-space such that more bits are available to express more meaningful attributes in content filters. Moreover, these techniques complement each other and may be combined for enhanced effectiveness. The techniques proposed in this paper result in significant updates to the network, i.e., the data plane, which must be performed in a consistent manner to ensure the desired behavior of the system. Therefore, the contributions of this paper also include the design of a light-weight approach that ensures data plane consistency in the presence of dynamic network updates when each of the proposed techniques is employed. Our evaluations show that a significant amount of irrelevant traffic (up to 97%) can be avoided by employing each of the techniques, designed to improve bandwidth efficiency, independently or in combination while benefiting from the advantages of SDN in terms of reduced end-to-end latency, high throughput, etc. Moreover, evaluation results confirm that the designed light-weight approach preserves data plane consistency during dynamic network updates and also manages to do so in a more resource-efficient manner as compared to state-of-the-art solutions.

II. Preliminaries and Limitations

In this section, we provide an overview of PLEROMA [35], [9], a content-based pub/sub middleware realized on SDN, followed by a discussion on the limitations it faces.

A. The PLEROMA Middleware

A content-based in-network filtering solution using SDN, such as PLEROMA, follows the same principles of the pub/sub paradigm which consists of two participants—publisher and subscriber. In PLEROMA, a publisher sends an advertisement to the controller of the software-defined network to specify the content it intends to publish. Similarly, a subscriber specifies the content it is interested in receiving by sending a subscription to the controller. Based on these advertisements and subscriptions, along with the global view of the physical network, the controller installs content filters represented by match fields of flow table entries (in this paper, we refer to flow entries as flows) on TCAM of switches along optimized paths between publishers and their interested subscribers. For example, in Figure 2, the publisher P and the subscriber S_1 send an advertisement and a subscription, respectively, to the controller which installs content filters (match fields on flows) along the path between them. Similarly, events are represented by header fields (corresponding to the selected match fields on flows) in the packet header. This enables header-based matching of event packets directly on TCAM of hardware switches, resulting in line-rate performance.

Content representation follows a content-based subscription model where published events are attribute-value pairs and advertisements and subscriptions (i.e., content filters) are conjunctions of filters on these attributes. An event matches a subscription only if it lies within the range of values of a subscription along each attribute. Therefore, in order to map events and subscriptions of such a model to packet header

fields and match fields of flows on switches, we need to, first, convert them to binary form. One could argue that since the controller is aware of all subscriptions in the system, it could assign each subscription a bit string and these bit strings could be attached to each event so that it can be forwarded to interested subscribers. However, in order to do so, (i) either each event will have to be sent to the control plane such that the controller can attach the necessary bit strings to the events, or (ii) the mapping between all subscriptions and their bit strings has to be known to every publisher. Case (i) is not an option as sending events to the control plane will incur additional latency in forwarding of events which will no longer happen purely on the network layer. As for case (ii), all publishers need to have the information of all subscriptions and their mapping, because when publishing an event, a publisher needs to send packets for that event to all subscribers interested in receiving it. This is of course not desirable in a content-based publish/subscribe system and impacts space decoupling between publishers and subscribers. Moreover, in such a scheme, a publisher will have to send multiple event packets, each represented by a bit string corresponding to a matching subscription, for the same event. This will significantly increase the bandwidth usage of the network and is of course not desirable. Therefore, we need to design a scheme that allows a publisher to encode information of all interested subscribers in a single event packet.

From the above discussion, we derive the requirements for mapping of content in an SDN-based pub/sub middleware. Firstly, events should be directly filtered and forwarded by switches on the data plane in order to achieve line-rate performance. Please note that switches only support matching operations (including prefix-based matching) at the match field of flows. As a result, the content mapping scheme should ensure that if an event matches a subscription, i.e., the event is contained by the subscription, then the binary strings of the subscription and the event should reflect this containment relation such that matching (more specifically prefix-based matching) of events is possible at the TCAM of switches.

Secondly, please note that in our system we must ensure the avoidance of false negatives. **False negatives** are those events that are not delivered to subscribers despite their interest in receiving them. As a result, the controller must install necessary content filters along all switches along the entire path between a publisher and its relevant subscriber. The content mapping scheme must ensure that no false negatives are introduced in the system.

Finally, in order to provide a bandwidth-efficient solution, a requirement to reuse paths in the network while forwarding events becomes necessary. This implies that if two subscriptions have similar interests, i.e., one is contained by the other in terms of the ranges of attributes comprising the subscriptions, then multiple copies of the same event should not be sent over the same links in order to reach the two subscribers. Instead, a single event packet matching a filter that represents related subscriptions should be forwarded over a link that is shared by paths to the two subscribers. This implies that containment relations between subscriptions must be preserved during content mapping.

Of all the available techniques for converting content into

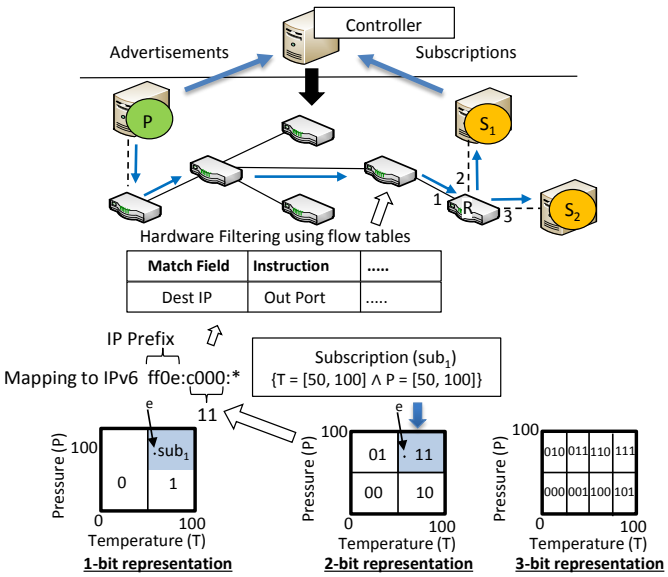


Fig. 2: SDN-based Pub/Sub Middleware

binary form, the above three requirements are fulfilled by the widely used spatial indexing technique [37], [28], [17]. As a result, PLEROMA, specifically, uses spatial indexing for content representation. In spatial indexing, an event-space (denoted by Ω) is modeled as an ω-dimensional space where each dimension represents a content attribute. Recursive binary decomposition of Ω generates regular subspaces that serve as enclosing approximations for advertisements, subscriptions, and events which are represented by binary strings known as dzs. Please note that the recursive binary decomposition of Ω is done along each attribute in a round-robin manner. This results in bits from different dimensions being interleaved in the resulting dz. Here, we look at an example from Figure 2 where the 2-dimensional event-space is first divided along the dimension *temperature* to obtain a dz of 1 bit. For a 2-bit representation, Ω is divided this time along *pressure*, and for a 3-bit representation, Ω is again divided along the first dimension *temperature*, and this continues till the closest approximation of a subspace, that needs to be represented by a dz, is obtained. This enables a subscription to introduce filtering on each dimension from the very first bits of the dz. With more and more bits, the filtering gets even more fine-grained on each dimension. To understand this better, we present certain characteristic properties of dzs based on the subspaces they represent. For example, the shorter a dz, the larger is the subspace it represents. This is visible in Figure 2 where the dz $\{00\}$ represents a subspace smaller than and contained by the subspace $\{0\}$. Therefore, for a more expressive content filter, the recursive binary decomposition is more fine granular, and the resulting dzs are longer. The previous example also points out another property of spatial indexing that the dz of a subspace has a prefix equivalent to the dz of the subspace containing it. We also refer to this containment relation between dzs as a covering relation (denoted by $\succ$). This property ensures that a longer dz representing an event (i.e., a point in Ω) is considered a match for all subspaces

(filters) containing it simply through a prefix match. Note that an increase in the number of attributes (i.e., dimensions) in the system increases the length of the dz required to accurately represent content.

After converting content to binary strings, the next step is to map dzs representing content filters to the selected match field in flow entries of TCAM and dzs representing events to the same field in packet headers to enable header-based matching. For this purpose, we choose a range of IP multicast addresses (e.g., IPv6) to use as destination IP addresses in the match field of flows as well as in the packet headers. The dzs are simply appended to a fixed prefix, e.g., ff0e (representing the IPv6 multicast address range available to pub/sub traffic), in the destination IP address. The prefix-based filtering operation is guaranteed in IP addresses with the help of Class-less Interdomain Routing (CIDR) style masking supported by SDN-compliant switches where masking operations are represented by the 'don't care' symbol (*). Please note that a flow consists of the match field (MF), in our case an IP multicast address representing a dz, and an instruction set (IS), which specifies the port through which an event should be forwarded on account of a match.

We further explain the two-step mechanism of content representation with the example from Figure 2. Let us assume that subscriber S_1 has a subscription $sub_1 : \{\text{T = [50, 100]} \wedge \text{P = [50, 100]}\}$. Spatial indexing yields the dz $\{11\}$ to represent it as illustrated in the 2-bit representation in the figure. This dz is then appended to a multicast IPv6 prefix (ff0e) and installed as a destination IP (ff0e:c000:*/18) in the match field of flows on the switches. Now, if the event e in Figure 2, lying within (matching) sub_1, is represented by the dz $\{110010\}$, then it is converted to an IP address ff0e:c800:: and header-based matching of this event packet takes place with the installed flows for sub_1.

To establish paths between publishers and subscribers, an efficient approach to topology reconfiguration is important for pub/sub on SDN. For this purpose, a spanning tree (comprising switches) is maintained to account for an acyclic dissemination structure on which paths are embedded between publishers and subscribers by installing appropriate flows (filters) on switches along these paths. A path is a sequence of switches (denoted as R) on which flows are deployed to ensure connectivity between the publisher and the subscriber. An acyclic spanning tree ensures that there is always only a single path between each publisher and each of its relevant subscribers, thus avoiding the possibility of cycles or loops in the network. The flows to be deployed on a switch depend largely on the already existing pub/sub flows on that switch and as a result it is important for the controller to identify the state of each switch in the network. In fact, the network state (denoted by NS) consists of (i) all switches constituting the network, (ii) all links connecting the switches in a spanning tree to account for an acyclic dissemination structure, and (iii) all pub/sub flows deployed on each switch. Please recall that, with the advent of (un)advertisement/(un)subscription requests, the controller performs reconfiguration of the network by (un)installing filters on switches to ensure correct dissemination of events from publishers to interested subscribers. Therefore, when an (un)advertisement/(un)subscription request arrives at the controller, it reads the current network state and accordingly decides on necessary changes to the flows on the network switches that are required to satisfy the current request (the details of exactly how the entry and exit of publishers (advertisements) and subscribers (subscriptions) in the system are handled are provided in [35], [9]).

This decision to make necessary changes to the network state, on advent of a request at the controller, largely depends on flow relations. Please note that the containment/covering relations between dzs is also reflected on flows. For example, a flow fl_i covers (or contains) another flow fl_j, denoted by $fl_i \succ fl_j$, if the following two conditions hold: (i) the dz associated with the destination IP address in the match field of fl_j is covered by the dz of fl_i, and (ii) the out ports to which a packet matching fl_j is forwarded are a subset of those of fl_i. Therefore, while establishing a route on advent of a new subscription, let us assume that a new flow fl_n needs to be installed on a switch R. Now, if an existing flow fl_e on R already covers fl_n , then no further actions are taken as fl_e already forwards the traffic of fl_n. Therefore, an additional flow fl_n will be redundant on this switch. Similarly, if an existing flow fl_e is covered by fl_n, then fl_n is added and fl_e is deleted from the flow table as, now, fl_e is redundant. Likewise, a partial containment relation ($\succsim$) can be defined between flows of a switch (or flows to be installed on a switch) which may impact the topology reconfiguration process as well, the details of which are provided in [35], [9]. Therefore, in a nutshell, the controller establishes paths, containing content filters, between publishers and subscribers in this manner to enable filtering of events on hardware switches in PLEROMA. Please note that PLEROMA takes advantage of the aforementioned flow relations to represent multiple similar subscriptions by a single flow. For example, let us assume that in Figure 2, both S_1 and S_2 subscribe for the same subspace sub_1. In such a scenario, a single flow representing sub_1 is installed along the path from publisher P to switch R. At R, again there is a single flow for the filter sub_1, however, this time the IS of this flow will include both outgoing port 2 and outgoing port 3 in order to forward a matching packet to both S_1 and S_2. Please note that, similar to traditional switches, an SDN-compliant switch can also forward multiple copies of an incoming packet through multiple ports.

B. Limitations of Content Representation

From the above description on content filter representation on switches in PLEROMA, we see that expressiveness or granularity at which spatial indexing can be performed is limited by the number of bits that can be appended to the destination IP address. Let us assume that instead of 2 bits only 1 bit can be accommodated in the IP address reserved for pub/sub traffic. In such a scenario, subscription sub_1 will be represented by the dz $\{1\}$ as depicted in the 1-bit representation in Figure 2. This implies that all events matching the entire subspace of $\{1\}$ in the figure will be received by subscriber S. Therefore, the path between P and S will be subjected to unnecessary traffic which we will henceforth refer to as *false positives*.

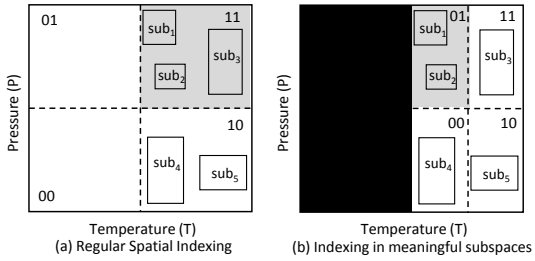


Fig. 3: Avoiding Empty Subspaces

More specifically, we define false positives as follows. **False positives** are those events which should be filtered out by the network but, nevertheless, are forwarded to uninterested subscribers due to limitations of content filters on switches.

Here, we also define the term false positive rate as follows. **False positive rate** is the percentage of total number of events received at the subscribers that are unnecessary (i.e., false positives).

We formally define our problem related to limitations of content representation as follows. Given a fixed number of bits for filter representation, the goal is to increase expressiveness of filters such that false positives are minimized in the system without incurring any false negatives. Here, we relate bandwidth efficiency directly with the occurrence of false positives. We do so as most pub/sub applications, e.g., stock exchange, network monitoring, environmental monitoring, etc., publish event packets where the payload is relatively small and the variation in packet size is, therefore, not significant. As a result, in literature false positives are used as a good indicator of unnecessary bandwidth usage [18], [22].

As a result, the remaining part of this paper is dedicated to the design of various techniques that would improve expressiveness of content filters installed on hardware switches, despite their limitations, and render content-based pub/sub realized on software-defined networks bandwidth-efficient. The presented techniques are workload dependent, i.e, they take decisions based on past events and/or subscriptions in the system. Please note that the distribution of events and subscriptions can change with time. We assume that the traffic distribution changes gradually with time as is typically the case in many IoT applications (e.g., sensor data measuring physical phenomena). As a result, we employ each of the following techniques periodically in the system such that decisions can be taken based on the most recent data. These techniques are implemented at the control plane. The controller already has a knowledge of all the subscriptions in the system and has to additionally collect statistics of events periodically and modify flows on switches accordingly. In the context of pub/sub, the control plane may be scaled to distribute this additional overhead among multiple controllers while guaranteeing the notion of a logically centralized controller as achieved in [9]. Please note that the most common notations used in the following techniques are presented in Table I.

III. Workload-based Indexing

The effectiveness of the previous attempts to encode content into binary form has primarily depended on the size of the event-space. The only parameters that play a role in the mapping process are the number of available bits and the size of Ω. However, in this section, we design two mapping techniques—selective indexing and adaptive spatial indexing—that consider the previous two parameters and also look into the workload of the system (i.e., events and subscriptions) to encode content to binary strings.

TABLE I: Important Notations

$\mathbb{M}$	Set of MBRs
ss	Subspace in Ω
S	Set of current subscriptions
s	$\lvert S \rvert$
E^t	Set of past events in consideration
ψ	$\lvert E^t \rvert$
$\mathbb{D}$	Set of original dimensions
ω	$\lvert \mathbb{D} \rvert$
$\mathbb{SD}$	Set of selected dimensions $\in \mathbb{D}$
n	$\lvert \mathbb{SD} \rvert$
ϱ	Selectivity factor of a dimension
$\mathbb{C}$	Covariance matrix
sf	Similarity factor between two dimensions

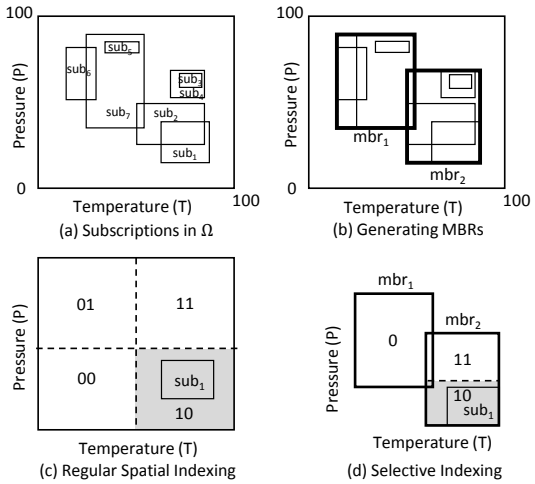


Fig. 4: Selective Indexing

A. Selective Indexing

In-network filtering may result in significant number of false positives depending on the size of Ω, i.e., number of dimensions and range of values along each dimension. This is mainly due to the fact that with a fixed number of bits available for a dz (e.g., 23 bits for IPv4 multicast addresses), larger the size of Ω, less fine granular is the indexing. However, it should be noted that regular spatial indexing partitions the entire space into subspaces, even those subspaces that are of no interest to any subscriber. Here, we introduce the notion of avoiding the indexing of the entire event-space Ω such that all empty subspaces with respect to subscription distribution in Ω are left out and the bit strings earlier assigned to these empty spaces are used for more fine granular indexing of the populated subspaces. Please note, we do not specifically consider the event distribution as, in any case, only those events that lie within the subscriptions are important from filtering point of view and those lying in other subspaces can be ignored. To understand the effectiveness of such selective indexing of Ω, we look at an example from Figure 3. We,

specifically, focus on subscription sub_1 in a 2-dimensional event-space comprising the dimensions temperature (T) and pressure (P). For the sake of simplicity, let us assume that only 2 bits are available to represent sub_1 through spatial indexing. Figure 3(a) shows that when the entire event-space is indexed, then sub_1 is represented by $\{11\}$ and it receives all events lying within this subspace (highlighted in gray). Now, since there are no subscriptions in subspaces $\{00\}$ and $\{01\}$, we completely neglect these empty spaces and use the available strings for finer indexing in the populated subspaces as illustrated in Figure 3(b). Therefore, in Figure 3(b), sub_1 is represented as $\{01\}$, and receives only the events lying within this subspace which is much smaller than the subspace representing sub_1 in Figure 3(a). Due to more fine granular indexing in the latter case, the false positives received by sub_1 will also be lower compared to the former case. Therefore, to this end, we introduce the selective indexing approach where the main idea is to identify meaningful subspaces with respect to subscriptions in Ω and only index those subspaces instead of the entire event-space.

The first step in the selective indexing approach is to select subspaces in Ω populated with subscriptions while identifying the empty spaces to be neglected. To identify meaningful subspaces, we benefit from the widely used mechanism of similarity-based subscription clustering [30], [10]. Once subscriptions are clustered into groups, we generate polyspace rectangles which serve as the closest enclosing approximation of each of these clusters. These polyspace rectangles are known as minimum bounding rectangles or MBRs. The set of generated MBRs encloses all subscriptions in the system such that every subscription can be represented by a binary filter (or set of filters) and attempts to leave out as much empty space as possible. To understand the concept of an MBR, we provide an example from Figure 4. Here, the subscriptions are distributed in the event-space as illustrated in Figure 4(a). Figure 4(b) shows two MBRs covering all subscriptions in the system clustered together in two groups on the basis of similarity. Please note that even though two MBRs may partially overlap as in 4(b), a subscription strictly belongs to a single MBR. Let us suppose that the controller chooses to have 2 MBRs for the system. Therefore, for the purposes of our example, we proceed with the next phase of this approach with the two MBRs, mbr_1 and mbr_2, obtained from the first phase.

Having identified the MBRs, the next phase is the actual mapping of subscriptions to dzs. We again employ spatial indexing for the binary conversion of content however, of course, now, with a difference. Spatial indexing is not employed on the entire range of values along each dimension to arrive at the dz of a subscription. Instead, spatial indexing is performed only on the range of values along each dimension of the MBR (i.e., subspace in Ω) which contains the subscription in question. This means that two subscriptions belonging to two different MBRs may end up with the exact same dz as they occupy the same relative position in their respective MBRs. However, this would be incorrect as the two subscriptions occupy different positions relative to the actual event-space. This problem is mitigated by assigning unique IDs to MBRs. First, each MBR is assigned an MBR ID which is in binary form and which depends on the total number of MBRs in the system. Therefore, if $\mathbb{M}$ is the set of MBRs in the system, then the total bits required to uniquely identify each MBR is $log_2|\mathbb{M}|$. Next, the dz representing a subscription generated by the recursive decomposition of the MBR is appended to the MBR ID that the subscription belongs to. The unique ID prefix makes a dz different from that of another MBR.

The selective indexing approach allows for more fine granular spatial indexing as it can avoid assigning bits to the subspaces in Ω that are not part of any subscription in the system, thus allowing the use of more bits to represent more meaningful subspaces. We illustrate our point in Figure 4(c) and Figure 4(d). Let us focus on the subscription sub_1 that needs to be converted to a binary string. Let us assume that again only 2 bits are available for representing content filters. Now, since there are two MBRs, a bit is required to uniquely represent them. However, this bit represents a smaller subspace as compared to what it would represent in regular spatial indexing in Ω as the empty spaces have been removed. The next step is to perform spatial indexing within mbr_2 till the closest approximation of the subscription is reached with the available number of bits. In this case, the subscription can afford just one more bit that will be appended to the MBR ID 1 for mbr_2. Therefore, for sub_1, the generated dz is $\{10\}$ as depicted in Figure 4(d). However, when spatial indexing is performed on entire Ω as depicted in Figure 4(c), false positives are more as the same dz of $\{10\}$ represents a much larger subspace in this context.

Of course, for header-based matching of packets to work, events will also need to be mapped to the selected packet header field using the selective indexing approach. For this purpose, publishers need to have information about the MBRs and their respective bounding values. As a result, the controller sends this information to each publisher whenever there is a change in MBR values. The mapping of events to the selected header field works similar to the mapping of subscriptions to match fields. However, it should be noted that MBRs may overlap. For example, in Figure 4(b), mbr_1 and mbr_2 overlap. In such a scenario, an event that lies in the overlapping subspace must be indexed with respect to both MBRs as it can match subscriptions from both. This ensures the avoidance of false negatives. Also, please note that all events that do not lie within any MBR are simply ignored by the publisher and do not need to be indexed. However, this does not mean that false negatives are introduced in the system as the events that lie outside of all MBRs are those that no subscriber is interested in receiving. As a result, these events, in any case, should not be forwarded in the network.

B. Adaptive Spatial Indexing

The selective indexing approach uses regular spatial indexing to finally convert filters and events to dzs. As discussed before, spatial indexing divides the event-space repeatedly to achieve subspaces of maximum possible granularity where each decomposition divides the current subspace equally into two halves. Question is whether the employed spatial indexing technique itself can be modified to obtain more expressive

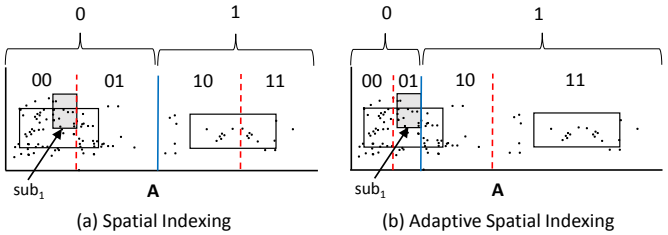


Fig. 5: Adaptive Spatial Indexing

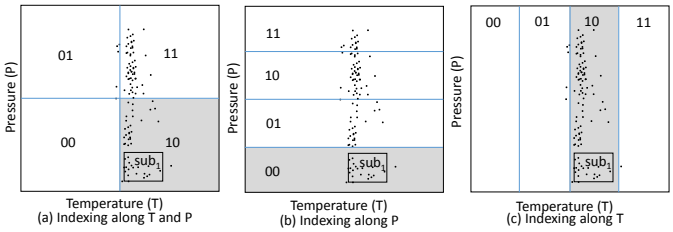


Fig. 6: Effects of event distribution

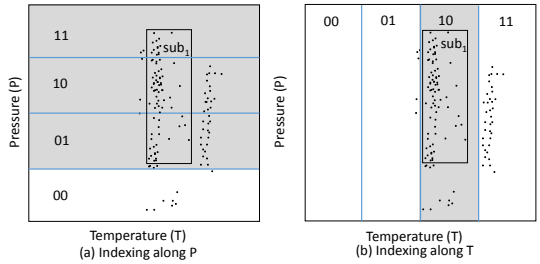


Fig. 7: Event-based Selection

filters and, therefore, less false positives in the system. In this section, we design an adaptive spatial indexing (ASI) approach to answer the same.

The spatial indexing technique, essentially, performs disjoint event-space partitioning. We employ a similar technique but with a difference. For each recursive decomposition, instead of dividing a subspace into two equal halves in terms of range of values along dimensions, the basic idea is to divide it into two subspaces with balanced workload with respect to events and subscriptions, i.e., number of events matching subscriptions. This allows indexing to have finer granularity in subspaces with higher workload in Ω. Here, it is extremely important to first define the term workload. We define workload of a subspace ss_i as $W_{ss_i} = \sum_{e_k \in E^t} |S_{e_k}^{ss_i}|$ where $S_{e_k}^{ss_i}$ represents the set of subscriptions within ss_i matched by an event e_k. Therefore, when a subspace is further divided during spatial indexing along a dimension, the workload of it along that dimension is calculated and the division is made such that the resultant two subspaces have equal workload, i.e., they are not necessarily equal in terms of range of values along dimensions.

We explain the above indexing strategy with the help of an example from Figure 5 which depicts a 2-dimensional event-space with events and subscriptions. For the sake of simplicity, we only explain indexing along one dimension, i.e., dimension A. Let us assume that, again, only 2 bits are available for indexing. Now, while performing indexing to represent sub_1, in regular spacial indexing, the dimension range is divided equally into two subspaces $\{0\}$ and $\{1\}$ as depicted by the blue solid line in Figure 5(a). However, in our adaptive spatial indexing technique, the division is made such that the workload in the resultant subspaces is equal. Let the blue solid line in Figure 5(b) illustrate this workload-based division. This allows for more fine-grained partitioning in the subspace denoted by $\{0\}$ where matching traffic is heavy as compared to $\{1\}$. Further divisions in both cases, as depicted by the red dotted lines in Figure 5(a) and Figure 5(b), clearly indicate that sub_1 is represented by a much smaller subspace $\{01\}$ in adaptive spatial indexing as compared to $\{00\}$ in regular indexing. As a result, sub_1 suffers from fewer false positives when represented by adaptive spatial indexing.

All dimensions are divided in the exact same manner to arrive at the final dz for a subscription or an event in a multi-dimensional system. By allowing more bits to be assigned to more meaningful parts of Ω, false positives can be reduced in adaptive spatial indexing.

The efficiency of the workload-based indexing approaches with respect to reducing false positives may still be limited when the number of attributes (dimensions) in the system is large. As a result, the next section is dedicated to mechanisms that influence the number of dimensions to be encoded into content filters while performing in-network filtering.

IV. Dimension Selection

As discussed before, more the number of dimensions in a system, longer are the dzs. As a result, in this section, we discuss the notion of avoiding the indexing of every dimension and using the available bits to perform fine granular spatial indexing only on a subset of dimensions that prove to be more promising with respect to bandwidth efficiency. While this notion was briefly introduced in [35], it was not thoroughly explored and left open questions. In this paper, we use this notion to propose and thoroughly evaluate a set of algorithms that select dimensions that are beneficial for reducing false positives and discuss their applicability, complexity, and performance with respect to realistic workload distributions. Please note that, while performing dimension selection, our mapping scheme ensures that, for all dimensions which do not get selected for indexing, all subscribers have subscribed for the entire range of values along these dimensions (all non-indexed dimensions are effectively represented by '*' in the filters). As a result, no expressive filtering is done along these dimensions as, based on the non-selected dimensions, all events would be forwarded to all subscribers.

A. Event Variance

The distribution of events in Ω plays a major role in determining the importance of each dimension for filtering in the system. To this end, the spread of events along a dimension is an important metric to determine the importance of that dimension. More spread would require more fine granular indexing to avoid false positives, rendering the dimension worthy of being considered for selection. More specifically, we use variance of events to measure this spread. If E^t

Algorithm 1 Event Variance-based Selection

1: $\mathbb{D} \rightarrow$ Set of original dimensions
2: $E^t \rightarrow$ Set of all events
3: $\mathbb{SD} = \emptyset$ // Set of selected dimensions
4: $\varrho \rightarrow$ Set of ω selectivity factors for ω dimensions, where $\omega = |\mathbb{D}|$
5: **for all** $d \in \mathbb{D}$ **do**
6: $\quad \varrho_d = (\sum_{i=1}^{|E^t|} (e_i^d - \overline{e^d})^2)/|E^t|$
7: $\mathbb{SD} \leftarrow$ Select dimensions corresponding to n highest values in ϱ

denotes the set of all events in Ω, then event variance along a dimension d is measured as $(\sum (e_i^d - \overline{e^d})^2)/|E^t|$ where e_i^d represents the value of the i^{th} event along dimension d. We illustrate this with a very simple example in Figure 6 with respect to a single subscription sub_1, where the variance of events along dimension P is far greater than that along dimension T. Let us assume that only 2 bits are available for spatial indexing. Figure 6(a) shows spatial indexing along both dimensions according to which sub_1 is represented by the subspace $\{10\}$ which means that sub_1 receives all events lying in this subspace. Now, if only dimension P, with a high variance value for events, is selected for indexing, then sub_1 gets represented by the subspace $\{00\}$ and receives all events lying within it as shown in Figure 6(b). In Figure 6(a) sub_1 suffers from far more false positives as compared to the false positives received when only P is selected for indexing. This is because, the latter can take advantage of the fact that dimension P has a significantly high variance value for events as compared to dimension T and thus has the liberty of more fine granular indexing along P. As a result, most events that are irrelevant for sub_1 can be partitioned out into other subspaces. Since event variance is low along dimension T, ignoring it does not cost sub_1 much. However, if the dimension with low variance value for events, i.e., dimension T is selected for indexing, Figure 6(c) clearly shows that sub_1 would be subjected to more false positives as compared to indexing along dimension P and, also, indexing along both dimensions. This example clearly indicates the importance of event distribution within Ω in dimension selection.

Therefore, the very first dimension selection algorithm that we present is Event Variance-based Selection (EVS). This algorithm is, also, formally described in Algorithm 1. EVS calculates the variance of events along each dimension. Let $\mathbb{D}$ be the set of ω dimensions in Ω and E^t be the set of ψ events that are being considered for the algorithm in the current time window t. Let $\mathbb{SD}$ be a subset of n dimensions of $\mathbb{D}$, i.e., $\mathbb{SD} \subseteq \mathbb{D}$ and $|\mathbb{SD}| = n$. We assign, to each dimension $d \in \mathbb{D}$, a selectivity factor denoted as ϱ_d, which determines the importance of the dimension in terms of reduction of false positives if chosen for spatial indexing. Higher the value of ϱ_d, higher is the importance (selectivity) of d with respect to the ability to reduce false positives. For EVS, the selectivity factor ϱ_d of a dimension d is given by the variance of events along that dimension (cf. Algorithm 1, line 6). EVS selects dimensions for $\mathbb{SD}$ by selecting n dimensions in $\mathbb{D}$ with the highest variance/selectivity factor values. Spatial indexing commences now on $\mathbb{SD}$.

The main advantage of this approach lies in its low computation overhead with a complexity of $O(\omega * \psi)$. However, the consideration of only event distribution may not be enough in every scenario. For example, in Figure 7(a), since event variance along dimension P is high, the subscription sub_1, when indexed along P, is represented by the subspaces $\{01\}$, $\{10\}$, and $\{11\}$ and will receive all events lying within these subspaces. However, if indexed along dimension T, with lower event variance, sub_1 is represented by the subspace $\{10\}$ and receives events lying within it as depicted in Figure 7(b). Here, false positives are lower in the latter case. This clearly indicates that both events as well as subscriptions play a major role in the selection process.

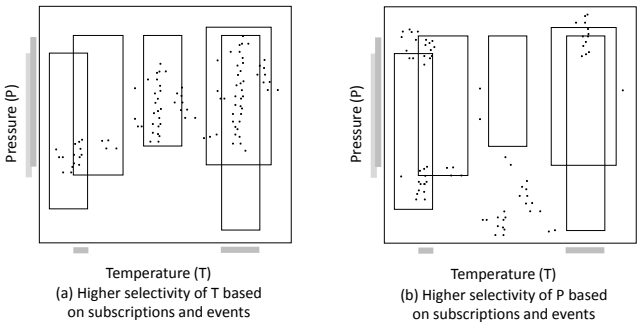


Fig. 8: Subscription-based Selection

Algorithm 2 Event Match Count-based Selection

1: $\mathbb{D} \rightarrow$ Set of original dimensions
2: $S \rightarrow$ Set of all subscriptions
3: $E^t \rightarrow$ Set of all events
4: $\mathbb{SD} = \emptyset$ // Set of selected dimensions
5: $\varrho \rightarrow$ Set of ω selectivity factors for ω dimensions, where $\omega = |\mathbb{D}|$
6: **for all** $d \in \mathbb{D}$ **do**
7: $\quad matches = 0$
8: $\quad$ **for all** $e \in E^t$ **do**
9: $\quad\quad matches+ = |S_e^d|$ // No. of subscriptions that e matches along d
10: $\quad \varrho_d = 1 - (matches/(|E^t| * |S|))$
11: $\mathbb{SD} \leftarrow$ Select dimensions corresponding to n highest values in ϱ

B. Subscription Matching

It would be interesting to investigate the role played by subscriptions in the process of dimension selection. In fact, in doing so, we identified the importance of subscription overlaps. Dimensions where subscriptions have a lot of overlaps are less important for filtering because if an event matches a subscription along this dimension, then it matches majority of the subscriptions along this dimension, thus reducing its importance with respect to the ability to reduce false positives. For example, Figure 8(a) shows a scenario where there is a significant overlap of subscriptions along dimension P (the gray lines indicate overlaps). According to the figure, selection of dimension T would reduce more false positives than if P is selected. If indexing is performed along T, events are matched by interested subscriptions as most events are matched by disjoint subscriptions. On the contrary, if indexing is performed along P, then false positives will be high as most events match multiple overlapping subscriptions on this dimension but not along T. Note that an event is matched by

Algorithm 3 Correlation-based Selection

1: $\mathbb{D} \rightarrow$ Set of original dimensions
2: $S \rightarrow$ Set of all subscriptions
3: $E^t \rightarrow$ Set of all events
4: $\mathbb{SD} = \emptyset$ // Set of selected dimensions
5: $\mathbb{C} \leftarrow$ Initialize $\omega * \omega$ covariance matrix, where $\omega = |\mathbb{D}|$
6: **for** i=0 to $\omega - 1$ **do**
7: **for** j=0 to $\omega - 1$ **do**
8: $sf_{i,j}$ = 0.0 // Similarity factor between dimension *i* and dimension *j*
9: **for all** $e_k \in E^t$ **do**
10: $sf_{e_k}^{i,j} = (|S_{e_k}^{d_i} \cap S_{e_k}^{d_j}|)/|S|$
11: $sf_{i,j} += sf_{e_k}^{i,j}$
12: $c_{i,j}$ = 1.0 - ($sf_{i,j}$ / $|E^t|$) // covariance value at $(i,j)^{th}$ index of $\mathbb{C}$
13: $V \leftarrow$ Calculate eigenvectors of $\mathbb{C}$
14: $\Lambda \leftarrow$ Calculate eigenvalues of $\mathbb{C}$
15: $princComp \leftarrow$ Select eigenvector $\in V$ corresponding to highest eigenvalue $\in \Lambda$
16: $\mathbb{SD} \leftarrow$ Select n dimensions $\in \mathbb{D}$ with highest coefficients in $princComp$

a subscription if and only if it is matched on all dimensions. However, again, the selection decision cannot be taken based on subscription overlaps alone. The reason why the selectivity of T is higher is because of not only fewer overlaps but also the distribution of events. For example, in Figure 8(b), we have the same subscription overlaps as before, however, due to the distribution of events, the selectivity of T is not too high.

Therefore, it is necessary to consider the combination of both subscriptions and events to determine selectivity of dimensions. As a result, we introduce another algorithm known as Event Match Count-based Selection (EMCS) which has higher computational complexity than Event-based Selection but considers both events and subscriptions to take the selection decision, rendering it more generic with respect to the distribution of events and subscriptions in Ω.

The main idea of EMCS is to deem dimensions where event traffic matches most subscriptions as less important for dimension selection. Considering S to be the total set of s subscriptions in the system, this algorithm determines the set of subscriptions that each event $e \in E^t$ matches, i.e., S_e^d, along each dimension d and calculates the number of matches in each case, i.e., $|S_e^d|$. Now, for each $d \in \mathbb{D}$, the selectivity factor is calculated as $\varrho_d = 1 - (\sum_{e \in E^t} |S_e^d|)/(|E^t| * |S|)$ where the sum of all the matches of all events matching subscriptions is calculated and represented as a fraction of the maximum value possible for matches, i.e., $|E^t| * |S|$. Having calculated ϱ^d for each d, a value between 0.0 and 1.0, n dimensions with highest values of selectivity factor are added to $\mathbb{SD}$. The steps of EMCS are more formally presented in Algorithm 2. This algorithm is more generic than the previous one. This is mainly because it manages to capture the amount of event variance within the meaningful subspaces of Ω, i.e, within subscriptions, thus addressing the aforementioned problem of EVS. However, it has a higher time complexity of $O(\omega * \psi * s)$.

C. Correlation

Most application domains handle a large amount of data with numerous attributes. Quite often, such data has redundancy among its attributes. Redundancy in data may occur due to underlying relations (i.e., correlations) between the attributes (i.e., dimensions) of the system such that the change in values in one dimension is positively or inversely correlated to the change in values in another dimension. Quite often, subscriptions and the events matching them have dimensions that are correlated or inversely correlated rendering the selection of these dimensions redundant because if an event matches a subscription in one dimension, it would also do so in the others. Such correlation between attributes exists across most applications. For instance, in IoT, most sensors detect and measure changes in various physical phenomena (i.e., dimensions) where correlations exist. For example, the sensor data set provided by the Intel Research Berkeley Lab [2] that has a 54 node sensor network measuring values for temperature, humidity, and light shows positive correlation among all the 3 attributes [16]. Again, in a traffic monitoring scenario, for certain time periods, there may exist an inverse correlation between car speed and density. In a completely different domain, i.e., in stock exchange, it is well established that there exists a correlation between volume and stock prices [1]. Redundancy in data can be utilized to avoid less meaningful dimensions without loss of much information while selecting dimensions. However, because of the sheer amount, data is often fuzzy making it difficult to identify such redundancy.

As a result, our next algorithm, Correlation-based Selection (CS) tries to take advantage of any redundancy in data, in the form of correlation, that may exist between dimensions while also considering the previous two factors, i.e., event variances and subscriptions across dimensions. In the previous two algorithms, the selectivity factor ϱ was independently calculated for each dimension d. However, in order to consider correlation as well, we construct a covariance matrix, $\mathbb{C}$, which captures relations between dimensions as well as within them with respect to selectivity. This algorithm, formally described in Algorithm 3, consists of primarily two steps—(i) calculating the covariance matrix and (ii) performing principal component analysis (PCA) on the calculated matrix.

The basis of this approach is the calculation of the covariance matrix $\mathbb{C}$. A covariance matrix holds covariances representing relations between two random variables, in this case, dimensions. As before, considering ω to be the number of dimensions, $\mathbb{C}$ is an (ω*ω) matrix where an element at position (i,j) represents covariance of the i^{th} and the j^{th} dimensions. $\mathbb{C}$ captures two types of information—the relation between dimensions with respect to selectivity as well as the amount of variance within each dimension. The diagonal of $\mathbb{C}$ captures the latter. For dimension selection, both of these information are crucial as the former highlights correlated dimensions and the latter highlights selectivity of each independent dimension. Quite naturally, it is crucial to identify the metric representing the covariances, i.e., $c_{i,j} \in \mathbb{C}$, depending on the type of relation between dimensions that needs to be captured. In the context of this algorithm, we define covariances between dimension pairs with respect to events consumed by subscriptions along each dimension.

We provide the steps to calculate the covariance matrix more formally as follows (cf. Algorithm 3, lines 5-12). While

calculating the covariance $c_{i,j}$ between a pair of dimensions d_i and d_j, first, for each event $e_k \in E^t$, we calculate a factor called the similarity factor which calculates the set of subscriptions that the event e_k matches along both dimensions of the dimension pair. Therefore, the similarity factor of a dimension pair d_i and d_j for an event $e_k \in E^t$ is calculated as $sf_{e_k}^{i,j} = (|S_{e_k}^{d_i} \cap S_{e_k}^{d_j}|)/|S|$ (cf. Algorithm 3, line 10). As before, here, $S_{e_k}^{d_i}$ represents the set of subscriptions matched by event e_k along dimension d_i. As a result, an intersection of set $S_{e_k}^{d_i}$ and set $S_{e_k}^{d_j}$ provides the set of only those subscriptions that e_k matches along both d_i and d_j. The number of subscriptions in this resultant subscription set contributes to the similarity factor between the two dimensions for this event. To calculate the aggregated similarity factor ($sf_{i,j}$) between the dimension pair d_i and d_j, the similarity factors of all events are calculated as mentioned above and aggregated (cf. Algorithm 3, lines 8-11). Then, the inverse effect of this summed up (or aggregated) value is considered to measure the dissimilarity between the two dimensions in order to calculate the covariance between them. Therefore, finally, $c_{i,j}$ is calculated as 1.0 - $\sum_{e_k \in E^t} sf_{e_k}^{i,j}/|E^t|$ (cf. Algorithm 3, line 12). This value indicates the covariance between a dimension pair with respect to the number of times events match subscriptions along both dimensions of a dimension pair. Along the diagonal of $\mathbb{C}$, the variance of the match of events with subscriptions within each dimension gets captured.

Once $\mathbb{C}$ is calculated, the technique of principal component analysis (PCA) is applied [23]. Without going into much mathematical details, we describe the main steps required to select dimensions using PCA (cf. Algorithm 3, line 13-16). First, $\mathbb{C}$ is subjected to spectral analysis through the process of eigendecomposition, i.e., $\mathbb{C} = V\Lambda V^T$, where $\Lambda = \{\lambda_1, \ldots, \lambda_\omega\}$ is a diagonal matrix of eigenvalues and $V = \{v_1, \ldots, v_\omega\}$ is the matrix whose columns are orthogonal eigenvectors of $\mathbb{C}$. Eigendecomposition projects the original dimensions (in Ω) onto an orthogonal basis of vectors called eigenvectors. This transformation makes the highest variance by any projection of the dimensions to lie on the very first axis (i.e., first principal component). In fact, an eigenvector v with largest eigenvalue represents the dimension (in the orthogonal basis) along which variance is maximized (i.e., first principal component), and thus this eigenvector v is used to rank the original dimensions [27]. In more detail, a higher absolute value of i^{th} coefficient of v indicates that the dimension d_i is more important to be used for filtering. Thus, the dimensions (in the original space) that correspond to the first n coefficients with higher magnitude are selected for filtering. CS efficiently chooses dimensions based on the idea of reducing redundancy in data while maximizing variance of events matched by subscriptions. The time-complexity of the calculation of the covariance matrix itself is $O(\omega^2 * \psi * s)$, rendering the algorithm more complex than the previous two.

D. Evaluation-based Techniques

The previous algorithms, though effective in their own ways, do not give an indication of an ideal value of n. As a result, in this subsection, we introduce two algorithms which significantly reduce false positives in the system and also provide the most suitable value for n. Since the controller has knowledge of both S and E^t, we can implement evaluation-based techniques to simulate false positives in the system for various combinations of dimensions and choose the most beneficial one, thus obtaining even a suitable value for n. The performances of these techniques are more optimal as compared to the previous three algorithms. However, these techniques have relatively higher computational complexities.

Ideally, in order to obtain an optimal set $\mathbb{SD}$, a brute force technique must be employed which calculates the false positives for all combinations of dimensions and finally selects the one producing least false positives. In order to do so, a complete simulation of the entire filtering process must be performed at the logically centralized controller, given a fixed value of the number of available bits for filter representation. With the information of the actual subscription and event values, their corresponding mappings to binary strings, the false positive rate can be determined for each combination of dimensions. However, running such a simulation has exponential computation overhead of $O(2^\omega * \omega * s * \psi)$.

We reduce the complexity of the brute force algorithm by using a greedy strategy which is also based on simulation, however it does not evaluate every combination of dimensions. We describe the steps of this algorithm as follows and, also, provide a formal description in Algorithm 4. Initially, the combination with all ω dimensions in $\mathbb{D}$ is considered and the resulting false positive rate noted. Then, all combinations with ω-1 dimensions are evaluated, i.e., each combination has ω-1 dimensions, however in each combination a different dimension is removed. The combination with the lowest false positive rate is selected and in the process one dimension gets removed (cf. Algorithm 4, lines). The next cycle uses this selected combination with ω-1 dimensions as input and evaluates all combinations with ω-2 dimensions to arrive at the most beneficial combination for ω-2 dimensions. The process continues till the number of dimensions being considered for the combinations is reduced to 1 by incrementally removing

Algorithm 4 Greedy Strategy

1: $\mathbb{D} \rightarrow$ Set of original dimensions
2: $S \rightarrow$ Set of all subscriptions
3: $E^t \rightarrow$ Set of all events
4: $\mathbb{SD} \rightarrow$ Set of selected dimensions
5: $combinations_\omega = \mathbb{D}$
6: originalFPR = evaluateFPR($\mathbb{D}, S, E^t$) // Evaluate false positive rate ranging from 0 to 1 with original dimensions
7: **while** $\omega > 1$ **do**
8: lowestFPR$_{\omega-1}$ = 1
9: selIgnoredDim$_{\omega-1}$ = $\emptyset$
10: $temp\mathbb{D} = combinations_\omega$
11: **for all** $d \in combinations_\omega$ **do**
12: $temp\mathbb{D} = temp\mathbb{D} \setminus d$
13: FPR$_{\omega-1}$ = evaluateFPR($temp\mathbb{D}, S, E^t$) // Evaluate false positive rate when dimension d is ignored
14: **if** FPR$_{\omega-1}$ < lowestFPR$_{\omega-1}$ **then**
15: lowestFPR$_{\omega-1}$ = FPR$_{\omega-1}$
16: selIgnoredDim$_{\omega-1} = d$
17: $temp\mathbb{D} = temp\mathbb{D} \cup d$
18: $combinations_{\omega-1}$ = $combinations_\omega \setminus$ selIgnoredDim$_{\omega-1}$
19: $\omega- = 1$
20: $\mathbb{SD} \leftarrow$ Select combination $\in combinations$ with lowest FPR

one dimension in every step. As a result, we have a total of ω combinations where the first combination consists of ω dimensions, the second consists of ω-1, and so on till the last (ω^{th}) combination contains 1 dimension. Quite often, with decreasing number of dimensions, the false positive rate decreases till the redundancies in data are removed, after which the rate increases again due to loss of important information with further reduction in dimension count. As a result, different combinations with different dimension counts can be expected to reduce different number of false positives. Therefore, of all the aforementioned ω combinations, the one producing least false positives is chosen for $\mathbb{SD}$. By employing such a technique, we essentially also obtain the most suitable value of n. The greedy strategy has a time complexity of $O(\omega^3 * \psi * s)$.

V. Handling Dynamic Network Updates

All of the above discussed methods rely heavily on past event traffic and subscription distributions. However, the event distribution and the current subscriptions in the system may change over time, degrading the effectiveness of the proposed techniques. Hence, the controller must periodically collect workload information over time to monitor the recent distribution, execute proposed techniques, and deploy necessary changes in the network. For example, in the case of dimension selection algorithms, the event traffic distribution may change over time and the dimensions that were selected previously by the dimension selection algorithm may need to be replaced in the next period. This implies that the indexing of content will be done for a different set of dimensions now, resulting in completely different dzs. As a result, a new set of flows would need to be deployed in the network. Therefore, all the techniques described in this paper require periodic updates to flows in the network (i.e., removal of existing flows and deployment of new flows that replace the existing ones) according to the current indexing decisions.

However, with the need for network updates comes the problem of ensuring consistency in the data plane. Please recall, from Section II, that a network state (NS) consists of all flows on all switches in the network. Therefore, when the transition from one network state to another is being performed, the event packets in transition in the network may be incorrectly dropped or forwarded. Let us consider an example of a system where indexing is performed on 4 dimensions, A, B, C, and D, resulting in a network state NS_o. Let us assume that dimension selection is employed to improve the bandwidth efficiency of the system and now spatial indexing is performed on only 3 dimensions, B, C, and D. In such a scenario, the existing flows need to be removed and new flows according to the new indexing (resulting in a network state NS_n) must be deployed as the old dzs, representing all 4 dimensions, are semantically different from the new dzs, representing only 3 dimensions. Therefore, while the transition from NS_o to NS_n is being performed, event packets in transition and targeted to follow NS_o, may no longer find a path through NS_o or/and be incorrectly forwarded by NS_n which is semantically different from the event packet in question. The same applies to event packets targeted at NS_n. The difference in semantics can be further explained through an example depicted in Figure 6 where when indexed along both dimensions, temperature and pressure, sub_1 has the dz $\{10\}$ (cf. Figure 6(a)). However, on indexing only along the dimension pressure, it has a dz $\{00\}$ (cf. Figure 6(b)). Clearly, in the context of the new index, the old one has completely different semantics and an event published during transition, say with a dz $\{10001\}$ lying within sub_1 and indexed according to the old dimension set will no longer find a match in the newly deployed flow representing sub_1 which now matches $\{00*\}$. As a result, this event may be dropped due to the absence of any flow matching it or may be incorrectly forwarded by a flow matching the event but representing a different subscription according to the new index. Therefore, additional mechanisms must be employed to ensure that packets are not lost or incorrectly forwarded in the data plane during transitions. Here, we would like to point out that, in this paper, we address the aforementioned problems that arise in the case where the entire network state gets replaced by a semantically different network state. It is important to note that modifications to the network state might be performed due to arrival and departure of publishers and subscribers. However, these modifications are completely different (with respect to those required in complete transition) as, in these modifications, the old state and the new state of the network are semantically the same and, therefore, the consistency issues that we face during a complete transition do not arise. As a result, it is important to mention that the consistency issues that we discuss and address in the remaining part of this section are specific to the case where the entire network state needs to be replaced by a semantically different network state.

A. Data Plane Consistency in PLEROMA

A lot of work has already been dedicated to ensuring data plane consistency in SDN [29], [21]. Most works attempt to provide a general solution to data plane consistency for any application and are, therefore, computation intensive and/or resource intensive. However, in our case, we can design a middleware-specific solution, i.e., a light-weight approach, that targets only those data-plane consistency issues that affect the functional requirements of our specific system. Data plane consistency in SDN is primarily characterized by three properties—(i) blackhole-freedom, i.e., a packet that should be forwarded by a switch should not be dropped during the transition, (ii) loop-freedom, i.e., no packet should loop in the network, and (iii) packet coherence, i.e., no packet should see a mix of old and new flows belonging to the old (NS_o) and new (NS_n) network states, respectively.

Please recall from Section II that the PLEROMA middleware installs paths between publishers and subscribers by first creating an acyclic spanning tree that covers all switches in the network and then embedding content filters along these paths. This ensures the existence of only a single path between two hosts of a network. Also, here, when we talk about transitions from one network state to another, we only talk about changing the content filters that are embedded along the path connecting

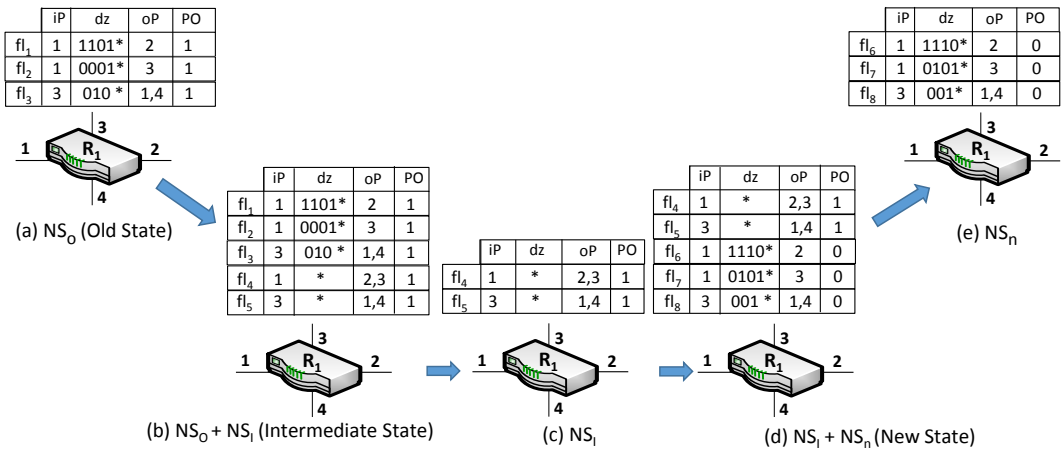


Fig. 9: Light-Weight Approach

a publisher to a subscriber, i.e., the path itself between two hosts remains the same in a transition. As a result, there is no possibility of cycles or loops in the network due to the transition because of which no additional measures need to be taken to ensure the inherent loop-freedom property of our system. Therefore, in the context of our middleware, the main consistency properties that we try to ensure are blackhole-freedom and packet coherence. Ensuring these two properties is essential for the system as both of these can lead to false negatives which is not tolerated in our system.

B. Light-Weight Approach

To ensure the above two properties, the main idea is to continue to have a path connecting a publisher to a subscriber when flows are being updated while also ensuring that a packet sees only one network state while being forwarded to its destination. To achieve this, Reitblatt et al. [29] propose the method of versioning, which allows both the old as well as the new network states to be installed in the network simultaneously with different version numbers. A packet with one of the two version numbers is forwarded by the old or the new network state depending on its version number but never forwarded by a mixture of both. However, this implies that each switch will require almost double the number of flows to accommodate both the old as well as the new version. Please note that TCAM is a very expensive and power-hungry resource and current vendors design SDN-compliant switches that can accommodate only a few thousand flows [15]. Moreover, as these flows are shared between applications, only a fraction may be available to pub/sub traffic to represent its content filters. Clearly, there may not be any TCAM space available to install another version for the same filters. Therefore, we attempt to avoid such a resource-intensive solution by designing a light-weight approach which is loosely based on the versioning method. The main idea of the light-weight approach (LWA) is, also, to always have a path connecting each publisher to its relevant subscribers so that false negatives, due to packet drops or incorrect forwarding of event packets, can be avoided. In fact, we use a temporary intermediate network state that can be used to forward events which are targeted at either the old network state or the new one while network updates are being performed. Let us take an example depicted in Figure 9 where there is a need to transition from NS_o (cf. Figure 9(a)) to NS_n (cf. Figure 9(e)) on switch R_1. In Figure 9 each flow in the flow table of R_1 is represented by its incoming port (iP), match field (represented by dz), outgoing ports (oP) which specifies the ports through which matching events are forwarded, and flow priority (PO). The priority of a flow may be important in certain cases as, please note that, when an event satisfies the matching criteria of multiple flows, the flow with the highest priority is allowed to forward it.

Once the decision to make the switch to NS_n is taken, first, a resource-efficient intermediate network state, i.e., NS_I, comprising flows matching any pub/sub event is installed along all paths connecting publishers to their relevant subscribers. The purpose of this temporary intermediate state is to always have paths for any pub/sub event no matter which network state it is targeted at. More specifically, for each incoming port, say, iP, on a switch, all flows on the switch, belonging to the old network state NS_o, which have iP as their incoming port are identified and a set of outgoing ports oP is created from the union of all outgoing ports to which the identified flows forward packets on account of a match. For example, in Figure 9(b), for the incoming port iP=1, the flows fl_1 and fl_2 are identified and a union of the outgoing ports to which these flows forward events is performed yielding the outgoing port set oP=$\{2,3\}$. Next, a single flow that forwards all pub/sub traffic (representing the entire event-space Ω) through all ports in oP is installed on the switch for this incoming port. In Figure 9(b), this is represented by fl_4 which forwards all incoming pub/sub traffic through the outgoing ports $\{2,3\}$. Please note that as this flow represents the dz $\{*\}$, covering entire Ω, it forwards any pub/sub traffic, irrespective of the semantics of the event and as long as it is part of the pub/sub traffic. This is done for every incoming port on each switch of the network during the transition. Therefore, in the example in Figure 9, the same is done for the incoming port 3 as there exists a flow fl_3 in the old network state where incoming traffic arrives at port 3. The flows, covering the entire event-space, constitute the intermediate network state NS_I. For each incoming port of a switch, replacing fine-grained filters of NS_o with a single flow covering Ω ensures the use of

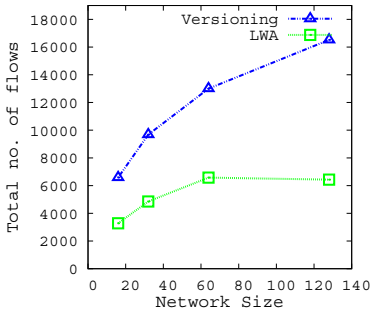


Fig. 10: Versioning vs light-weight approach (LWA)

minimum additional flows during the transition to maintain data plane consistency. In fact, in our light-weight approach, at any given time, the maximum number of additional flows installed on a switch to avoid false negatives is the total number of incoming ports of the switch. This is because at most a temporary flow belonging to NS_I may be added for each incoming port. This is in sharp contrast to the traditional versioning method [29] which would result in almost double the number of old flows during the transition to guarantee consistency. The impact on number of flows in the pub/sub network for our light-weight approach as compared to the versioning method is depicted in Figure 10. Please recall that TCAM space is limited. Therefore, where TCAM is scarce, the figure clearly shows the advantage of our designed approach over the traditional versioning method. Please note that the effectiveness of the light-weight approach lies in ensuring data plane consistency during network updates in a resource-efficient manner as compared to traditional versioning which would require double the number of flows to ensure the same when the network state is transitioning. However, when the network state is not transitioning, the light-weight approach is not a means to reduce filters that should exist on switches for correct forwarding of events to interested subscribers as this is a completely different problem that has been addressed in [6].

Once NS_I is deployed in the network, the flows constituting NS_o can be removed as there is already an alternate set of flows connecting each publisher to at least its relevant subscribers through NS_I. This step is depicted in Figure 9(c). Once NS_o is removed, flows constituting NS_n are added to the switches of the network (cf. Figure 9(d)). However, the priority of the flows in NS_n is kept lower than the priority of the flows in NS_I such that any event that was targeted at NS_o matches the flows in NS_I and never those in NS_n. We do so to satisfy the packet coherence property in the context of our approach that prohibits a packet to see a mix of old and new flows belonging to NS_o and NS_n, respectively. Please note that in our approach we provide a form of packet coherence where packet coherence is not considered to be violated if a packet sees a mix of old ($\in NS_o$) and intermediate ($\in NS_I$) flows or a mix of intermediate ($\in NS_I$) and new ($\in NS_n$) flows. This is mainly because the intermediate state semantically applies to both NS_o and NS_n and an event belonging to either is a valid event for NS_I. Once all flows in NS_n have been deployed, the publishers are notified to index events according to the new chosen indexing approach such that events can now be targeted towards the already deployed new network state NS_n. Therefore, at this point the publishers start indexing events according to the new chosen indexing scheme. Please note that the subscribers do not need to be aware of any indexing changes as the design of PLEROMA ensures that subscribers remain oblivious to the underlying details of the middleware [35]. After a given time bound (depending on the bounds on the forwarding latency of the longest path in the network) that ensures that all events matching the old network state NS_o have been delivered to the subscribers, all flows constituting NS_I are removed (cf. Figure 9(e)).

In this way a transition from NS_o to NS_n is performed in the network without violating the blackhole-freedom and packet coherence properties. As a result, the pub/sub system does not suffer from any false negatives. Of course, as, for each incoming port, the fine-grained content filters are temporarily substituted by a single flow representing the entire event-space, the pub/sub system experiences additional false positives temporarily during the transition (cf. Figure 14(c)).

VI. Performance Evaluations

This section is dedicated to evaluating and analyzing the performances of each of the presented approaches. We conduct a series of experiments to measure and compare the overall false positive rate at the subscribers of an SDN-based publish/subscribe system for all the techniques. We, especially, show the impact of different types of workload on the performance of each of the approaches in order to highlight their applicability in various scenarios.

Experimental Setup : For our experiments, we have used a prominent tool for emulating software-defined networks, namely, Mininet [25]. Based on the concept of OS-level lightweight virtualization for network emulation, Mininet enables users to experiment with various topologies and application traffic. We use Mininet to experiment with up to *128* switches and *256* end-hosts on different topologies. At the control plane, we use the popular Floodlight controller that collects subscriptions and events from the data plane and runs the techniques proposed in this paper. Our evaluations include up to 10,000 subscriptions and up to 100,000 events. In order to generate workload, i.e., events and subscriptions we use both synthetic as well as real world data. In synthetic data, a content-based schema containing up to 8 attributes is used where the domain of each attribute varies between the range [0,4095]. With regards to the distributions of subscriptions and events for synthetic data, experiments are performed on two models that have been predominantly used to evaluate pub/sub solutions in the past [28], [12]. The uniform model generates subscriptions and events independent of each other, whereas, the interest popularity model chooses up to 8 hotspot regions around which it generates subscriptions and events using the widely used zipfian distribution. We use synthetic data to especially highlight certain properties of the designed algorithms by adjusting correlation between dimensions, matching traffic variances, number of correlated dimensions, etc. We also use real world workload in the form of stock quotes procured from Yahoo! Finance containing a stock's daily closing

prices [13] to show the performance of our algorithms in a realistic environment. In the following evaluations, we show the effectiveness of our techniques even when the number of available bits for spatial indexing is restricted to just 23 bits as available in IPv4 multicast addresses.

Workload-based Indexing : The first set of experiments evaluates the behavior of the selective indexing (SI) approach when subjected to both uniform as well as zipfian data. Figure 11(a) plots the false positive rate with increasing number of subscriptions for both selective indexing as well as regular indexing (RI) when uniform data is used. Figure 11(b) shows the same when zipfian data is used instead. These plots show that indexing within MBRs has significant benefits over regular indexing. In Figure 11(b), the benefit of indexing within MBRs for fewer number of subscriptions is even more significant as compared to uniform distribution. This is because, in the case of zipfian distribution, precise MBRs can be generated due to the similarity of subscriptions concentrated around hotspots. However, with a large number of subscriptions, it becomes comparable to the benefits of uniform distribution. This is because, when a large number of subscriptions are concentrated around a hotspot, even a slight approximation in representing a subscription may generate a lot of false positives as the number of events around a hotspot is extremely high in the generated zipfian data.

We, also, conducted a set of experiments to evaluate the impact of adaptive spatial indexing (ASI) on false positive rate of a system. We evaluated the effect of this technique when both uniform and zipfian data are used to generate events and subscriptions. Figure 11(c) (uniform) and Figure 11(d) (zipfian) clearly show that in-network filtering gains from the use of adaptive spatial indexing as opposed to regular indexing. For both uniform and zipfian data and for every subscription count, ASI results in a lower false positive rate when compared to RI and the plots behave similar to SI.

Dimension Selection : We conducted a series of experiments to evaluate the behavior of all presented dimension selection algorithms when subjected to various types of workload. In the following experiments, we primarily calculate the false positive rate at the subscribers of the system when the number of selected dimensions are gradually reduced for a specific workload. We also evaluate the runtime of each approach to compare their complexities.

While generating workload (i.e., subscriptions and events), we mainly specify two factors. The first is the variance factor which can be either random or uniform. **Random variance factor** means that the variance of events in certain dimensions may be high whereas they may be low in others and this is decided at random. **Uniform variance factor** signifies similar variance of events across all dimensions. The second factor that we define is the correlation factor. Here, a **high correlation factor** implies high correlation between multiple dimensions while very few dimensions are independent. A **low correlation factor** signifies low correlation between very few dimensions while most dimensions are independent.

The first set of experiments is dedicated to evaluating the performance of the least complex algorithm, Event Variance-based Selection (EVS). These experiments highlight the benefits of dimension selection on reduction of false positives and also show that even a simple approach like EVS performs better than a random dimension selection (RS) approach. Figure 12(a) plots false positive rate when EVS and random selection approaches are employed on multiple datasets having 8 dimensions with a random variance factor. The figure shows that, when EVS is used, reducing dimensions up to a point reduces false positives, however, after that false positives rise again. This is because, for example, in the case of Figure 12(a), EVS benefits by removing 3 less selective dimensions and assigning the additional bits to the 5 more selective dimensions. However, ignoring one or more of these 5 dimensions implies major information loss which again increases the false positive rate. EVS performs better than a random selection approach as it takes advantage of the random variance factor which allows certain dimensions to have higher selectivity than the others.

We evaluated the next set of experiments, however, with uniform variance factor instead of a random variance factor as before. We again plot the performance of EVS in such a scenario and as expected, due to uniform event variance in all dimensions, it does not succeed in reducing false positives as can be seen in Figure 12(b). In fact, its performance can be compared to random selection. However, in such a scenario, the Event Match Count-based Selection (EMCS) approach performs much better than EVS, providing a significant benefit in terms of reduction of the false positive rate (cf. Figure 12(b)). When event distribution alone cannot differentiate between selectivity of dimensions, then it is necessary to look at both events and subscriptions to determine selectivity and this is the reason why EMCS performs much better in this case.

EMCS works very well in the previous scenario. However, in the following experiments we compare its performance to Correlation-based Selection (CS) when the correlation factor is both high and low. Figure 12(c) plots false positive rate when selected dimensions are gradually reduced for data with high correlation factor. The figure clearly shows that CS gains significantly over EMCS in the presence of high correlation as CS is designed to remove the redundancy in data due to correlation. When the correlation factor is low, quite understandably EMCS and CS perform similarly as depicted in 12(d). However, please note that even with low correlation CS does not perform worse than EMCS.

The next set of experiments compares the performance of the greedy selection (GS) algorithm with CS when a high correlation factor is used while data generation. Figure 12(e) shows that GS outperforms CS even in the very best case for CS, i.e., high correlation. Since GS is an evaluation-based technique, it performs in most cases better than the other techniques and is very close to the performance of ideal selection, i.e., Brute-Force Selection (BRS), as can be seen in Figure 12(f) and Figure 12(g) for uniform and zipfian data, respectively. BRS, of course, produces the most optimal set of dimensions, however, as can be seen from the results, the performance of GS is almost equivalent to this optimal.

The dimension selection evaluation results show the performance of the selection algorithms in increasing order of effectiveness, i.e., EVS, EMCS, CS, GS, and BRS. However, better the performance, higher is the time complexity of

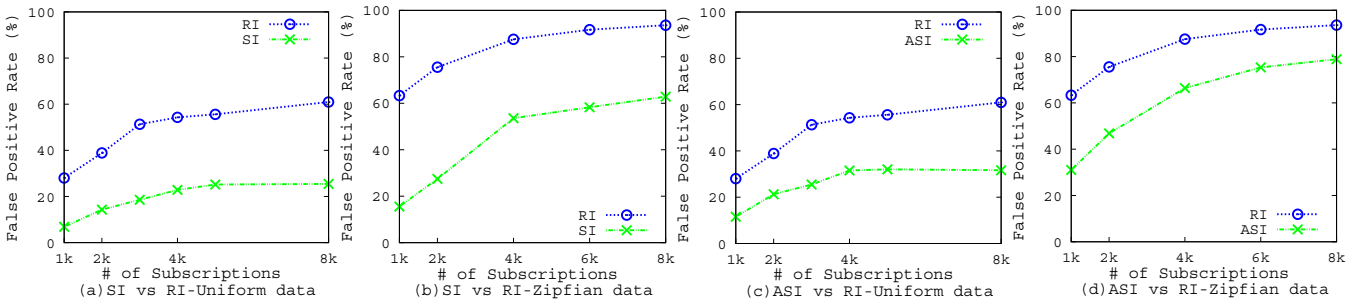


Fig. 11: Performance Evaluations: Workload-based Indexing

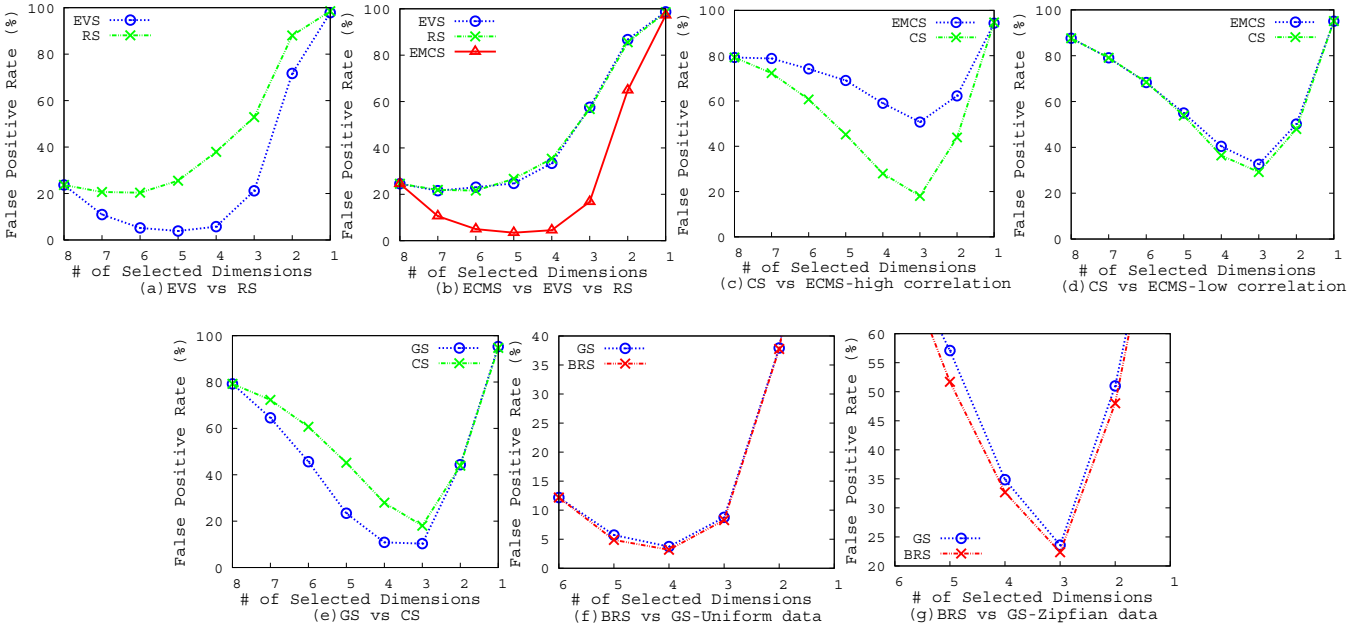


Fig. 12: Performance Evaluations: Dimension Selection - False Positive Rate

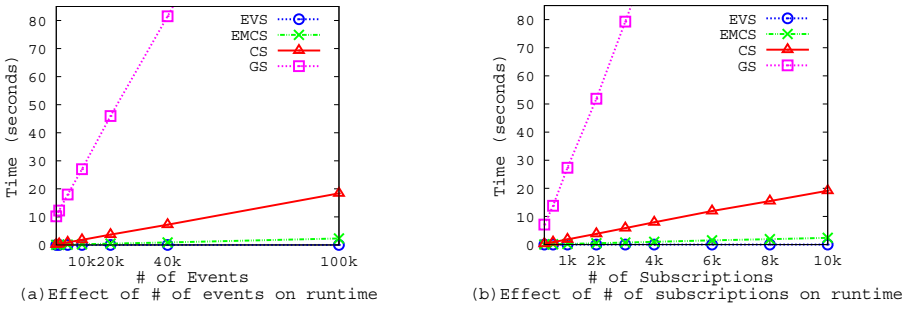


Fig. 13: Performance Evaluations: Dimension Selection - Runtime Overhead

the selection algorithm. This is visible in the next set of experiments that we conducted. The first set of experiments shows the impact of increasing the number of events on the time required to select a set of 4 dimensions from a set of 8 dimensions when the number of subscriptions is fixed to 1000. Figure 13(a) clearly shows that EVS and EMCS require least computation time (in the order of milliseconds), whereas CS takes significantly more time than them with GS requiring most. Similarly, the impact of number of subscriptions on computation time, with the event count set to 100,000, can be seen in Figure 13(b). As expected, again EVS performs fastest, followed by EMCS, CS, and finally GS.

Combining Approaches : The next set of experiments are dedicated to highlighting the effect of combining various algorithms. We used zipfian distribution to generate data for these experiments with a random selectivity factor. Figure 14(a) shows the performance of CS and GS both independently and when combined with adaptive spatial indexing. As expected, the combinations perform much better than CS or GS alone. In fact, for GS+ASI, the false positive rate goes down from 80% (if regular indexing is performed on 8 dimensions) to merely 3.33%.

To ensure that our approaches are effective in realistic

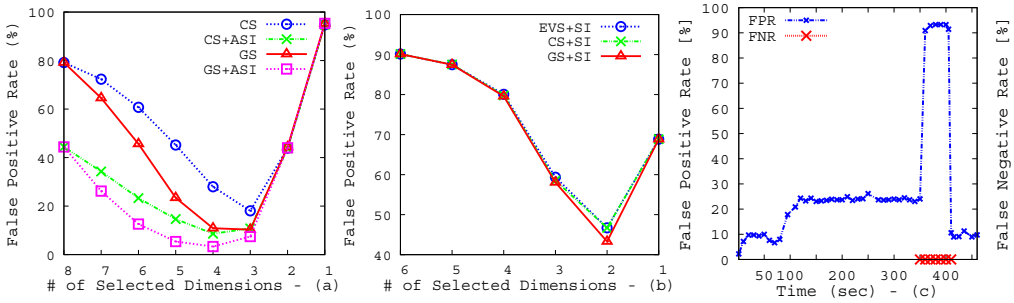


Fig. 14: Performance Evaluations: Combined Approaches and Dynamism

scenarios, we conducted experiments to show their effects on real world stock data. As can be seen in Figure 14(b), our algorithms are capable of significantly reducing false positives in a real world system. This time we combine EVS, CS, and GS with selective indexing. The plots show that even an approach combined with EVS reduces false positive rate by 48% when 2 dimensions are selected. Also, in this case, GS, CS, and EVS, when combined with SI, have very similar performances. GS successfully reduces the false positive rate by up to 53%. These evaluation results further highlight the applicability of the approaches presented in this paper.

Our evaluations show that out of the various combinations, GS+ASI and GS+SI have the best performance (we do not consider BRS as it is not a scalable solution). This is expected considering the fact that, irrespective of the event and subscription distribution, GS always has the best performance among the dimension selection techniques. However, between SI and ASI, the event distribution and current subscriptions determine the better technique.

Handling Dynamics : The final set of experiments have been conducted to show the dynamic behavior of the system with the passage of time in Figure 14(c). Therefore, we start evaluating false positive rate for a system on which CS has been recently employed and then plot its behavior with changing dynamics over time. Initially, the false positive rate is quite low due to the recent execution of CS that selected dimensions based on the recent traffic distribution. However, around the 95^{th} second the traffic distribution changes because of which the dimensions chosen in the previous period become less effective. As a result, the false positive rate goes up significantly in the system. Please note that, in our system, the dimension selection algorithms may be periodically executed on the system or may be executed once the false positive rate in the system exceeds a threshold value. In fact, in this set of experiments, we consider a system where CS is periodically executed. As a result, around the 350^{th} second, CS is again executed and indexing is done based on the current selected dimensions chosen according to the current traffic distribution. Now, new flows are installed in the system following the light-weight approach. Here, we define the term false negative rate as follows. **False negative rate** is the percentage of events dropped in the network that should have been forwarded to interested subscribers. Our evaluation results confirm that there are no false negatives in the system when LWA is employed. As expected, during this process, the false positive rate is very high as the fine-grained filters are temporarily replaced by filters representing the entire event space. However, around the 410^{th} second, the deployment of the new flows is complete and the false positive rate goes down significantly as indexing is now according to the current traffic distribution.

VII. Related Work

The past decade has seen a significant amount of effort devoted to the realization of scalable and efficient pub/sub systems [30], [12], [18], [32]. The primary focus of most of these systems has been efficient communication that ensures scalability and, also, preserves expressiveness of content in order to avoid unnecessary traffic in the system. A very widely used technique employed in overlay networks is subscription clustering where events are flooded within clusters [30], [12], [31]. These approaches have very significant problems. Clusters have to be recomputed in the presence of subscription/advertisement churn and, depending upon the subscriptions, there might be a large number of false positives in the system (e.g., in the scenario of dissimilar subscriptions) [36]. Moreover, these approaches do not allow fine-grained filtering within a cluster. For example, Riabov et al. perform clustering for content-based pub/sub systems by grouping subscribers into multicast channels and performing IP multicast thereafter [30]. However, this approach largely depends on the similarity of subscriptions within generated clusters and may fail to ensure minimal false positives as multicasting is eventually employed within a cluster. Please note that, in this paper, we are the first to combine the concept of subscription clustering (essentially an overlay-level mechanism) with in-network filtering on a software-defined network to avoid unnecessary traffic. Such a combination largely preserves expressiveness of a content-based subscription model.

Linearizing content-space (e.g. hashes, bit strings, etc.) for fast matching of events with subscriptions while balancing the load in structured P2P overlay network has been much researched in the past [4], [28]. Most of these works are based on distributed hash tables (DHT) that are load-balanced and self-organizing. Baldoni et al. in [4], realize a Chord-based [33] publish/subscribe where events and subscriptions are mapped to bit strings. Where on one hand, [4] maps subscriptions and events to multiple nodes, on the other hand, Muthusamy et al. in [28] design a protocol that primarily indexes subscriptions at a single node. Of course, linearizing content-space is also of extreme relevance to content-based routing in software-defined networks. However, in order to

directly employ the above techniques, the SDN-compliant switches would have to support far more expressive operations. As a result, none of these linearizing techniques can be directly deployed in an SDN-based pub/sub system.

While attempting efficient content-based routing, considerable work has been dedicated to subscription summarization techniques that compact subscription information. With regards to this, various data structures and matching algorithms have been developed. For example, Jerzak et al., in [20], use Bloom filters [11] to encode subscriptions and events. While this expedites content-based routing, it suffers from the inherent limitations of a Bloom filter with respect to presence of considerable amount of false positives in the system. Again, the system MICS [18] uses Hilbert space filling curve to generate a one-dimensional representation of events and subscriptions. However, MICS too suffers from false positives in the system.

The above systems primarily work on overlay networks. However, the recent past has seen the use of networking technologies such as NetFPGA and SDN to realize filtering of events in the network layer. For example, LIPSIN [22] uses Bloom filters to encode the routing path of an event in its packet header. This enables a packet to be routed directly on the network layer. However, since a packet header is limited in size, LIPSIN uses a limited fixed length Bloom filter for encoding, which results in false positives. Similarly, systems [35] that exploit the capabilities of SDN to achieve line-rate forwarding of events also suffer from the limitations of hardware and are subjected to unnecessary traffic. Recently, Bhowmik et al. [8] have proposed a hybrid pub/sub middleware which may be used to offload some of the content filters from switches in a software-defined network to the application layer, resulting in filtering of events in both software and hardware. However, such a middleware loses some of the advantages of a pure network layer implementation. In comparison, the techniques in this paper improve bandwidth efficiency and also benefit from line-rate forwarding.

VIII. Conclusion

In this paper, we address the limitations of an SDN-based publish/subscribe middleware with respect to bandwidth efficiency. We present a series of algorithms that improve the performance of such a system and provide extensive evaluation results to analyze their behavior. The proposed techniques complement and can build on top of each other to considerably impact unnecessary traffic in the middleware. Our evaluation results show that these strategies can significantly reduce false positive rate in the system (up to 97%) when subjected to various kinds of workload. Each of these algorithms preserve the benefits of using SDN for pub/sub by ensuring line-rate forwarding of events directly on switches while also preserving the benefits of content-based routing by focusing on bandwidth-efficient communication. Moreover, evaluation results, also, show the effectiveness of our proposed light-weight approach in completely avoiding false negatives in the system during dynamic network updates.

References

[1] Correlation in Stock Exchange Data. http://www.investopedia.com/articles/technical/02/010702.asp.

[2] Intel Research Berkeley Lab Sensor Data Set. http://www.cs.cmu.edu/~guestrin/Research/Data/.

[3] Report from Open Networking Summit: Achieving Hyper-Scale with Software Defined Networking, 2015.

[4] R. Baldoni, C. Marchetti, A. Virgillito, and R. Vitenberg. Content-based publish-subscribe over structured overlay networks. In *Proc. of the 25th IEEE Int. Conf. on Distributed Computing Systems*, ICDCS '05, 2005.

[5] R. Barazzutti, P. Felber, C. Fetzer, E. Onica, J.-F. Pineau, M. Pasin, E. Rivière, and S. Weigert. Streamhub: A massively parallel architecture for high-performance content-based publish/subscribe. In *Proc. of the 7th ACM Int. Conf. on Distributed Event-based Systems*, 2013.

[6] S. Bhowmik, M. A. Tariq, A. Balogh, and K. Rothermel. Addressing TCAM limitations of software-defined networks for content-based routing. In *Proceedings of the 11th ACM International Conference on Distributed and Event-based Systems*, DEBS, 2017.

[7] S. Bhowmik, M. A. Tariq, J. Grunert, and K. Rothermel. Bandwidth-efficient content-based routing on software-defined networks. In *Proc. of the 10th ACM Int. Conf. on Distributed and Event-based Systems*, DEBS, 2016.

[8] S. Bhowmik, M. A. Tariq, L. Hegazy, and K. Rothermel. Hybrid content-based routing using network and application layer filtering. In *Proc. of the 36th IEEE Int. Conf. on Distributed Computing Systems*, 2016.

[9] S. Bhowmik, M. A. Tariq, B. Koldehofe, F. Dürr, T. Kohler, and K. Rothermel. High performance publish/subscribe middleware in software-defined networks. In *IEEE/ACM Trans. on Networking*, 2016.

[10] S. Bianchi, P. Felber, and M. G. Potop-Butucaru. Stabilizing distributed r-trees for peer-to-peer content routing. *IEEE Trans. Parallel Distrib. Syst.*, 21(8):1175–1187, 2010.

[11] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Comm. of the ACM*, 1970.

[12] C. Chen, Y. Tock, and H. Jacobsen. Overlay design for topic-based publish/subscribe under node degree constraints. In *36th IEEE International Conference on Distributed Computing Systems, ICDCS*, 2016.

[13] A. K. Y. Cheung and H. Jacobsen. Green resource allocation algorithms for publish/subscribe systems. In *2011 International Conference on Distributed Computing Systems, ICDCS*, 2011.

[14] O. M. E. Committee. *Software-defined Networking: The New Norm for Networks*. Open Networking Foundation, 2012.

[15] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee. Devoflow: Scaling flow management for high-performance networks. In *Proc. of the ACM SIGCOMM 2011 Conf.*

[16] C. Dong, Q. Xiuquan, J. Gelernter, L. Xiaofeng, and M. Luoming. Mining data correlation from multi-faceted sensor data in the internet of things. In *China Comm.*, 2011.

[17] V. Gaede and O. Günther. Multidimensional access methods. *ACM Comput. Surv.*, 30(2), 1998.

[18] H. Jafarpour, S. Mehrotra, N. Venkatasubramanian, and M. Montanari. MICS: An Efficient Content Space Representation Model for Publish/Subscribe Systems. In *Proc. of the 3rd ACM Int. Conf. on Distributed Event-Based Systems*, DEBS '09.

[19] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat. B4: Experience with a globally-deployed software defined wan. In *Proc. of the ACM SIGCOMM 2013 Conf.*

[20] Z. Jerzak and C. Fetzer. Bloom filter based routing for content-based publish/subscribe. In *Proc. of the 2nd Int. Conf. on Distributed Event-based Systems*, 2008.

[21] X. Jin, H. H. Liu, R. Gandhi, S. Kandula, R. Mahajan, M. Zhang, J. Rexford, and R. Wattenhofer. Dynamic scheduling of network updates. In *Proceedings of the 2014 ACM Conference on SIGCOMM*.

[22] P. Jokela, A. Zahemszky, C. Esteve Rothenberg, S. Arianfar, and P. Nikander. LIPSIN: line speed publish/subscribe inter-networking. *ACM SIGCOMM Computer Communication Review*, 2009.

[23] I. Jolliffe. *Principal Component Analysis*. Springer Verlag, 1986.

[24] K. LaCurts, S. Deng, A. Goyal, and H. Balakrishnan. Choreo: Network-aware task placement for cloud applications. In *Proc. of the 2013 Conf. on Internet Measurement*, IMC '13.

[25] B. Lantz, B. Heller, and N. McKeown. A network on a laptop: Rapid prototyping for software-defined networks. In *Proc. of 9th ACM Wshop. on Hot Topics in Networks*, 2010.

[26] M. Li, F. Ye, M. Kim, H. Chen, and H. Lei. A scalable and elastic publish/subscribe service. In *Proc. of IEEE Int. Parallel & Distributed Processing Symp.*, 2011.

[27] A. Malhi and R. X. Gao. PCA-based feature selection scheme for machine defect classification. *IEEE T. Instrumentation and Measurement*, 2004.

[28] V. Muthusamy and H.-A. Jacobsen. Infrastructure-free content-based publish/subscribe. *IEEE/ACM Trans. Netw.*, 22(5):1516–1530.

[29] M. Reitblatt, N. Foster, J. Rexford, C. Schlesinger, and D. Walker. Abstractions for network update. In *Proc of the ACM SIGCOMM Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communication*, 2012.

[30] A. Riabov, Z. Liu, J. L. Wolf, P. S. Yu, and L. Zhang. Clustering algorithms for content-based publication-subscription systems. In *Proc. of the 22nd Int. Conf. on Distributed Computing Systems*, 2002.

[31] P. Salehi, C. Doblander, and H. Jacobsen. Highly-available content-based publish/subscribe via gossiping. In *Proc. of the 10th ACM Int. Conf on Distributed and Event-based Systems, DEBS*, 2016.

[32] P. Salehi, K. Zhang, and H. Jacobsen. Popsub: Improving resource utilization in distributed content-based publish/subscribe systems. In *Proc. of the 11th ACM Int. Conf. on Distributed and Event-based Systems, DEBS*, 2017.

[33] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proc. of the 2001 Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '01, 2001.

[34] M. A. Tariq. *Non-functional requirements in publish, subscribe systems*. PhD thesis, University of Stuttgart, 2013.

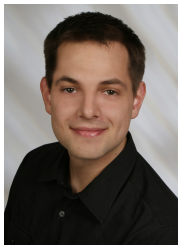
[35] M. A. Tariq, B. Koldehofe, S. Bhowmik, and K. Rothermel. PLEROMA: A SDN-based high performance publish/subscribe middleware. In *Proc. of the 15th ACM/IFIP/USENIX International Middleware Conference*, Middleware '14, pages 217–228, 2014.

[36] M. A. Tariq, B. Koldehofe, G. G. Koch, and K. Rothermel. Distributed spectral cluster management: A method for building dynamic publish/subscribe systems. In *Proc. of the 6th ACM Int. Conf. on Distributed Event-Based Systems*, 2012.

[37] X. Yang and Y. Hu. A DHT-based infrastructure for content-based publish/subscribe services. In *Proc. of Int. Conf. on Peer-to-Peer Computing (P2P)*, 2007.

Sukanya Bhowmik received her doctoral degree from University of Stuttgart, Germany, in 2017. She is currently working as a postdoctoral researcher at the Distributed Systems research group, University of Stuttgart. Her research interests include high performance communication middleware using software-defined networking with focus on line-rate performance, bandwidth efficiency, and control plane distribution.

Muhammad Adnan Tariq received his doctoral degree from the University of Stuttgart, Germany. He worked as a postdoctoral researcher at the Distributed Systems department of the University of Stuttgart, where he was involved in the projects related to data stream processing, complex event processing, software-defined networking, and geo-distributed cloud computing. Currently he is an Assistant Professor at National University of Computing and Emerging Sciences, Pakistan.



Jonas Grunert studied Bachelor of Science and Master of Science in Software Engineering at the University of Stuttgart. During that time he worked as research and teaching assistant for the distributed systems and reliable software departments. He was involved at university open source projects and worked as free software developer in 3D graphics. Now he is working as a software engineer at Google in Munich.

Deepak Srinivasan is an M.Sc. graduate from University of Stuttgart. He is a software developer interested in Distributed Systems and Cloud Computing & Automation.

Kurt Rothermel received his doctoral degree in Computer Science from University of Stuttgart in 1985. From 1986 to 1987 he was a Post-Doctoral Fellow at IBM Almaden Research Center in San José, U.S.A., and then joined IBM's European Networking Center in Heidelberg. Since 1990 he is a Professor for Computer Science at the University of Stuttgart. From 2003 to 2011 he was head of the Collaborative Research Center Nexus (SFB 627), conducting research in the area of mobile context-aware systems. His current research interests are in the field of distributed systems, computer networks, and mobile systems.