Bandwidth-Efficient Content-Based Routing on Software-Defined Networks

Sukanya Bhowmik, Muhammad Adnan Tariq, Jonas Grunert, Kurt Rothermel
University of Stuttgart, Germany
{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 datacenters is extremely desirable. In this respect, the very popular Software-defined Networking (SDN), 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 w.r.t. no. 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 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. Moreover, these techniques complement each other and can be combined together to further enhance performance. Our detailed performance evaluations show the potential of these techniques in reducing unnecessary traffic when subjected to different workloads.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Distributed networks; C.2.4 [**Distributed Systems**]: Distributed applications

Keywords

SDN, Publish/Subscribe, Content-based Routing

1. 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 datacenters 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 datacenter (DC) WAN, B4 [10]. Microsoft, too, has been using SDN to flexibly and reliably operate Microsoft Azure [2]. 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 [7]. While SDN has been extensively considered for dynamic resource sharing, WAN VPN, etc., across datacenters, the potential of SDN to realize content-based pub/sub, the backbone of datacenter communication, has also been explored in literature [17, 13]. The SDN-based pub/sub middleware PLEROMA [17] uses the power of SDN to enable in-network filtering of published events directly on SDN-compliant switches, resulting in line-rate performance.

Even though the dissemination of content between publishers and subscribers in a software-defined network proves to be resource-efficient, 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 fil-

DEBS '16, June 20-24, 2016, Irvine, CA, USA
© 2016 ACM. ISBN 978-1-4503-4021-2/16/06. . . $15.00
DOI: http://dx.doi.org/10.1145/2933267.2933310

tering 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 the Ternary Content Addressable Memory (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 [12], 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 [14]. As a result, this paper focuses on exploring techniques that address concerns with bandwidth efficiency in the context of content-based filtering on hardware switches. First, we propose a technique—workload-based indexing—that considers workload in the system, in terms of 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. Our evaluations show that a significant amount of unnecessary traffic can be avoided by employing each of these techniques while benefiting from the advantages of SDN in terms of reduced end-to-end latency, high throughput, etc.

2. PRELIMINARIES AND LIMITATIONS

In this section, we provide an overview of PLEROMA [17], a content-based pub/sub middleware realized on SDN, followed by a discussion on the limitations it faces.

An SDN-based Pub/Sub 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 flow table entries on TCAM of switches along optimized paths between publishers and their interested subscribers. For example, in Figure 1, the publisher P and the subscriber S send an advertisement and a subscription respectively to the controller which installs content filters along the path between them. This enables header-based matching of 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

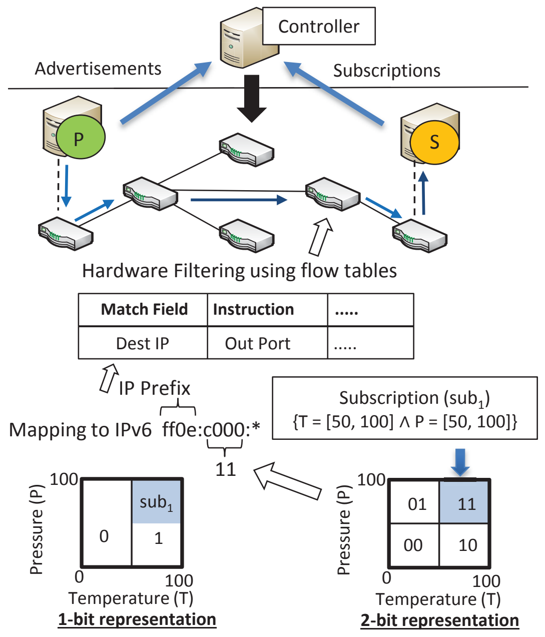

Figure 1: SDN-based Pub/Sub Middleware

and advertisements and subscriptions (i.e., content filters) are conjunction of filters on these attributes. The mapping of content filters to the flow table entries is an integral part of the system and primarily consists of two steps.

In order to represent values and ranges of values (filters) along attributes in flow table entries, the first step is to convert content into binary form. This can be achieved using various techniques such as bloom filters [6], spatial indexing [17], etc. In this paper, we specifically look at spatial indexing where an event-space (denoted by Ω) is modeled as a 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.

These dzs have certain characteristic properties based on the subspaces they represent. For example, the shorter a dz, the larger is the subspace it represents. This is visible in Figure 1 where the dz $\{00\}$ represents a subspace smaller than and contained by the subspace $\{0\}$. So, more expressive a content filter, more fine granular is the recursive binary decomposition and longer are the resulting dzs. 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. 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 (*).

We further explain the two-step mechanism of content representation with an example from Figure 1. Let us assume that subscriber S has a subscription sub_1 : {T = [50, 100] $\wedge$ 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 converted into an IPv6 address (ff0e:c000:*) and installed as a destination IP in the match field of flows on the switches, enabling hardware filtering of events along the path between publisher P and subscriber S.

Limitations of Content Representation: From the above description, 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 1. This implies that all events matching the entire subspace of {1} in the figure will be received by subscriber S. So, the path between P and S will be subjected to a lot of unnecessary traffic which we will henceforth refer to as *false positives*. More specifically, we define false positives as those events which should be filtered out by the network but, nevertheless, are received by uninterested subscribers due to limitations of content filters on switches. Here, we also define the term *false positive rate* as the percentage of total number of events received at the subscribers that are unnecessary (i.e., false positives). Note, the length of dzs, required to accurately represent content, increases with the increase in the number of dimensions in the system.

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 and are implemented by the controller. 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 up/out to distribute this additional overhead among multiple controllers while guaranteeing the notion of a logically centralized controller as achieved in [4].

Note that, although we focus on spatial indexing, other indexing techniques (e.g., Bloom filters, hashes) will encounter the same problems and the proposed techniques in this paper are applicable in general to all indexing mechanisms.

3. WORKLOAD-BASED INDEXING

The effectiveness of encoding content into binary form has primarily depended on two parameters—the size of event-space and the number of available bits. 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, larger the size of Ω, less fine granular is the indexing. However, it

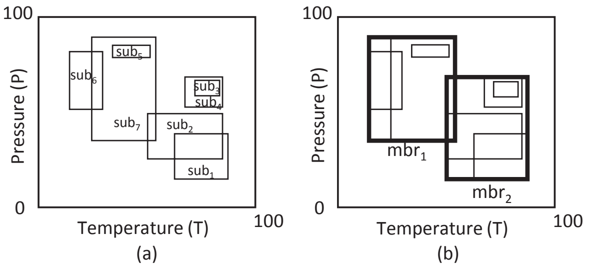

Figure 2: Workload-based Indexing

should be noted that regular spatial indexing partitions the entire space into subspaces, even those subspaces that are of no interest to any subscriber. What if the entire event-space Ω does not get indexed? What if empty subspaces w.r.t. subscription distribution in Ω are left out and the bit strings earlier assigned to these empty spaces used for more fine-granular indexing of the populated subspaces? So, to this end, we introduce the workload-based indexing approach (WI) where the main idea is to identify meaningful subspaces w.r.t. subscriptions in Ω and only index those subspaces instead of indexing the entire event-space.

The first step in WI 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 [16, 5]. 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 termed minimum bounding rectangles or MBRs. The set of generated MBRs encloses all subscriptions in the system and attempts to leave out as much empty space as possible. Figure 2 illustrates the concept of MBR. Here, the subscriptions are distributed in Ω, as illustrated in Figure 2(a). Figure 2(b) shows two MBRs covering all subscriptions in the system clustered together in two groups on the basis of similarity. Note that, even though two MBRs may partially overlap, as in Figure 2(b), a subscription strictly belongs to a single MBR.

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 but, 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, it 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. So, 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.

This approach allows for more fine granular spatial indexing as it can avoid assigning bits to the subspaces in Ω

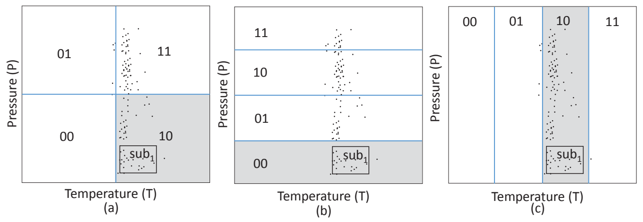

Figure 3: Effects of event distribution

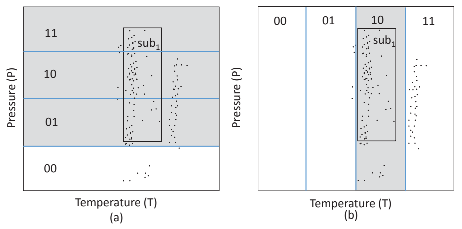

Figure 4: Event-based Selection

that are not part of any subscription in the system, thus allowing the use of more bits to represent more meaningful subspaces. 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 workload-based 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 2(b), mbr_1 and mbr_2 overlap. In such a scenario, an event that lies in the overlapping subspace must be indexed w.r.t. both MBRs as it can match subscriptions from both.

4. DIMENSION SELECTION

As discussed before, more the number of dimensions in a system, longer are the dzs. However, what if there was no need to index every dimension? What if the available bits could be used to perform fine granular spatial indexing only on a subset of dimensions that prove to be more promising w.r.t. bandwidth efficiency? While this notion was briefly introduced in [17], 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 w.r.t. realistic workload distributions.

4.1 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 denotes the set of all events in Ω, then event variance along a dimension d is measured as $(\sum(x_i^d - \overline{x^d})^2)/|E^t|$ where x_i^d represents the value of the i^{th} event along dimension d. We illustrate this with a very simple example in Figure 3 w.r.t. 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 3(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 3(b). In Figure 3(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 3(c) clearly shows that sub_1 would be subjected to more false positives as compared to not only indexing along dimension P but also indexing along both dimensions. This example clearly indicates the importance of event distribution within Ω in dimension selection.

So, the very first dimension selection algorithm that we present is Event Variance-based Selection (EVS). 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 w.r.t. 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. 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 4(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 4(b). Here, false positives are lesser in the latter case. This clearly indicates that both events as well as subscriptions play a major role in the selection process.

4.2 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 re-

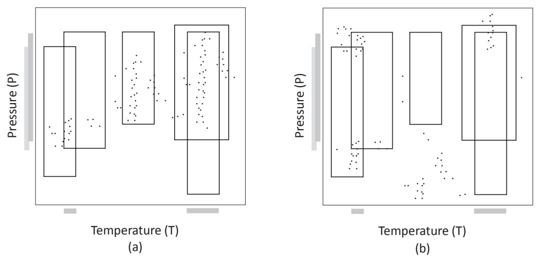

Figure 5: Subscription-based Selection

ducing its importance w.r.t. the ability to reduce false positives. For example, Figure 5(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 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 lesser overlaps but also the distribution of events. For example, in Figure 5(b), we have the same subscription overlaps as before, however, due to the distribution of events, in this case, 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 w.r.t. 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 $\mathbb{SD}$. This algorithm is more generic than the previous one but has a higher time complexity of $O(\omega * \psi * s)$.

4.3 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 in a system 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 [1] that has a 54 node sensor network measuring values for temperature, humidity, and light shows positive correlation among all the 3 attributes [8]. 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 w.r.t selectivity.

The main steps of the algorithm are outlined as follows. 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 $(\omega * \omega)$ 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 w.r.t. 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 w.r.t. events consumed by subscriptions along each dimension. 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 an event matches along both dimensions of the dimension pair. More formally, the similarity factor (sf) is calculated as $sf_{e_k}^{i,j} = (|S_{e_k}^{d_i} \cap S_{e_k}^{d_j}|)/|\mathbb{S}|$. As before, here, $S_{e_k}^{d_i}$ represents the set of subscriptions matched by event e_k along dimension d_i. The similarity factors of all events are aggregated for the dimension pair and the inverse effect of the summed up value is considered to measure the dissimilarity between the two dimensions in order to calculate the covariance between them. So, finally, $c_{i,j}$ is calculated as 1.0 - $\sum_{e_k \in E^t} sf_{e_k}^{i,j}/|E^t|$. This value indicates the covariance between a dimension pair in terms of number of events matching subscriptions along dimension pairs. Along the diagonal of $\mathbb{C}$, the variance of this same match of events with subscriptions within each dimension gets captured in the process.

Once $\mathbb{C}$ is calculated, the technique of principal component analysis (PCA) is applied. $\mathbb{C}$ is subjected to spectral analysis through the process of eigendecomposition. 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) followed by the other variances in decreasing order lying on the following axes. The required n dimensions in $\mathbb{SD}$ can be selected by choosing the dimensions from $\mathbb{D}$ that correspond to the first n principal components. 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. Moreover, PCA has a time-complexity of $O(\omega^3)$.

4.4 Evaluation-based Techniques

The previous algorithms, though effective in their own ways, do not give an indication of an ideal value of n. By defining a threshold for the amount of variability of data to be retained, CS does attempt to arrive at an estimate for n (cf. [15] : Step 3). However, the threshold has to be provided by the system administrator and the method is merely a heuristic with not much direct impact on false positives. So, in this subsection, we introduce two algorithms which not only significantly reduce false positives in the system, but 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 but 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 but does not evaluate every combination of dimensions. 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 but 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. 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 one dimension in every step. So, 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. So, 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 $O(\omega^3 * \psi * s)$.

Note, the event distribution and the current subscriptions in the system may change over time, degrading the effectiveness of the proposed techniques. So, the controller must periodically collect workload information, execute proposed techniques, and deploy necessary changes in the network.

5. PERFORMANCE EVALUATIONS

This section is dedicated to evaluating and analyzing the performances of each of the presented techniques. We conduct a series of experiments to measure and compare the overall false positive rate of an SDN-based pub/sub system for all the techniques. We, especially, show the impact of different types of workload on the performance of each of the techniques in order to highlight their applicability in various scenarios. Our evaluations include up to 10,000 subscribers and up to 100,000 events. In order to generate workload, a content-based schema containing up to 8 attributes is used where the domain of each attribute varies between the range [0,4095]. We primarily use two models for the distribution of subscriptions and events to generate data. 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. 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.

The first set of experiments evaluates the behavior of the workload-based indexing (WI) approach when subjected to zipfian data. Figure 6(a) plots the false positive rate with increasing number of subscriptions for both workload-based indexing as well as regular indexing (RI) when zipfian data is used. These plots show that indexing within MBRs has significant benefits over regular indexing. The benefit of indexing within MBRs is quite significant as, due to the similarity of subscriptions concentrated around hotspots in the case of zipfian distribution, precise MBRs can be generated.

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 when the number of selected dimensions are gradually reduced for a specific workload. We also evaluate the runtime of each algorithm 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. In random variance factor, 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

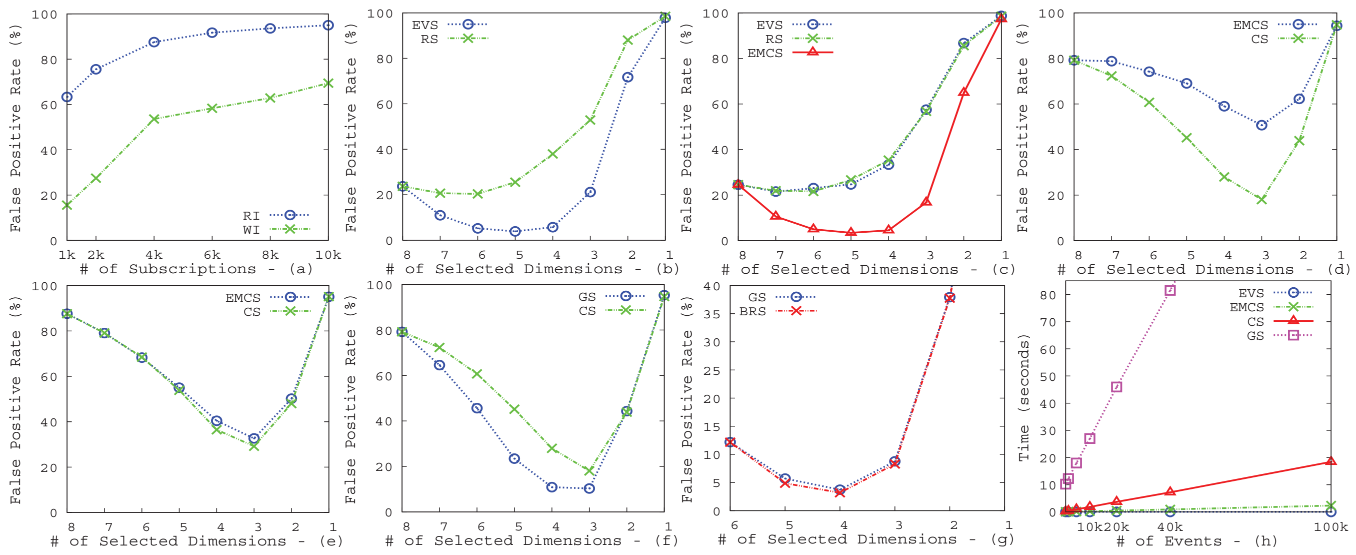

Figure 6: Performance Evaluations

the correlation factor. Here, a high correlation factor implies high correlation between multiple dimensions while very few dimensions are independent whereas a low correlation factor signifies low correlation between very few dimensions while most dimensions are completely independent.

The first set of experiments is dedicated to evaluating the performance of the least complex algorithm, Event Variance-based Selection (EVS). These experiments not only highlight the benefits of dimension selection on reduction of false positives but also show that even a simple approach like EVS performs better than a random dimension selection (RS) approach. Figure 6(b) 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, but, after that false positives rise again. This is because, for example, in the case of Figure 6(b), 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 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 6(c). 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 6(c)). 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 that is 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 6(d) 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. When the correlation factor is low, quite understandably EMCS and CS perform similarly as depicted in 6(e). 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 6(f) 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 6(g). BRS produces the most optimal set of dimensions but, as can be seen from the evaluation 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 the selection algorithm. This is visible in the next set of experiments that we conducted. The experiments show 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 6(h) 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, our experiments also showed that with increasing number of subscriptions or original dimensions, again, EVS performs fastest, followed by EMCS, CS, GS, and finally BRS. In fact, with increasing number of original dimensions, runtime for BRS grows exponentially.

6. RELATED WORK

The past decade has seen a significant amount of effort being devoted to the realization of scalable and efficient pub/sub systems [5, 9, 11, 18]. The primary focus of most of these systems has been efficient communication that not only ensures scalability, but also preserves expressiveness of content in order to avoid unnecessary traffic in the system. A very widely used technique employed to reduce false positives in overlay networks is subscription clustering [18] where events are flooded within clusters. Riabov et al. [16] perform clustering for content-based pub/sub systems by grouping subscribers into multicast channels and performing IP multicast thereafter. However, this approach largely depends on the similarity of subscriptions within generated clusters and may fail to ensure minimal false positives as multicasting is employed eventually within a cluster.

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. [11] use Bloom filters [6] to encode subscriptions and events. While this expedites content-based routing, it suffers from the inherent limitations of a Bloom filter w.r.t. presence of false positives in the system. Again, the system MICS [9] uses Hilbert space filling curve to generate a one-dimensional representation of events and subscriptions. However, MICS too suffers from false positives.

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 on the network layer. For example, LIPSIN [12] 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 such as PLEROMA [17], 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. A hybrid approach to filtering has also been realized in literature where some events are filtered at the application layer while the remaining at the network layer [3]. However, even though such an approach reduces false positives, it has to compromise on line rate performance for paths involved with application layer filtering.

7. CONCLUSION

In this paper, we attempt to mitigate the limitations of an SDN-based pub/sub middleware w.r.t. bandwidth efficiency. The proposed workload-based indexing technique and dimension selection algorithms complement each other and considerably impact unnecessary traffic in the middleware. These techniques 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. Our evaluation results show that each of these techniques can significantly reduce false positive rate in the system when subjected to various kinds of workload.

8. REFERENCES

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

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

[3] 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.

[4] S. Bhowmik, M. A. Tariq, B. Koldehofe, A. Kutzleb, and K. Rothermel. Distributed control plane for software-defined networks: A case study using event-based middleware. In *Proc of the 9th ACM Int. Conf. on Distributed Event-Based Systems*, 2015.

[5] S. Bianchi, P. Felber, and M. Gradinariu. Content-based publish/subscribe using distributed R-trees. In *Proc. of 13th Int. Euro-Par Conf.*, 2007.

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

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

[8] 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.

[9] 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.

[10] 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.*

[11] 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.

[12] 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.

[13] B. Koldehofe, F. Dürr, and M. A. Tariq. Tutorial: Event-based systems meet software-defined networking. In *Proc. of the 7th ACM Int. Conf. on Distributed Event-based Systems*, DEBS '13.

[14] 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.

[15] Y. Lu, I. Cohen, X. S. Zhou, and Q. Tian. Feature selection using principal feature analysis. In *Proc. of the 15th ACM Int. Conf. on Multimedia*, MM '07.

[16] 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.

[17] M. A. Tariq, B. Koldehofe, S. Bhowmik, and K. Rothermel. PLEROMA: A SDN-based high performance publish/subscribe middleware. In *Proc. of 15th Int. Middleware Conf.*, 2014.

[18] 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.