Efficient Content-based Routing with Network Topology Inference

Muhammad Adnan Tariq, Boris Koldehofe, Kurt Rothermel
University of Stuttgart
{first name.last name}@ipvs.uni-stuttgart.de

ABSTRACT

Content-based publish/subscribe has gained high popularity for large-scale dissemination of dynamic content. Yet it is highly challenging to enable communication-efficient dissemination of content in such systems, especially in the absence of a broker infrastructure. This paper presents a novel approach that exploits the knowledge of event traffic, user subscriptions and topology of the underlying physical network to perform efficient routing in a publish/subscribe system. In particular, mechanisms are developed to discover the underlay topology among subscribers and publishers in a distributed manner. The information of the topology and the proximity between the subscribers to receive similar events is then used to construct a routing overlay with low communication cost. Our evaluations show that for internet-like topologies the proposed inference mechanisms are capable of modelling an underlay in an efficient and accurate manner. Furthermore, the approach yields a significant reduction in routing cost in comparison to the state of the art.

Categories and Subject Descriptors

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

Keywords

Content-based, Publish/Subscribe, P2P, Underlay, QoS

1. INTRODUCTION

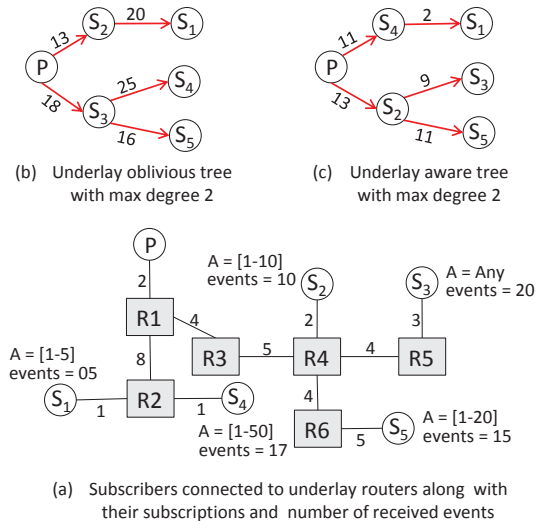
Publish/Subscribe (pub/sub) is an important many-to-many communication paradigm for building large-scale distributed applications such as news distribution, service discovery, stock exchange, electronic auction, network monitoring, environmental monitoring and others. In a pub/sub system, messages are not given explicit destination addresses and are routed according to their content. This enables loose coupling between the producers (*publishers*) and consumers (*subscribers*) of information. Publishers inject information into the system in the form of events without the knowledge of the relevant set of subscribers. Likewise, subscribers express their interest in certain events by issuing subscriptions without the need to know the set of publishers.

In a content-based pub/sub – which provides the most expressive way to specify events of interest, where subscriptions define restrictions on message content – an important concern is to route events from the publishers to the relevant subscribers with low delay and message overhead [25]. Traditionally, a dedicated network of brokers is used for the intermediate routing of events. However, recent systems use peer-to-peer (in short P2P) model, where subscribers and publishers arrange themselves in a broker-less overlay network and participate in forwarding events. The efficiency of event routing in a broker-less environment is very sensitive to the organization of subscriber and publisher peers in an overlay network. Typically, two approaches are used for organizing the peers in such systems. The first approach uses DHT-based overlays [29] to arrange peers. The content-based filtering is implemented as a separate layer on top of DHTs [13]. The efficiency of this approach is restricted as the overlay network is oblivious to the dynamics in the upper content-based layer [14].

The second approach on the other hand uses the information about the event traffic and user subscriptions to organize peers in semantic (or interest) communities [28, 32, 31], i.e., peers matching similar events are placed in close proximity in an overlay network. Such a semantic organization of peers reduces the pure forwarders, i.e., the peers which participate in forwarding an event without a matching subscription, and as a consequence results in a decrease of overall message overhead as well as an improvement in the average number of overlay hops to deliver events. However, the overlay-level mechanisms alone may not provide desired benefits without the knowledge of the underlying physical network topology (in short underlay). For instance, two apparently independent overlay paths may share common underlying physical links [18] and therefore, the selection of overlay paths solely based on the semantic similarities between the peers may lead to multiple copies of the same messages on the shared underlay links resulting in higher message overhead (bandwidth utilization) and higher end-to-end delays, as illustrated in the following simple example.

EXAMPLE 1. *Figure 1(a) shows a small scenario where five subscribers and a publisher are connected to different routers in an IP (*Internet Protocol*) network. The subscribers have non-identical but overlapping subscriptions to*

DEBS'13, June 29–July 3, 2013, Arlington, Texas, USA.



(b) Underlay oblivious tree with max degree 2

(c) Underlay aware tree with max degree 2

(a) Subscribers connected to underlay routers along with their subscriptions and number of received events

Figure 1: A simple example illustrating the benefits of underlay awareness.

a numeric attribute A and thus receive intersecting (or overlapping) sets of event messages. The overlay in Figure 1(b) organizes the subscribers according to the containment relationship between their subscriptions without the knowledge of the underlying network, similar to the work of Tariq et al. [30]. Clearly, no extra message overhead is incurred (to deliver the events received by each subscriber) at an overlay-level, each subscriber only receives the matching events and forwards a subset of those events to its child subscribers. However, the communication cost in terms of number of packets (or messages) travelled on each underlay link is high (i.e., 327 *messages are forwarded in total) because overlay paths induce duplicate messages on common underlay links. Moreover, the delay penalty, i.e., delay in comparison to the unicast delay from the publisher, for individual subscribers is very high, e.g., s_4 experience* 3.9 *times more delay. In contrast, the overlay in Figure 1(c) takes into account both the underlay topology and the containment relationships between the subscriptions for its organization. Here, the subscriber s_2 has to receive and forward additional events to satisfy the subscriptions of s_3 and s_5 resulting in an overhead of* 10 *overlay-level messages. However, the delay penalty of subscribers is reduced to at most* 1.2 *and the communication cost is lowered to* 246 *underlay messages, which clearly shows that the semantic arrangement alone is not always beneficial in reducing message overhead and delay.*

In this paper, we present a novel scheme that exploits the knowledge of event traffic, user subscriptions, and the router-level topology of the underlying network to construct an efficient routing overlay, which minimizes the overall cost of disseminating events in a content-based pub/sub system. Our main contributions can be summarized as follows. First, we develop methods to discover the underlay topology among the peers participating in a pub/sub system. The proposed methods incur low overhead and maintain the topology information in a distributed manner by constructing a *Topology Discovery Overlay* (in short TDO) (cf. Section 4). Second, we propose different strategies to construct a cost efficient pub/sub routing overlay on a TDO, taking into account the underlay topology as well as the event traffic matched between the subscriber peers (cf. Section 5). Finally, through extensive evaluations, we show that the proposed approach performs well on Internet-like topologies, by leading to substantially low routing cost in terms of message overhead and end-to-end delays (cf. Section 6).

2. SYSTEM MODEL AND PROBLEM FORMULATION

We consider a content-based pub/sub system consisting of a set of N distinct peers. The peers have unique identifiers and are connected to the underlying (IP) network at different points through access links. The underlay is modelled as an undirected graph $G^U = (\mathcal{V}^U, \mathcal{E}^U)$, where $\mathcal{V}^U = (\mathcal{R} \cup \mathcal{P})$ represents the set of routers $\mathcal{R}$ and peers $\mathcal{P}$, and $\mathcal{E}^U \subseteq (\mathcal{P} \times \mathcal{R}) \cup (\mathcal{R} \times \mathcal{R}) \cup (\mathcal{R} \times \mathcal{P})$ represents the set of physical links. Each link $\varepsilon^u_{i,j} = (i,j) \in \mathcal{E}^U$ from $i \in \mathcal{V}^U$ to $j \in \mathcal{V}^U$ is associated with a delay value $d^U(\varepsilon^u_{i,j})$.

The overlay network is the virtual topology induced by the peers on the underlay. It can be modelled as a graph $G^O = (\mathcal{P}, \mathcal{E}^O)$, where $\mathcal{E}^O \subseteq \mathcal{P} \times \mathcal{P}$ is the set of overlay links. An overlay link is the point-to-point connection between two peers p and q, it is mapped to an *underlay route* consisting of a sequence of physical routers $r_i \in \mathcal{R}$ and physical links determined by IP routing, i.e., $\langle p,q \rangle = \{(p,r_i),(r_i,r_j).....(r_m,q)\}$. The delay of an overlay link $\langle p,q \rangle$ corresponds to the unicast path delay from peer p to q, i.e., $d^O(p,q) = \sum_{\varepsilon^u \in \langle p,q \rangle} d^U(\varepsilon^u)$. Note that the IP network routing is usually asymmetric [18] and therefore, the underlay route from a peer p to another peer q (i.e., $\langle p,q \rangle$) may not be the reverse of the route followed by $\langle q,p \rangle$. The methods developed in this paper, work with asymmetry by treating $\langle p,q \rangle$ and $\langle q,p \rangle$ as two separate overlay links. However, for the ease of presentation, we assume that the underlay routes are symmetric and $d^O(p,q) = d^O(q,p)$. A prefix relation can be defined between peers w.r.t. the overlay links from a common ancestor peer. Let $\langle p,q \rangle = \{(p,r_{q,1}),(r_{q,1},r_{q,2}),....,(r_{q,j},q)\}$ and $\langle p,s \rangle = \{(p,r_{s,1}),(r_{s,1},r_{s,2}),....,(r_{s,j+1},s)\}$ represent overlay links of p with q and s respectively. Then $\langle p,q \rangle$ is said to be a prefix of $\langle p,s \rangle$ w.r.t. p, denoted by $q \rightsquigarrow^p s$, iff $r_{q,i} = r_{s,i}, \forall i \in \{1,...,j\}$ holds. Moreover, the neighbourhood of a peer p in the overlay network is defined as the set of peers to which this peer has direct point-to-point connections (overlay links) and is denoted by $NG(p)$.

Peers act as publishers and/or subscribers, and run a content-based pub/sub protocol to exchange events. The pub/sub overlay is maintained as a spanning tree T of G^O, i.e., $T = (\mathcal{P}, \mathcal{E}^O_T)$, where $\mathcal{E}^O_T \subseteq \mathcal{E}^O$. In the following, we will refer to T as pub/sub overlay or tree. A *path* in T connects a peer p to another peer q over intermediate peers and is defined by a set of overlay (tree) links, i.e., $path(p,q) = \{\langle p,p_i \rangle, \langle p_i,p_j \rangle, ..., \langle p_m,q \rangle\}$. Furthermore, each path is associated with a delay value $D(p,q) = \sum_{\langle i,j \rangle \in path(p,q)} d^O(i,j)$. The dissemination of an event over T induces *routing cost* in terms of delay and message overhead. The routing cost is influenced by three factors, i) rate of false positives, ii) stress on physical links, and iii) relative delay penalty.

False positives measure the excess bandwidth consumption (in terms of extra messages) induced by the pub/sub tree T during the dissemination of events. They are defined as the rate of events that peers receive and forward with-

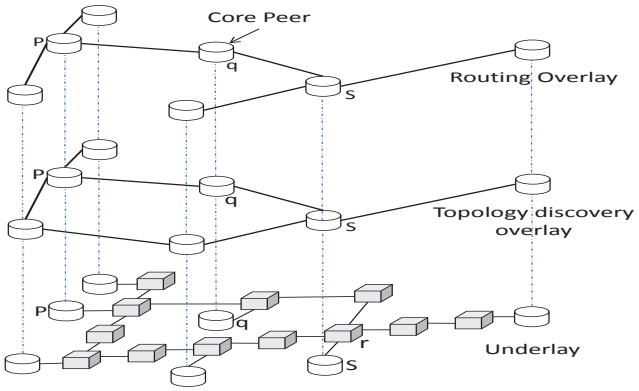


Figure 2: Decomposition into different layers.

out a matching subscription. Let E_ξ be the set of last ψ events published in the system. Each event $e \in E_\xi$ published by a peer p_e is delivered to a set of matching (subscriber) peers S_e by traversing only those links in T that lie on the paths between p_e and S_e. The *routing load* of a link $\langle i,j\rangle \in \mathcal{E}_T^O$, denoted by $\Upsilon(i,j)$, is defined as the number of overlay paths that pass through the link for the dissemination of all the events in E_ξ, i.e., $\Upsilon(i,j) = \sum_{e\in E_\xi} |\{q \in S_e : \langle i,j\rangle \in path(p_e,q)\}|$. Ideally to avoid false positives, the routing load on each link should be induced by the dissemination of only those events that are matched by the subscriptions of both of its adjacent peers.

The *Stress* of a physical link (i,j) is defined as the number of identical copies of a message sent over that link. Similar to false positives, stress measures the excess bandwidth usage and is influenced by the organization (topology) of peers in the pub/sub tree T. Ideally, a message should traverse each physical link at most once.

Relative delay penalty (in short RDP) measures the additional delay introduced by the pub/sub tree T on the delivery of an event from a publisher to all relevant subscribers. It is defined as the ratio of the delay experienced when sending events using the overlay to the delay experienced when sending events using the direct unicast path in the underlay [15]. The RDP value of 1.0 means that the delay at the overlay and the underlay is exactly same.

Given a dynamic set of (subscriber and/or publisher) peers $\mathcal{P}$ and continuously evolving event traffic E_ξ, our objective is to maintain a pub/sub tree T such that

1. the stress induced by the pub/sub tree on the links in the underlay is minimized, and
2. the cost for disseminating events is minimized in terms of false positives and delay, i.e., $\text{cost}(T,\mathcal{P}) = \min \sum_{\langle p,q\rangle\in\mathcal{E}_T^O} \Upsilon(p,q)\ d^O(p,q)$.

3. APPROACH OVERVIEW

Meeting the objectives presented in Section 2 amounts to finding a good trade-off between three contradicting goals: i) to lower the relative delay penalty (RDP), ii) to reduce the rate of false positives, and iii) to minimize stress on underlay links. For instance, lowering RDP may increase stress on some links in the underlay. Consider a mesh overlay where all the peers have point-to-point connections (overlay links) with each other. In this case, RDP between all pairs of peers is 1.0, however, the stress on the physical links close to the peers is very high [15]. Similarly, organizing subscribers (according to the similarity of their received events) to avoid false positives conflicts with the other two goals (cf. Example 1).

We therefore propose to solve the problem by decomposing it into two layers, the topology discovery layer and the routing layer, as shown in Figure 2. The topology discovery layer focuses on minimizing RDP and link stress without taking into account the subscriptions of peers and the event traffic matched by them. This layer maintains a topology discovery overlay network (TDO), which connects all the participants of a pub/sub system. In general, TDO can use any overlay link from $\mathcal{E}^O = \mathcal{P}\times\mathcal{P}$. However, to lower the delay penalty and limit the duplicate packets on the underlay, only those overlay links are selected, which minimize the overlaps between the mapped underlay routes, i.e., minimize sharing of underlay links or in other words connect peers according to their location in the underlay topology (G^U). Figure 2 shows that the peers are arranged in TDO such that the underlay routes mapped by the corresponding overlay links have minimum overlap. For instance, $q \rightsquigarrow^p s$ and therefore $path(p,s)$ (overlay path from p to s) in TDO passes through q. Connecting subscribers according to the underlay topology may induce higher stress on the last mile links e.g., (s,r) has a stress of 3 in Figure 2. To alleviate this problem, peers actively monitor the bandwidth of their last mile links and impose limits on their degrees (neighbours in TDO).

In general, organizing peers in TDO according to the underlay topology requires tools (or techniques) to infer the underlay route among all pairs of peers in the system. A number of underlay route inference tools and techniques are discussed in Section 4.3. However, these tools are expensive in terms of time and control traffic. Therefore, the topology discovery layer employs methods to limit the underlay route inferences between the peers without much degradation in the quality of TDO in terms of minimizing stress and RDP.

The routing layer runs on top of the TDO and maintains a spanning tree to distribute events (using a subset of overlay links from the TDO), as shown in Figure 2. To reduce the cost of event routing, the selection of links from the TDO is based on end-to-end delays between the peers as well as event traffic consumed or produced by them. In particular, a core-based approach is employed, whereby a small set of peers that experience higher routing load and have low delay paths (in the TDO) act as *cores*. The remaining non-core peers connect to their closest cores by using the lowest cost paths (in terms of dissimilarity of received events and end-to-end delays) in the TDO. The cost of event routing is sensitive to the selection of cores and therefore, various core selection strategies with different performance benefits are developed with complexity ranging from the use of global knowledge to only local neighbourhood-based voting mechanisms.

In the subsequent sections, we first describe the construction of TDO (cf. Section 4) and afterwards present the maintenance of a cost efficient pub/sub routing tree on the TDO (cf. Section 5).

4. TOPOLOGY DISCOVERY OVERLAY

In this section, we tackle the problem of constructing a topology discovery overlay (TDO) with low stress and RDP. In particular, we will describe two approaches to maintain

the TDO in a dynamic manner, the landmark approach and the random walk approach. The landmark approach extends the work of Kwon et al. [24] on underlay-aware single source multicast to support multiple sources in a scalable manner, whereas the random walk approach is more sophisticated, it overcomes the limitations of the landmark approach and addresses the trade-off between stress and RDP.

4.1 Landmark Approach

The landmark approach is inspired from the work of Kwon et al. [24], which addresses underlay-aware overlay creation with respect to only a single source. However, in a pub/sub system, every peer can be a publisher and a subscriber at the same time, and hence the TDO should reflect (discover) the underlay topology w.r.t. all the peers participating in the system. For this reason, the direct use of the approach of Kwon et al. [24] is not suitable and would result in an overhead of $O(N^2)$ underlay route inferences, i.e., detection of router-level underlay path between all pairs of peers.

It is a known fact that the router-level network forms a sparse graph [9] and therefore, underlay route inference between all pairs of peers is not necessary to construct a TDO that reflects a highly accurate underlay topology among publishers and subscribers [18]. For this reason, the landmark approach selects a small set of $k_{\mathcal{L}}$ peers as pivots (or landmarks), i.e., $k_{\mathcal{L}} \ll N$, and the underlay topology between the peers is discovered only with respect to the selected landmarks.

The set of landmarks is fixed and globally known to all peers in the system. Moreover, landmarks in the set are selected uniformly and independently from each other. Each peer individually picks a random number ρ in $[0, 1]$ and decides to become landmark if $\rho < \frac{k_{\mathcal{L}}}{N}$. To estimate the total number of peers N in a distributed and scalable manner, a gossip-based aggregation algorithm [16] is used. The same algorithm serves the purpose of distributing the set of landmarks among other peers.

The TDO is maintained as a virtual forest of $k_{\mathcal{L}}$ logical trees, where each tree is associated with a landmark. Each landmark acts as the root of its associated tree while all other peers (including other landmarks) join the tree.

4.1.1 Joining a Landmark Tree

In order to connect to a tree associated with a landmark s_k, a newly arriving peer s_n sends a connection request to the root of the tree (i.e., s_k). On receiving the request, the root s_k discovers the underlay route mapped by the overlay link $\langle s_k, s_n \rangle$ by using one of the underlay route inference techniques described later in Section 4.3. Once the underlay route for $\langle s_k, s_n \rangle$ is discovered, the peer s_n is placed in the tree such that the newly discovered underlay route has minimum overlap with the underlay routes of other members of the tree. This is accomplished by comparing the underlay route of $\langle s_k, s_n \rangle$ with the routes of other neighbours (child peers) at each level of the tree and performing one of the following three mutual exclusive actions [24] until the desired parent is reached.

Let s_t be the current peer in the tree which is processing the connection request from s_n. The algorithm deals with three possible cases as follows:

- *Case 1:* $\exists p \in NG(s_t) : p \rightsquigarrow^{s_k} s_n$

  In this case, the connection request is forwarded to the peer p as it shares a part of the underlay route from s_k to s_n and hence is a more appropriate parent (for s_n) than s_t. For instance, if $\langle s_k, p \rangle = \{(s_k, r_i), (r_i, r_j), (r_j, p)\}$ and $\langle s_k, s_n \rangle = \{(s_k, r_i), (r_i, r_j), (r_j, r_o), (r_o, s_n)\}$, then placing s_n as a child of p in the tree reduces overlapping among the underlay links.

- *Case 2:* $\exists p \in NG(s_t) : s_n \rightsquigarrow^{s_k} p$

  In this case, the underlay route of s_n is a prefix of the route mapped by $\langle s_k, p \rangle$ and hence, s_n should become a child of s_t and adopt p as its own child.

- *Case 3:* Neither Case 1 nor Case 2 evaluates to true.

  If no prefix relationship can exist between the underlay routes of s_n and the neighbours of s_t, then s_n joins s_t as a child.

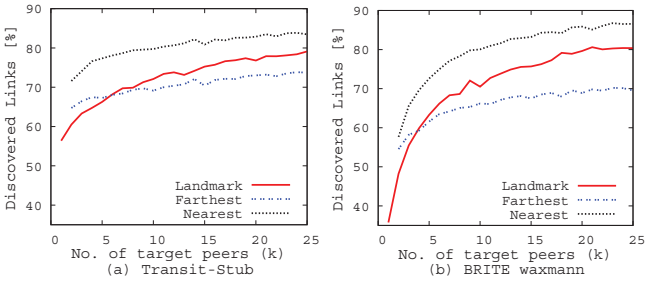


Figure 3: Selection of target peers vs. accuracy of discovered topology.

4.2 Random Walk Approach

The TDO maintained by the landmark approach reflects the underlay topology among the participating peers with reasonable accuracy (cf. Figure 3). However, there are some drawbacks associated with the landmark approach. First, the stress induced by the TDO on the underlay links close to the landmark peers is high. Second, the landmark approach is not very resilient against churn and the failure of an existing landmark requires all the peers to join a new landmark tree.

The above drawbacks can be avoided, if the landmarks are not fixed. Each peer is allowed to infer underlay routes to $k_{\mathcal{L}}$ different peers (termed as *target peers*) and use the inferred routes to find a suitable position (to connect) in the TDO. An important consideration in this regard is how each peer selects its target peers? Clearly, the selection of target peers affects the discovery of the underlay topology (among the peers participating in the TDO) and thereby influences the stress and RDP. Figures 3(a) and (b) show the impact of the selection of target peers on the accuracy of discovered underlay topology. The figures depict the percentage of the discovered underlay links on GT-ITM Transit-Stub [34] and BRITE Multi-level [26] topologies, for two target peer selection criteria and the landmark approach. The nearest and the farthest criteria make use of round trip delays for the selection of target peers. The figures show that selecting $k_{\mathcal{L}}$ peers with the lowest delays as targets consistently performs better than the landmark approach. The farthest target selection criterion on the other hand performs worst because the peers with high delays are usually in different

Algorithm 1 Random walk approach

```
1:  upon event Receive(RANDWALK, p, D(p,q), k_L , TTL) at peer q do
2:    TTL = TTL −1
3:    selectionCriteria()
4:    if |NG(q)| > ℓ(q) then
5:      peerToRemove = {s1 : ∀a ∈ NG(q)  d^O(q,s1) > d^O(q,a)}
6:      trigger Send(DISCONNECT, peerToRemove)
7:    if k_L > 0 ∧ TTL > 0 then
8:      peers[] = randomlySelectNeighbours (NG(q), σ × |NG(q)|)
9:      for all t ∈ peers do
10:       delay= D(p,q) + d^O(q,t) − 2× last mile delay of q
11:       trigger Send(RANDWALK, t, delay, ⌊k_L/|peers|⌋)

12:   procedure selectionCriteria do
13:     switch (heuristic) do
14:       case DWorst:
15:         delay = {d^O(q,s2) : ∀a ∈ NG(q)  d^O(q,s2) > d^O(q,a)}
16:       case DBest:
17:         delay = {d^O(q,s1) : ∀a ∈ NG(q)  d^O(q,s1) < d^O(q,a)}
18:       case MLink:
19:         delay = D(p,q) // Obtained in the RANDWALK message
20:     if d^O(q,p) <delay then
21:       k_L = k_L − 1
22:       Infer underlay route ⟨q,p⟩ using tools from Section 4.3.
23:       if ∃s1 ∈ NG(q) : s1 ⇝^q p then // Case 1
24:         trigger Send(CONNECT, s1, p, ⟨q,p⟩)
25:       else if ∃s1 ∈ NG(q) : p ⇝^q s1 then // Case 2: Swap s1 and p
26:         NG(q) = {NG(q) \ s1} ∪ p
27:         trigger Send(NEWPARENT, s1, p)
28:         trigger Send(DISCONNECT, s1)
29:       else // Case 3
30:         NG(q) = NG(q) ∪ p
31:     if p ∈ NG(q) then
32:       trigger Send(ACK, p)
```

Autonomous systems (AS). Inferring underlay routes between the peers in different AS discover the same backbone links repeatedly and consequently, the TDO may results in higher stress on the underlay links as well as an increase in RDP [19]. The links within a single AS are more diverse and therefore, peers with low delays should be preferred as targets.[1]

The evaluations shown in Figures 3(a) and (b) use global knowledge to select the peers with lowest delays. However, finding $k_{\mathcal{L}}$ nearest peers of a particular peer accurately in a distributed system is very expensive [5]. For this reason, our distributed approach does not focus on finding $k_{\mathcal{L}}$ nearest peers, but rather proposes three simple heuristics to select the target peers using random walk on the TDO.

A newly arriving peer p joins the TDO by sending the connection request to any existing peer. The connection request follows the protocol mentioned in Section 4.1.1 to find an appropriate place (neighbours in the TDO) for the new peer. Once joined, the peer p initiates a random walk (RANDWALK message) on the TDO. The number of peers visited by the random walk is controlled by the Time-to-Live (TTL) value and the neighbourhood selectivity factor (σ). The TTL value determines the number of forwarding hops, while the selectivity factor ($0 < \sigma \leq 1$) specifies the percentage of the neighbours to be included in the random walk at each forwarding hop.

Upon the reception of a random walk (RANDWALK) message from a peer p, the peer q employs one of the following three heuristics to determine its feasibility as a target

[1]Similar conclusions are drawn by the work of Jin et al. [19].

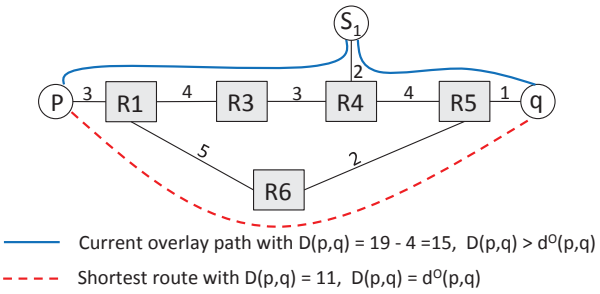


Figure 4: MLink heuristic.

to infer the underlay route to p (cf. Algorithm 1, *lines 13 - 19*):

- *DWorst Heuristic:* This heuristic selects q as a target if the unicast path delay from q to p (i.e., $d^O(q,p)$) is lower than the delay $d^O(q,a)$ to any existing neighbour a of q.
- *DBest Heuristic:* This heuristic is more restrictive than DWorst and selects q as a target only if $d^O(q,p)$ improves on the existing neighbour of q with the lowest delay.
- *MLink Heuristic:* This heuristic selects q as a target, if some underlay links between p and q with lower delays are not discovered yet, i.e., not mapped by the overlay paths in the TDO. The peers are arranged in the TDO according to the prefix relation between their underlay routes and therefore, existence of undiscovered underlay links between p and q can be checked by comparing the delay on the $path(p,q)$ in the TDO (excluding the last mile delays of the intermediate peers) and the unicast delay $d^O(q,p)$, as shown in Figure 4. A smaller unicast delay $d^O(q,p)$ predicts the existence of undiscovered underlay links, which if included in the TDO lead to smaller delay on the $path(q,p)$.

In case peer q is selected as a target, it infers the underlay route of $\langle q,p\rangle$ using tools described in Section 4.3, compares it with the routes to other neighbours, and follows one of the three mutually exclusive cases mentioned in Section 4.1.1 (cf. Algorithm 1, *lines 22 - 30*). Following case 2 or 3 results in the acceptance of peer p as a neighbour of q. However, accepting peer p as a neighbour may violate the degree constraints (i.e., $\ell(q)$) imposed by peer q (to avoid bandwidth bottleneck on the last mile link) and therefore, an existing neighbour with the highest delay may be selected for disconnection (cf. Algorithm 1, *lines 4 - 6*).[2] Finally, the random walk message is forwarded to $\sigma \times |NG(q)|$ randomly selected neighbours along with the number of remaining targets to be selected. (cf. Algorithm 1, *lines 7 - 11*).

4.3 Techniques for Underlay Route Inference

Until now, we assume the availability of some tool to infer the underlay route between the peers. In this section, we give an overview of different techniques and tools which can be used for this purpose.

In general, the techniques to infer the underlay route (or topology) between a pair (or group) of peers mainly fall in

[2]To avoid partitions in the TDO, each peer additionally maintains a small set of overlay links to distant peers.

two categories [35]: tomography-based and router-assisted. The tomography-based techniques only discover the logical topology between the peers, induce high computation and communication overhead, and have limited accuracy due to certain assumptions about the statistical properties of the underlying network [35, 18]. We therefore propose to use router-assisted techniques. These techniques take advantage of the router's response to the Internet Control Message Protocol (ICMP) based probe messages to infer the underlay route between the pair of peers. In particular, a peer can find the underlay route to other peers using tools, such as *traceroute*, *tracepath*, *tcptraceroute* etc. These tools have been extensively used for underlay topology discovery [24, 6]. Recently, many new tools are proposed to overcome the limitations (e.g., due to load balancing in ISPs [23] or Multi-protocol Label Switching – MPLS) and increase the efficiency (especially the probe redundancy) of the standard tools [7]. A problem with router-assisted techniques is that roughly 1/3 of routers (termed as *anonymous routers*) do not respond to ICMP messages [24, 35, 18]. These anonymous routers appear distinct in different underlay routes and therefore, inflate the discovered underlay topology [18]. Several merging techniques [18, 12] are proposed to detect and collapse the anonymous occurrences of the same router. All these techniques are orthogonal to our work in this paper and can be integrated. Moreover, a reasonable percentage of routers respond to ICMP messages [24, 35, 18] and thereby the inferred underlay routes (with distortions induced by anonymous routers) can still suffice to arrange peers in the TDO minimizing RDP and stress.

Another possibility is to rely on the data available from the internet topology discovery projects [10]. Topology servers such as OSPF [11] can also be used to obtain ISP level (intra-domain) underlay routes by using simple network management protocol (SNMP). Moreover, as pointed out by Kwon et al. [24], the periodic logs of router configuration can be employed. However, underlay route information obtained from these techniques may not be very accurate due to dynamic nature of the internet [35].

5. EVENT ROUTING

Until now, we have described the maintenance of the topology discovery layer. In this section, we address the problem of constructing a cost efficient routing tree T on a TDO, additionally taking into account the end-to-end delay and the event traffic matched between the subscriber peers.

The basic idea of our approach is to reduce the distance (delay) between the peers that consume or produce similar events by placing them nearby in the routing tree T (or in other words, to place peers matching dissimilar events away from each other). Therefore, the first step is to quantify the (dis)similarities between the peers to produce or consume similar events. Intuitively the similarity between two peers increases with the increase in overlapping events and decreases with the non-overlapping event traffic. More precisely, let $\overline{E}_{p_i}$ and $\overline{E}_{p_j}$ denote the events matched (or produced) from the set E_ξ by the peers p_i and p_j respectively. The similarity between peers p_i and p_j is calculated by using the Jaccard function[3] [32], i.e., $\text{sim}(p_i, p_j) = \frac{|\overline{E}_{p_i} \cap \overline{E}_{p_j}|}{|\overline{E}_{p_i} \cup \overline{E}_{p_j}|}$. Accordingly, the dissimilarity between the peers can be derived as: $\text{dsim}(p_i, p_j) = [1 - \text{sim}(p_i, p_j)] \times \alpha$. The dissimilarity values are normalized in the range $[0, \alpha]$, where 0 means that peers consume/publish exactly same events and α is a penalization constant.

Consequently, each overlay link in the TDO is weighted according to the delay (induced by the underlay) as well as the dissimilarity of its adjacent peers p_i and p_j to receive/publish same events, i.e., $w(p_i, p_j) = d^O(p_i, p_j) + \text{dsim}(p_i, p_j)$. Similarly, the weight of a path in the TDO, i.e., $path(p_i, p_n)$, is defined as: $W(p_i, p_n) = \sum_{\langle i,j \rangle \in path(p_i, p_n)} w(i, j)$. The links with smaller weights $w(p_i, p_j)$ should be preferred in T to reduce the routing cost. However, taking into account the weights of the links alone, for instance, by connecting peers in a minimum (weighted) spanning tree (MST), may results in a very high routing cost (cf. Section 6.2). The reason is that the structure of a tree, i.e., organization of peers, influences the routing load (defined in Section 2) on each (overlay) link and is therefore crucial for achieving lower routing cost. Similarly, source-based trees (i.e., separate shortest weighted path tree for each publisher peer) are not desirable as they impose serious scalability issues in terms of control overhead and size of the routing tables.

We therefore employ a core-based approach to build the pub/sub routing tree T. The approach selects a small set of peers, denoted by $\mathcal{C}$, as cores. In general, any peer in the system can be selected as a core. However, only those $k_\mathcal{C}$ peers which have low delay paths in the TDO and participate in the dissemination of many events are selected as cores. The core peers are connected with each other using the minimum weighted paths in the TDO. A separate shortest weighted path tree is maintained by each core peer c, denoted as SPT_c. Each non-core peer p selects one of the core peers as its *relay*, denoted as $rel(p)$, such that $W(p, rel(p)) = min_{c \in \mathcal{C}}\ W(p, c)$, and joins the shortest path tree rooted at $rel(p)$ (i.e., $SPT_{rel(p)}$). A path in T between two peers p and q with different relays is composed of three sub-paths: from p to $rel(p)$, from $rel(p)$ to $rel(q)$ and, finally from $rel(q)$ to q.

Clearly, to realize the core-based approach three main issues should be addressed: i) selection of good candidate peers to act as cores (cf. Section 5.1), ii) discovery and connection to the closest relays by non-core peers (cf. Section 5.2) and, iii) maintenance of the routing tree T in the presence of dynamics that arise due to changes in the event traffic and the subscriptions as well as churn/failures of the core and the non-core peers (cf. Section 5.3).

5.1 Core Selection

The core selection in a distributed environment mandates: i) election of a leader to decide the set of cores ($\mathcal{C}$) and, ii) a structure (e.g., tree) to communicate between the leader and the peers in the system. For this purpose, the peers maintain a spanning tree, denoted as $T_\mathcal{C}$, on the TDO such that the root of the spanning tree acts as a leader. Maintaining a distributed spanning tree in dynamic conditions is a well researched topic [2]. For this reason, we will not discuss the maintenance algorithm in this paper.

In the following, we present various core selection strategies with the variations in selection criteria, required knowledge and performance under different workload scenarios.

[3]Jaccard function assigns similarities in metric space [4] i.e., similarities are non-negative, symmetric and obey triangular inequality.

5.1.1 Strategies Based on Global Knowledge

These strategies assume that the leader has knowledge about the shortest (weighted) paths between all pair of peers in the system (All pairs shortest path – APSP). To acquire APSP knowledge at the leader we used the algorithm of Kanchi et al. [22]. The Kanchi's algorithm works on $T_{\mathcal{C}}$ and has a complexity of $O(N)$ message overhead and $O(N^2)$ message size. Once APSP information is available at the leader, two different core selection strategies can be employed.

Maximum Path Count (MaxPath): This strategy selects the peers with higher routing load as cores. In particular, first $k_{\mathcal{C}}$ highest loaded peers which participate in the dissemination of most events, i.e., a number of shortest paths pass through the peers, are selected as cores.

Shortest Path Cost (SPath): This strategy prefers those peers as cores, whose shortest path trees result in lower routing cost. More precisely, for each peer p, the weights along the shortest paths to all other peers are summed (i.e., $\sum_{\forall q \in \mathcal{P}} W(p,q)$) and the first $k_{\mathcal{C}}$ peers with the lowest values are selected as cores.

5.1.2 Strategies Based on Local Knowledge

Maintenance of APSP information (required by the strategies based on global knowledge) utilizes messages of very large size, i.e., $O(N^2)$, especially on the links near the root of $T_{\mathcal{C}}$ [22]. To overcome the problem, we employ voting-based mechanism where each peer votes for potential candidates to be selected as cores according to its local knowledge. In particular, each peer either votes for itself or its neighbour. The votes are aggregated towards the root (leader) of $T_{\mathcal{C}}$. To reduce the message size, a small (constant) number of core candidates with higher votes are kept at each aggregation step (each level of $T_{\mathcal{C}}$). Finally, the root selects the $k_{\mathcal{C}}$ peers with the highest votes as cores. Two different voting strategies are considered.

Closest Neighbour (CNeigh): This strategy promotes peers which are connected by low delay links and receive events similar to their neighbours (in the TDO) as cores. In this strategy, each peer votes for its neighbour with the lowest weight (i.e., neighbour with low delay and receiving similar events).

Selection Potential (SPotent): This strategy is inspired from the centralized heuristic developed by Campos et al. [1] for the construction of a minimum cost multicast tree. In this strategy, each peer votes for itself by measuring its potential to be selected as a core. The selection potential of a peer p is determined by considering three characteristics: i) number of neighbours in the TDO (i.e., $NG(p)$) , ii) sum of weights to all the neighbours (i.e., $\sum_{q \in NG(p)} W(p,q)$) and, iii) worst delay to a neighbour (i.e., $D_W = \{d^O(p,q) : \forall a \in NG(p) \;\; d^O(p,q) > d^O(p,a)\}$). More precisely, the selection potential of a peer p is calculated as follows:

$$NG(p) + \frac{NG(p)}{\sum_{q \in NG(p)} W(p,q)} + \frac{1}{D_W}$$

5.2 Core Discovery and Connection

Once the core peers are selected and distributed by the leader (root of $T_{\mathcal{C}}$), the next step is to connect all the non-core peers to their relays (i.e., closest cores) using minimum weighted paths in the TDO. To accomplish this, each core c_i initiates the construction of a shortest path tree (SPT_{c_i}) by sending CORE_SPT message to its neighbours in the TDO. Upon the reception of a CORE_SPT message for SPT_{c_i} from a peer p, the peer q checks whether the minimum weight path to c_i improves on the weight of the shortest paths to other cores. If there is no improvement then the message is dropped from further forwarding (to reduce control overhead), otherwise the peer q joins SPT_{c_i} by sending join (JOIN_CORE) message to its parent p on SPT_{c_i}. Moreover, the CORE_SPT message for SPT_{c_i} is propagated to only those neighbours which have not received it yet (cf. Algorithm 2).

Algorithm 2 Core discovery and connection

```
1: ∀_{c∈C} w(c) = ∞ // Initialization performed at each peer q
2:   upon event Receive(CORE_SPT, c_i, W(c_i,q), p ) at peer q do
3:     w(c_i) = W(c_i,q)
4:     Rcv(c_i) = Rcv(c_i) ∪ p // msg for core c_i is received from p
5:     if ∀_{t∈C} w(c_i) < w(t) then
6:       trigger Send(JOIN_CORE, c_i, p )
7:       for all v ∈ NG(q) : v ∉ Rcv(c_i) do
8:         weight = W(c_i,q) + w(q,v)
9:         trigger Send(CORE_SPT, c_i, weight, q , v)
```

To avoid partitions in T, all the cores should also be connected with each other. During the core selection, the leader selects a core with the highest votes (similarly highest load or lowest cost in the case of MaxPath and SPath respectively) as the root core. All other cores connect to the shortest path tree of the root core similar to the non-core peers, as described in Algorithm 2.

5.3 Handling Dynamics

P2P systems are very dynamic in nature and therefore, complete recalculation of routing tree for every minor change in the event traffic or the set of peers ($\mathcal{P}$) is not feasible.

A newly arrived peer requests its neighbours on the TDO to forward the CORE_SPT messages and joins the SPT of the closest core (cf. Algorithm 2, *lines 1 - 6*). Similarly, disconnections (due to leaves/failures) of existing core and non-core peers are handled locally. A peer p on discovering the disconnection of parent q (which may be core itself) on $SPT_{rel(p)}$, joins the SPT of the second closest core. However, if the information about the minimum weight paths to other cores is not available (due to pruning of CORE_SPT message in Algorithm 2), the peer p requests its neighbours on the TDO to forward the CORE_SPT messages and joins similar to the arrival of a new peer. Moreover, the CORE_SPT message of the selected core is forwarded to the child peers (on the SPT of previous $rel(p)$) to detect cycles and update weights to the new core.

The dynamic changes in the subscription and event workload as well as changes in the TDO due to arrivals/disconnections of peers, accumulate over time, so that the current routing tree T becomes suboptimal. In order to adapt to the changes, the leader of $T_{\mathcal{C}}$ periodically starts the core selection process and a new routing tree T is constructed. The time period for the construction of new routing tree T is a system parameter, which is specified by the administrator of the system.

6. PERFORMANCE EVALUATIONS

Experiments are performed using PeerSim [17]. Physical network topologies are generated using BRITE [26] and GT-

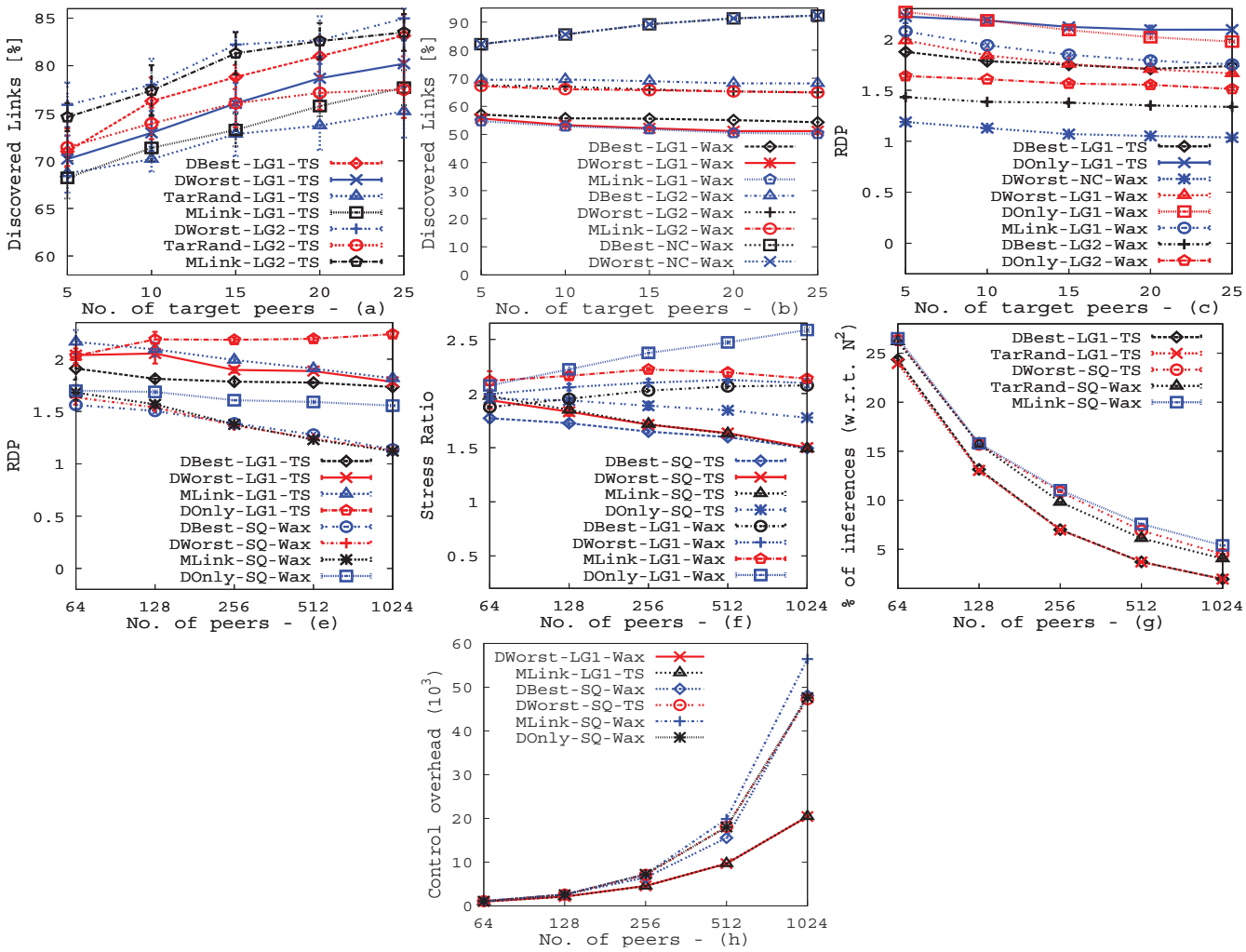


Figure 5: Evaluations for topology discovery overlay (TDO)

ITM [34] tools. BRITE generates a non-hierarchical Waxman topology (in short Wax) with 1640 routers. GT-ITM generates two layer hierarchy of transit and stub domains (in short TS) comprising 1584 routers. Up to $N = 1024$ peers are used in the experiments, which are connected to random routers (stub domain routers for TS topology). The accuracy of the topology discovery increases with the increase in the number of peers (cf. Section 6.1.2) and therefore, to mimic conservative (strict) settings moderate number of peers are used in the experiments. The last mile delays (between the peers and their access routers) are set to be $5 - 10\%$ of the average delay between all the routers in the system. To avoid bandwidth bottlenecks on the last mile links, the limits on the degree constraints of the peers are chosen as $log_2 N$ (denoted as LG1), $2 \times log_2 N$ (denoted as LG2) and $\sqrt{N}$ (denoted as SQ). Moreover, NC represents the scenario where no degree constraints are imposed on the peers. For all the experiments, each peer performs only a single random walk. Unless otherwise stated, the neighbourhood selectivity factor is set to 0.1 and the number of target peers are chosen as 10.

The content-based schema contains up to 5 integer attributes, where the domain of each attribute is in the range $[0, 10]$. Experiments are performed on two different models for the distributions of subscriptions and events. The uniform model (in short UD) generates random subscriptions/events independent of each other. The interest popularity model (in short ZD) chooses five hotspot regions around which subscriptions/events are generated using the widely used zipfian distribution. For the experiments, up to 5000 events are used and each event matches 5% of subscriptions.

6.1 Performance of TDO

We compare our work with two baseline approaches. The first approach maintains TDO by inferring underlay routes to $k_{\mathcal{L}}$ arbitrary target peers (chosen randomly) and is denoted as *TarRand*. The second approach (denoted as *DOnly*) organizes peers in the TDO solely based on their end-to-end delays to the target peers without using the underlying topology information.

6.1.1 *Influence of Target Peers*

Figures 5(a) and (b) show the percentage of discovered underlay links versus the number of target peers ($k_{\mathcal{L}}$), for different (target) selection heuristics and degree constraints.

As expected, the percentage of discovered underlay links increases by relaxing the degree constraints, for both types of topologies and all selection heuristics. Moreover, DBest heuristic performs slightly better by discovering higher percentage of underlay links. It is also worth noting that DBest and DWorst heuristics show similar results in the absence of degree constraints. TarRand on the other hand performs poor, which shows that the proposed heuristics are beneficial for the selection of good target peers. Figures 5(a) and (b) depict that for transit-stub topologies, the percentage of discovered underlay links improves by increasing the number of target peers under all degree constraints. In contrast, the percentage of discovered links decreases very slightly in the presence of degree constraints, for waxmann topologies (Wax). An explanation for such a behaviour lies in the structure of the topologies. However, in the absence of any degree constraints, the percentage of discovered underlay links for Wax improves with the increase in the number of target peers similar to TS, as shown in Figure 5(b). Nevertheless, the slight loss in accuracy for Wax in the presence of degree constraints does not effect the corresponding RDP and Stress. Figure 5(c) plots RDP achieved by the TDO for different numbers of target peers and selection heuristics. RDP is measured as the ratio of delay experienced between all pairs of peers on the TDO to the delay experienced between them using the direct unicast paths in the underlay. It is clear from the figure that for Wax topologies, RDP decreases slightly with the increase in the number of target peers. Moreover, RDP is lowered quiet significantly by relaxing the degree constraints as expected. For instance, in the absence of degree constraints and 10 target peers, the RDP is 1.19 for Wax topologies. Furthermore, DOnly approach which does not utilize topology information results in higher RDP.

6.1.2 *Effect of Number of Peers Participating in the System*

Figures 5(e) and (f) illustrate the effectiveness of TDO to lower RDP and minimize stress, for different size of peers participating in the system. It is clear from the Figure 5(e) that RDP decreases gradually with the increase in the number of peers. The reason is that the percentage of access routers (i.e., routers directly attached to the peers through the last mile links) increases with the increase in the number of peers and therefore, the organization of peers in the TDO resembles underlay topology more accurately, enabling messages to follow overlay paths which are similar to the underlay routes (except the last mile links to intermediate peers in the TDO), decreasing the delay penalty. For the same reason, only moderate number of peers (up to 1024) are used in the evaluations. Figure 5(f) shows the impact of the number of peers (participating in the system) on stress. Clearly, the total stress introduced by the TDO increases in proportion to the number of participating peers, since more identical messages traverse physical links when more peers join the TDO. In order to compare the effectiveness of the TDO to reduce stress across different peer sizes, Figure 5(f) plots the ratio of total stress introduced (as a result of communication between all pairs of peers) by sending messages on the TDO to the stress introduced when the messages are routed directly on the underlay. The lower value of stress ratio represents higher effectiveness in minimizing stress. Figures 5(e) and (f) show that the performance of all three selection heuristics becomes almost similar with the increase in the number of peers and the relaxation of degree constraints. The DOnly approach which does not take into account the underlay topology experiences up to 39% higher RDP and 30% rise in stress.

Figure 5(g) shows the influence of the number of peers on the control overhead due to the inference (discovery) of underlay routes between the peers. Underlay route inference techniques are very expensive in terms of computation and communication cost (cf. Section 4.3) and therefore, number of underlay route inferences should be minimized. The figure plots the percentage of underlay route inferences in comparison to the naive approach, whereby underlay routes are discovered between all pairs of peers. The percentage of inferences decreases with the increase in the number of peers mainly because the number of targets are kept constant. Figure 5(h) displays the control overhead due to peer joins during the construction of the TDO. In general, MLink performs slightly more underlay route inferences and produces higher control overhead. This is because in MLink target peers are selected according to the difference between the delays on the paths in the TDO (excluding the last mile delays) and the unicast delays. This may results in TDO connections between arbitrary peers which have high delays or are not located nearby in the underlay topology and hence, resolves in comparatively higher number of disconnections (e.g., when a peer with lower delay sends join message, the neighbour with the highest delay is disconnected), producing slightly more underlay route inferences and join messages.

6.2 Performance of Event Routing

We evaluate three aspects of our event routing approach: i) capability of the proposed core selection strategies to reduce cost of event dissemination (routing cost) in comparison to the baseline approaches, ii) impact of increase in number of cores on the routing cost and the control overhead and, iii) adaptability to dynamic changes in the workload.

We compare our work against three baseline approaches: i) random (*Rand*), ii) similarity-based (*Sim*), and iii) delay-based (*MST*). The first two are core-based approaches. Rand selects k_C random peers as cores, whereas Sim selects cores solely based on the similarity between the peers to produce/consume similar events. The third approach MST maintains a minimum delay spanning tree for the routing of events. Moreover, we implement an optimal routing (*OPT*) algorithm that uses separate shortest path routing trees for the dissemination of each published event [25].

6.2.1 *Core Selection Strategies*

Figure 6(a) shows the routing efficiency of the proposed core selection strategies and the baseline approaches in comparison to OPT. More precisely, the efficiency of a routing approach (or a strategy) is defined as the ratio of the event routing cost (cf. Section 2) incurred by the approach (or the strategy) to the cost of routing events using OPT. To measure the routing cost around 2000 events are disseminated in the system. Clearly, a lower value for routing efficiency means better approach. The figure depicts that SPath performs better than all other strategies mainly because it uses the global knowledge. However, interestingly SPotent strategy which only exploits the local knowledge to select cores, performs almost similar to the global knowledge based MaxPath strategy. Moreover, all the approaches perform consis-

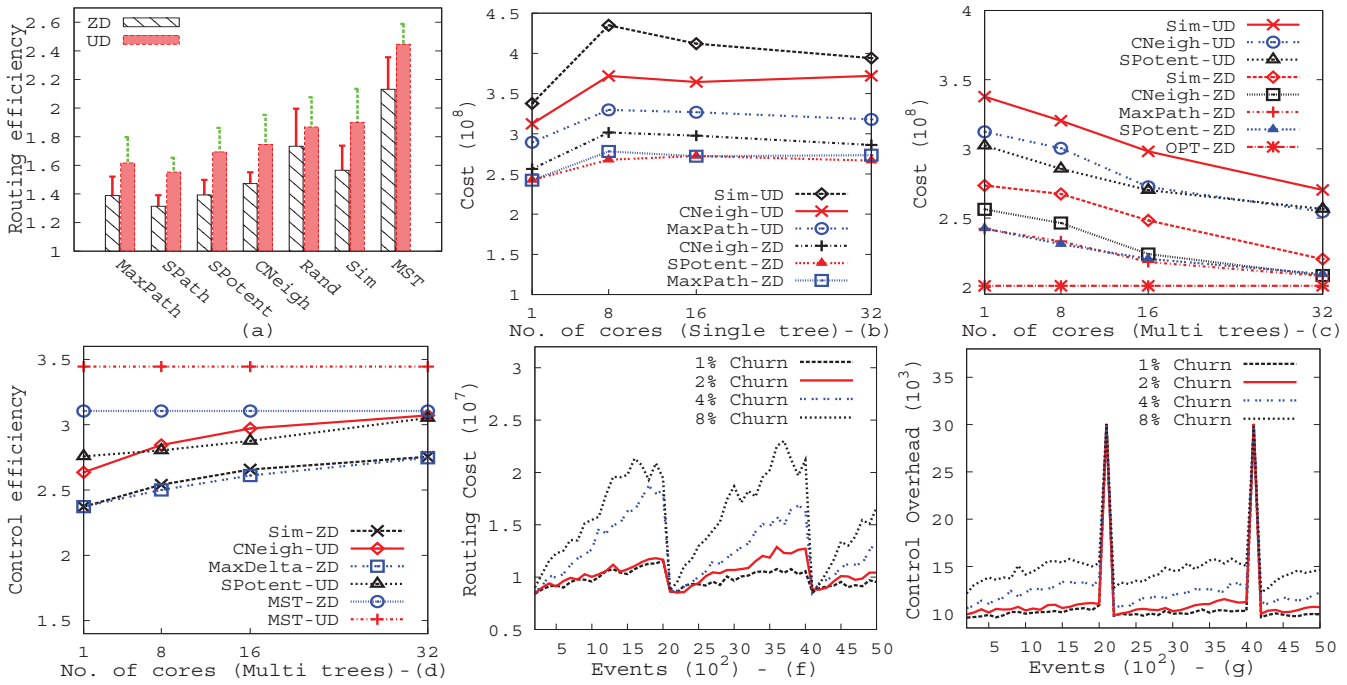


Figure 6: Evaluations for event routing

tently better in case the subscription/event workload follows interest popularity model (i.e., ZD). This is because the subscriptions associated with an interest hotspot consume similar events and placing such subscriptions nearby in the routing tree limits event dissemination to a certain region saving routing cost, whereas for uniformly distributed workload the improvement from nearby placement of the subscriptions consuming similar events is relatively small. For the same reason, Sim achieves better efficiency than Rand for zipfian workload (ZD), whereas both Sim and Rand perform almost same in the case of uniform workload (UD). Another interesting observation is that MST performs even worse than Rand. This is because MST only considers the TDO links with lowest delays, while Rand additionally takes into account the even traffic consumed by the peers (for connecting them to the closest cores) to further reduce the cost of event dissemination.

6.2.2 *Impact of Core Size*

Figure 6(b) depicts the influence of the number of cores on the communication cost to forward subscriptions as well as route events from the publishers to the interested subscribers. The trend shows that the cost increases with the increase in the number of cores. This is because all the cores in $\mathcal{C}$ are connected with each other through the shortest weighted path tree rooted at the core c_h with the highest votes (cf. Section 5.2). The SPT_{c_h} may not represents shortest (weighted) paths between all pairs of cores in $\mathcal{C}$ and therefore, the communication (such as event dissemination or subscription forwarding) between two peers p and q with different relays (cores) incurs higher cost due to higher weight (delay and dissimilarity in event traffic) on the path between $rel(p)$ and $rel(q)$. Certainly, the communication cost can be decreased, if the cores are connected with each other through shortest weighted paths. Figure 6(c) illustrates this point by showing that in the presence of APSP (All pairs shortest weighted paths) between the cores, the communication cost decreases significantly with the increase in the number of cores. Clearly, this reduction in cost comes at the expense of additional control overhead due to the maintenance of $k_{\mathcal{C}}$ shortest weighted path trees. It is worth mentioning that the cores represent only a small fraction of peers in the system (i.e., $k_{\mathcal{C}} \ll N$) and therefore, control overhead is much lower in comparison to the maintenance of APSP w.r.t. all peers in the system. Figure 6(d) depicts the control efficiency of the proposed core selection strategies w.r.t. different core sizes. The control efficiency is defined as the ratio of the complete message traffic generated in the system to the number of matching events received by the peers (i.e., events which are matched by the subscriptions of the peers). The traffic includes all the control overhead induced in the system during leader election, selection of cores, maintenance of SPTs to connect peers and cores in the routing overlay, subscription forwarding and event dissemination. The control efficiency of 1.0 indicates ideal system, whereby events are directly delivered to all the peers with matching subscriptions without incurring any unnecessary message overhead. Figure 6(d) shows that the control efficiency decreases with the increase in core size, mainly due to the maintenance of $k_{\mathcal{C}}$ shortest path trees. However, the control efficiency of the proposed core selection strategies is still better than MST approach. This is because MST does not take into account the event traffic consumed/produced by the peers during the construction of routing overlay and therefore, experiences high control overhead during subscription forwarding and event dissemination.

6.2.3 *Adaptability to the Changes in Workload*

Figures 6(f) and (g) show the behaviour of the system in the presence of continuously arriving and leaving sub-

scribers. The churn is introduced in the system after the dissemination of every 100 events. The percentage of churn is relative to the total number of peers in the system. During the experiment, SPotent is used as the core selection strategy and subscriptions/events workload is generated using zipfian distribution (ZD). Nevertheless, similar trends are observed for other core selection strategies and uniformly distributed subscriptions/events workload (UD).

Figure 6(f) shows that the cost of event routing increases as more and more events (up to 2000) are disseminated in the system. This is because the joins and leaves of peers (due to churn) are only handled locally to reduce the control overhead (cf. Section 5.3) and as a consequence, the routing overlay degrades overtime resulting in higher routing cost. Figure 6(g) shows the control overhead in terms of the number of overall messages in the system. A slight increase in the control overhead for higher percentages of churn is due to the local handling of peer dynamics, as mentioned above. After every 2000 events, complete recalculation of the routing overlay is initiated by the leader, which significantly lowers the routing cost, as shown in Figure 6(f). Moreover, Figure 6(g) depicts that immediately after the leader initiated recalculation of the routing overlay, the control overhead rises considerably for a small transient period. However, the new routing overlay obtained after the transient period consumes almost identical control traffic.

7. RELATED WORK

Publish/Subscribe Systems: In the recent past, several content-based pub/sub systems have been proposed with the aim to provide communication efficient routing of events from the publishers to the subscribers. Many of these systems [3, 28, 32] focus on minimizing bandwidth usage by clustering subscribers according to their interests, without taking into account the properties of the underlying physical network. As shown in Example 1, these systems could be suboptimal w.r.t. the communication overhead and end-to-end delays.

Few systems [25, 14, 27] consider underlay related QoS metrics such as end-to-end delay, data rate, loss rate etc., to optimize pub/sub overlay for efficient routing of events. Majumder et al. [25] propose an approach that constructs multiple trees to efficiently distributed events in a content-based pub/sub system. The construction of each tree is formulated as a generalized steiner tree problem and an approximation algorithm is developed to build trees with communication cost at most poly-logarithmic factor of the optimum. However, the proposed approach assumes the availability of the content-based workload (subscriptions and matched events) and the properties of the underlying network at a central coordinator, which hinders its scalability. Similarly, XPort [27] targets the construction of an event distribution tree that can be optimized according to the application-defined performance metrics, e.g., minimize average path delay to the root. However, XPort mandates that all the publishers are connected to the root of the event distribution tree.

Nevertheless, some existing pub/sub systems [8] address reliable delivery of events by explicitly taking into account the router-level topology of the underlying network. Generally, these systems rely on the redundancy in the underlay paths between publishers and subscribers to provide resilience against the network failures. In contrast to our work, these systems assume that the topology information is somehow available. Moreover, the QoS metrics such as delay and bandwidth, which are focused in this paper are not addressed.

Topology-aware Overlay Networks: Previous research in the area of Application Layer Multicast (ALM) has shown that the knowledge of the underlying (router-level) network topology is beneficial to achieve low physical link stress, low RDP and high bandwidth data dissemination [18, 21, 20, 24, 36]. Zhu et al. [36] address the problem of constructing a high bandwidth overlay for ALM. The authors prove that the problem is NP-hard and propose a distributed heuristic, which incrementally improves the bandwidth of the ALM tree by replacing the lower bandwidth overlay links with the higher bandwidth links. Similarly, Jin el al. [21] develop approximation algorithms to construct maximum bandwidth multicast tree (MBMT) and minimum stress multicast tree (MSMT). However, the approximation algorithms are centralized and assume the availability of the complete knowledge about the router-level topology of the underlying network. FAT (FAST Application-layer Tree) [20] uses underlay route inference tools such as traceroute, to discover router-level underlay topology and build a multicast tree on top of the discovered topology to achieve high bandwidth and low RDP. A heuristic, named Max-Delta, is employed to discover the underlay topology in an efficient and scalable manner. The MLink heuristic proposed in this paper is adopted from the Max-Delta heuristic. In particular, we modified the Max-Delta to operate in a completely decentralized environment and without the use of network coordinate system.

Lastly, a huge amount of graph theory literature is available on spanning tree related optimization problems [33] such as Minimum Routing Cost Spanning Tree (MRCT) or Optimum Communication Spanning Tree (OCRT). Nevertheless, all these theoretical approaches cannot be applied for event routing in a content-based pub/sub system, because their focus is to minimize the pairwise distances between the vertices in the input graph without any consideration to the traffic requirements between those vertices. Moreover, these approaches are centralized and are not targeted to handle continuously evolving workload as is the case in P2P-based systems.

8. CONCLUSION

In this paper, we have presented an approach that exploits the knowledge of event traffic, user subscriptions and the router-level topology of the underlying physical network to achieve scalable and communication efficient dissemination of events in a content-based pub/sub system. For this purpose, we have developed methods to discover underlay topology between subscribers and publishers in the system. The proposed methods construct a *Topology Discovery Overlay* (TDO), whereby peers are connected according to the overlapping in the underlay routes. Afterwards, the information of the discovered topology and the proximity between the peer to receive or produce similar events is used to build an event routing overlay. In particular, we have proposed different core selection strategies (exploiting global and local knowledge) to facilitate the construction of a communication efficient event routing overlay on top of TDO. Our evaluations show that for Internet-like topologies, TDO is capable of lowering physical link stress and reducing RDP. Moreover, the proposed core-based approach reduces the cost to

disseminate events up to 49% in comparison to the widely used baseline and related approaches.

9. REFERENCES

[1] R. Campos and M. Ricardo. A fast algorithm for computing minimum routing cost spanning trees. *Comput. Netw.*, 2008.

[2] C. Cheng, I. Cimet, and S. Kumar. A protocol to maintain a minimum spanning tree in a dynamic topology. *SIGCOMM Comput. Commun. Rev.*, 1988.

[3] A. K. Y. Cheung and H.-A. Jacobsen. Green resource allocation algorithms for publish/subscribe systems. In *Intl. conf. on distributed computing systems*, 2011.

[4] K. L. Clarkson. Nearest-neighbor searching and metric space dimensions. In *Nearest-neighbor methods for learning and vision: theory and practice*. 2006.

[5] M. Costa, M. Castro, A. Rowstron, and P. Key. PIC: Practical Internet coordinates for distance estimation. In *Intl. conf. on distributed computing systems*, 2004.

[6] B. Donnet and T. Friedman. Internet topology discovery: A survey. *IEEE Comm. Surv. Tuts.*, 2007.

[7] B. Donnet, P. Raoult, T. Friedman, and M. Crovella. Deployment of an algorithm for large-scale topology discovery. *IEEE J.Sel. A. Commun.*, 2006.

[8] C. Esposito, D. Cotroneo, and A. Gokhale. Reliable publish/ subscribe middleware for time-sensitive Internet-scale applications. In *DEBS*, 2009.

[9] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the Internet topology. *SIGCOMM Comput. Commun. Rev.*, 1999.

[10] C. T. C. A. for Internet Data Analysis. http://www.caida.org/home/, 2012.

[11] M. Goyal, M. Soperi, E. Baccelli, G. Choudhury, A. Shaikh, H. Hosseini, and K. Trivedi. Improving convergence speed and scalability in OSPF: A survey. *IEEE Commun. Surveys Tuts.*, 2011.

[12] M. H. Gunes and K. Sarac. Resolving anonymous routers in Internet topology measurement studies. In *IEEE INFOCOM*, 2008.

[13] A. Gupta, O. D. Sahin, D. Agrawal, and A. E. Abbadi. Meghdoot: Content-based publish/subscribe over P2P networks. In *Intl. conf. on middleware*, 2004.

[14] M. A. Jaeger, H. Parzyjegla, G. Muehl, and K. Herrmann. Self-organizing broker topologies for publish/subscribe systems. In *ACM sympos. on appl. comput. (SAC)*, 2007.

[15] S. Jain, R. Mahajan, D. Wetherall, and G. Borriello. Scalable self-organizing overlays. Technical Report UW-CSE 02-02-02, University of Washington, 2002.

[16] M. Jelasity, W. Kowalczyk, and M. v. Steen. An approach to massively distributed aggregate computing on peer-to-peer networks. In *Workshop on parallel, distributed and network-based processing (PDP)*, 2004.

[17] M. Jelasity, A. Montresor, G. P. Jesi, and S. Voulgaris. PeerSim: A peer-to-peer simulator. http://peersim.sourceforge.net/.

[18] X. Jin, W. Tu, and S. H. G. Chan. Scalable and efficient end-to-end network topology inference. *IEEE Trans. Parallel Distrib. Syst.*, 2008.

[19] X. Jin, W. Tu, and S.-H. G. Chan. Traceroute-based topology inference without network coordinate estimation. In *IEEE intl. conf. on commun.*, 2008.

[20] X. Jin, Y. Wang, and S.-H. G. Chan. Fast overlay tree based on efficient end-to-end measurements. In *IEEE intl. conf. on commun. (ICC)*, 2005.

[21] X. Jin, W. P. Yiu, S. H. Chan, and Y. Wang. On maximizing tree bandwidth for topology-aware peer-to-peer streaming. *IEEE Transactions on Multimedia*, 9:1580–1592, 2007.

[22] S. Kanchi and D. Vineyard. An optimal distributed algorithm for ALL-pairs shortest-path. *Intl. J. on Information Theories & App.*, 11:141–146, 2004.

[23] S. Kandula, D. Katabi, S. Sinha, and A. Berger. Dynamic load balancing without packet reordering. *ACM SIGCOMM Comput. Commun. Rev.*, 2007.

[24] M. Kwon and S. Fahmy. Path-aware overlay multicast. *Comput. Netw.*, 47:23–45, 2005.

[25] A. Majumder, N. Shrivastava, R. Rastogi, and A. Srinivasan. Scalable content-based routing in pub/sub systems. In *IEEE INFOCOM*, 2009.

[26] A. Medina, A. Lakhina, I. Matta, and J. Byers. BRITE: Universal topology generation from a user"s perspective. Technical report, Boston University, 2001.

[27] O. Papaemmanouil, Y. Ahmad, U. Çetintemel, J. Jannotti, and Y. Yildirim. Extensible optimization in overlay dissemination trees. In *Proc. of ACM SIGMOD intl. conf. on management of data*, 2006.

[28] O. Papaemmanouil and U. Cetintemel. SemCast:: Semantic multicast for content-based data dissemination. In *Intl. conf. on data engineering (ICDE)*, 2005.

[29] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: a scalable peer-to-peer lookup protocol for Internet applications. *IEEE/ACM Transactions on Networking*, 11:17–32, 2003.

[30] M. A. Tariq, G. G. Koch, B. Koldehofe, I. Khan, and K. Rothermel. Dynamic publish/subscribe to meet subscriber-defined delay and bandwidth constraints. In *Proc. of Intl. conf. on parallel computing (Euro-Par)*, 2010.

[31] M. A. Tariq, B. Koldehofe, G. G. Koch, I. Khan, and K. Rothermel. Meeting subscriber-defined QoS constraints in publish/subscribe systems. *Concurrency and Computation: Practice and Experience*, 2011.

[32] M. A. Tariq, B. Koldehofe, G. G. Koch, and K. Rothermel. Distributed spectral cluster management: A method for building dynamic publish/subscribe systems. In *Intl. conf. on dist. event-based systems (DEBS)*, 2012.

[33] B. Y. Wu and K.-M. Chao. *Spanning Trees and Optimization Problems*. Chapman and Hall, 2004.

[34] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee. How to model an internetwork. In *IEEE intl. conf. on comp. commun. (INFOCOM)*, 1996.

[35] X. Zhang and C. Phillips. A survey on selective routing topology inference through active probing. *IEEE Communications Surveys & Tutorials*, 2011.

[36] Y. Zhu, B. Li, and K. Q. Pu. Dynamic multicast in overlay networks with linear capacity constraints. *IEEE Trans. Parallel Distrib. Syst.*, 2009.