

# Dynamic Publish/Subscribe to Meet Subscriber-Defined Delay and Bandwidth Constraints\*

Muhammad Adnan Tariq, Gerald G. Koch, Boris Koldehofe,  
Imran Khan, and Kurt Rothermel

IPVS - Distributed Systems, University of Stuttgart  
{firstname.lastname}@ipvs.uni-stuttgart.de

**Abstract.** Current distributed publish/subscribe systems assume that all participants have similar QoS requirements and equally contribute to the system's resources. However, in many real-world applications, the message delay tolerance of individual peers may differ widely. Disseminating messages according to individual delay requirements not only allows for the satisfaction of user-specific needs but also significantly improves the utilization of the resources in a publish/subscribe system. In this paper, we propose a peer-to-peer-based approach to satisfy the individual delay requirements of subscribers in the presence of bandwidth constraints. Our approach allows subscribers to dynamically adjust the granularity of their subscriptions according to their bandwidth constraints and delay requirements. Subscribers maintain the publish/subscribe overlay in a decentralized manner by establishing connections to peers that provide messages meeting exactly their subscription granularity and complying to their delay requirements. Evaluations show that for practical workloads, the proposed system scales up to a large number of subscribers and performs robustly in a very dynamic setting.

## 1 Introduction

Publish/subscribe is an important many-to-many communication paradigm for applications with loosely coupled entities where providers of information *publish* events while recipients *subscribe* to them. The advantage of this paradigm is the decoupling of *publishers* and *subscribers*: Events can be published by providers without knowledge on the set of relevant recipients, while recipients express their interest in certain information without the need to know the actual set of its providers.

The evolution of publish/subscribe has followed two main objectives, namely an increased decentralization and an increased orientation on the participants' specific needs. Former static broker-based architectures were overcome by decentralized systems where publishers and subscribers contribute as peers to the

---

\* This work was partially funded by the SpoVNet project of Baden-Wuerttemberg Stiftung gGmbH.

dynamic maintenance of the publish/subscribe system and where they perform the dissemination of events collectively. Specific needs of subscribers were met by the transition from topic-based and channel-based publish/subscribe to content-based publish/subscribe. Its expressive way to subscribe allows the definition of subscriber-specific restrictions on the event message content.

There is still potential for the adaptation of publish/subscribe to peer-specific needs. For instance, many current systems assume that all subscribers expect the same quality of service (QoS) for their requested events. In fact, for many real-world settings, events are of different importance to individual subscribers which can therefore subscribe with different QoS requirements. Consider, for example, meteorological sensor information such as temperature and wind fields. The data itself is relevant for a large number of application entities such as news agencies, traffic monitoring, energy management, and rescue services. However, while local rescue services need to react fast and cannot tolerate large transmission delays, other recipients like a weather forecast service which has a large prediction window do not have that strict delay requirements. Accounting for individual QoS requirements is promising to better utilize the system's resources. Again, resources such as bandwidth should be considered peer-specific constraints for the maintenance of the system rather than system constants.

Considering peer-specific contributions, needs and constraints in publish/subscribe systems is severely complicated by its inherent decoupling. Therefore, in literature, only few approaches have addressed QoS for publish/subscribe. Solutions supporting message delay bounds either assume static topologies [20] or rely on complex management protocols such as advertisement and subscription forwarding to manage end-to-end state information with respect to each publisher [6,17] and therefore constrain the system's scalability. Peer-specific resource contribution and its inter-dependencies to user-specific delay requirements have not been discussed yet in literature.

In this paper, we present a broker-less content-based publish/subscribe system which satisfies the peers' individual message delay requirements and supports system stability by accounting for resources contributed by individual peers. Subscribers arrange in an overlay so that subscribers with tight delay requirements are served first and then forward messages to peers with lesser requirements. Peers are motivated to contribute some of their bandwidth on receiving and forwarding events which do *not* meet their own subscriptions (*false positives*) in exchange for an increased opportunity to satisfy their individual delay requirements. Therefore, peers with tight delay requirements also significantly contribute to the stability of the publish/subscribe system, while they are still in control of their individual permissible ratio of false positives and thus can consider their bandwidth constraints. The evaluations demonstrate the viability of the proposed system under practical workloads and dynamic settings.

## 2 System Model and Problem Formulation

We consider a broker-less content-based publish/subscribe system consisting of an unbounded set of peers. Peers leave and join the system at arbitrary time, and

they can fail temporarily or permanently. The peers act as publishers and/or subscribers which connect in an overlay and forward events to relevant subscribers. The set of overlay connections of a peer  $s$  can be classified into incoming connections  $F_{in}(s)$  and outgoing connections  $F_{out}(s)$ . We support event forwarding using an out-degree constraint  $m$ . It obliges peer  $s$  to be ready to forward received messages up to  $m$  times ( $F_{out}(s) \leq m$ ). The rate  $R(s)$  of events received over connections in  $F_{in}(s)$  is therefore constrained: it must not consume more than a fraction  $\frac{B(s)}{m+1}$  of the overall bandwidth  $B(s)$  provided by the access link that connects  $s$  with the physical network.

The basis for all events and subscriptions is the event space denoted by  $\Omega$ . It is composed of a global ordered set of  $d$  distinct attributes ( $A_i$ ):  $\Omega = \{A_1, A_2, \dots, A_d\}$ . Each attribute  $A_i$  is characterized by a unique *name*, its *data type* and its *domain*. The data type can be any ordered type such as integer, floating point and character strings. The domain describes the range  $[L_i, U_i]$  of possible attribute values.

The relations between events, subscriptions and advertisements can be demonstrated by modelling  $\Omega$  geometrically as a  $d$ -dimensional space so that each dimension represents an attribute. A publisher's advertisement is a sub-space of that space, and a published event is a single point  $\omega$  in the space. A subscription is a hyper-rectangle in  $\Omega$ . An event is *matched* by a subscription, iff the point  $\omega$  defined by the event is located within the hyper-rectangle defined by the subscription. A subscription  $sub_1$  is *covered by* a subscription  $sub_2$ , iff the hyper-rectangle of  $sub_1$  is enclosed in the hyper-rectangle of  $sub_2$ .

Apart from that, we allow a subscriber  $s$  to specify the delay  $\Delta(s)$  that it is willing to tolerate when receiving events from any of its relevant publishers.

In the publish/subscribe system described above, a peer clearly has two concerns. The first is to receive all relevant messages in compliance with its delay requirements. The second is, for the sake of saving bandwidth, to receive and forward only messages that exactly match the peer's subscription.

More precisely, let  $S$  be a set of subscribers and  $P_S$  the set of publishers that publish events matching the subscriptions of  $S$ .  $E$  denotes the set of all overlay links and  $path(p, s) = \{(p, i_1), (i_1, i_2), \dots, (i_m, s)\} \subseteq E$  defines the set of overlay links on the path from a publisher  $p \in P_S$  over intermediate nodes  $i_j$  to a subscriber  $s \in S$ . The delay on this path is defined as  $D(p, s) = \sum_{e \in E: e \in path(p, s)} d(e)$  where  $d(e)$  denotes the link delay on a link  $e \in E$ . The objective is to maintain the publish/subscribe overlay network in the presence of *dynamic* sets of publishers  $P_S$  and subscribers  $S$ , so that

1. the delay constraints of a large number of subscribers are satisfied w.r.t. the sets of their relevant publishers (ideally, in the presence of sufficient resources,  $\forall s \in S, \forall p \in P_S : D(p, s) \leq \Delta(s)$ ), and
2. each subscriber can dynamically adjust the rate of false positives it receives so that its bandwidth constraints are not violated, i.e.  $\frac{B(s)}{m+1} \geq R(s)$ .

Our approach can work with any monotonically increasing delay metric. However, for simplicity, in our algorithm description we use the hop count as delay metric, i.e.  $D(p, s) = |\{e \in E | e \in path(p, s)\}|$ .

### 3 Approach Overview

Meeting the objectives presented in *Section 2* amounts to finding a good trade-off between two contradicting goals: to minimise resource usage by avoiding false positives (i.e., a subscriber  $s$  receives and therefore forwards only messages that match its own subscription), and to ensure scalability by balancing the contribution of the peers according to their available resources.

Fulfilling the first goal affects the scalability of the overall system especially in the presence of out-degree constraints. In the content-based model, subscriptions often intersect with each other rather than being in a containment relationship. Hence, the complete removal of false positives may require subscribers to maintain large number of incoming connections in order to cover their subscriptions [15]. Therefore, false positives cannot be completely avoided and peers need to contribute resource in terms of false positives to ensure scalability. However, allowing individual peers to induce false positives by arbitrarily coarsening their subscriptions without any regularity is unrewarding due to the fact that coarser subscriptions may still intersect instead of being in a containment relationship.

We therefore propose to coarsen subscriptions systematically by distinguishing between two levels of subscriptions: user-level and peer-level, as shown in *Figure 1*. The user-level subscription represents the original subscription as defined by the application. The peer-level subscription is an approximation of the user level subscription and defines which events a peer actually receives.

The peer-level subscription is created by spatial indexing [9,14]. The event space is divided into regular sub-spaces which serve as enclosing approximations for user-level subscriptions. The sub-spaces are created by recursive binary decomposition of the event space  $\Omega$ . The decomposition procedure divides the domain of one dimension after the other and recursively starts over in the created sub-spaces. *Figure 2* visualizes the advancing decomposition. Sub-spaces can be identified by *dz-expressions*. A *dz-expression* is a bit-string of “0”s and “1”s, which is empty ( $\epsilon$ ) for  $\Omega$ . Each time a sub-space is divided, its *dz-expression* is inherited as prefix for the *dz-expressions* of the newly created sub-spaces.

The peer-level subscription of a peer  $p$  can be composed of several sub-spaces and is therefore represented by a set of *dz-expressions* denoted by  $DZ(p)$  with  $DZ(p) = \{dz_i \mid i \geq 1\}$ . For instance, in *Figure 2*, the accurate mapping of  $sub_1 = \{humidity = [0, 25] \wedge Temp = [0, 100]\}$  requires two sub-spaces in its peer-level subscription. The mapping is  $sub_1 \mapsto \{000, 010\}$ .

If the mapping between the subscriptions at user and peer level is identical, the peer will only receive events matching its user-level subscription. In general, however, a peer can coarsen its peer-level subscription in a regular manner so that additional events can occupy a share of its bandwidth. For example,  $sub_1$  in *Figure 2* can be coarsened by mapping it to the sub-space 0, i.e.  $sub_1 \mapsto \{0\}$ .

The regularity of sub-spaces created by spatial indexing is advantageous due to the fact that overlapping sub-spaces are always in a containment relationship, which can be directly mapped to the overlay structure as discussed in *Section 4*. Additionally, subscriptions can be coarsened or refined in a regular manner. This lesser degree of freedom in the selection of false positives also helps in the

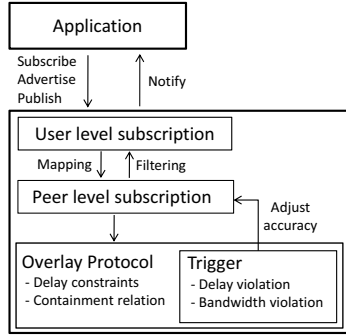


Fig. 1. Architecture

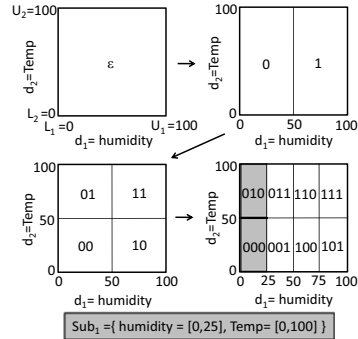


Fig. 2. Spatial indexing

anticipated bandwidth estimation of the sub-spaces that are considered for use in coarser subscription as detailed in *Section 5*.

## 4 Overlay Protocol

Subscribers maintain the overlay in a decentralized manner by connecting and disconnecting other peers. In particular, subscribers satisfy their peer-level subscriptions and delay requirements by connecting to subscribers or publishers that have covering subscriptions and tighter delay requirements. Thereby, subscribers just rely on the subscription and the delay constraints of the peers they are connecting to, and on the fact that these in turn connect to suitable peers.

For the satisfaction of its subscription, a peer  $p$  needs to discover a suitable parent for each of its  $dz_i$  in  $DZ(p)$ . Furthermore, dynamic conditions such as churn, failures and changes in the delay requirements may require a previously suitable parent to be replaced. Therefore, each subscriber maintains a peer view  $pView^1$  that caches information about peers which are relevant because they have covering subscriptions.

**Overlay maintenance:** Periodically, each peer  $p$  runs the *connectionManagement* procedure (cf. *Algorithm 1*, lines 1-6) to check whether each  $dz_i$  in  $DZ(p)$  is covered either by the subscription of a subscriber or by all of the relevant publishers in  $F_{in}(p)$ .<sup>2</sup> If any  $dz_i$  is not covered, the *findBestParent* routine selects a suitable parent from  $pView$ . Peer  $p$  sends a connection request to this potential parent once it is selected.

**Connection request:** Upon reception of a connection (CONNECT) request from a peer  $p$ , the potential parent  $q$  will normally acknowledge the connection, but it will reject the request if  $\Delta(p) > \Delta(q)$  does not hold. In this case,  $q$  sends a hint about the most suitable parent for  $p$  according to  $q$ 's knowledge.

<sup>1</sup> In our implementation we modified an epidemic protocol for maintaining  $pView$ .

<sup>2</sup> The set of relevant publishers is maintained similar to  $pView$ .

**Algorithm 1.** Publish/subscribe overlay maintenance

---

```

1: procedure connectionManagement do
2:   while true do
3:     if  $\exists dz_i \in DZ(p) | dz_i$  is not covered then
4:        $parent = \mathbf{findBestParent}(pView, dz_i)$ 
5:        $pView = pView - parent$ 
6:       trigger Send(CONNECT, p, parent,  $dz_i$ ,  $\Delta(p)$ )
7:   upon event Receive(CONNECT, p, q,  $dz(p)$ ,  $\Delta(q)$ ) do
8:     if  $\Delta(p) > \Delta(q)$  then
9:        $F_{out}(q) = F_{out}(q) \cup p$ 
10:      if  $|F_{out}(q)| > m$  then
11:         $peer[] = \mathbf{peersToDisconnect}()$ 
12:        for all  $t \in peer$  do
13:           $parent = \mathbf{findBestParent}(pView \cup F_{out}(q), dz(t))$ 
14:          trigger Send(DISCONNECT, t)
15:          trigger Send(POTENTIALPARENT, t, parent)
16:        if  $p \notin peer$  then
17:          trigger Send(ACK, q)
18:      else //  $\Delta(p) \leq \Delta(q)$ 
19:         $parent = \mathbf{findBestParent}(pView \cup F_{in}(q), dz(p))$ 
20:        trigger Send(POTENTIALPARENT, p, parent)
21:   upon event Receive(ACK, q) do
22:      $F_{in}(p) = F_{in}(p) \cup q$ 
23:      $iCon = \{dz(a_i) | a_i \in F_{in}(p) \wedge \Delta(a_i) \neq 0\}$ 
24:     //  $DZ(p)$  should be covered exactly once, therefore remove unnecessary parents
25:     for all  $dz(a_i) \in F_{in}(p)$  do
26:       for all  $dz(a_j) \in iCon : j \neq i$  do
27:         if  $dz(a_i) \prec dz(a_j)$  then //  $dz(a_i)$  is covered by  $dz(a_j)$ 
28:            $iCon = iCon - dz(a_i)$ 
29:            $F_{in}(p) = F_{in}(p) - a_i$ 
30:           trigger Send(DISCONNECT,  $a_i$ )

```

---

Accepting peer  $p$  as a child may violate the out-degree constraints of the peer  $q$ . In this case, the *peersToDisconnect* routine prepares the disconnection from children with a highly selective subscriptions and a large  $\Delta$ . If  $p$  is chosen for disconnection, it will receive a hint (POTENTIALPARENT) message instead of a connection acknowledgement.

Upon reception of a hint (POTENTIALPARENT) message, a peer will add the hint to its  $pView$  and consider it as a potential parent in its next iteration of the *connectionManagement* procedure.

*Connection acknowledgement:* Upon reception of an acknowledgement (ACK) message, a peer  $p$  ensures that its peer-level subscription is covered exactly once by parent subscribers. This ensures that  $p$ 's bandwidth is not wasted in receiving duplicate events. For sub-spaces of  $p$ 's subscription that cannot be covered by parent subscribers, coverage must be accomplished by connecting to *all* relevant publishers. Thus, for each of such sub-spaces that are only covered by one or more publishers,  $p$  continues to search for relevant publishers or subscribers.

*Placement of publishers:* Similar to subscriptions, an advertisement of a publisher is represented by a set of  $dz$ -expressions ( $DZ$ ). This allows the automatic discovery and inclusion of the publishers in the overlay network, as a result of connection requests (CONNECT) from subscribers. Publishers maintain their  $F_{out}$  connections similar to subscribers (cf. *Algorithm 1, lines 9-17*).

### 5 Triggers for Changes in Accuracy

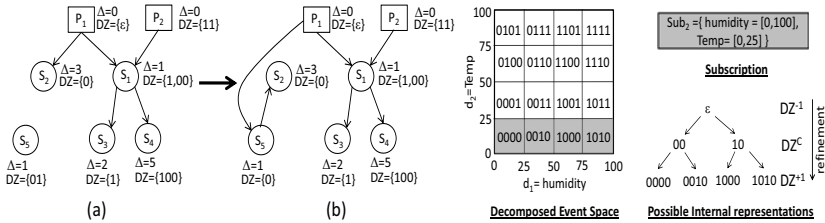
Until now we have described the organization and maintenance of the publish/subscribe overlay in the presence of subscriber-specified delay requirements. Nevertheless, we need additional mechanisms to ensure the scalability of the scheme. Sometimes a peer cannot find any potential parent to satisfy its delay constraints. In *Figure 3(a)*, for instance, subscriber  $S_5$  has a rather selective subscription and tight delay requirements. If the publisher  $P_1$  cannot accommodate more child subscribers, then  $S_5$  can only connect to  $S_2$  according to *Algorithm 1*. However, doing so violates the delay constraints of  $S_5$ . In this case  $S_5$  can coarsen its peer-level subscription according to its bandwidth constraints in order to be placed between  $P_1$  and  $S_2$ . This is possible because the overlay maintenance strategy places subscribers with less selective subscriptions higher in the dissemination graph (cf. *Algorithm 1, lines 7-17*). Therefore, subscribers can improve the probability to satisfy their delay requirements by agreeing to a coarser subscription as shown in *Figure 3(b)*.

Similarly, if changes in the event rate violate the bandwidth constraints, a subscriber refines its subscription accordingly. In this case, however, there will be no change in the set of existing parents as the new subscription is covered by the previous subscription (cf. *Algorithm 2, lines 4-6*).

In the following sections, we describe the mechanisms to adjust the accuracy of the mapping between user and peer level subscriptions according to subscriber-specific bandwidth constraints.

#### 5.1 Accuracy of Subscription Mapping

A subscriber can reduce the accuracy of the peer-level subscription by using a coarser mapping  $\xrightarrow{c}$  from the user-level subscription  $sub$  to a smaller set



**Fig. 3.** Example Scenario with  $m = 2$       **Fig. 4.** Subscriber-defined Accuracy

---

#### Algorithm 2. Triggers for change in accuracy

---

- 1: **upon event** Timeout **do**
  - 2:    **if**  $\exists dz_i \in DZ(p) | dz_i$  is not covered **then**
  - 3:     reduce accuracy of peer-level subscription by coarsening
  - 4: **upon event** BandwidthViolated **do**
  - 5:    increase accuracy of peer-level subscription accordingly
  - 6:    remove subscribers in  $F_{out}(p)$  which are not covered by the new  $DZ(p)$ .
-

of coarser dz-expressions  $DZ^C$ . Reduced accuracy causes false positives and increases bandwidth usage. Therefore, a condition for selecting a  $\vdash^c$  mapping on peer  $s$  is that the reduction of accuracy does not violate the peer's bandwidth constraint  $B(s)$ . The subscriber can ensure this by iteratively selecting another coarse mapping, thereby refining or coarsening individual dz-expressions and thus controlling the overall rate of received events.

The bandwidth usage induced by each dz-expression depends on the rate of events matched by the expression. Therefore, for each sub-space represented by a dz-expression in  $DZ^C$ , the subscriber continuously studies the event rates in the sub-space that is divided once less ( $DZ^{-1}$ ) and in the sub-spaces that are divided once more ( $DZ^{+1}$ ). The latter can be calculated by counting the received messages, while the event rate in the coarser sub-space is estimated by means of statistical aggregation [11]. The estimation of the event rate in the coarser sub-space relies on the measurements of other subscribers that are currently subscribed to the coarser sub-space or a part of it. The measurements appear in the messages of the protocol used to maintain *pView* (Section 4).

Figure 4 shows the possible mappings from a user-level subscription. If the subscription is currently mapped to  $DZ^C = \{00, 10\}$  then the subscriber keeps track of the event rates in the sub-spaces  $DZ^{-1} = \{\epsilon\}$  and  $DZ^{+1} = \{0000, 0010, 1000, 1010\}$ . If there is a high rate of false positives in a sub-space of the current peer-level subscription, the subscriber will drop it and select the relevant of the finer sub-spaces from  $DZ^{+1}$  instead. Similarly, the subscriber can select one sub-space from  $DZ^{-1}$  instead of multiple previous enclosed sub-spaces and receive additional false positives. If the event rate and the subscriptions in the system remain constant, this strategy will allow the subscriber to converge to a state where its dz-expressions no longer need to be adjusted.

## 5.2 Optimized Spatial Indexing

For an event space with a large set of attributes, the number of dz-expressions for an accurate subscription representation can be very large. As described in Section 5.1, a coarse subscription mapping reduces the number of dz-expressions. However, it induces false positives and hence its applicability depends on the bandwidth constraints of the subscriber.

A simple modification in the representation of dz-expressions can reduce their number without changing their accuracy. A *dz-expression* is redefined to include the wild-card  $*$  which stands for “0 and 1”. Two dz-expressions that differ in only one place can be combined by replacing this place by “\*”. For example, the subscription in Figure 4 can be represented by one dz-expression “\*0\*0”.

Dz-expressions of that form are created by a modified spatial indexing mechanism (Section 3). The decomposition procedure works mainly as before. Only if the subscription covers the complete domain of the dimension to be divided, then instead of creating two dz-expression for the smaller sub-spaces (ending with 0 and 1),  $*$  is added to the dz-expression.

The containment relationship defined on dz-expressions as well as the subscription mapping and bandwidth estimation mechanisms work with the

modified technique. Furthermore, the modification allows subscribers to define constraints only on a subset of attributes in the event space.

## 6 Evaluation

In this section, we evaluate the performance of the presented algorithms according to the following criteria: i) convergence to subscription and delay constraint satisfaction, ii) control overhead, iii) adaptability to dynamic conditions, iv) scalability in terms of number of peers and attributes, and v) effect of bandwidth consumption on the satisfaction of subscribers.

**Experimental Setup:** Simulations are performed using PeerSim [12]. Each peer relies on *Gossip-based peer sampling service* [13] to maintain its partial view (*pView*) of 5% other peers in the system. Unless otherwise stated, all the simulations are performed for  $n = 1,024$  peers. The out-degree constraints of the peers are chosen as  $m = \log_2(n)$ . The event space has up to 10 different attributes. The data type of each attribute is Integer, and the domain of each attribute is the range  $[1, 128]$ . We evaluated the system performance under uniform ( $W_1$ ) and skewed ( $W_2$ ) subscription workloads; and with a uniform event distribution. Skew is simulated by twenty randomly chosen hot spots in the event space, around which subscriptions are generated using the widely used 80%-20% Zipfian distribution. We use the following performance metrics in our evaluations:

- 1) *Percentage of converged peers:* The fraction of peers out of the total population which have found a suitable set of parents that cover their subscription.
- 2) *Percentage of notified peers:* The fraction of peers which are receiving events from all the relevant publishers without violating their delay constraints.
- 3) *Control Messages:* The control overhead in terms of number of connection request messages that a peer sent before finding its appropriate set of parents.
- 4) *Construction time:* The time needed to complete the construction of the overlay topology.

**Convergence:** In this experiment, moderate delay requirements are assigned to the peers such that convergence can be achieved. *Figures 5(a)-(b)* show the construction time for the overlay topology. For all of the workloads, the percentage of notified peers is always less than that of converged peers until 100% convergence is achieved. The reason is that the peers opportunistically connect to other peers in order to cover their subscriptions and satisfy their delay constraints. Therefore, during the evolution of the overlay topology, many separate isolated groups of peers may exist. Some of these groups may not have found a connection to the relevant publishers. Eventually, all the groups converge to one overlay topology.

The overlay construction time for workload  $W_2$  is higher due to the fact that the subscription distribution is highly skewed with very little overlap between the subscriptions of peers assigned to different hot-spots. This results in subscribers with coarser subscriptions occupying all the places near the publishers, forwarding events that only correspond to a portion of the event space. Therefore, the

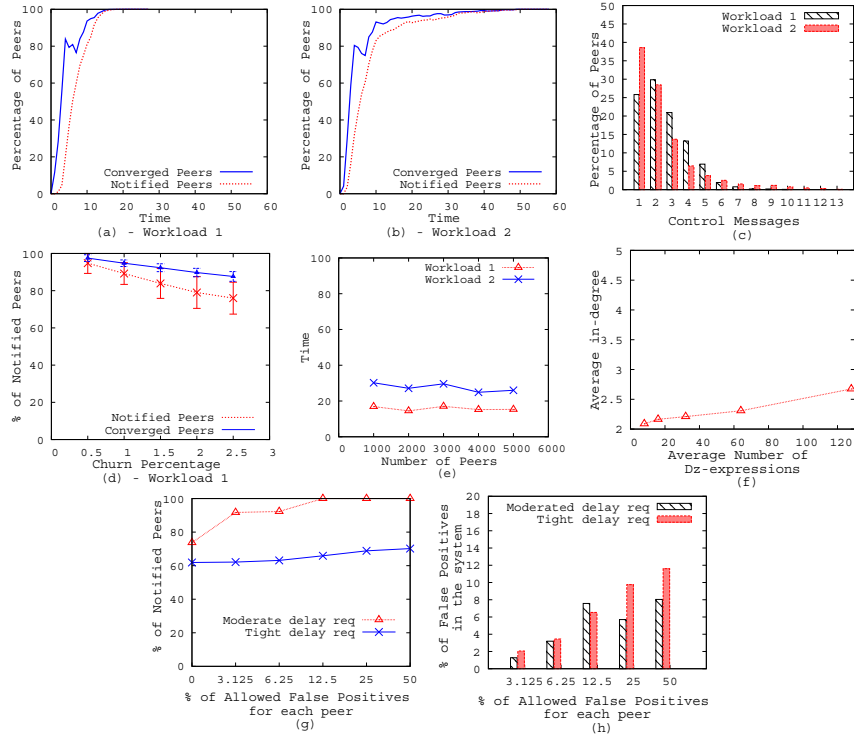


Fig. 5. Evaluations of the presented algorithms

subscribers with finer subscriptions (to uncovered portions of the event space) have to increase their subscription to compete with subscribers with coarser subscriptions. *Figure 5(c)* shows the control overhead incurred by the peers in order to find suitable parents. It shows the percentage of the affected peers as a function of the number of connection request messages sent by them.

**Adaptability:** This experiment evaluates the dynamic resilience of the system in the presence of continuously joining and leaving subscribers. The percentage of churn is relative to the total of all peers in the system. For instance, a churn of 2.5% means that in each time step, 25 on-line peers leave and the same number of new peers with different subscriptions and delay requirements join the system. *Figure 5(d)* shows the percentage of converged and notified peers for different percentages of churn along with the standard deviation for  $W_1$ . The reason for the gradual degradation in the percentage of notified and converged peers is due to the fact that a high churn rate increases the probability that peers placed near the publishers leave the system, affecting the delay constraint satisfaction of all

their descendant subscribers. The evaluation results obtained from  $W_2$  shows similar trend.

**Scalability:** First, we study the scalability with respect to the number of peers in the system. In all the experiments the out-degree constraints are chosen as  $\log_2(n)$  of the total number of peers  $n$ . *Figure 5(e)* shows that up to 5000 peers the overlay construction time almost stays the same. Furthermore, the overlay construction time for  $W_2$  is in general higher due to the fact that the subscription distribution is skewed and some subscribers may need to increase their subscriptions as discussed in the convergence evaluations.

Next, we study the effect of the number of attributes in the event space on the system's scalability. The number of dz-expressions needed for the accurate representation of a user-level subscription generally increases with the number of attributes. A peer maintains a suitable parent for each of its dz-expressions. Therefore, we study the effect of an average increase in the number of dz-expressions on the average in-degree for  $W_1$  as shown in *Figure 5(f)*. The averages are taken over all the peers in the system and the out-degree constraints of the peers are kept constant during the experiment. The results show a slight increase in the average in-degree with the number of dz-expressions, i.e., increasing the average number of dz-expressions from 8 to 128 increases the average in-degree by just 0.7 to 2.7.

**Effect of bandwidth on the satisfaction of delay requirements:** In this experiment two scenarios are evaluated: one where the subscribers are assigned moderate delay requirements ( $S_1$ ) and the other with tight delay requirements ( $S_2$ ). In both the scenarios, delay requirements of all the subscribers cannot be satisfied without inducing false positives. All the subscribers are assigned the same bandwidth constraints, specified in terms of allowed false positives as a percentage of the overall event rate. For example, 3.125% of allowed false positives means that subscribers can increase their subscription till they are receiving 3.125% of overall events in the system as false positives. *Figure 5(g)* shows the percentage of notified peers for different percentages of allowed false positives, and *Figure 5(h)* shows the actual percentage of false positives in the system for the scenarios  $S_1$  and  $S_2$ . In case of  $S_1$ , only 73.7% of subscribers are notified in the absence of false positives. However, allowing peers to receive up to 12.5% of overall events as false positives increases the percentage of notified peers by 26.3% to 100% with only 7.5% increase in the overall rate of false positives in the system. In contrast, in scenario  $S_2$ , even when the subscribers are allowed to increase their false positives up to 50% of overall event rate, the percentage of notified peers increases by only 8.3% to 70.2%. The reason is that the delay requirements of subscribers in  $S_2$  are very tight and that it is not possible to satisfy all of them. In this case, the unsatisfied subscribers coarsen their subscriptions to get a better place in the overlay. However, as all the subscribers have similar bandwidth constraints and there are limited places to satisfy delay requirements, coarsening subscriptions does not give them any competitive advantage. It just raises the overall rate of false positives.

## 7 Related Work

Over the last decade, many content-based publish/subscribe systems have been proposed with scalability as the main design criterion [7,10,5,3]. In order to achieve scalability, a large number of unnecessary events (false positives) are clearly undesirable and should be avoided [15]. Many recent systems address scalability by clustering the subscribers with similar interests [1,8]. Sub-2-Sub [19] clusters subscribers with non-intersecting subscriptions into rings and completely avoids false positives. However, even for a moderate number of subscribers, the number of clusters may quickly grow to a very large number, limiting the scalability of the approach [15]. Similarly, techniques from data mining have been used to group subscriptions in a limited number of clusters [16], but this requires central coordination. Apart from the stated drawbacks, existing approaches [2,4] only focus on the overall reduction of false positives without taking into account the heterogeneity of subscribers in terms of QoS requirements to better utilize resources in a publish/subscribe system.

Only few publish/subscribe systems address issues related to QoS. IndiQoS [6] addresses individual delay requirements, but it relies on subscription and advertisement forwarding mechanisms to maintain end-to-end delay bounds and to reduce false positives. These mechanisms introduce an extra overhead and have a restricted efficiency with widely dispersed subscribers [15]. Some of the problems stated above are addressed by the system presented in [17] which clusters subscribers into groups in order to reduce false positives. However, within each group, subscription and advertisement forwarding is used to maintain end-to-end delay bounds. The solution presented in the paper at hand goes a step forward, as it avoids advertisement flooding and takes into account the inter-dependencies between peer-specific resource contribution and delay requirements.

## 8 Conclusion

In this paper we have shown how the individual delay requirements of a large dynamic set of subscribers in a content-based publish/subscribe system can be satisfied without violating their bandwidth constraints. In particular, subscribers are given the flexibility to define their permissible rate of false positives according to their individual bandwidth constraints. Additionally, we propose a subscriber-driven decentralized algorithm to connect publishers and subscribers in an overlay network according to their delay requirements so that subscribers with tight delay requirements are located closer to the relevant publishers. The evaluation shows that the proposed algorithm converges to the satisfaction of subscriber-specific delay constraints even in a very dynamic setting. The ideas presented in this paper are applied to support a peer-to-peer based gaming application in the SpoVNet project where link delay information is provided by a cross-layer information framework [18].

## References

1. Anceaume, E., Gradinariu, M., Datta, A.K., Simon, G., Virgillito, A.: A Semantic Overlay for Self-Peer-to-Peer Publish/Subscribe. In: ICDCS (2006)
2. Baldoni, R., Beraldi, R., Querzoni, L., Virgillito, A.: Efficient Publish/Subscribe Through a Self-Organizing Broker Overlay and its Application to SIENA. *The Computer Journal* (2007)
3. Bhola, S., Strom, R.E., Bagchi, S., Zhao, Y., Auerbach, J.S.: Exactly-once Delivery in a Content-based Publish-Subscribe System. In: Intl. Conf. on Dependable Systems and Networks (2002)
4. Bianchi, S., Datta, A., Felber, P., Gradinariu, M.: Stabilizing Peer-to-Peer Spatial Filters. In: ICDCS (2007)
5. Briones, J.A., Koldehofe, B., Rothermel, K.: Spine: Adaptive publish/subscribe for wireless mesh networks. *Studia Informatika Universalis* 7 (2009)
6. Carvalho, N., Araujo, F., Rodrigues, L.: Scalable QoS-Based Event Routing in Publish-Subscribe Systems. In: Intl. Symposium on Network Computing and Applications (2005)
7. Carzaniga, A., Rosenblum, D.S., Wolf, A.L.: Design and evaluation of a wide-area event notification service. *ACM Trans. Comput. Syst.* (2001)
8. Chand, R., Felber, P.: Semantic Peer-to-Peer Overlays for Publish/Subscribe Networks. In: Cunha, J.C., Medeiros, P.D. (eds.) Euro-Par 2005. LNCS, vol. 3648, pp. 1194–1204. Springer, Heidelberg (2005)
9. Gaede, V., Günther, O.: Multidimensional access methods. *ACM Comput. Surv* (1998)
10. Gupta, A., Sahin, O.D., Agrawal, D., Abbadi, A.E.: Meghdoot: Content-Based Publish/Subscribe over P2P Networks. In: Intl. conf. on Middleware (2004)
11. Jelasity, M., Kowalczyk, W., van Steen, M.: An approach to massively distributed aggregate computing on peer-to-peer networks. In: Workshop on Parallel, Distributed and Network-Based Processing (2004)
12. Jelasity, M., Montresor, A., Jesi, G.P., Voulgaris, S.: PeerSim: A Peer-to-Peer Simulator, <http://peersim.sourceforge.net/>
13. Jelasity, M., Voulgaris, S., Guerraoui, R., Kermarrec, A.-M., van Steen, M.: Gossip-based peer sampling. *ACM Trans. Comput. Syst* (2007)
14. Ohsawa, Y., Sakauchi, M.: A New Tree Type Data Structure with Homogeneous Nodes Suitable for a Very Large Spatial Database. In: Proc. of the Intl. Conf. on Data Engineering (1990)
15. Querzoni, L.: Interest clustering techniques for efficient event routing in large-scale settings. In: Intl. Conf. on Distributed Event-Based Systems (2008)
16. Riabov, A., Liu, Z., Wolf, J.L., Yu, P.S., Zhang, L.: Clustering Algorithms for Content-Based Publication-Subscription Systems. In: ICDCS (2002)
17. Tariq, A., Koldehofe, B., Koch, G., Rothermel, K.: Providing probabilistic latency bounds for dynamic publish/subscribe systems. In: Proceedings of the 16th ITG/GI Conference on Kommunikation in Verteilten Systemen (KiVS). Springer, Heidelberg (2009)
18. The SpoVNet Consortium. Spontaneous Virtual Networks: On the road towards the Internet's Next Generation. *it - Information Technology* (2008)
19. Voulgaris, S., Rivire, E., Kermarrec, A.-M., van Steen, M.: Sub-2-sub: Self-organizing content-based publish and subscribe for dynamic and large scale collaborative networks. In: Int'l Workshop on Peer-to-Peer Systems (2006)
20. Wang, J., Cao, J., Li, J., Wu, J.: Achieving Bounded Delay on Message Delivery in Publish/Subscribe Systems. In: Intl. Conf. on Parallel Processing (2006)