CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2011; **23**:2140–2153
Published online 12 May 2011 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cpe.1751

Meeting subscriber-defined QoS constraints in publish/subscribe systems

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

IPVS—Distributed Systems, Universität Stuttgart, Stuttgart, Germany

SUMMARY

Current distributed publish/subscribe systems consider all participants to have similar QoS requirements and contribute equally to the system's resources. However, in many real-world applications, the message delay tolerance of individual participants 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 that participants contribute to a publish/subscribe system. In this article, 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 overlay in a decentralized manner, exclusively establishing connections that satisfy their individual delay requirements, and that provide messages exactly meeting their subscription granularity. The evaluations show that for many practical workloads, the proposed publish/subscribe system can scale up to a large number of subscribers and performs robustly in a very dynamic setting.

Received 15 November 2010; Revised 28 February 2011; Accepted 14 March 2011

KEY WORDS: content-based; publish/subscribe; QoS

1. INTRODUCTION

The publish/subscribe communication paradigm has gained high popularity because of its inherent decoupling of publishers from subscribers in terms of time, space and synchronization. Publishers inject information into the publish/subscribe system, and subscribers specify the events of interest by means of subscriptions. Published events are routed to their relevant subscribers, without the publishers knowing the relevant set of subscribers, or vice versa.

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 [1, 2] were overcome by decentralized systems [3, 4] where publishers and subscribers contribute as peers to the 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 [1, 5]. 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

*Correspondence to: Muhammad Adnan Tariq, IPVS—Distributed Systems, Universität Stuttgart, Stuttgart, Germany.
†E-mail: adnan.tariq@ipvs.uni-stuttgart.de

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 constraints in publish/subscribe systems is severely complicated by its inherent decoupling. Therefore, in the literature, only few approaches have addressed QoS for publish/subscribe. Solutions supporting message delay bounds either assume static topologies [6] or rely on complex management protocols such as advertisement and subscription forwarding to manage end-to-end state information with respect to each publisher [7, 8]. Peer-specific resource contribution and its inter-dependencies to user-specific delay requirements have not been discussed yet in the literature.

This article is a revised and extended version of [9]. 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 before subscribers with looser ones. Peers 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 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 a peer-specific out-degree constraint $m(s)$. It obliges peer s to be ready to forward received messages up to $m(s)$ times ($F_{out}(s) \leq m(s)$). The rate $R(s)$ of events received over connections in $F_{in}(s)$ is therefore constrained: it must not consume more than a fraction $B(s)/(m(s)+1)$ 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 Ω. It is composed of a global ordered set of d distinct attributes $(A_i):\Omega=\{A_1, A_2, \ldots, A_d\}$. 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 Ω 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 ω in the space. A subscription is a hyper-rectangle in Ω. An event is *matched* by a subscription, iff the point ω 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, denoted as $sub1 \prec sub2$, iff the hyper-rectangle of sub_1 is enclosed in the hyper-rectangle of sub_2.

DOI: 10.1002/cpe

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 $path(p,s) = \{(p,i_1),(i_1,i_2),\ldots,(i_m,s)\} \subseteq E$ 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 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. $B(s)/(m(s)+1) \geq R(s)$.

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 minimize 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 a large number of incoming connections in order to cover their subscriptions [10]. 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 [5, 11]. 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 Ω. 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 with the aid of a binary tree.

[‡]If round trip time (RTT) is used as a delay metric, each s should maintain end-to-end delay information w.r.t to relevant publishers. This information can be easily maintained as a part of the event dissemination strategy (cf. Section 4).

DOI: 10.1002/cpe

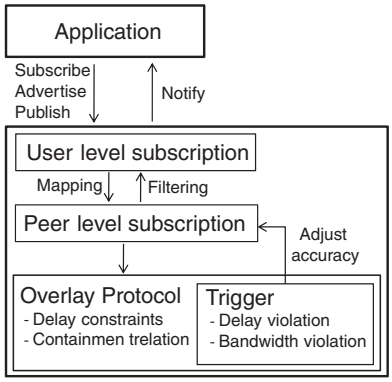

Figure 1. Architecture.

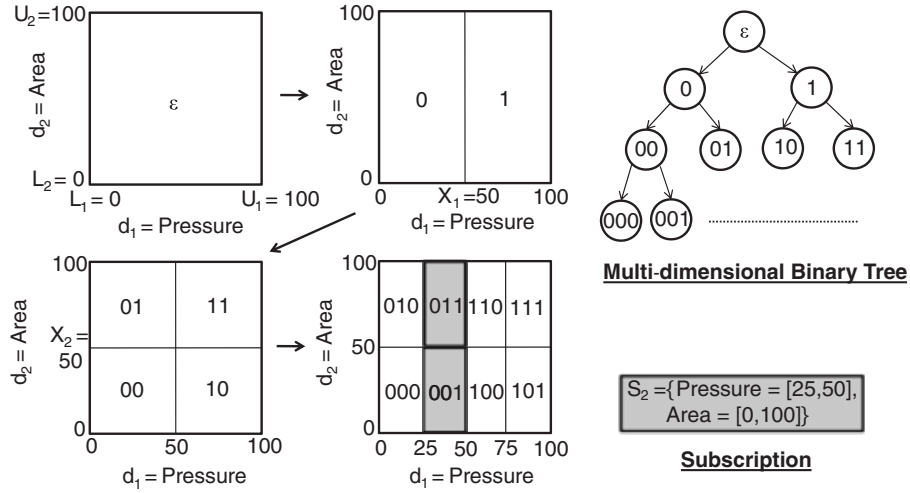

Figure 2. Spatial indexing.

Sub-spaces can be identified by *dz-expressions*. A dz-expression is a bit-string of '0's and '1's, which is empty (ε) for Ω. 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 $DZ(p)=\{dz_i | i \geq 1\}$. For instance, in Figure 2, the accurate mapping of $sub_1 = \{Pressure = [25,50] \wedge Area = [0,100]\}$ requires two sub-spaces in its peer-level subscription. The mapping is $sub_1 \longmapsto \{001, 011\}$.

If the mapping between the subscriptions at user-level 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. $sub_1 \longmapsto \{0\}$.

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 helps to take into account the rate and distribution of events in the decomposition process. In particular, it allows anticipated bandwidth estimation of the sub-spaces that may be included in coarser subscription as detailed in Section 5.

DOI: 10.1002/cpe

4. PUBLISH/SUBSCRIBE 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*[§] that caches information about peers which are relevant because they have covering subscriptions (cf. Algorithm 1, *lines 8–15*).

Overlay maintenance: Periodically, each peer p runs the *connectionManagement* procedure (cf. Algorithm 1, *lines 1–7*) 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 $F_{in}(p)$[¶]. If any dz_i is not covered, the *findBestParent* routine selects a suitable parent from *pView*, whose subscription covers dz_i. 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 $\Delta(p) \geq \Delta(q)$ does not hold. In this case, q sends a hint about the most suitable parent for p according to q's knowledge.

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 the child with highly selective subscription. Ties between the candidate children are resolved by selecting the one with a large delay. 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 acknowledgment (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.

Accepting connection from a parent may introduce a cycle in the system. To preserve acyclic topology, whenever a peer p establishes an incoming connection, a control (CYCLEDETECT) message is passed down to the peers in $F_{out}(p)$. Peers always connect to the parents with covering subscriptions and thereby cycles can only occur between the peers with similar subscription and delay constraints. This fact is used to reduce the overhead of control messages by forwarding them to only those peers, which have same subscription and delay constraints. The reception of the control message by the originator signals the presence of a cycle and the connection to the respective parent is disconnected.

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 18–26*).

Event dissemination: On reception of an event, a peer p runs Algorithm 2. Usually, a peer receives events from F_{in} and forwards them to all connections in F_{out} with a dz that covers the event. However, a newly arrived publisher may not be initially visible to all relevant peers and is connected by the peers down in the dissemination tree. To handle such cases, event dissemination

[§] In our implementation we modified an epidemic protocol for maintaining *pView*.
[¶] The set of relevant publishers is maintained similar to *pView* (cf. Algorithm 1, *lines 8–16*).

DOI: 10.1002/cpe

Algorithm 1 Publish/subscribe overlay maintenance

```
1:  procedure connectionManagement do
2:   while true do
3:      updatePviewAndPublisherList()
4:      if ∃dz_i ∈ DZ(p)|dz_i is not covered then
5:         parent = findBestParent(pView, dz_i)
6:         pView = pView − parent
7:         trigger Send(CONNECT, p, parent, dz_i , Δ(p) )
8:  procedure updatePviewAndPublisherList do
9:   for all q ∈ GossipList do // Uniform peer sample derived from epidemic protocol
10:     if ∃dz_p ∈ DZ(p), ∃dz_q ∈ DZ(q): dz_p ≺ dz_q ∨ dz_q ≺ dz_p then // p and q have covering subscriptions
11:        if Δ(q)==0 then // q is publisher so add to publisher list
12:           publisherList = publisherList ∪ q
13:        else
14:           pView = pView ∪ q
15:  Increase age and remove old elements from pView and publisherList
16:  upon event Receive(CONNECT, p, q, dz(p), Δ(p)) at peer q do
17:  if Δ(p) ≥ Δ(q) then
18:     F_out(q) = F_out(q) ∪ p
19:     if |F_out(q)|>m then
20:        peer[] = peersToDisconnect()
21:        for all t ∈ peer do
22:           parent = findBestParent(pView ∪ F_out(q), dz(t))
23:           trigger Send(DISCONNECT, t)
24:           trigger Send(POTENTIALPARENT, t , parent)
25:        if p ∉ peer then
26:           trigger Send(ACK,q)
27:  else // Δ(p)<Δ(q)
28:     parent = findBestParent(pView ∪ F_in(q), dz(p))
29:     trigger Send(POTENTIALPARENT, p, parent)
30:  upon event Receive(ACK, q) at peer p do
31:  F_in(p) = F_in(p) ∪ q
32:  iCon = {dz(a_i): a_i ∈ F_in(p) ∧ Δ(a_i) ≠ 0} // current list of subscriptions from non publisher parents
33:  Remove all dz from iCon which are covered by dz(q) // DZ(p) should be covered exactly once
34:  for all a ∈ F_out(p): Δ(a) = Δ(p) = Δ(q) ∧ dz(a) = dz(p) = dz(q) do
35:     trigger Send(DETECTCYCLE, p, dz(p), Δ(p))
36:  upon event Receive(DETECTCYCLE, originator, dz(q), Δ(q)) at peer p do
37:  if originator = p then
38:     trigger Send(DISCONNECT, q) // cycle is present through parent q
39:  else
40:     if Δ(q) = Δ(p) ∧ dz(p) = dz(q) then // delay constraints and subscriptions are same
41:        for all a ∈ F_out(q): Δ(a) = Δ(p) ∧ dz(a) = dz(p) do
42:           trigger Send(DETECTCYCLE, originator, dz(p), Δ(p))
```

Algorithm 2 Event dissemination

```
1:  upon event Receive( EVENT , msg, publisher, p, q ) at peer p do
2:  if q ∈ F_out(p) then
3:     if dz(publisher) is not covered by any peer in F_in(p) then // dz(publisher) ⊀ {⋃_{a_i ∈ F_in(p)} dz(a_i)}
4:        trigger Send(CONNECT, p, publisher, dz(p), Δ(p) )
5:  Δ(msg) = Δ(msg) + δ(q, p)
6:  if q ∈ F_out(p) ∨ q == publisher then
7:     for all a_i ∈ F_in(p): Δ(a_i) ≠ 0 ∧ msg.dz ≺ dz(a_i) do
8:        trigger Send(EVENT, msg, publisher, a_i, p)
9:  if q ∈ F_out(p) ∨ q ∈ F_in(p) then
10:    for all a_i ∈ F_out(p): a_i ≠ q ∧ Δ(a_i) ≠ 0 ∧ msg.dz ≺ dz(a_i) do
11:       trigger Send(Event, msg, publisher, a_i, p)
```

requires an additional rule: Each event received from a publisher or from F_{out} connections is forwarded to all other matching F_{in} and F_{out} connections.

If a peer p receives an event from its F_{out} connections that originates from a publisher, and publisher's advertisement is not completely covered by the dz of the p's parents (i.e. p is the top most peer), then p sends a connection request to the publisher. This strategy speeds up the placement of new publishers in the overlay network and allows the peers with the most tight delay constraints to connect directly to publishers.

DOI: 10.1002/cpe

If the delay requirements of the peer p and its parents are strictly ordered then $\Delta(p)$ will always be satisfied as long as the delay constraints of the parents are not violated. However if both p and the parent have same delay requirements then $\Delta(p)$ may be violated. In this case, peer p disconnects its parent if the delay encountered by the received event violates $\Delta(p)$.

5. TRIGGERS FOR CHANGE 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 children, 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 16–26*). 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 lead to violation of the bandwidth constraints, a subscriber will refine its subscription accordingly (cf. Algorithm 3, *lines 4–6*).

In addition to individual delay requirements, coarser subscriptions increase the overall satisfaction of subscriptions. Peers (subscribers or publishers) with out-degree constraints cannot satisfy a large number of fine-grained disjoint subscriptions. However, disjoint subscription *hotspots* are common when accessing content with internet-like popularity. In that case, even a small number of coarser subscriptions allows subscribers to find intermediary parent peers that satisfy their subscriptions, instead of being rejected by an overloaded publisher with exhausted outgoing connections (cf. Section 6, Convergence).

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

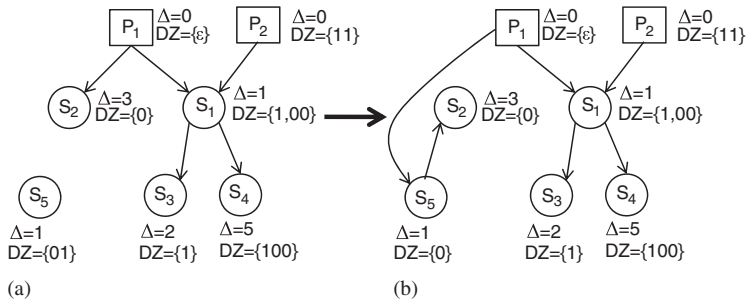

Figure 3. Example scenario with $m = 2$.

Algorithm 3 Triggers for change in accuracy

```
1: upon event TimeOut do
2:   if ∃dz_i ∈ 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 DZ(p).
```


DOI: 10.1002/cpe

5.1. Subscriber-defined accuracy

A subscriber can reduce the accuracy of the peer-level subscription by using a coarser mapping $\overset{c}{\longmapsto}$ from the user-level subscription *sub* to a smaller set of coarser dz-expressions DZ^C. Reduced accuracy causes false positives and increases bandwidth usage. Therefore, a condition for selecting a $\overset{c}{\longmapsto}$ mapping on node s is that the reduction of accuracy does not violate the node'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 [12]. 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 $DZ^{-1} = \{\varepsilon\}$ 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.

Special rules apply to new subscribers that do not know about event rates yet and cannot decide about the permissible degree of coarseness of their mapping. A new subscriber initially selects a parameter r that may for example represent the number of incoming connections that the subscriber can maintain. Then it approximates the accurate subscription DZ^A by at most r dz-expressions in DZ^C. This initial coarse mapping has the following properties:

1. $|DZ^C| \leq r$.
2. $\forall dz_a \in DZ^A \exists dz_c \in DZ^C : dz_a \prec dz_c$.

A coarse transformation of a subscription into r dz-expressions using breadth-first exploration of the binary tree of dz-expressions (cf. Figure 2) is given in Algorithm 4. The algorithm generates less

Algorithm 4 dz expression generation

```
1: DZ^C ← ∅ // Final holder of dz-expressions
2: L^d d-dimensional vector with lower bounds for each dimension
3: U^d d-dimensional vector with upper bounds for each dimension
4: Queue q ← {dz = *, L^d, U^d}
5: while !q.empty() do
6:     subSpace ← q.dequeue()
7:     i ← len(subSpace.dz) mod d // Next dimension to decompose/split.
8:     x ← (subSpace.L_i + subSpace.Ui)/2 // mid-point of the subSpace along the dimension i.
9:     if ∀_j(sub.L_j ≤ subSpace.L_j ∧ sub.U_j ≥ subSpace.U_j) then
10:        DZ^C ← subSpace.dz // completely covered
11:    else if (sub.L_j < x ∧ sub.U_j > x) ∧ (|q| + |DZ^C| < r) then // sub occupies both sides of midpoint.
12:        q ← {subSpace.dz‖0, L^d, U^d(U_i = x)}
13:        q ← {subSpace.dz‖1, L^d(L_i = x), U^d}
14:    else if sub.L_j < m ∧ sub.U_j < m then
15:        q ← {subSpace.dz‖0, L^d, U^d(U_i = x)}
16:    else if sub.L_j > x ∧ sub.U_j > x then
17:        q ← {subSpace.dz‖1, L^d(L_i = x), U^d}
18:    else
19:        DZ^C ← subSpace.dz
20: Merge terminal dz-expressions
```


DOI: 10.1002/cpe

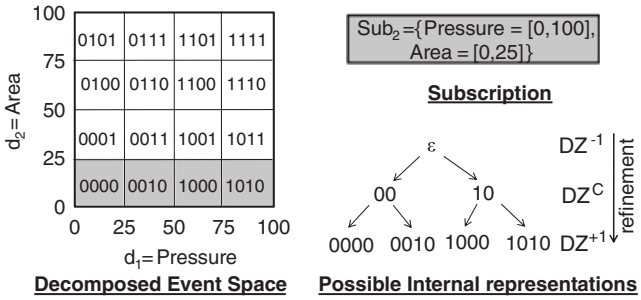

Figure 4. Subscriber-defined accuracy.

than r dz-expressions if the terminal dz-expressions generated by the splitting of a sub-space into two smaller sub-spaces can be merged into a single dz-expression without affecting the accuracy of the subscription approximation.

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 denotes '0 and 1'. Two dz-expressions that differ in only one place can be combined by replacing this place with '$*$'. 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. 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-expressions 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.

6. PERFORMANCE 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 and out-degree constraints on the satisfaction of subscribers.

Experimental setup: Simulations are performed using PeerSim [13]. Each peer relies on *Gossip-based peer sampling service* [14] to maintain its partial view (*pView*) of 5% other peers in the system. Simulations are performed for up to $n = 7000$ peers. Unless otherwise stated, 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 skewed and uniform event distributions. Skew is simulated using the widely used 80–20% Zipfian distribution with 4–6 hot spots.

[‖]Spatial indexing can work with any ordered data type with a known domain [5]. Evaluation results are not sensitive to the choice of data type and therefore, similar to [5] only integer data types are considered.

DOI: 10.1002/cpe

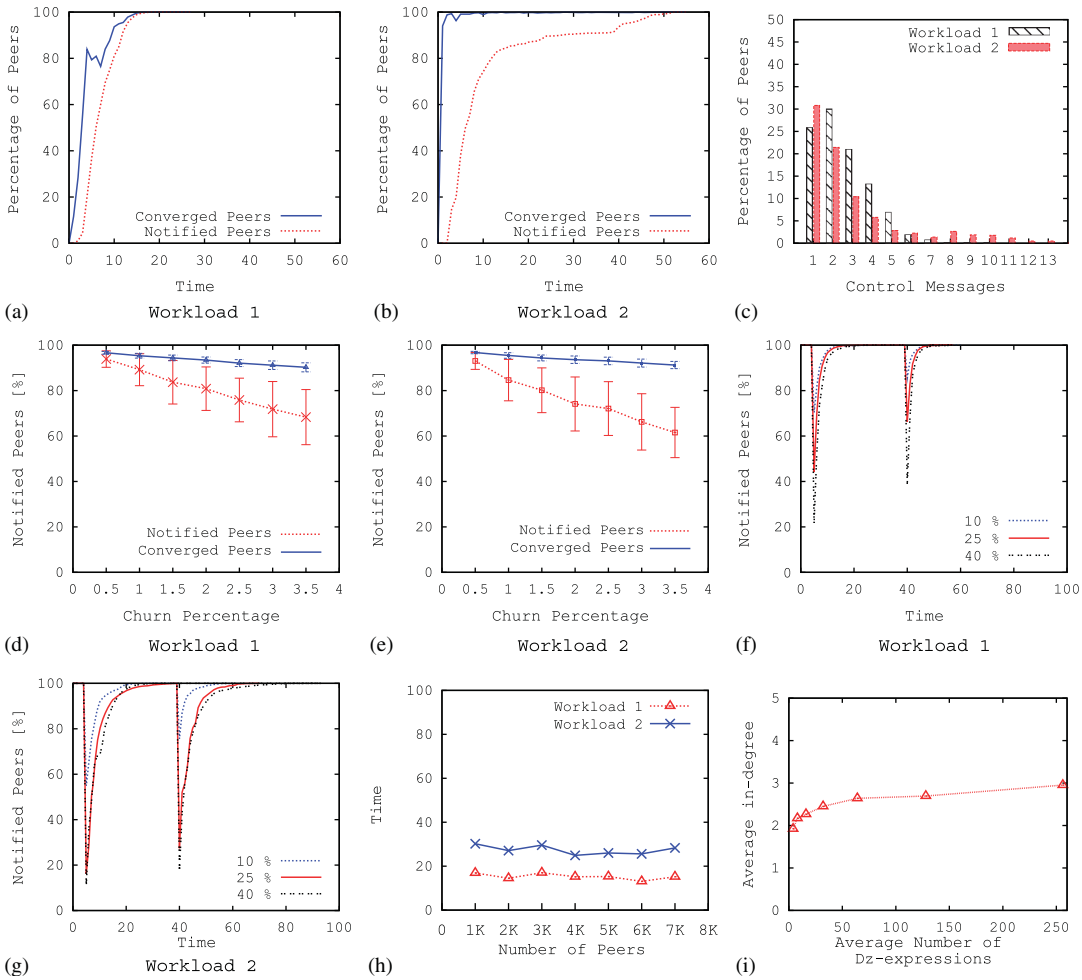

Figure 5. Evaluations to study the convergence, adaptability and scalability of the proposed algorithms.

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 that are receiving events from all the relevant publishers without violating their delay constraints.
(3) *Construction time*: Logical 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. Figure 5(a) and (b) shows the construction time for the overlay topology. For all 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 there is 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

DOI: 10.1002/cpe

only correspond to a portion of the event space. Therefore, the 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: First, we study 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, for a total number of 1000 peers, a churn of 2.5% means that in each time step, 25 online peers leave and the same number of new peers with different subscriptions and delay requirements join the system. Figure 5(d) and (e) shows the percentage of converged and notified peers for different percentages of churn along with the standard deviation. The reason for the gradual degradation in the percentage of notified and converged peers is 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 lower percentage of notified peers in W_2 is 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 evaluate the dynamic resilience of the system to sudden massive churn. Once the system converges to a stable state, massive churn is introduced in the system, i.e. 10, 25 and 40% of the online peers leave and an equal number of peers join the system. Figure 5(f) and (g) shows that the system can tolerate and recover from massive occurrences of churn. The reason for sudden degradation in the percentage of notified peers is the same as in the case of continuous churn.

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(h) shows that up to 7000 peers the overlay construction time almost stays the same. Furthermore, the overlay construction time for W_2 is in general higher because 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(i). The averages are taken over all the peers in the system. 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 4 to 256 increases the average in-degree by just 1.2–3.1.

Effect of bandwidth and out-degree constraints on the satisfaction: 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, 3.1% of allowed false positives means that subscribers can increase their subscription till they are receiving 3.1% of the overall events in the system as false positives.

Figure 6(a) shows the percentage of notified peers for different percentages of allowed false positives and out-degree constraints. Figure 6(b) shows the actual percentage of false positives in the system for the scenarios S_1 and S_2. The figures show that it is of advantage to increase the rate of false positives since it raises the overall percentage of satisfied subscribers. In the case of S_1 with $m=6$, only 65% of subscribers are notified in the absence of false positives. However, allowing peers to receive up to 6.2% of overall events as false positives increases the percentage of notified peers by 27.6% to 83% with only 1.2% increase in the overall rate of false positives in the system. Moreover, the figures show that the out-degree constraints have more profound impact on the satisfaction of subscribers especially in the presence of tight delay requirements. For

DOI: 10.1002/cpe

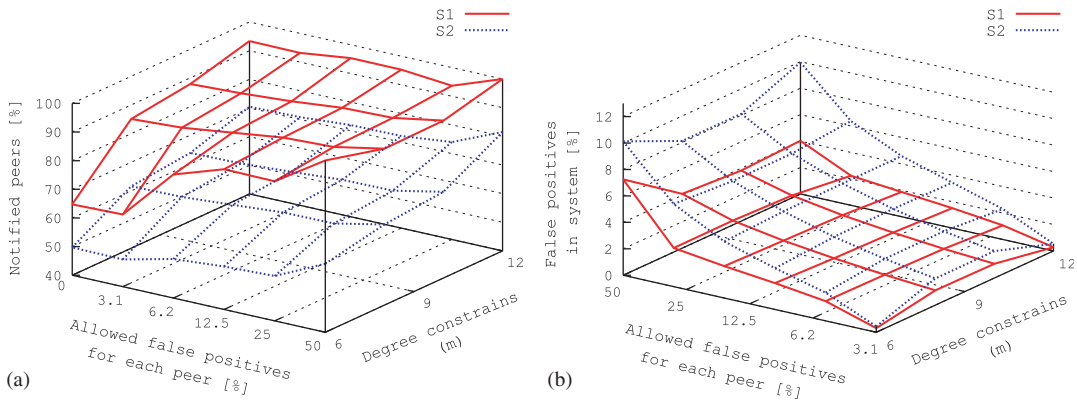

Figure 6. Effect of bandwidth and out-degree constraints on satisfaction of delay requirements.

example, in scenario S_2 with $m=6$, even when the subscribers are allowed to increase their false positives up to 50% of the overall event rate, only 64% peers are notified, mainly due to the lack of available places to connect. However, increasing the value of m to 12, raises the percentage of notified peers by 28% to 82%. Figure 6(b) also shows that the overall rate of false positives in the system is higher for scenario S_2. The reason is that the delay requirements of subscribers in S_2 are very tight and 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 [1, 2, 5, 15]. In order to achieve scalability, a large number of unnecessary events (false positives) are clearly undesirable and should be avoided [10]. Many recent systems address scalability by clustering the subscribers with similar interests [3, 16]. Sub-2-Sub [4] 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 [10]. DR-trees [17] extend R-trees to support content-based filtering in a distributed fashion. DR-trees inherit beneficial properties of R-trees such as a low rate of false positives and a logarithmic event dissemination time. However, the approach induces higher load on the peers close to the root of the DR-tree: Peers closer to the root experience more false positives without any possibility to change their rate dynamically. Similarly, techniques from data mining have been used to group subscriptions into a limited number of clusters [18], but this requires central coordination. Apart from the stated drawbacks, existing approaches [3, 16, 17, 19] 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 [20]. Bounded delays on event delivery can be achieved by employing message scheduling strategies at each broker [6]. However, this requires a static broker topology and cannot be applied to provide QoS bounds in the presence of dynamic conditions such as churn. IndiQoS [7] addresses individual delay requirements, but it relies on resource reservation mechanism to guarantee end-to-end QoS. Resource reservation protocols are not available on a global scale, which limits the applicability of the approach in heterogeneous network environments. Some of the problems stated above are addressed by the

DOI: 10.1002/cpe

system presented in [8]. Instead of relying on reservation protocols, it provides probabilistic delay bounds. To reduce false positives, subscribers are clustered into groups. However, the clustering strategy is only sketched without any evaluations. The solution presented in this article goes a step forward, as it takes into account the inter-dependencies between peer-specific resource contribution and delay requirements.

8. CONCLUSION

In this article 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. This article extends [9] to handle scenarios where all (or large number of) subscribers have the same delay constraints. Algorithmic parts of the paper are enhanced to give more insight into the system's working such as event dissemination. Finally, new evaluation results are included and previous results are extended with more realistic zipfian event distribution. The ideas presented in this article are successfully applied to support a peer-to-peer-based gaming application in the SpoVNet project [21].

ACKNOWLEDGEMENTS

This work was partially funded by the SpoVNet project of Baden-Wurttemberg Stiftung gGmbH.

REFERENCES

1. Carzaniga A, Rosenblum DS, Wolf AL. Design and evaluation of a wide-area event notification service. *ACM Transactions on Computer Systems* 2001; **19**:332–383.
2. Bhola S, Strom RE, Bagchi S, Zhao Y, Auerbach JS. Exactly-once delivery in a content-based publish–subscribe system. *International Conference on Dependable Systems and Networks*, Bethesda, MD, U.S.A., 23–26 June 2002.
3. Anceaume E, Gradinariu M, Datta AK, Simon G, Virgillito A. A semantic overlay for self-peer-to-peer publish/subscribe. *ICDCS*, Lisboa, Portugal, 4–7 July 2006.
4. Voulgaris S, Rivire E, Kermarrec A-M, van Steen M. Sub-2-sub: Self-organizing content-based publish subscribe for dynamic large scale collaborative networks. *P2P Systems Workshop*, Santa Barbara, CA, U.S.A., 2006.
5. Muthusamy V, Jacobsen H-A. Infrastructure-less content-based publish/subscribe. *Technical Report*, Middleware Systems Research Group, University of Toronto, 2007.
6. Wang J, Cao J, Li J, Wu J. Achieving bounded delay on message delivery in publish/subscribe systems. *International Conference on Parallel Processing*, Columbus, OH, U.S.A., 2006.
7. Carvalho N, Araujo F, Rodrigues L. Scalable QoS-based event routing in publish–subscribe systems. *International Symposium on Network Computing and Applications*, Cambridge, MA, U.S.A., 27–29 July 2005.
8. Tariq A, Koldehofe B, Koch G, Rothermel K. Providing probabilistic latency bounds for dynamic publish/subscribe systems. *Proceedings of the 16th ITG/GI Conference on Kommunikation in Verteilten Systemen* (*KiVS*). Springer: Berlin, 2009.
9. Tariq MA, Koch GG, Koldehofe B, Khan I, Rothermel K. Dynamic publish/subscribe to meet subscriber-defined delay and bandwidth constraints. *The 16th International Conference Parallel Computing* (*Euro-Par*) (*Lecture Notes in Computer Science*, vol. 6271), Springer: Berlin, 2010; 458–470.
10. Querzoni L. Interest clustering techniques for efficient event routing in large-scale settings. *International Conference on Distributed Event-based Systems*, Rome, Italy, 1–4 July 2008.
11. Gaede V, Günther O. Multidimensional access methods. *ACM Computing Surveys* 1998; **30**:170–231.
12. Jelasity M, Kowalczyk W, van Steen M. An approach to massively distributed aggregate computing on peer-to-peer networks. *Parallel*, *Distributed and Network-based Processing*, A Coruna, Spain, 11–13 February 2004.
13. Jelasity M, Montresor A, Jesi GP, Voulgaris S. PeerSim: A Peer-to-Peer Simulator. Available at: http://peersim.sourceforge.net/ [28 February 2011].
14. Jelasity M, Voulgaris S, Guerraoui R, Kermarrec A-M, van Steen M. Gossip-based peer sampling. *ACM Transactions on Computer Systems* 2007; **25**; DOI: 10.1145/1275517.1275520.

15. Briones JA, Koldehofe B, Rothermel K. Spine: Adaptive publish/subscribe for wireless mesh networks. *Studia Informatika Universalis* 2009; **50**:367–375.
16. Chand R, Felber P. Semantic peer-to-peer overlays for publish/subscribe networks. *Euro-Par* Lisbon, Portugal, 30 August–2 September 2005.
17. Bianchi S, Datta A, Felber P, Gradinariu M. Stabilizing peer-to-peer spatial filters. *ICDCS*, Toronto, Ontario, Canada, 25–29 June 2007.
18. Riabov A, Liu Z, Wolf JL, Yu PS, Zhang L. Clustering algorithms for content-based publication–subscription systems. *ICDCS*, Vienna, Austria, 2–5 July 2002.
19. 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; **50**:444–459.
20. Mahambre SP, SD, MK, Bellur U. A taxonomy of QoS-aware, adaptive event-dissemination middleware. *IEEE Internet Computing* 2007; **11**:35–44.
21. The SpoVNet Consortium. Spontaneous Virtual Networks: On the road towards the Internet's Next Generation. *IT—Information Technology* 2008; **50**:367–375.

DOI: 10.1002/cpe