Soft State in Publish/Subscribe

Zbigniew Jerzak*
Dresden University of Technology
Systems Engineering Group
01062 Dresden, Germany

Christof Fetzer
Dresden University of Technology
Systems Engineering Group
01062 Dresden, Germany
Christof.Fetzer@tu-dresden.de

ABSTRACT

Building survivable content-based publish/subscribe systems is difficult. Every node in a distributed publish/subscribe system stores a significant amount of routing state which can be easily corrupted due to message omissions, link and node failures. In this paper, we show how to build a soft state content-based publish/subscribe system where the whole state is stored at the edge of the publish/subscribe network, at the entity which is utilizing the state. This results in a robust and resilient system, as the routing state is permanently lost or corrupted only if the endpoint entity associated with the given state permanently fails.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed Applications*; D.1.3 [**Programming Techniques**]: Concurrent Programming—*Distributed Programming*

General Terms

Algorithms, Design

Keywords

soft state, publish/subscribe, routing, round-trip time, upper bound on message transmission delay

1. INTRODUCTION

In publish/subscribe (pub/sub) systems participants are decoupled with respect to space, time and synchronization [8]. This implies that neither information producers (publishers) know who and when is going to receive information they publish, nor information consumers (subscribers) are aware of the sources of information they subscribe to. The decoupling properties of publish/subscribe systems and their inherent many-to-many communication style have contributed to widespread adoption of the pub/sub paradigm in distributed systems [4, 26].

However, the specific environment of distributed systems, with high probability of communication and node failures [20, 33], implies the need for an approach which could be used to build survivable publish/subscribe systems. A survivable pub/sub system continues to provide service despite components and links failures [1].

In this paper, we propose a soft state [5, 27] content-based publish/subscribe system. Based on [5], we define the soft state as a system state which can be lost due to a failure without a *permanent* disruption of system features. Following the recently expressed need for the robust pub/sub systems [25], we propose a soft state-based design which allows content-based pub/sub systems to recover from temporal node, link and timing failures so that a correct system state is never permanently lost.

1.1 Background

The basic model of communication in content-based pub/sub systems involves subscriptions and events. An entity interested in a given information issues a subscription, which contains a filter summarizing entity's interest. The pub/sub system disseminates the subscription across all nodes (brokers) in the network. The goal of the subscription dissemination is to create a spanning tree rooted in the subscribing entity. The spanning tree forms a reverse path system which, once set up, is used to route events to the interested subscriber. Specifically, messages in the content-based pub/sub system do not contain any source or destination addresses. Instead, events published into the network are matched based on their content against subscriptions stored at the nodes.

Many pub/sub systems [4, 21, 17, 31] use advertisements in order to limit the propagation of subscriptions. The idea of advertisements is similar to that of subscriptions. A publisher issues an advertisement which contains a filter summarizing the content it is going to publish. Advertisements are broadcasted into the network forming spanning trees rooted at the respective publishers. Subsequent subscriptions, instead of being broadcasted into the network, are forwarded only on the reverse paths of the matching advertisements. A subscription matches an advertisement if the interest of the subscriber summarized by its subscription and the set of events summarized by the advertisement form a non-empty intersection. Advertisements allow to lower the total number of subscriptions which need to be sent into and stored in the

*Author is currently with the SAP Research (CEC Dresden) and can be reached at zbigniew.jerzak@sap.com

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
DEBS'09, July 6-9, Nashville, TN, USA.
Copyright 2009 ACM X-XXXXX-XX-X/XX/XX ...$10.00.

system, which is important for pub/sub systems with high subscription dynamism.

1.2 Motivation

The goal of the proposed soft state approach is the creation and the maintenance of the soft logical routing state in the brokers of the pub/sub system. The logical routing state is created using advertisements and subscriptions and it is stored in the routing tables of pub/sub brokers. Distributed routing tables, in turn, form advertisement and subscription trees which are the foundation upon which the actual information exchange takes place. The lack of or incomplete advertisement and subscription information implies that nodes are not able to forward events and thus it is not possible for the publishers and subscribers to communicate. It is therefore of vital importance for the survivability of the publish/subscribe system to ensure the eventually proper establishment of subscription and advertisement trees.

To better illustrate the possible issues, let us consider a simple pub/sub network presented in Figure 1. Publishers **P1** and **P2** issue advertisements $a1$ and $a2$ at 12:00, real-time. The propagation delay between the publishers **P1** and **P2** and their connecting brokers **B1** and **B2** equals 0:01. Therefore, both advertisements arrive at respective brokers at 12:01, real-time. Simultaneously, at 12:01, subscriber **S** issues a subscription s matching both advertisements $a1$ and $a2$. Subscription s arrives at the broker **B1** at 12:02, real-time – see the upper part of Figure 1. When subscription s arrives at the broker **B1**, the advertisement $a1$ from publisher **P1** is already present in the routing table of the broker **B1**.

However, due to a larger propagation delay between brokers **B1** and **B2** (equal to 0:02) the advertisement $a2$ from the publisher **P2** did not reach the broker **B1** yet. As a result subscription s will only be visible to the publisher **P1**. Specifically, the subscription s will not be forwarded to broker **B2** as the matching advertisement $a2$ has not been yet delivered to the broker **B1**. The above problem will be further strengthened if the advertisement $a2$ gets dropped or corrupted on its way from the broker **B2** to the broker **B1**.

Moreover, due to the decoupling properties of pub/sub systems, the subscriber **S** will never be able to tell that it is missing events matching its subscription s and issued by the publisher **P2**. In other words, the publisher **P2** in the above scenario will never be able to send its messages to the subscriber **S**, as the subscription s will never be propagated to the broker **B2** – see the lower part of Figure 1.

From the above sketch, we can conclude that even transient failures can have a significant impact on the functioning of the pub/sub systems. Additionally the detection of such errors is difficult due to the decoupled nature of the pub/sub systems.

1.3 Contribution

In this paper, we show how to use a soft state approach for the creation and maintenance of the routing state stored in the routing tables of a content-based publish/subscribe system deployed in the Wide Area Network environment of the PlanetLab [28].

We implement the soft state pub/sub approach on top of the XSiena [16] content-based pub/sub system. In our approach, both publishers and subscribers determine the validity (lease time) of periodically published advertisements and subscriptions. Such advertisements and subscriptions

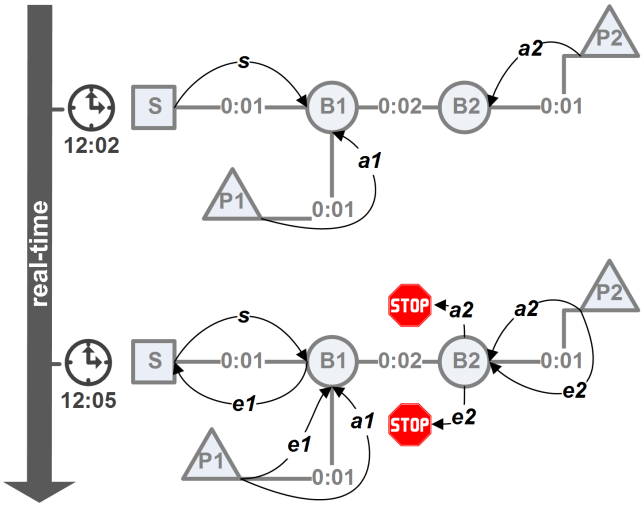


Figure 1: Subscription propagation issues due to timing relations with advertisements

need to be refreshed by the publishers and subscribers before they become invalid (the lease time expires), otherwise they are removed by the brokers from their routing tables.

Our approach is generic in nature in that it can be applied to any pub/sub system, including: topic-based [24], type-based [7], content-based [29], rule-based [18] and coverage-based [4]. Specifically, we do not restrict in any way the expressiveness of advertisements and subscriptions, including the coverage-based routing optimizations. The presented approach is also independent of the pub/sub system architecture and can be applied to systems based on the cyclic [19] and acyclic [4] architectures as well as those based on the DHT approach [26].

In this paper, we show how to use the fail-aware pub/sub approach [14] to adaptively calculate the validity of subscriptions and advertisements. Specifically, we do not require any prior information regarding the diameter of the pub/sub network nor any a priori known bounds on the message transmission delays. Our approach is fully decoupled with respect to space and synchronization and lightweight in that we do not require any kind of clock synchronization (global clock) among the nodes of the pub/sub network. In our system both periodic re-advertisements and re-subscriptions are idempotent with respect to the correct system state, thus ensuring the soft-state properties of the content-based pub/sub system.

Our approach guarantees that the pub/sub system recovers from transient failures and reaches an eventual stability in that advertisements and subscriptions are delivered to all connected and correct (non-faulty) participants. Specifically, in the scenario depicted on Figure 1 after the expiry of the advertisement $a2$ lease, the publisher **P2** will re-issue it, ensuring that it is delivered to the broker **B1**, which in turn will trigger the propagation of the subscription s. This in turn will allow events published by the publisher **P2** to reach the subscriber **S**.

An interesting property [21] of the soft state pub/sub system is that it automatically handles the effects of subscribers' and publishers' failures and unannounced departures from the system. In traditional, hard state pub/sub systems such

failures would result in the pollution of the brokers' routing tables. In hard state systems subscriptions and advertisements are stored by the brokers in their routing tables until a matching unsubscription or unadvertisement is received. If a subscriber or a publisher which issued a subscription or an advertisement crashes, neither a matching unsubscription nor a matching unadvertisement is issued. This results in a broker routing table indefinitely holding entries for inactive subscribing and publishing nodes. Soft state approach allows for the automatic expiration of such entries, thus alleviating the need for explicit unsubscriptions and unadvertisements. This in turn results in a more flexible system, as the departures of subscribers and publishers are lightweight in that no messages need to be exchanged upon such event.

2. RELATED WORK

Timed subscriptions and events have been first addressed in [9]. Authors assume that subscriptions and events are associated with time intervals, which limit their validity, however, the authors do not consider the renewal of subscriptions or advertisements which leads to the issues presented on Figure 1 in Section 1.2. Another approach for the creation of the soft state system has been presented in [3] where authors assumed a fixed subscription structure for each of the publish/subscribe system nodes and focused on the exactly once delivery of publications. Specifically, authors did not consider dynamic content-based routing using the coverage relation. The authors of [30] designed a self-stabilizing publish/subscribe system. The proposed system assumes however, that all subscriptions issued by subscribers are always broadcasted into the network. Such an algorithm is not efficient, as similar or identical subscriptions need not to be resent on links where a subscription subsuming the given one has already been sent. Moreover, the authors proposed to exchange whole routing tables in order to detect potential inconsistencies, which can be expensive in publish/subscribe systems with large number of subscribers and publishers.

The author of [21] proposes a subscription leasing scheme to achieve eventual stability. Author assumes the existence of a global clock, i.e., the availability of the internal or external clock synchronization providing a notion of global time accessible to all participants of the pub/sub network. Author assumes a synchronous system in that the message transmission time is always bounded and stays within $[\delta_{\min}, \delta_{\max}]$. It has been shown that such system model is impossible to satisfy [12], which implies the need for a weaker set of assumptions regarding the system model, e.g., such as those made in this paper. In [35] authors show how to provide a reliable (in-order, gapless) delivery in a content-based publish/subscribe system. In contrary to solutions presented in [35] we do not assume a redundant overlay network and propose an active countermeasure for the problem of subscription and advertisement propagation failures. Moreover, our approach can deal with complete broker failures and does not require a persistent storage on the nodes of the pub/sub system.

Another approach [34] to the soft state pub/sub system has been proposed in context of the Pastry DHT overlay. Authors assume that both subscriptions and advertisements are assigned timeouts which are evaluated with respect to the node's local clock. However, the presented approach does not account for the influence of the propagation delays on the timeout values which makes it impractical in typical distributed systems with varying communication latencies.

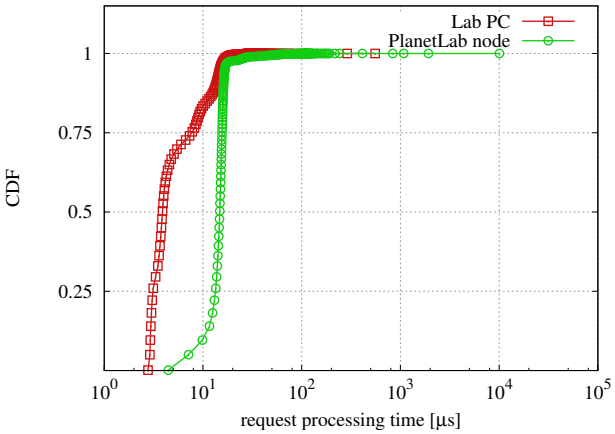


Figure 2: UDP-based ping message processing time

A related approach [26] proposes a soft state design in which publish/subscribe brokers periodically exchange heartbeat messages in order to maintain the state of the routing tables. The presented approach does not allow to cope with the issues presented in Figure 1 as the heartbeat messages and not data carrying messages (subscriptions and advertisements) are used to refresh the state.

In a more recent approach [23, 13] authors simulate a pub/sub system achieving eventual stability using subscription and advertisement leases. Authors do not use advertisement-based routing. Similarly to [21], a globally uniform subscription timeout value is assumed as well as an a priori known upper bound on the communication delays. Our approach does not make such assumptions. Instead, we propose per subscription and per advertisement timeout values which allow us to adaptively account for varying communication delays. Moreover, we do not assume any upper bounds on the communication delays in the system – as such assumption is not possible to satisfy in an asynchronous system [15]. Our approach, unlike the one presented in [23, 13] does not assume a known pub/sub network diameter. Due to the decoupled nature of pub/sub systems a network diameter value is never available to all participants of the system. Therefore, our algorithm does not make any assumptions on the pub/sub network diameter.

3. SYSTEM MODEL

In this section we summarize the system model in which we implement our soft state publish/subscribe system. Our system model is based on the Timed Asynchronous Distributed System Model [6]. It has been shown [14, 15] that the Timed Asynchronous Distributed System Model is suitably weak to be used for building of loosely coupled, distributed systems and simultaneously it is sufficiently strong to be able to reflect the real-world environment of the distributed systems. Specifically, we assume that all processes in our system are timed, i.e., there exists a time interval $\sigma_{\max}$ within which every process is supposed to respond to a request sent to it.

However, we assume that there are no guarantees that a request is indeed processed and answered within $\sigma_{\max}$. Figure 2 shows two cumulative distribution functions for one million processing times of a UDP ping request. The distribution functions are plotted for two hosts Ű- `Lab PC` is a host

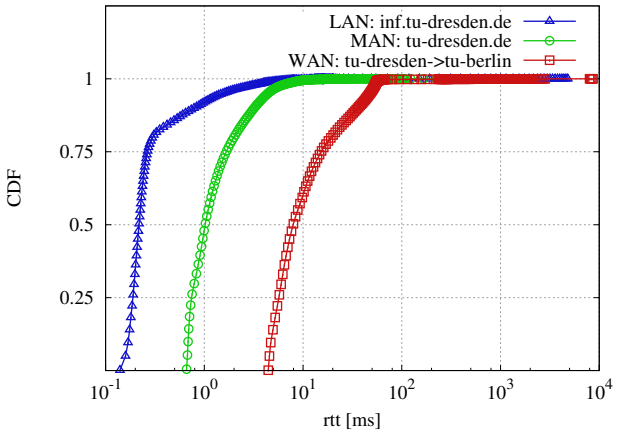


Figure 3: UDP-based ping message latencies (RTT)

in our laboratory and `PlanetLab node` is the `planet1.inf.tu-dresden.de` node of PlanetLab consortium. It can be observed that for both hosts the response times are characterized by a significant long tail and that it is not possible to set a reasonable upper bound for the processing time $\sigma_{\max}$.

We assume that processes may suffer performance failures. Specifically, a process might not respond within $\sigma_{\max}$ seconds because it is either slow or it has crashed. We assume processes do not suffer Byzantine failures. Such assumption can be fulfilled by converting Byzantine failures into crash failures using different software [32] and hardware [2] techniques.

We assume that processes communicate using unreliable transport protocol with omission and/or performance (non-Byzantine) failure semantics. Messages sent between processes may be arbitrarily delayed or might get dropped. Specifically we do not assume a priori any upper bound on the message transmission delay. Our experience shows (see Figure 3) that independently of the network environment it is not possible to determine a practical upper bound for message transmission delays.

Figure 3 shows the long-tail cumulative distribution functions of the round-trip times for one million UDP ping messages between two hosts in different network environments. The experiment has been conducted using three network setups: (1) the Local Area Network (LAN) Ű- between two computers in our laboratory, (2) the Metropolitan Area Network (MAN) -Ű between two computers at the Dresden University of Technology campus and (3) the Wide Area Network (WAN) -Ű between two computers at the Dresden and Berlin Universities of Technology. Based on the above measurements, we assume that there exists no upper bound on the frequency of communication and process failures.

Every process in our system has access to a local hardware clock. The term $H(t)$ denotes the value of the hardware clock at the real-time t. We assume that all hardware clocks have a drift rate bounded by the a priori known constant $\rho_{\max}$. It has been shown in [14] that for commercial, off-the-shelf components the value of $\rho_{\max}$ usually stays within $[1ppm, 100ppm]$. We say that a hardware clock is correct if:

$$H(t) - H(s) \leq (t-s)(1+\rho_{\max}) \quad (1)$$

and

$$H(t) - H(s) \geq (t-s)(1-\rho_{\max}) \quad (2)$$

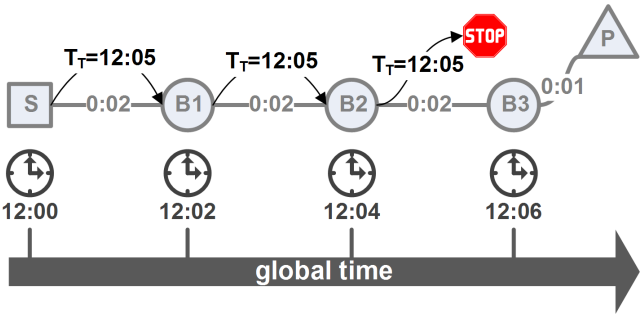


Figure 4: Propagation of a subscription using the validity time approach

Specifically, we assume that hardware clocks have crash-stop failure semantics. [10] shows how to achieve this property despite possible arbitrary value failures.

4. SOFT STATE

The main issue in designing a soft state pub/sub system is the determination of the validity T and the re-issue period τ of subscriptions and advertisements. The validity T of a subscription or advertisement can be expressed either as a *validity time* T_T or *validity interval* T_I and determines the life span of a given message. A validity time T_T describes a point in time when a given subscription or advertisement should expire. In order to be meaningful to all nodes the validity time is expressed using a common time reference – the synchronized clock.

A synchronized clock is shared by all nodes in the pub/sub system and ensures that every node has the same view of the time – called global time. Validity time describes the global time instance at which a given subscription or advertisement should expire. An exemplary T_T value could read 12:05, meaning that the subscription associated with it should expire on every node at 12:05 global time. Assuming all processes in the pub/sub system have their clocks synchronized within a given imprecision $\Delta_{\max}$[1], such subscription would indeed expire on all nodes within $[12{:}05 - \Delta_{\max},\ 12{:}05 + \Delta_{\max}]$. However, the practical application of the validity time approach is difficult due to two issues: (1) it is hard to provide a low imprecision, internal clock synchronization in large distributed systems and (2) message propagation delays significantly influence the validity time.

To better illustrate the latter case let us consider the scenario depicted on Figure 4. For the clarity of presentation we ignore the imprecision $\Delta_{\max}$ of the clock synchronization: $\Delta_{\max} = 0$. The subscriber **S** subscribes at 12:00 (global time) with a subscription which has the validity time T_T set to 12:05 (global time). The subscription propagation delay between subscriber **S** and the connecting broker **B1** equals 0:02, which implies that the subscription issued by the subscriber **S** will arrive at broker **B1** at 12:02, global time. Analogously, the next broker **B2** will receive the subscription at the 12:04, global time. The last broker **B3** cannot accept the subscription as upon its reception at the 12:06, global time, the subscription will be already expired as $12{:}05 < 12{:}06$. As a result events published by the publisher **P** will never

[1]we define the imprecision $\Delta_{\max}$ as the maximum difference between the synchronized clocks of all processes

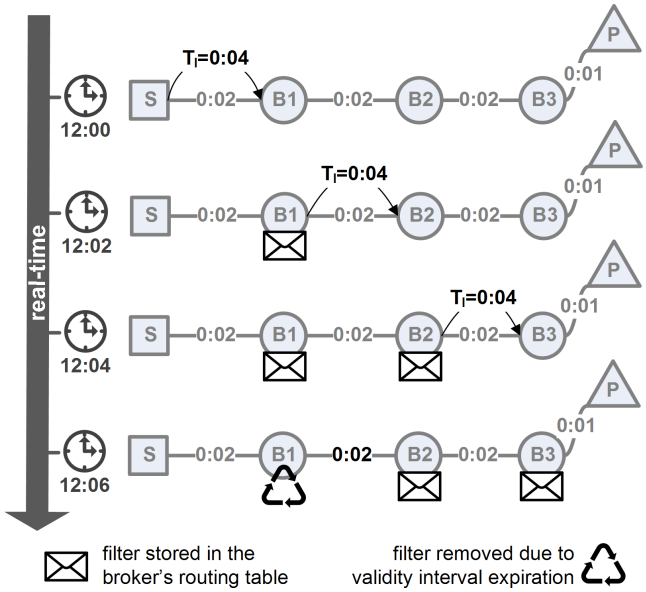


Figure 5: Propagation of a subscription using the validity interval approach

reach subscriber **S**, due to the missing subscription at the broker **B3**. From the above we can observe that the straightforward validity time approach is not applicable to large networks (with many brokers interconnecting publishers and subscribers) or to networks with varying transmission delays – see Figure 3.

Therefore, in this paper we use an approach based on the validity interval. The validity interval T_I does not rely on the synchronized clocks. Instead, every process in the pub/sub network calculates the validity interval attached to a subscription or an advertisement independently. The validity interval specifies the amount of time each broker should keep a subscription or an advertisement until it expires and is removed from its routing tables.

Figure 5 shows the influence of the subscription propagation delay on the routing tables. It can be observed that subscription issued by the subscriber **S** at real-time 12:00 has the validity interval T_I set to 0:04. At real time instance 12:02 it reaches the broker **B1** which installs it in its routing table and starts decrementing the validity interval of the subscription using its local hardware clock. Subsequently, it forwards the subscription to the broker **B2**, which repeats the procedure. At real-time instance 12:06 subscription issued by the subscriber **S** reaches the broker **B3**. However, at that time broker **B1** removes the subscription from its routing table since the validity interval T_I (equal to 0:04) has been decremented (since 12:02) to 0. This in turn results in publisher's **P** events never being delivered to the subscriber **S**.

It is therefore required to set the re-issue period τ of the subscription to a value which is lower than the specified validity interval T_I. However, even the setting of the re-issue period τ to low values does not guarantee that the refresh messages arrive before the expiry of the validity interval T_I. Varying link latencies (see Figure 3) and link overload may lead to significant delays of the refresh messages which in turn leads to the expiry of advertisements and subscriptions.

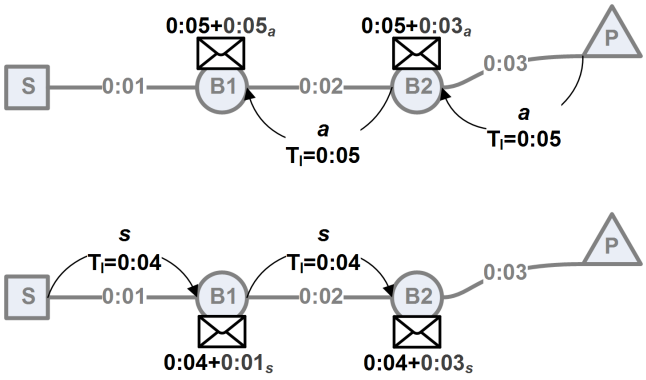


Figure 6: Extending validity interval using the propagation delay

4.1 Extending Validity Interval

In order to cope with the issues presented in Figure 5 we correlate the message propagation delays with the validity interval approach for both subscriptions and advertisements. Specifically, we estimate the advertisement and subscription propagation delays for every link traversed by those messages. The estimated delays are added to the validity interval T_I of each subscription and advertisement and are stored with subscriptions and advertisements at the processing brokers. As a result the validity interval of subscriptions and advertisements is *extended* by the amount equal to the expected link and processing delays. This in turn ensures that subscriptions and advertisements do not expire before the arrival of the re-subscription or re-advertisement. Moreover, such approach is transparent to both subscribers and publishers.

We define the propagation delay $pd(m)$ of the message m as a sum of the transmission delay $td(m)$ and processing time $pt(m)$ of the message m. The transmission delay is the actual time spent by the message in transit between two nodes. Processing time is the time spent by the message at the given node from its reception until its dispatching. Specifically, if the current node is the Nth node encountered by the message m, the propagation delay of message $pd_N(m)$ calculated by that node equals:

$$pd_N(m) = \sum_{i=1}^{i=N} td_i(m) + \sum_{i=0}^{i=N} pt_i(m) \qquad (3)$$

where $pt_0(m)$ is the time between the scheduled creation of the message m and its dispatching at the producing node, $pt_N(m)$ is the time between the reception of the message m at the destination node and the placement of the message m in the routing table of the destination node and $pt_i(m)$ for $i \in \{1, \ldots, N-1\}$ is the time between the reception of the message m and its dispatch at the node i. The transmission delay $td_i(m)$ is the transmission delay between nodes $i-1$ and i.

As an example let us consider the scenario presented in Figure 6. For the simplicity of the presentation we assume all processing times pt_i are equal to zero. The upper part of the Figure 6 shows the process of the propagation of the advertisement a issued by the publisher **P**. The advertisement has been assigned a validity interval T_I equal to 0:05. Upon the reception of the advertisement a broker **B2** calculates the propagation delay of the advertisement a. The propaga-

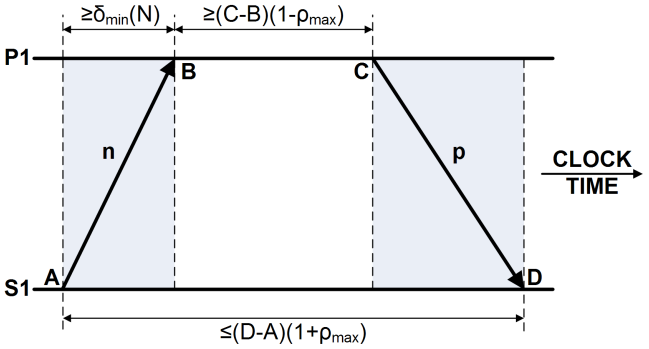


Figure 7: Upper bound on the propagation delay of the message p

tion delay $pd_{\mathbf{B2}}(a)$ is the sum of the processing time of the advertisement a (equal to zero) and the transmission delay between the publisher **P** and the broker **B2** equal to 0:03. The calculated propagation delay $0{:}03_a$ of the advertisement a is subsequently added to the publisher set validity interval $T_I = 0{:}05$ of the advertisement a and stored in the routing table of the broker **B2**. Broker **B2** will remove the advertisement a from its routing table only after the expiry of the updated validity interval $0{:}05 + 0{:}03_a$. The advertisement a is subsequently propagated by the broker **B2** towards the broker **B1**. Broker **B1** performs the analogous operations to those of the broker **B2**, the only difference being the value of the transmission delay of the advertisement a including now both transmission delays between the publisher **P** and the broker **B2** (0:03) and between the broker **B2** and the broker **B1** (0:02) plus the processing delays at the three nodes **P**, **B2** and **B1**. The propagation and calculation of the validity interval T_I for subscriptions is done in the analogous way – see lower part of the Figure 6.

The above algorithm allows subscriptions and advertisements to remain valid at every node despite the varying latencies on the communication links and despite the varying processing latencies. The algorithm does not guarantee that a subscription or advertisement will never expire before the arrival of the refresh message. However, such guarantee is impossible to satisfy with unbounded link and processing delays. Nonetheless, subsequent refresh messages will adapt to the changed link and processing characteristics so that the correct operation of the content-based pub/sub network will be eventually restored, providing the processing times and transmission delays do not grow infinitely.

4.2 Upper Bound

In order to allow the nodes of the pub/sub system to calculate the transmission delay of messages using only local hardware clock and not relying neither on the internal nor the external clock synchronization we use the upper bound on the message transmission delay technique [14].

In order to calculate the upper bound on the transmission delay of the message p sent from a publisher **P1** to the subscriber **S1** there must exists an earlier message n sent from the subscriber **S1** to publisher **P1** – see Figure 7.

The upper bound $\mathrm{ub}_{\mathbf{S1}}(p)$ on the transmission delay of the message p can be calculated by subscriber **S1** as the difference between the message processing times on both publisher $C-B$ and subscriber $D-A$. In order to calculate the upper bound subscriber has to make a pessimistic assumption regarding the drift rate of the hardware clocks of both publisher $(1-\rho_{\max})$ and its own $(1+\rho_{\max})$. The upper bound can be further refined by subtracting the minimum transmission delay of the message n equal to $\delta_{\min}(n)$, which results in:

$$\begin{aligned}\mathrm{ub}_{\mathbf{S1}}(p) &= (D-A)(1+\rho_{\max}) - \\ &\quad -(C-B)(1-\rho_{\max}) - \\ &\quad -\delta_{\min}(n) \end{aligned} \tag{4}$$

The calculation of the upper bound does not require synchronized clocks between communicating processes and the only requirement is the *a priori* known bound on the maximum clock drift rate $\rho_{\max}$. In order for the calculation to be performed locally by the subscriber **S1**, values of C, B and A are piggybacked on the publication p.

The Equation 4 implies that the calculated upper bound becomes less precise (tight) as the helper message n ages, i.e., its reception by the publisher **P1** lies further apart from the sending of the message p. In our scheme, however, periodic subscriptions and advertisements, as well as constant event flow present themselves as a valuable source of new helper messages, periodically improving the calculated upper bound.

The local processing time of a message n can be estimated by the publisher **P1** using its own local hardware clock, including the compensation for the hardware clock drift rate $(1+\rho_{\max})$:

$$\mathrm{ub}(pt_{\mathbf{P1}}(n)) = (1+\rho_{\max})pt_{\mathbf{P1}}(n) \tag{5}$$

Using Equations 4 and 5 we can rewrite the Equation 3 from Section 4.1 as:

$$pd_N(m) = \sum_{i=1}^{i=N} \mathrm{ub}_i\,(m) + \sum_{i=0}^{i=N} \mathrm{ub}\,(pt_i\,(m)) \tag{6}$$

For a more thorough explanation of the upper bound technique and a set of possible improvements we refer the reader to [11, 14].

4.3 Utilization and Uncertainty

The validity interval extension technique presented in Section 4.1 makes a pessimistic assumption regarding the propagation of refresh messages. The assumption being made is that a refresh message propagation time might span the range from zero to the calculated upper bound on the propagation delay. Such assumption while providing large margin of safety is certainly an overestimation. Therefore, we propose an alternative algorithm for the calculation of the validity interval extension.

In order to evaluate the new algorithm we first define a validity interval utilization metric. The validity interval utilization describes the amount of the extended validity interval $T_I(m)$ of the message m which has elapsed before the arrival of the refresh message:

$$U_{T_I} = \frac{T_I(m) - \mathrm{wait}(m)}{T_I(m)} \tag{7}$$

where the $\mathrm{wait}(m)$ is the amount of time message m has spent in the routing table before the arrival of the refresh message. The validity interval utilization U_{T_I} is expressed in percent. Intuitively, the higher the validity interval utilization, the less bandwidth is wasted for the unnecessary (too early) refresh messages. On the other hand, high validity utilization implies that the given filter or advertisement was more likely to expire

before the arrival of the refresh message – as the delay of the refresh message might not have been compensated by the small remaining validity utilization. Specifically validity interval utilization exceeding 100% indicates that the given message (subscription or advertisement) has expired before the arrival of the refresh message.

The validity interval extension presented in Section 4.1 lowers the values of the validity interval utilization for filters and advertisements by extending the validity interval with the upper bound on the respective message propagation delay. Moreover, the further the given message travels from the source (in terms of latency) the lower will the validity interval utilization be.

The above observation is the motivation for the introduction of the new algorithm for the extension of the validity interval. The new algorithm is based on the upper bound on the propagation delay, however its main goal is the estimation of the *uncertainty* of the upper bond for the given message. The uncertainty ub_Δ of the upper bound on the propagation delay for a given message m is defined as the difference between the minimum and maximum upper bound on the propagation delay values over time:

$$\mathrm{ub}_\Delta(m) = \frac{\mathrm{ub}_{\max}(\mathrm{pd}(m)) - \mathrm{ub}_{\min}(\mathrm{pd}(m))}{\Delta t} \qquad (8)$$

where Δt is the period of time (window) over which the uncertainty is calculated. Intuitively, the uncertainty serves as an estimation of the dispersion of the latency for the given path limited to a certain time window. Unlike the extension calculation algorithm presented in Section 4.1 for stable paths the value of the uncertainty-based extension will be low as the difference between the minimum and maximum upper bound on the propagation delay will remain small. This in turn will increase the values of the utilization for the given path.

The window Δt for which the uncertainty ub_Δ is calculated can be expressed either in terms of time units or in terms of messages. The choice of the window size and calculation method is left to the application programmer and can be set as one of the node's parameters. The calculation and management of the upper bound uncertainty is performed on a per link and per subscription/advertisement basis as different subscriptions/advertisements arriving on the same link might originate at different hosts and their upper bounds my differ significantly.

4.4 Coverage

In typical pub/sub systems often multiple subscribers are interested in identical or similar content. Analogously, often multiple advertisements describe similar sets of events. In content-based pub/sub systems these facts are used to limit the propagation of subscriptions and advertisements. Whenever an advertisement $a1$ about to be published on a given link describes a set of events $\mathbb{E}_1$ and there exists an advertisement $a2$ which has been previously published on that link and it describes a set of events $\mathbb{E}_2$ and the event set $\mathbb{E}_1$ is a subset of the event set $\mathbb{E}_2$ than the publishing of the advertisement $a1$ is obsolete. We can also say that advertisement $a2$ covers, i.e., is more general than advertisement $a1$ – denoted as $a1 \prec a2$. The above observation when applied to both advertisements and subscriptions allows to conserve both bandwidth and resources in pub/sub systems.

Figure 8 shows an example pub/sub network where three

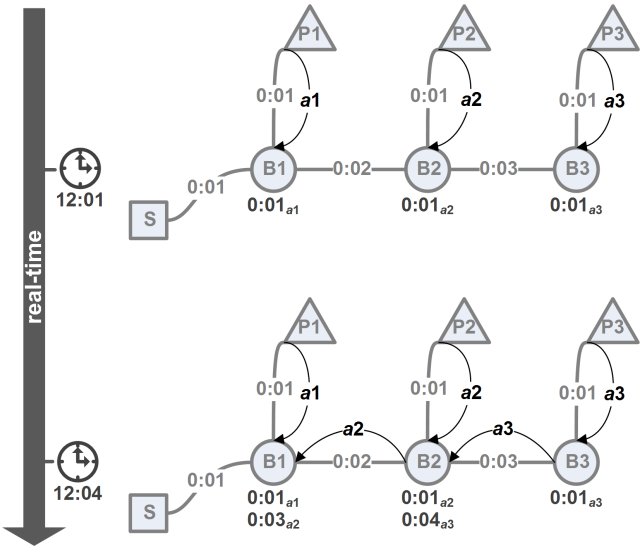


Figure 8: Multiple publishers publishing for the covering advertisements

publishers **P1**, **P2** and **P3** publish advertisements $a1$, $a2$ and $a3$ respectively. Advertisements are in the following coverage relation to each other: $a3 \prec a2 \prec a1$. In the upper part of the Figure 8 we can observe the pub/sub network shortly after all advertisements have been published. The bottom part shows the pub/sub network after the advertisements $a1$, $a2$ and $a3$ have been fully propagated throughout the network. We can observe that since advertisement $a3$ is covered by $a2$ ($a3 \prec a2$) it will not be propagated past the broker **B2**.

The validity extension for the advertisement $a3$ calculated at the broker **B2** contains both the propagation delays between publisher **P3** and broker **B3** and the brokers **B3** and **B2**.

The only issue which needs special consideration when content-based routing with coverage relation is used is the case of identical subscriptions and advertisements. If two subscriptions $s1$ and $s2$ are identical $s1 \equiv s2$ than subscription $s1$ covers $s2$ and vice versa [22]:

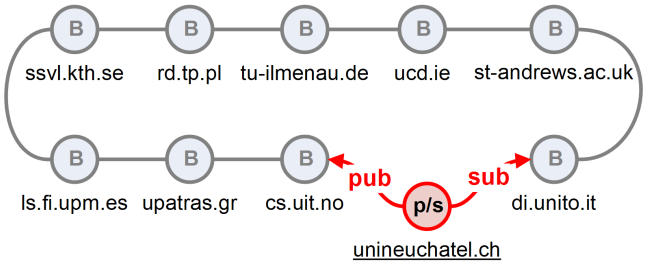
$$s1 \equiv s2 \quad \Rightarrow \quad s1 \prec s2 \ \wedge \ s2 \prec s1 \qquad (9)$$

The same applies to advertisements. This in turn implies that re-issuing the same subscription or advertisement in order to refresh the one previously sent into the pub/sub network would result in the dropping of such message by the first broker which encounters it. Therefore, instead of creating a special refresh message type, we choose to modify the coverage relation to not include identical subscriptions and advertisements:

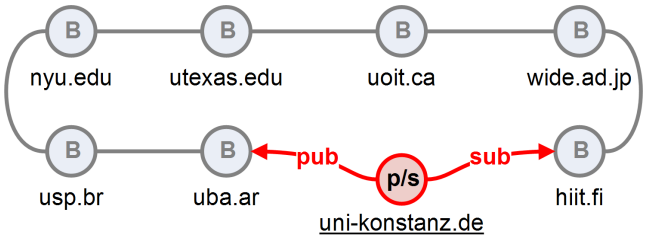
$$s1 \equiv s2 \quad \Rightarrow \quad s1 \not\prec s2 \ \wedge \ s2 \not\prec s1 \qquad (10)$$

5. EVALUATION

We have performed the evaluation of the implementation of the soft state pub/sub system in two basic scenarios – see Figure 9. The *ipeurope* scenario (Figure 9(a)) involves 10 PlanetLab hosts placed in Europe (Norway, Greece, Spain, Sweden, Poland, Germany, Ireland, Scotland, Italy and Switzerland), connected via TCP/IP. We use one host (`unineuchatel.ch`) to host both publisher and subscriber, so that we can com-



(a) *ipeurope* scenario – European PlanetLab hosts



(b) *ipworld* scenario – planetary PlanetLab hosts

Figure 9: Different network scenarios for the soft state pub/sub experiments

pare the send and receive times of the messages using the host's local hardware clock. This scenario can be seen as an example distribution path in a large scale, content-based pub/sub system. Similarly the *ipworld* scenario (Figure 9(b)) includes hosts distributed across multiple continents: Europe (Germany and Finland), South America (Argentina and Brazil), North America (USA and Canada) and Asia (Japan).

The first experiment (see Figure 10) illustrates the upper bound on the message propagation delay (`ub propagation delay`) which is equal (see Equation 6) to the sum of the upper bound on the message transmission delay (`ub transmission delay`) and the upper bound on the processing time (`ub processing time`) along with the real-time propagation delay (`rt propagation delay`) for events sent and received by the `unineuchatel.ch` node in the *ipeurope* scenario.

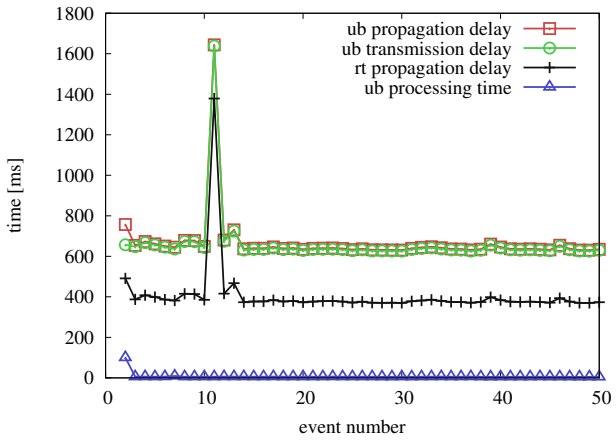
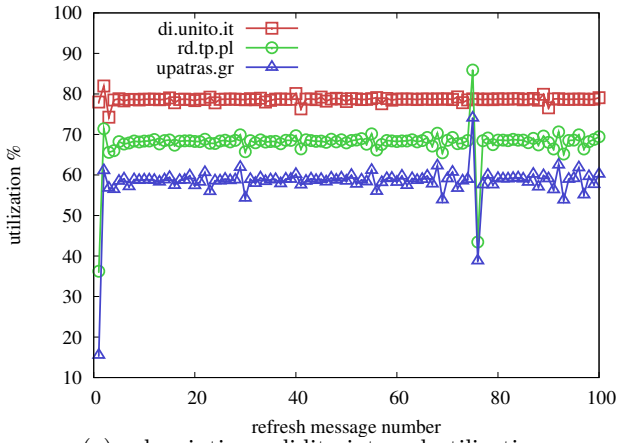
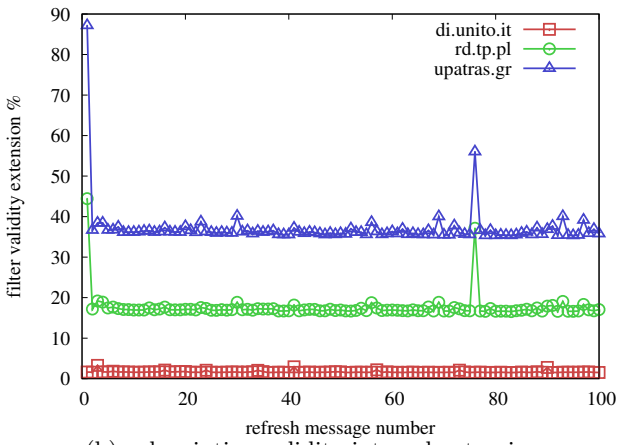


Figure 10: Upper bound on the propagation and transmission delays and processing time



(a) subscription validity interval utilization



(b) subscription validity interval extension

Figure 11: The subscription statistics for hosts in the *ipeurope* scenario

The real-time propagation delay was calculated using the local hardware clock of the `unineuchatel.ch` node by subtracting the send and receive time stamps of every event. We can observe that the upper bound holds in that it is never lower than the real-time propagation delay. We can also observe that the network latency dominates the overall propagation delay. The sudden spike for the event number 10 indicates that it has suffered an increased latency on one of the links between the nine nodes which are traversed by every event sent and received by the `unineuchatel.ch` node. Based on the above we can state that the upper bound technique is valid and can be used for the validity interval extension.

Figure 11 shows the validity interval utilization U_{T_I} (see Equation 7) and validity interval extension (see Section 4.1) for different hosts in the *ipeurope* scenario. The experiment data has been plotted for one subscription message. Intuitively, the higher the validity interval utilization (see Figure 11(a)) the less bandwidth is wasted for the unnecessary, i.e., arriving too soon, refresh messages. On the other hand, high validity utilization implies that the given message (subscription or advertisement) is more likely to expire before the arrival of the refresh message – as the delay of the refresh message will not be compensated by the remaining validity

utilization.

In the experiment presented in Figure 11 we have set the subscription validity interval T_I to 1 second and the refresh period τ to 0.8 seconds. In this experiment we extend the validity interval by adding to it the calculated upper bound on message propagation delay. The extension is calculated on the per message basis – the current message validity interval is extended using the propagation delay of the previous message. We can observe that according to expectations the utilization of the subscription at the first broker encountered by the subscription (`di.unito.it`) is close to the 80% as the upper bound on the subscription propagation delay is much lower than the actual validity interval T_I. The utilization of the same subscription at one of the last brokers on its path (`upatras.gr`) is around 60%. This can be explained by the fact that upper bound on the processing delay which is added to the subscription's validity interval is much larger (due to the larger number of brokers traversed by the subscription) and therefore the upper bound overestimation also grows. On the other hand, growing overestimation for the brokers further from the message source is compensating for a higher probability of a sudden increase in the message propagation delay, which is proportional to the number of hops a message has to traverse.

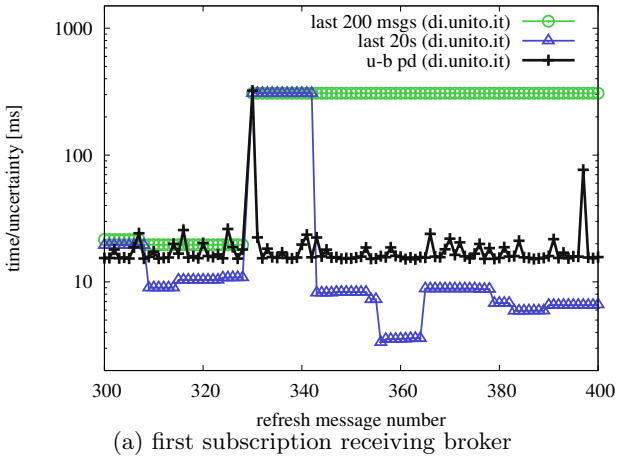
Figure 11(b) shows how for the same experiment (as in Figure 11(a)) different brokers extend the validity interval of a single subscription. We can observe that the first broker (`di.unito.it`) encountered by the subscription barely extends its validity, while one of the last ones (`upatras.gr`) extends it by almost 40%. If we remember that the validity interval of the subscription was set to 1 second, we can conclude that the average upper bound on the propagation delay of that subscription when it arrived at `upatras.gr` node was equal to 0.4 seconds.

On Figure 11(b) we can also observe that the refresh message number 75 was significantly slowed down (upward spike) between the `upatras.gr` and `rd.tp.pl` nodes. This translates to the utilization of the subscription refreshed by that message growing proportionally to the refresh message delay – see Figure 11(a). Subsequent refresh message arrived timely and therefore the utilization calculated for it significantly dropped – see downward spike in Figure 11(a). Since the utilization for message 75 remained under 100% no unsubscription due to the timeout of the validity interval has taken place.

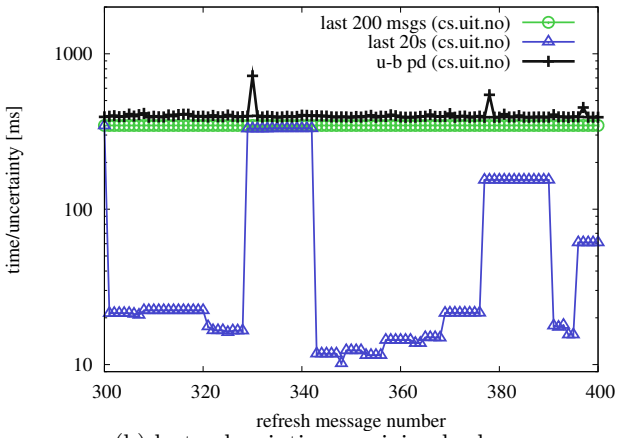
Figure 12 illustrates the uncertainty-based approach presented in Section 4.3. It compares the upper bound (u-b pd) and the calculated uncertainty ub_Δ using time (`last 20s`) and counting (`last 200 msgs`) based sliding windows. The comparison has been performed for subscription messages in the *ipeurope* scenario. We have plotted the results obtained from the first (Figure 12(a)) and the last (Figure 12(a)) broker encountered by subscriptions issued by the `unineuchatel.ch` node in the *ipeurope* scenario.

It can be observed that unlike in case of the Figure 11 the uncertainty based approach calculated extensions do not expose a bias based on the location of the node within the publish/subscribe network. The last subscription receiving broker (Figure 12(b)) exposes a few more spikes in the calculated extension, however the average values are comparable with the first subscription receiving broker.

Figure 13 compares the new, uncertainty-based approach ub_Δ using the sliding window of 20 seconds with the upper



(a) first subscription receiving broker



(b) last subscription receiving broker

Figure 12: Validity interval extension using uncertainty-based extension

bound-based approach (cf. Figure 11) for the last broker encountered by subscription messages. For subscriptions we have chosen the validity interval T_I to equal 1 second and the reissue period τ to equal 0.8s. We can observe that relatively high upper bound values (0.4 seconds) in combination with the original approach have contributed to the drop of the utilization from the application programmer specified 80% to about 60%. On the other hand, the sliding window based upper bound uncertainty estimation method retains the designer specified utilization, despite upper bound values reaching 40% of the originally specified validity.

Figure 14 illustrates the comparison between the uncertainty-based interval extension calculated using the upper bound on the propagation delay (`ub-based extension`) and the interval extension based on the real-time propagation delay (`rt-based extension`) for the `unineuchatel.ch` node in the *ipeurope* scenario. We can observe that both calculated values are virtually identical, which confirms the upper bound as the right technique for use in our soft state publish/subscribe system. For comparison we have also included the actual values of the upper bound on (`ub pd`) and the real-time (`rt pd`) subscription propagation delays. Both left and right y axes have logarithmic scale.

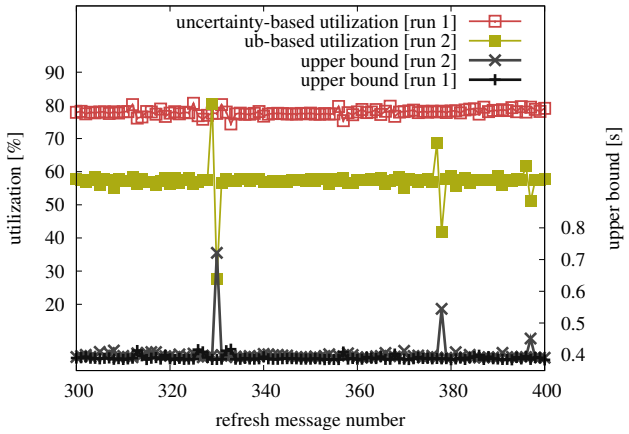


Figure 13: Comparison of subscription utilization using the upper bound (`ub-based`) and sliding window (`uncertainty-based`) approach for the `cd.uit.no` host in the *ipeurope* scenario

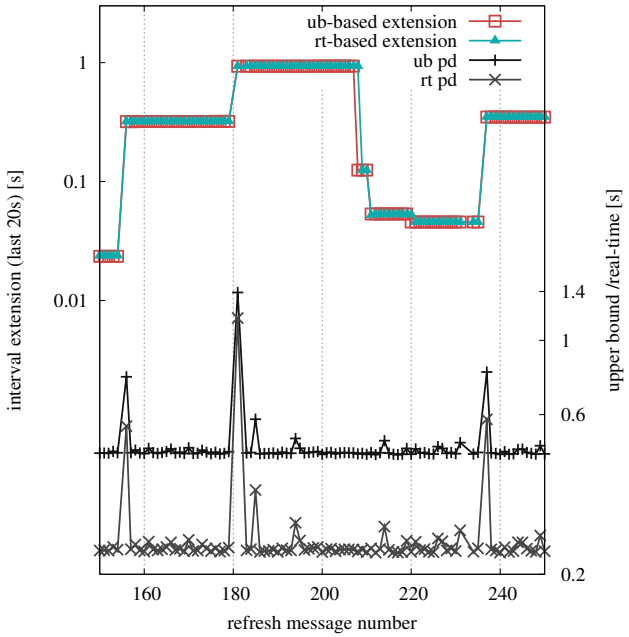
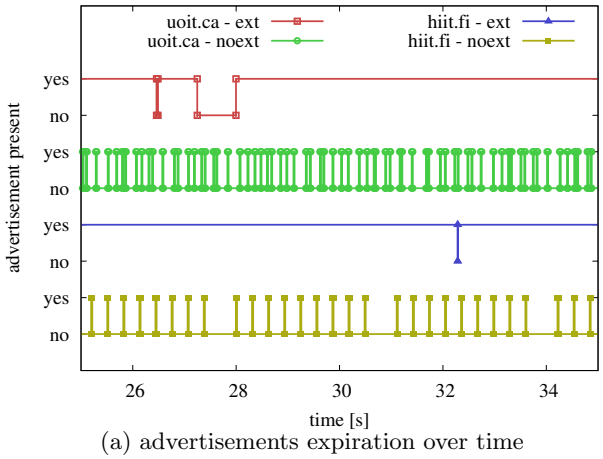
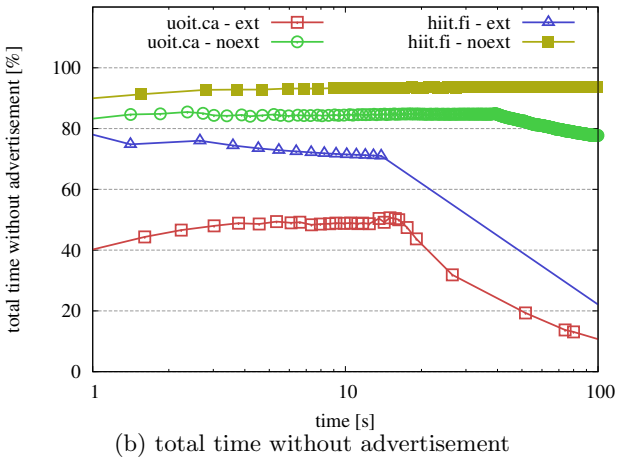


Figure 14: Real-time and upper bound based interval extension



(a) advertisements expiration over time



(b) total time without advertisement

Figure 15: Soft state pub/sub against hard state pub/sub

The last experiment compares the efficiency of the validity extension technique with the naive approach where no extension is used. For that experiment the *ipworld* setup has been used. During the experiment all brokers were connected using the TCP protocol. The measurement has been performed for the advertisement messages. Advertisement messages have been assigned the validity interval T_I of 20 milliseconds and the refresh period τ has been set to 16 milliseconds, which results in the expected utilization of 80%.

Figure 15(a) illustrates the presence of advertisements in the two brokers of the *ipworld* setup. It can be observed that when no validity interval extension (`noext`) is used the advertisement presence in the brokers is very brief, with the validity interval expiration quickly causing an unadvertisement. In contrast, the validity interval extension technique (`ext`) assures that the expiration is postponed until the refresh message is able to arrive. Figure 15(a) illustrates also the fact that the further away from the source of the advertisement a given broker is, the lower the amount of time an advertisement is present at that broker when no validity interval extension is used.

Figure 15(b) plots the data from Figure 15(a) across a longer period of time as a percentage of time a given broker

has spent without an advertisement. Time without advertisement implies the amount of time when broker is not able to forward filters towards matching publishers and thus it is not able to deliver events to the interested subscribers. It can be observed that without the extension technique the total broker time without valid advertisement remains very high, while the validity interval extension technique quickly helps to extend the advertisement intervals so as to accommodate for the large and varying latencies experienced by the refresh messages.

6. SUMMARY

The soft state publish/subscribe system proposed in this paper offers a set of interesting properties and trade-offs. On one hand it automatically recycles the state of crashed subscribers (subscriptions) and publishers (advertisements) across the whole content-based pub/sub system using communication by time. It frees the pub/sub application developers from the concerns regarding the issues related to timing (ordering) of messages and it upholds the decoupled nature of the pub/sub systems. Moreover, the processing overhead of unsubscriptions and unadvertisements is significantly reduced as it is always local to a single broker and never triggers any additional messages. On the other hand, the periodic re-subscriptions and re-advertisements consume more bandwidth than in case of the traditional hard state pub/sub systems. However, one has to consider that the hard state pub/sub systems might also impose high message overhead if they persistently try to remove orphaned state from unreachable or malfunctioning brokers.

The soft state pub/sub system is lightweight in that no costly protocol for time synchronization is required. This is especially important if we consider scenarios when time synchronization is not possible, e.g., due to restrictive firewall policies or due to the decoupling properties of pub/sub systems which do not provide users with the "whole system view". Therefore it is especially important to mention that our implementation upholds all decoupling properties of the pub/sub systems.

We do not claim that the soft state pub/sub system is a one fit for all solution. However, we believe that it is an interesting alternative for hard state systems in large-scale, high latency setups. Moreover, it significantly simplifies the deployment of the dynamic pub/sub networks – an experience we have gained and confirmed multiple times when running our test suites on PlanetLab hosts.

7. ACKNOWLEDGMENTS

Authors would like to thank Robert Fach for inspiring discussions regarding the soft state approach in the context of the XSiena publish/subscribe system. Zbigniew Jerzak's work has been partially supported by the Polish Ministry of Science and Higher Education grant number N N516 375034.

8. REFERENCES

[1] Algirdas Avizienis, Jean-Claude Laprie, Brian Randell, and Carl E. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33, Jan.–Mar. 2004.

[2] David Bernick, Bill Bruckert, Paul Del Vigna, David Garcia, Robert Jardine, Jim Klecka, and Jim Smullen. Nonstop advanced architecture. In *DSN '05: International Conference on Dependable Systems and Networks*, pages 12–21, Yokohama, Japan, June 2005. IEEE Computer Society.

[3] Sumeer Bhola, Robert E. Strom, Saurabh Bagchi, Yuanyuan Zhao, and Joshua S. Auerbach. Exactly-once delivery in a content-based publish-subscribe system. In *DSN 2002: International Conference on Dependable Systems and Networks*, pages 7–16, Bethesda, MD, USA, June 2002. IEEE Computer Society.

[4] Antonio Carzaniga, David S. Rosenblum, and Alexander L. Wolf. Design and evaluation of a wide-area event notification service. *ACM Trans. Comput. Syst.*, 19(3):332–383, 2001.

[5] D. Clark. The design philosophy of the DARPA internet protocols. In *SIGCOMM '88: Symposium proceedings on Communications architectures and protocols*, pages 106–114, New York, NY, USA, 1988. ACM.

[6] Flaviu Cristian and Christof Fetzer. The timed asynchronous distributed system model. *IEEE Transactions on Parallel and Distributed Systems*, 10(6):642–657, June 1999.

[7] Patrick Eugster. Type-based publish/subscribe: Concepts and experiences. *ACM Transactions on Programming Languages and Systems*, 29(1):1–50, January 2007.

[8] Patrick Th. Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The many faces of publish/subscribe. *ACM Comput. Surv.*, 35(2):114–131, 2003.

[9] Francoise Fabret, H. Arno Jacobsen, Francois Llirbat, Joao Pereira, Kenneth A. Ross, and Dennis Shasha. Filtering algorithms and implementation for very fast publish/subscribe systems. In *SIGMOD '01: Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, pages 115–126, New York, NY, USA, 2001. ACM Press.

[10] Christof Fetzer and Flaviu Cristian. Building fault-tolerant hardware clocks from COTS components. In *Proceedings of the Seventh IFIP International Working Conference on Dependable Computing for Critical Applications*, pages 67–86, San Jose, CA, USA, Nov 1999.

[11] Christof Fetzer and Flaviu Cristian. A fail-aware datagram service. In Iain Bate and Alan Burns, editors, *IEE Proceedings - Software Engineering*, volume 146, pages 58–74. IEE, April 1999.

[12] Christof Fetzer and Flaviu Cristian. Fail-awareness: An approach to construct fail-safe systems. *Journal of Real-Time Systems*, 24(2):203–238, March 2003.

[13] Michael A. Jaeger. *Self-Managing Publish/Subscribe Systems*. PhD thesis, Technische Universität Berlin, 2007.

[14] Zbigniew Jerzak, Robert Fach, and Christof Fetzer. Fail-aware publish/subscribe. In *Sixth IEEE International Symposium on Network Computing and Applications (NCA 2007)*, pages 113–125, Cambridge, MA, USA, July 2007. IEEE Computer Society.

[15] Zbigniew Jerzak, Robert Fach, and Christof Fetzer. Adaptive internal clock synchronization. In *SRDS 2008: 27th International Symposium on Reliable*

Distributed Systems, pages 217–226, Naples, Italy, October 2008. IEEE Computer Society.

[16] Zbigniew Jerzak and Christof Fetzer. BFSiena: a communication substrate for StreamMine. In *DEBS '08: Proceedings of the second international conference on Distributed event-based systems*, pages 321–324, Rome, Italy, July 2008. ACM.

[17] Guoli Li, Shuang Hou, and Hans-Arno Jacobsen. A unified approach to routing, covering and merging in publish/subscribe systems based on modified binary decision diagrams. In *ICDCS '05: Proceedings of the 25th IEEE International Conference on Distributed Computing Systems*, pages 447–457, Washington, DC, USA, 2005. IEEE Computer Society.

[18] Guoli Li and Hans-Arno Jacobsen. Composite subscriptions in content-based publish/subscribe systems. In *Middleware*, volume 3790/2005 of *Lecture Notes in Computer Science*, 2005.

[19] Guoli Li, Vinod Muthusamy, , and Hans-Arno Jacobsen. Adaptive content-based routing in general overlay topologies. In Valérie Issarny and Richard E. Schantz, editors, *Middleware '08: Proceedings of the ACM/IFIP/USENIX 9th International Middleware Conference*, volume 5346 of *Lecture Notes in Computer Science*, pages 1–21, Leuven, Belgium, December 2008. Springer.

[20] Alan Mislove, Ansley Post, Andreas Haeberlen, and Peter Druschely. Experiences in building and operating a reliable peer-to-peer application. In Yolande Berbers and Willy Zwaenepoel, editors, *EuroSys*, pages 147–159, Leuven, Belgium, April 2006. ACM.

[21] Gero Mühl. *Large-Scale Content-Based Publish/Subscribe Systems*. PhD thesis, Technisches Universität Darmstadt, 2002.

[22] Gero Mühl, Ludger Fiege, and Peter Pietzuch. *Distributed Event-Based Systems*. Springer-Verlag New York, Inc., 2006.

[23] Gero Mühl, Michael A. Jaeger, Klaus Herrmann, Torben Weis, Andreas Ulbrich, and Ludger Fiege. Self-stabilizing publish/subscribe systems: Algorithms and evaluation. In *Euro-Par 2005 Parallel Processing*, volume 3648/2005, pages 664–674. Springer Berlin / Heidelberg, 2005.

[24] Brian M. Oki, Manfred Pflügl, Alex Siegel, and Dale Skeen. The information bus – an architecture for extensible distributed systems. In B. Liskov, editor, *Proceedings of the 14th Symposium on the Operating Systems Principles*, pages 58–68. ACM Press, December 1993.

[25] Peter Pietzuch, David Eyers, Samuel Kounev, and Brian Shand. Towards a common api for publish/subscribe. In Hans-Arno Jacobsen, Gero Mühl, and Michael A. Jaeger, editors, *DEBS '07: Proceedings of the 2007 inaugural international conference on Distributed event-based systems*, volume 233 of *ACM International Conference Proceeding Series*, pages 152–157, Ontario, Canada, June 2007. ACM.

[26] Peter R. Pietzuch. *Hermes: A Scalable Event-Based Middleware*. PhD thesis, Computer Laboratory, Queens' College, University of Cambridge, February 2004.

[27] Suchitra Raman and Steven McCanne. A model, analysis, and protocol framework for soft state-based communication. In *SIGCOMM '99: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, pages 15–25, New York, NY, USA, 1999. ACM.

[28] Timothy Roscoe. The planetlab platform. In Ralf Steinmetz and Klaus Wehrle, editors, *Peer-to-Peer Systems and Applications*, volume 3485 of *Lecture Notes in Computer Science*, pages 567–581. Springer, 2005.

[29] David S. Rosenblum and Alexander L. Wolf. A design framework for internet-scale event observation and notification. In Mehdi Jazayeri and Helmut Schauer, editors, *6th European Software Engineering Conference Held Jointly with the 5th ACM SIGSOFT Symposium on Foundations of Software Engineering*, volume 1301 of *Lecture Notes in Computer Science*, pages 344–360, Zurich, Switzerland, September 1997. ACM.

[30] Zhenhui Shen and S. Tirthapura. Self-stabilizing routing in publish-subscribe systems. In *International Workshop on Distributed Event-Based Systems*, 2004.

[31] Sasu Tarkoma. Chained forests for fast subsumption matching. In Hans-Arno Jacobsen, Gero Mühl, and Michael A. Jaeger, editors, *DEBS '07: Proceedings of the 2007 inaugural international conference on Distributed event-based systems*, volume 233 of *ACM International Conference Proceeding Series*, pages 97–102, Toronto, Ontario, Canada, June 2007. ACM.

[32] Ute Wappler and Christof Fetzer. Software encoded processing: Building dependable systems with commodity hardware. In Francesca Saglietti and Norbert Oster, editors, *SAFECOMP '07: 26th International Conference on Computer Safety, Reliability, and Security*, volume 4680 of *Lecture Notes in Computer Science*, pages 356–369, Nuremberg, Germany, September 2007. Springer.

[33] Timo Warns, Christian Storm, and Wilhelm Hasselbring. Availability of globally distributed nodes: An empirical evaluation. In *SRDS '08: IEEE Symposium on Reliable Distributed Systems*, pages 279–284, Naples, Italy, October 2008.

[34] Tao Xue, Boqin Feng, and Zhigang Zhang. P2pens: Content-based publish-subscribe over peer-to-peer network. In Hai Jin, Yi Pan, Nong Xiao, and Jianhua Sun, editors, *GCC 2004: Third International Conference on Grid and Cooperative Computing*, volume 3251 of *Lecture Notes in Computer Science*, pages 583–590, Wuhan, China, October 2004. Springer.

[35] Yuanyuan Zhao, Daniel Sturman, and Sumeer Bhola. Subscription propagation in highly-available publish/subscribe middleware. *Lecture Notes in Computer Science*, 3231:274 – 293, 2004.