Stochastic Performance Analysis and Capacity Planning of Publish/Subscribe Systems

Arnd Schröter[1], Gero Mühl[2], Samuel Kounev[3], Helge Parzyjegla[1], Jan Richling[1]

[1] Communication and Operating Systems Group, Berlin University of Technology, Germany
[2] Architecture of Application Systems, University of Rostock, Germany
[3] Institute for Program Structures and Data Organization, Karlsruhe Institute of Technology, Germany
{arnd.schroeter,g_muehl,skounev,parzyjegla,richling}@acm.org

ABSTRACT

Publish/subscribe systems are used increasingly often as a communication mechanism in loosely-coupled distributed applications. With their gradual adoption in mission critical areas, it is essential that systems are subjected to a rigorous performance analysis before they are put into production. However, existing approaches to performance modeling and analysis of publish/subscribe systems suffer from many limitations that seriously constrain their practical applicability. In this paper, we present a set of generalized and comprehensive analytical models of publish/subscribe systems employing different peer-to-peer and hierarchical routing schemes. The proposed analytical models address the major limitations underlying existing work in this area and are the first to consider all major performance-relevant system metrics including the expected broker and link utilization, the expected notification delay, the expected time required for new subscriptions to become fully active, as well as the expected routing table sizes and message rates. To illustrate our approach and demonstrate its effectiveness and practicality, we present a case study showing how our models can be exploited for capacity planning and performance prediction in a realistic scenario.

Keywords: Event-based Systems, Publish/Subscribe, Performance Modeling and Prediction

1. INTRODUCTION

Publish/subscribe systems were originally motivated by the need for loosely-coupled and asynchronous dissemination of information in distributed event-based applications [19]. With the advent of ambient intelligence and ubiquitous computing, many new applications of publish/subscribe systems have emerged [12], for example, in the areas of transport information monitoring [2, 23], event-driven supply chain management [1, 21], location-based services [6, 9] and ubiquitous sensor-rich environments [16, 20]. The main advantage of publish/subscribe is that it makes it possible to decouple communicating parties in time, space, and flow [10].

The clients of a publish/subscribe system can take over the roles of publishers or subscribers depending on whether they act as producers or consumers of information. *Publishers* publish information in the form of *notifications*, while *subscribers* express their interest in specific notifications by issuing subscriptions. Subscriptions are normally defined as a set of constraints on the type and content of notifications which are often referred to as *filters*. A *notification service*, interposed between the clients, delivers published notifications to all subscribers that have issued matching subscriptions. In many cases, the notification service is implemented by a set of *brokers* each managing a set of *local clients*. The brokers are connected by *overlay links* and a published notification is routed stepwise from the broker hosting the publisher over intermediate brokers to all brokers that host subscribers with matching subscriptions. To achieve this, each broker manages a *routing table* that is used to forward incoming notifications to neighbor brokers and local clients. The routing tables are updated according to a *routing algorithm* by propagating newly issued or canceled subscriptions in the broker network. This information is included in *control messages* exchanged by the brokers.

With the increasing popularity of publish/subscribe systems and their gradual adoption in mission critical areas [12], performance and scalability issues are becoming a major concern. System developers and deployers are often faced with questions such as: What performance would the system exhibit for a given deployment topology, configuration and workload scenario? What would be the expected notification and subscription delays as well as the utilization of the various system components (brokers, network links, etc)? What maximum load (number of publishers and subscribers, notification publication rates) would the system be able to handle? What would be the optimal number of brokers and the optimal system topology? Which components would be most utilized as the load increases and are they bottlenecks? To answer such questions, techniques for predicting the system performance as a function of its configuration and workload are needed.

Unfortunately, most existing approaches to performance analysis of publish/subscribe are focused on specific system configurations that are evaluated by means of time- and resource-intensive simulations. Such evaluations are expensive especially when large-scale systems with multiple

DEBS'10, July 12-15, 2010, Cambridge, UK.

alternative configurations and workloads have to be considered. While a few analytical models of publish/subscribe systems have been proposed (e.g., [4, 8, 17, 11]), they impose a number of restrictive assumptions which limit their practical applicability. Furthermore, most approaches typically do not consider important performance-relevant system metrics such as the end-to-end notification and subscription delays, but instead are focused on lower level metrics such as the routing table sizes of brokers and their message throughput.

In this paper, we propose a set of generalized analytical models of publish/subscribe systems that employ peer-to-peer or hierarchical routing schemes. In contrast to existing work, the proposed models cover a variety of different routing algorithms and are thus applicable to a large class of systems. The models capture the performance-relevant aspects of system behavior and can be used to predict the system performance for a given workload and configuration scenario. This allows to evaluate trade-offs across multiple scenarios with minimal overhead. The analytical models proposed in this paper have several novel aspects including:

1. All major performance-relevant system metrics are considered: the routing table sizes, the message rates, the expected notification delay, the expected time required for new subscriptions to become fully active, the expected broker utilization and the expected utilization of overlay links and physical links.
2. A variety of different routing algorithms are supported including simple and identity-based routing, as well as advanced algorithms, such as covering- and merging-based routing [16].
3. All routing algorithms are supported in their peer-to-peer variant in addition to their hierarchical variant.
4. For each broker and physical link, arbitrary service time distributions of notifications and control messages are supported. Furthermore, the physical network and its effect on the traffic generated by the overlay network is modeled explicitly.

To evaluate our approach in a realistic scenario, we apply it to a case study of a representative publish/subscribe system. The case study shows how the proposed modeling approach allows to analyze the impact of design decisions on the system performance without the need for expensive and time-consuming simulations. For example, we show how the models can be used to determine the optimal number of brokers and their position for a given workload and configuration scenario. Furthermore, we demonstrate how important insights into the dynamic system behavior can be gained.

The proposed analytical models significantly extend existing work on modeling of publish/subscribe systems providing a powerful tool for performance prediction and capacity planning. At system design time, the models can be exploited for comparing alternative system designs with different communication and messaging patterns. At system deployment time, models help to detect system bottlenecks and to ensure that sufficient resources are allocated to meet performance and Quality of Service requirements. Finally, during operation, the models can be exploited to dynamically reconfigure the system adapting its resource allocations to reflect changes in the workload intensity.

The remainder of the paper is organized as follows: In Sect. 2, we describe how routing is done in the type of systems we consider. Section 3 describes our analytical models in detail and Sect. 4 presents the case study. Section 5 discusses several ways in which the proposed models can be extended. In Sect. 6, related work is reviewed and the paper is wrapped up in Sect. 7.

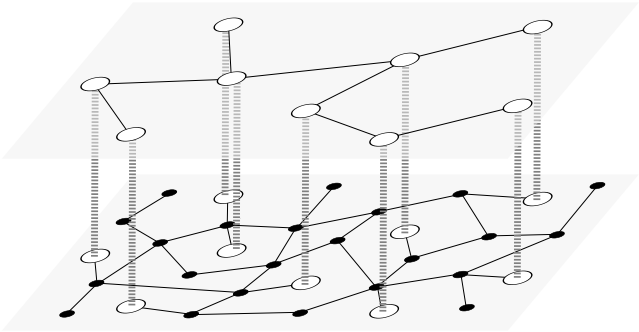

Figure 1: Overlay and physical network.

2. PUBLISH/SUBSCRIBE ROUTING

Routing in publish/subscribe systems takes place on two levels as depicted in Figure 1. On the upper level, messages traverse the broker overlay topology based on their content and the currently active subscriptions. On the lower level, messages are routed through the physical network between neighboring brokers in the overlay topology. The *physical network* consists of physical routers and links connecting brokers in the overlay topology. The *overlay network* is a logical network whose nodes are the brokers and whose links are paths in the physical network. When a broker sends a message to another broker over an overlay link, the message is injected into the physical network and forwarded stepwise over physical links and routers to the receiving broker. Therefore, overlay links sharing physical links or routers also share their physical bandwidth.

The routing of notifications and subscriptions in the broker overlay network is done according to a publish/subscribe *routing algorithm*. Normally, the overlay topology is required to be acyclic in order to enable the use of advanced routing algorithms introduced later. The most basic routing algorithm is *notification flooding*, where notifications are forwarded along all paths in the system reaching all brokers, whereas subscriptions are never propagated beyond the subscriber hosting brokers. This is in contrast to *filtering-based* routing algorithms which propagate subscriptions into the broker overlay network and forward notifications selectively utilizing the information about the currently active subscriptions. Filtering-based routing algorithms are available in two main variants: based on the hierarchical and the peer-to-peer routing scheme, respectively [7]. A number of specific routing algorithms are available for both schemes [16]. While the presentation focuses on peer-to-peer routing, the proposed models are applicable to both routing schemes and we consider both of them in the presented case study.

2.1 Peer-to-Peer Routing Scheme

With the *peer-to-peer routing scheme*, brokers are arranged in an acyclic overlay topology. Please note that "peer-to-peer" in the context of publish/subscribe systems [7] usually only means that neighboring brokers are peers in the sense that they behave symmetrically, which is in contrast to hierarchical routing, where neighboring brokers behave asym-

metrically. Whether a peer-to-peer substrate (e.g., Pastry, or Chord) is used for communication is an orthogonal issue, but incorporating such a substrate into the models of this paper would clearly be interesting.

With peer-to-peer routing, subscriptions issued by clients are propagated into the broker overlay network, and while being propagated, they establish routing entries which are stored in the broker routing tables to form the reverse delivery paths of notifications from the producers to the subscribers. The propagation of a subscription may be suspended if a point is reached at which the delivery of all notifications matching the subscription is already guaranteed by another active subscription that has been propagated previously. If a subscription is revoked by a client, information about this revocation is propagated in the same way as the information about new subscriptions.

Notifications published by clients are forwarded in the broker network using the reverse delivery paths set up by the subscription propagation process: a notification is forwarded by a broker to a neighbor broker only if there is a matching routing entry for this neighbor, representing a client with a matching subscription in the respective subtopology.

2.2 Hierarchical Routing Scheme

With the *hierarchical routing scheme*, brokers are arranged in a tree-based overlay topology, where one of the brokers is designated as *root broker* and broker relations are asymmetrical: for two adjacent brokers, the broker nearer to the root broker is called the *parent broker* of the other broker and brokers having the same parent broker are called *child brokers* of the respective broker. These asymmetrical relations affect subscription and notification forwarding.

With hierarchical routing, subscriptions are propagated only upwards in the broker tree from the broker hosting the subscriber towards the root broker. As in the case of peer-to-peer routing, this is done to form the reverse delivery paths for matching notifications. To compensate for the fact that subscriptions are only propagated upwards, notifications are always propagated all the way up to the root broker. In addition, notifications are propagated downwards towards subscribers whenever they meet a matching routing entry on their way to the root broker. Thus, notifications are filtered only at one end of an overlay link in contrast to peer-to-peer routing where they are filtered at both ends.

2.3 Considered Routing Algorithms

We consider several routing algorithms which are each available for both routing schemes. The algorithms differ in terms of whether and if yes in which cases the propagation of a subscription is suspended. At this point the delivery of all notifications matching the subscription is already guaranteed by other active subscriptions. In the case of hierarchical routing, the propagation of a subscription towards the root broker is suspended in the above case, while for peer-to-peer routing, propagation is suspended for individual neighbor brokers only.

With *simple routing*, propagation is never suspended resulting in each subscription always being propagated all the way up to the root broker, in the case of hierarchical routing, and being flooded into the broker overlay network, in the case of peer-to-peer routing. With *identity-based routing*, the propagation of a subscription is suspended if a broker is reached that has already propagated a subscription matching the *same* set of notifications to its parent broker in the case of hierarchical routing or to the considered neighbor broker in the case of peer-to-peer routing. Similarly, with *covering-based routing*, the propagation of a subscription is suspended if a broker is reached that has already propagated a subscription matching the *same* set of notifications or a superset of them. Finally, *merging-based routing* works like covering-based routing but additionally supports the merging of several subscriptions into a single subscription replacing the original subscriptions that is propagated towards the root broker in the case of hierarchical routing or to the considered neighbor broker in the case of peer-to-peer routing.

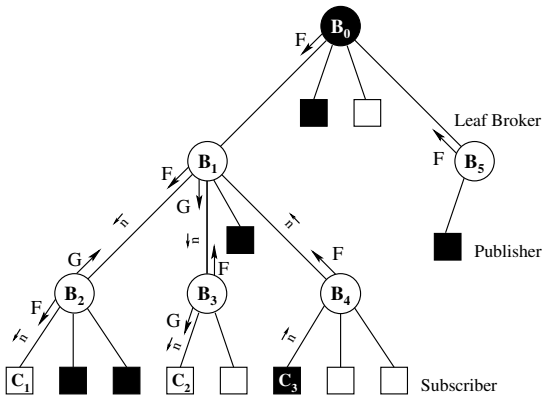

Figure 2: Peer-to-peer covering-based routing.

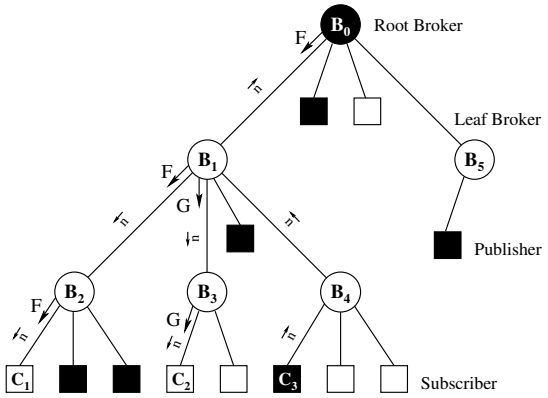

Figure 3: Hierarchical covering-based routing.

As an example, we describe peer-to-peer and hierarchical covering-based routing in more detail. We start with the peer-to-peer variant (Fig. 2): Client C_1 subscribes to a filter F and the subscription is propagated installing corresponding routing entries in the routing tables of all brokers (depicted as solid arrows). After that, client C_2 subscribes to filter G which matches a subset of the notifications matched by filter F. The subscription is propagated to B_1, where it meets the already installed routing entry for filter F. Because of that, G is not forwarded to the neighbor brokers B_0 and B_4, but only to B_2 installing a routing entry there. Then, client C_3 publishes a notification n matching filter G (and thus also F). The notification is propagated from B_4 to B_1 because of the routing entry for filter F. Broker B_1

forwards this notification to B_2 because of the routing entry for filter F and to B_3 due to the entry for filter G. Brokers B_2 and B_3 then deliver n to clients C_1 and C_2, respectively.

The hierarchical variant of covering-based routing (Fig. 3) differs in that (i) less routing entries are installed because subscriptions are only propagated upwards and (ii) the notification n is additionally forwarded to the root broker because notifications are always propagated to the root broker. In general, hierarchical routing is forwarding more notifications but requires smaller routing tables and less subscription forwarding to update routing tables.

3. ANALYTICAL MODELS

In this section, we present our modeling approach considering the different routing algorithms discussed in the previous section. We derive the following performance metrics of the publish/subscribe system (some of them are related to the whole system, others to individual brokers and links) in a step-by-step fashion:

- The *routing table sizes* (Sect. 3.2) which are a measure for the matching overhead.
- The *message rates* (Sect. 3.3) which can be used to estimate the overall system communication costs.
- The *utilization of brokers and links* (Sects. 3.4 and. 3.5) usable to detect performance bottlenecks.
- The *notification delay* (Sect. 3.6) and the *subscription delay* (Sect. 3.7) which are commonly used as Quality of Service (QoS) metrics in publish/subscribe systems.

In the following, we focus on peer-to-peer routing and describe later in Sect. 3.8 how to obtain the same results for hierarchical routing.

3.1 System Model

Before deriving the performance metrics, we describe the basic system model in the following subsections.

3.1.1 Data and Filter Model

Notifications in publish/subscribe systems are typically represented as a set of attributes where each *attribute* is a name/value pair. One of the attributes is often distinguished as *notification class*. For example, the notification $\{(stock, Foo), (price, 37)\}$, where *stock* is the notification class, can be used to indicate that a share of the corporation *Foo* is currently traded at $\$37$. Subscriptions are normally defined as conjunctions of attribute filters, where each *attribute filter* places a constraint on the value of a given attribute. For example, the subscription $\{stock = Foo, price > 35\}$ is matched by the above notification. We use this common model for notifications and subscriptions but make some restrictions for the sake of compactness of the presentation. Without loss of generality, we assume that subscriptions are placed for individual notification classes only. We also speak of a *filter class* referring to the set of all possible subscriptions for a given notification class. In addition to the notification class, we assume that each notification has a numeric attribute whose value v is a real number between 0 and 1. A subscription *matches* notifications of a single class whose numeric attribute's value lies in a given interval $[a, b]$, where a and b are real numbers and $0 \leq a \leq b \leq 1$. We assume that v as well as a and b are randomly chosen. These assumptions are again made for compactness of the presentation. As discussed in Sect. 5, the proposed modeling approach can be extended to support subscriptions and notifications with multiple additional attributes and other than uniform distributions.

In our model, in the case of simple and covering-based routing the complete information about a subscription (i.e., its class and interval) is propagated in the broker network, whereas in the case of identity-based routing only the notification class is propagated. Thus, in the latter case, routing is purely class-based. Finally, in the case of merging-based routing, the notification class is propagated together with an interval $[a_{min}, b_{max}]$, where a_{min} and b_{max} are the minimum a and maximum b of all active subscriptions of the respective filter class in a subtopology. Thus, we investigate a version of merging-based routing that can be seen as an extension of class-based identity-based routing.

3.1.2 Client Model

Instead of dealing with clients directly, we assume independent arrivals of new subscriptions at the brokers for each filter class. For simplicity, the subscription interarrival times are modeled using exponential distributions. The subscription lifetimes, on the other hand, can have arbitrary distributions. We denote with $\lambda^f(B)^{-1}$ the mean *interarrival time* and with $\mu^f(B)^{-1}$ the mean *lifetime* of subscriptions at broker B for filter class f. Effectively, this means that we model the subscription arrivals and departures using a $M/G/\infty$ queuing system for each broker and filter class. We denote with $\omega^f(B)$ the *publication rate* of broker B for filter class f. Let $\mathcal{B}$ be the set of brokers and $\mathcal{F}$ be the set of filter classes. The overall publication rate ω^f for a filter class f within the whole publish/subscribe system is then $\omega^f = \sum_{B \in \mathcal{B}} \omega^f(B)$ and the total publication rate ω over all filter classes is $\omega = \sum_{f \in \mathcal{F}} \omega^f$.

3.1.3 Broker Model

Brokers are modeled as $M/G/1$ queuing systems. For each broker, we allow to set the service time distributions of notifications and control messages individually for each filter class. We denote with $S_n^f(B)$ and $S_c^f(B)$ the service time distributions of broker B for notifications and control messages of filter class f, respectively. This basic model for the service time distributions can easily be extended to include the matching overhead, which depends on the routing table size of the respective broker. This can, e.g., be done by weighting the service time by the routing table size.

3.1.4 Model for Physical Links and Routers

Physical links and routers are also modeled as $M/G/1$ queuing systems. For physical links, we additionally introduce the *latency* $\delta(l)$ of link l that occurs in addition to the queuing delay due to the limited signal propagation speed and the distance between the connected routers. For routers, we set $\delta(l) = 0$. There is currently no need to further differentiate between physical links and routers for the purpose of our analysis because we consider both of them as *physical components* that introduce a delay for messages. Thus, in the rest of the paper we will speak of physical components without distinguishing between links and routers. If a physical component l is part of an overlay link (B_i, B_j), we denote this by $l \in (B_i, B_j)$. Finally, we define $S_n^f(l)$ and

$S_c^f(l)$ as the service time distributions of l for notifications and control messages of filter class f, respectively.

3.2 Routing Table Sizes

The routing table of a broker consists of *local routing entries* used to deliver notifications to local subscribers and *remote routing entries* used to forward notifications to neighbor brokers along delivery paths. The expected number of local routing entries at broker B for filter class f is given by:

$$x_l^f(B) = \lambda^f(B) \cdot \mu^f(B)^{-1} \tag{1}$$

Given that client communication is local, the local routing entries are of minor interest because they do not cause messages to be sent across the network. We therefore concentrate on the remote routing entries.

Let $\mathcal{N}(B)$ be the set of all neighbor brokers of a broker B and let $\mathcal{T}(B_i, B_j)$ be the subtopology containing all brokers that can be reached from a broker B_i via a neighbor broker B_j including B_j itself. Furthermore, let $\mathcal{S}(B_i, B_j) = \mathcal{N}(B_j) \setminus \{B_i\}$. Then, the accumulated number of subscriptions $x_a^f(B_i, B_j)$ for filter class f in $\mathcal{T}(B_i, B_j)$ is:

$$x_a^f(B_i, B_j) = x_l^f(B_j) + \sum_{N \in \mathcal{S}(B_i, B_j)} x_a^f(B_j, N) \tag{2}$$

To calculate the number of remote routing entries $x_r^f(B)$ a broker B has for filter class f, we have to take every neighbor N of B into account, where $x_r^f(B, N)$ is the number of remote routing entries that broker B has for its neighbor N:

$$x_r^f(B) = \sum_{N \in \mathcal{N}(B)} x_r^f(B, N) \tag{3}$$

The probability of having k active subscriptions in the subtopology $\mathcal{T}(B_i, B_j)$ for filter class f is then:

$$p_k^f(B_i, B_j) = \frac{e^{-x_a^f(B_i, B_j)}}{k!} \cdot x_a^f(B_i, B_j)^k \tag{4}$$

Let $E_r^f(k)$ be the expected number of remote routing entries for filter class f if there are k active subscriptions for the same filter class in the subtopology $\mathcal{T}(B_i, B_j)$. Then,

$$x_r^f(B_i, B_j) = \sum_{k=0}^{\infty} p_k^f(B_i, B_j) \cdot E_r^f(k) \tag{5}$$

Now, we derive $E_r^f(k)$ for the considered routing algorithms.

Flooding. For flooding, $E_r^f(k) = 0$ for all k and therefore $x_r^f(B_i, B_j) = 0$.

Simple Routing. For simple routing $E_r^f(k) = k$ for all k implying $x_r^f(B_i, B_j) = x_a^f(B_i, B_j)$.

Identity-Based and Merging-Based Routing. With both routing algorithms, a broker B_i has a single remote routing entry for its neighbor broker B_j and filter class f if there is at least one active subscription for this filter class in the subtopology $\mathcal{T}(B_i, B_j)$. Thus, $E_r^f(k) = 0$ for $k = 0$ and $E_r^f(k) = 1$ for $k > 0$ implying $x_r^f(B_i, B_j) = 1 - e^{-x_a^f(B_i, B_j)}$.

Covering-Based Routing. Here, each remote routing entry for a neighbor broker B_j corresponds to a subscription existing in the subtopology $\mathcal{T}(B_i, B_j)$ that is not covered by any other subscription existing in this subtopology. Let $p_{cs}^f(k)$ be the probability that a subscription for filter class f is covered by any of k other subscriptions for the same filter class. Then, $E_r^f(k) = k \cdot (1 - p_{cs}^f(k-1))$.

We now calculate $p_{cs}^f(k)$. The probability that a given subscription $[a, b]$ for filter class f is covered by another randomly chosen subscription for the same filter class is $2a \cdot (1 - b)$. Thus, the probability that a given subscription is not covered by any out of k subscriptions is $(1-2a(1-b))^k$. The complement of this probability is then the probability that the subscription is covered by at least one of the k subscriptions. We have to average this probability by integration over all a and b since both are chosen randomly:

$$p_{cs}^f(k) = 2 \cdot \int_{a=0}^{1} \int_{b=a}^{1} 1 - (1 - 2a\,(1-b))^k \,\mathrm{d}b\,\mathrm{d}a \tag{6}$$

$$= 1 - 2 \cdot \sum_{l=0}^{k} (-2)^l \frac{k!}{(k-l)!} \frac{l!}{(2l+2)!} \tag{7}$$

$$\approx \frac{2}{m(m-1)} \sum_{u=1}^{m} \sum_{v=1}^{m-u+1} \left[1 - \left(\frac{1 - (u - \frac{1}{2})(v - \frac{1}{2})}{m(m-1)/2} \right)^k \right] \tag{8}$$

While Eq. 7 gives the exact solution, Eq. 8 gives an approximation derived using numerical integration, which can be computed more efficiently. In Eq. 8, m is the number of interpolation points.

3.3 Message Rate

The *message rate* is defined as the sum of the notification rate and the control message rate which are now derived.

3.3.1 Notification Rate

Let $P_n^f(B_i, B_j)$ be the probability that a notification of filter class f is propagated from a broker B_i to a neighbor broker B_j. If a notification of filter class f is published at broker B_i, it is propagated into the subtopology $\mathcal{T}(B_i, B_j)$ causing the following notification traffic:

$$E_n^f(B_i, B_j) = P_n^f(B_i, B_j) + \sum_{N \in \mathcal{S}(B_i, B_j)} E_n^f(B_j, N) \tag{9}$$

The total notification rate generated by a broker B is then:

$$b_n^f(B) = \omega^f(B) \cdot \sum_{N \in \mathcal{N}(B)} E_n^f(B, N) \tag{10}$$

The total notification rate b_n^f for filter class f in the whole system and the total notification rate b_n over all filter classes in the whole system are:

$$b_n^f = \sum_{B \in \mathcal{B}} b_n^f(B) \qquad b_n = \sum_{f \in \mathcal{F}} b_n^f \tag{11}$$

Now, we derive $P_n^f(B_i, B_j)$. Let $p_n^f(k)$ be the probability that a notification is forwarded if there are k subscriptions of the same filter class f in the respective subtopology. Then,

$$P_n^f(B_i, B_j) = \sum_{k=0}^{\infty} p_k^f(B_i, B_j) \cdot p_n^f(k) \tag{12}$$

We now derive $p_n^f(k)$ for the different routing algorithms.

Flooding. Here, $p_n^f(k) = 1$ for all $k \geq 0$ and therefore $P_n^f(B_i, B_j) = 1$.

Identity-Based Routing. Here, $p_n^f(k) = 0$ for $k = 0$ and $p_n^f(k) = 1$ for $k > 0$ implying $P_n^f(B_i, B_j) = 1 - e^{-x_a^f(B_i,B_j)}$.

Simple and Covering-Based Routing. $p_n^f(k)$ does not differ for simple and covering-based routing since in both cases a notification is propagated if there is a matching subscription in the respective subtopology. The main difference from identity-based routing here is that both the notification class and the interval are required to match.

Again, $p_n^f(k) = 0$ for $k = 0$. We now derive $p_n^f(k)$ for $k > 0$. The probability that a point x randomly chosen from the interval $[0, 1]$ is covered by an interval whose limits are randomly chosen from $[0, 1]$ is $2 \cdot x(1 - x)$. The probability that x is not matched by any of k subscriptions is thus $(1 - 2 \cdot x(1 - x))^k$. The complement of this probability is the probability that a given notification is matched by at least one of k subscriptions. To obtain the desired result we have to average this probability by integration over all x since x is chosen randomly:

$$p_n^f(k) = \int_0^1 1 - (1 - 2 \cdot x\,(1 - x))^k \; \mathrm{d}x \tag{13}$$

$$= 1 - \sum_{l=0}^{k} (-2)^l \frac{k!}{(k-l)!} \frac{l!}{(2l+1)!} \tag{14}$$

$$\approx \frac{1}{m} \sum_{u=1}^{m} \left[1 - \left(\frac{1 - (u - \frac{1}{2})(m - u + \frac{1}{2})}{m(m-1)/2} \right)^k \right] \tag{15}$$

While Eq. 14 gives the exact solution, Eq. 15 gives an approximation derived using numerical integration using m interpolation points, which can be computed more efficiently.

Merging-Based Routing. Here, a notification of class f with value v of the numeric attribute is propagated from broker B_i to B_j if there is (i) a subscription $[a_1, b_1]$ for the same filter class in the subtopology $\mathcal{T}(B_i, B_j)$ with $a_1 \leq v$ and (ii) a subscription $[a_2, b_2]$ for the same filter class with $v \leq b_2$. These conditions can also be satisfied by a single subscription. Hence, $p_n^f(k) = 0$ for $k = 0$. If there are $k > 0$ subscriptions $[a_l, b_l]$ for a filter class f, then the probability that the minimum value a_{min} of all a_l's equals a is $(1-a)^{2k}$ and the probability that the maximum value b_{max} of all b_l's equals b is b^{2k}. The expected value of a_{min} is $a_{min}(k) = \int_0^1 a \cdot 2k \cdot (1-a)^{2k-1} \; \mathrm{d}a = (2k+1)^{-1}$ and the expected value of b_{max} is $b_{max}(k) = \int_0^1 b \cdot 2k \cdot b^{2k-1} \; \mathrm{d}b = 2k(2k+1)^{-1}$. The expected value of $b_{max} - a_{min}$ is $(2k-1)(2k+1)^{-1}$. Thus,

$$p_n^f(k) = \frac{2k-1}{2k+1} \tag{16}$$

3.3.2 Control Message Rate

The control message rate b_c covers all messages sent in the system to keep the routing tables up-to-date. Let $P_s^f(B_i, B_j)$ be the probability that a subscription is propagated from broker B_i to a neighbor broker B_j. If a subscription is issued at broker B_i, it is propagated into the subtopology $\mathcal{T}(B_i, B_j)$ causing the following control message traffic:

$$E_s^f(B_i, B_j) = P_s^f(B_i, B_j) + \sum_{N \in \mathcal{S}(B_i, B_j)} E_s^f(B_j, N) \tag{17}$$

The same message traffic is generated when a subscription is revoked at broker B_i. Thus, the control message rate generated by a broker B for filter class f is:

$$b_c^f(B) = 2 \cdot \lambda^f(B) \cdot \sum_{N \in \mathcal{N}(B)} E_s^f(B, N) \tag{18}$$

The total control message rate b_c^f for filter class f for the whole system and total control message rate b_c for the whole system over all filter classes are then:

$$b_c^f = \sum_{B \in \mathcal{B}} b_c^f(B) \qquad b_c = \sum_{f \in \mathcal{F}} b_c^f \tag{19}$$

Now, we determine $P_s^f(B_i, B_j)$. Let $p_s^f(k)$ be the probability that a subscription being issued or revoked is forwarded if there are currently k other subscriptions for the same filter class f active in the subtopology $\mathcal{T}(B_j, B_i)$. Then,

$$P_s^f(B_i, B_j) = \sum_{k=0}^{\infty} p_k^f(B_j, B_i) \cdot p_s^f(k) \tag{20}$$

Now, we derive $p_s^f(k)$ for the different routing algorithms.

Flooding. With flooding, a subscription is never propagated. Hence, $\forall k : p_s^f(k) = 0$ and, thus, $P_s^f(B_i, B_j) = 0$.

Simple Routing. With simple routing, a subscription issued or revoked is always propagated as a control message regardless how many other subscriptions are active in the subtopology $\mathcal{T}(B_j, B_i)$. Hence, $p_s^f(k) = 1$ for all k and, therefore, $P_s^f(B_i, B_j) = 1$.

Identity-Based Routing. With identity-based routing, a subscription issued or revoked leads to a control message propagated by broker B_i to its neighbor broker B_j if there is no other subscription for the same filter class in the subtopology $\mathcal{T}(B_j, B_i)$. Hence, $p_s^f(k) = 1$ for $k = 0$ and $p_s^f(k) = 0$ for $k > 0$ implying $P_s^f(B_i, B_j) = e^{-x_a^f(B_j,B_i)}$.

Covering-Based Routing. With covering-based routing, a subscription issued or revoked leads to a control message propagated by broker B_i to its neighbor broker B_j if this subscription is not covered by any existing subscription in the subtopology $\mathcal{T}(B_j, B_i)$. Thus, $p_s^f(k) = 1 - p_{cs}^f(k)$.

Merging-Based Routing. Here, a newly issued or canceled subscription for filter class f and interval $[a, b]$ at broker B_i causes a control message to be propagated to broker B_j if a is smaller than the current a_{min} or b is greater than the current b_{max}. The probability for the former is $a_{min}(k)$, for the latter it is $1 - b_{max}(k)$ provided that there are k subscriptions for filter class f active in the subtopology $\mathcal{T}(B_j, B_i)$. Thus, $p_s^f(k) = 1 - [(1 - a_{min}) \cdot b_{max}] = (4k+1)/(2k+1)^2$.

3.4 Performance Metrics for Brokers

In this section, we determine performance metrics for individual brokers including the expected broker traffic and broker utilization as well as the expected message delay.

3.4.1 Broker Traffic

The broker traffic consists of two parts: (a) traffic caused by notifications and (b) control traffic caused by control mes-

sages as well as local subscriptions and unsubscriptions. The the overall traffic is then the sum of these two parts.

Notification Traffic. The notification traffic on a broker B for filter class f consists of the notifications published by its local clients and those received from its neighbor brokers:

$$\nu_n^f(B) = \omega^f(B) + \sum_{N \in \mathcal{N}(B)} \nu_n^f(N, B) \tag{21}$$

The notification traffic $\nu_n^f(B_i, B_j)$ for filter class f that broker B_i forwards to B_j is:

$$\nu_n^f(B_i, B_j) = P_n^f(B_i, B_j) \cdot \omega_a^f(B_j, B_i) \tag{22}$$

where $\omega_a^f(B_j, B_i)$ is the accumulated publication rate for filter class f in the subtopology $\mathcal{T}(B_j, B_i)$:

$$\omega_a^f(B_j, B_i) = \omega^f(B_i) + \sum_{N \in \mathcal{S}(B_j, B_i)} \omega_a^f(B_i, N) \tag{23}$$

Control Traffic. The control traffic on a broker B consists of the local subscriptions and unsubscriptions as well as the control messages received from its neighbor brokers:

$$\nu_c^f(B) = 2 \cdot \lambda^f(B) + \sum_{N \in \mathcal{N}(B)} \nu_c^f(N, B) \tag{24}$$

The control traffic $\nu_c^f(B_i, B_j)$ for filter class f that broker B_i sends to B_j is:

$$\nu_c^f(B_i, B_j) = 2 \cdot P_s^f(B_i, B_j) \cdot \lambda_a^f(B_j, B_i) \tag{25}$$

where $\lambda_a^f(B_j, B_i)$ is the accumulated arrival rate of subscriptions of filter class f in the subtopology $\mathcal{T}(B_j, B_i)$:

$$\lambda_a^f(B_j, B_i) = \lambda^f(B_i) + \sum_{N \in \mathcal{S}(B_j, B_i)} \lambda_a^f(B_i, N) \tag{26}$$

Overall Broker Traffic. The overall broker traffic is then:

$$\nu(B) = \sum_{f \in \mathcal{F}} \left(\nu_n^f(B) + \nu_c^f(B) \right) \tag{27}$$

3.4.2 Broker Utilization

The overall utilization of broker B is caused by the notifications and control messages of all filter classes:

$$U(B) = \sum_{f \in \mathcal{F}} \left(\nu_n^f(B) \cdot E[S_n^f(B)] + \nu_c^f(B) \cdot E[S_c^f(B)] \right) \tag{28}$$

where $E[S_n^f(B)]$ and $E[S_c^f(B)]$ denote the expected values of the service time distributions for notifications and control messages of broker B. Note that if $U(B) > 1$ for some broker, the system is overloaded and not in a steady state.

3.4.3 Broker Delay

Modeling the brokers with $M/G/1$ queues, we can approximate the mean waiting time $W(B)$ of a message at broker B using the Pollaczek-Khinchin mean value formula [13]:

$$W(B) = \frac{\nu(B) \cdot E[S(B)^2]}{2(1 - U(B))} \tag{29}$$

In the equation above, $E[S(B)^2]$ is the second moment of the averaged service time distribution $S(B)$ of broker B. $S(B)$ can be derived by combining the individual $S_n^f(B)$ and $S_c^f(B)$ weighted by their relative shares of the overall traffic, i.e., $\nu_n^f(B)/\nu(B)$ and $\nu_c^f(B)/\nu(B)$, respectively.

The approximated expected delay for a notification $D_n^f(B)$ and for a control message $D_c^f(B)$ of filter class f at broker B is then given by:

$$D_n^f(B) = W(B) + E[S_n^f(B)] \tag{30}$$

$$D_c^f(B) = W(B) + E[S_c^f(B)] \tag{31}$$

Since we can use any service time distribution as long as its second moment $E[S^2(B)]$ and the expected processing times $E[S_n^f(B)]$ and $E[S_c^f(B)]$ can be computed, we are able to model graduated broker delays for the different message types. For example, we could use a hyper-exponential service time distribution consisting of two exponential distributions per filter class, one for the notifications and another for the control messages of each filter class.

3.5 Performance Metrics for Links

We now turn to performance metrics for physical links and overlay links. First, we determine the traffic on overlay links. After that, we derive the utilization of physical links and routers as well as the message delays they introduce. Finally, we derive the utilization and message delay of overlay links.

3.5.1 Overlay Link Traffic

The traffic on overlay link (B_i, B_j) is the notification and control message flow of filter class f from B_i to B_j:

$$\nu^f(B_i, B_j) = \nu_n^f(B_i, B_j) + \nu_c^f(B_i, B_j) \tag{32}$$

3.5.2 Traffic, Utilization and Delay of Physical Links and Routers

An overlay link (B_i, B_j) connects two brokers in the overlay topology by a path in the underlying physical network consisting of physical links and routers. The overall traffic per filter class $\nu^f(l)$ and the overall traffic $\nu(l)$ on a physical component l is then:

$$\nu^f(l) = \sum_{l \in (B_i, B_j)} \nu^f(B_i, B_j) \qquad \nu(l) = \sum_{f \in \mathcal{F}} \nu^f(l) \tag{33}$$

The notification and subscription traffic $\nu_n^f(l)$ and $\nu_c^f(l)$ can be determined using Eq. 33 by considering only $\nu_n^f(B_i, B_j)$ and $\nu_c^f(B_i, B_j)$, respectively.

The utilization $U(l)$ of component l is (cf. Eq. 28):

$$U(l) = \sum_{f \in \mathcal{F}} \left(\nu_n^f(l) \cdot E[S_n^f(l)] + \nu_c^f(l) \cdot E[S_c^f(l)] \right) \tag{34}$$

Note that if $U(l) > 1$ for some physical component, the system is overloaded and not in a steady state. We can approximate the mean waiting time $W(l)$ of a message at component l with (cf. Eq. 29):

$$W(l) = \frac{\nu(l) \cdot E[S(l)^2]}{2(1 - U(l))} \tag{35}$$

The approximated expected delay for a notification $D_n^f(l)$ and for a control message $D_c^f(l)$ of filter class f at component l is then given by (cf. Eqs. 30 and 31):

$$D_n^f(l) = W(l) + E[S_n^f(l)] + \delta(l) \tag{36}$$

$$D_c^f(l) = W(l) + E[S_c^f(l)] + \delta(l) \tag{37}$$

3.5.3 Overlay Link Delay

We now derive the expected delay $D_n^f(B_i, B_j)$ of a notification and the expected delay $D_c^f(B_i, B_j)$ of a control message of filter class f at overlay link (B_i, B_j) by adding up the delays at its constituting physical components:

$$D_n^f(B_i, B_j) = \sum_{l \in (B_i, B_j)} D_n^f(l) \tag{38}$$

$$D_c^f(B_i, B_j) = \sum_{l \in (B_i, B_j)} D_c^f(l) \tag{39}$$

The derived delays for the overlay links are used to calculate the notification delay as well as the subscription delay in the next two subsections.

3.6 Notification Delay

A notification published at some broker is forwarded stepwise through intermediate brokers until it reaches all brokers that have a local client with a matching subscription. The *delay of a notification* is the time between its publication at the initial broker and its latest delivery to one of the destination brokers. To determine the expected delay of a notification, the expected delays of the brokers and links on the delivery paths are not sufficient because the expected value of the maximum of a set of random variables is higher than the expected values of each random variable. In order to derive the expected notification delay, the probability density functions of the delays of all links and brokers would have to be used. For simplicity, we focus on deriving the *expected notification delay for the worst delivery path.*

In the following, we recursively derive the expected worst-path delay of a broker for a given filter class. We first calculate the delay excluding the publishing broker B_i for the delivery paths going through its neighbor broker B_j. As a next forwarding step, the neighbor broker N of B_j is chosen that maximizes the delay for this filter class considering all neighbor brokers to which notifications of the filter class are potentially forwarded (i.e., $P_n^f(B_j, N) > 0$):

$$\Delta_n^f(B_i, B_j) = D_n^f(B_i, B_j) + D_n^f(B_j) + \max_{N \in \mathcal{S}(B_i, B_j) : P_n^f(B_j, N) > 0} \Delta_n^f(B_j, N) \tag{40}$$

The expected worst-path notification delay $\Delta_n^f(B)$ for broker B and filter class f is then given by:

$$\Delta_n^f(B) = D_n^f(B) + \max_{N \in \mathcal{N}(B)} \Delta_n^f(B, N) \tag{41}$$

To derive a global measure, we determine the filter class generating the highest delay at some broker B that has a local publisher of this filter class (i.e., $\omega^f(B) > 0$):

$$\Delta_n = \max_{f \in \mathcal{F}} \left[\max_{B \in \mathcal{B} : \omega^f(B) > 0} \Delta_n^f(B) \right] \tag{42}$$

3.7 Subscription Delay

A subscription issued at some broker is propagated stepwise through the broker overlay network until it reaches other subscriptions that suspend its further propagation or until a leaf broker is reached. The *delay of a subscription* is the time between its issuing and the end of its propagation, i.e., the latest time a broker decides to not propagate the subscription further. Similar to the notification delay, we also derive the *expected subscription delay for the worst forwarding path.* This measure can again be calculated based on the expected delays of brokers and links. We recursively sum up the maximum delays of links and brokers if there is a non-zero subscription forwarding probability (i.e., $P_s^f(B_j, N) > 0$):

$$\Delta_s^f(B_i, B_j) = D_c^f(B_i, B_j) + D_c^f(B_j) + \max_{N \in \mathcal{S}(B_i, B_j)\} : P_s^f(B_j, N) > 0} \Delta_s^f(B_j, N) \tag{43}$$

The expected worst path subscription delay $\Delta_s^f(B)$ for broker B and filter class f is then given by:

$$\Delta_s^f(B) = D_c^f(B) + \max_{N \in \mathcal{N}(B)} \Delta_s^f(B, N) \tag{44}$$

To derive a global measure, we determine the filter class generating the highest delay at some broker B that may issue a subscription of this filter class (i.e., $\lambda^f(B) > 0$):

$$\Delta_s = \max_{f \in \mathcal{F}} \left[\max_{B \in \mathcal{B} : \lambda^f(B) > 0} \Delta_s^f(B) \right] \tag{45}$$

Both delays Δ_n and Δ_s are major performance metrics for publish/subscribe systems which are also used in the case study presented in Sec. 4.

3.8 Hierarchical Routing

Up to now, we concentrated on peer-to-peer routing. In order to derive the performance metrics for hierarchical routing, the forwarding of subscriptions has to be restricted such that subscriptions are *only* forwarded upwards in the broker tree (cf. change 1 below) and the forwarding of notifications has to be extended such that notifications are *always* propagated towards the root broker (cf. change 2 below). Furthermore, since subscriptions are not propagated downwards in the broker tree, only subscriptions originating in the subtree rooted at a broker can influence the routing tables, and, thus, also the subscription forwarding probabilities of this broker (cf. change 3 below). A consequence of this is, for example, that a broker has only remote routing entries for its child brokers but no entries for its parent broker. Incorporating these three changes into the equations presented is rather simple:

- **Change 1:** $P_s^f(B_i, B_j)$ is defined as before, but defined to be 0 if B_j is *not* the parent broker of B_i.
- **Change 2:** $P_n^f(B_i, B_j)$ is defined as before, but defined to be equal to 1 if B_j is the parent of B_i.
- **Change 3:** $x_a^f(B_i, B_j)$ is defined as before, but defined to be 0 if B_j is the parent broker of B_i.

4. CASE STUDY

To validate our approach and demonstrate its effectiveness in a realistic scenario, we now present a case study of a representative publish/subscribe system. We consider a scenario in which a company that operates world-wide needs to exchange data (e.g., stock quotes, trade orders, and business news) with its customers and field staff in an efficient and

timely manner. The company's activities are divided into different areas (e.g., countries) where each area runs some regional offices that may host publish/subscribe brokers. In order to plan the capacity of the system, the company needs to know the optimal number of brokers and their placement in the network, as well as the resulting utilization of links and brokers, and the expected system performance in terms of end-to-end notification and subscription delays.

A scenario is defined by a static and a dynamic part. The static part represents parameters that are usually fixed in reality and includes the network topology as well as the positions of clients and company offices that may host brokers. The dynamic part is used to describe concrete use cases and workloads characterized by message traffic and the locality of subscriptions and notifications.

The physical topology is created with 5 domains (representing the areas) and about 1000 routers (possible locations of clients) using the Brite [15] topology generator. To derive an Internet-like topology, the transit-stub model is applied. The capacity of the physical links varies between 30 000 and 60 000 messages per second, and the capacity of physical routers is set to 250 000 messages per second. Within this topology, 100 clients (representing customers and field staff that produce and consume notifications) are positioned by attaching them to randomly chosen routers in the physical topology. Next, two special locations per domain are selected that represent the company's regional offices that may be used to host a broker. For a given broker placement using some or all of these locations, each broker has a capacity of 1400 messages per second and is connected to the other brokers by a minimum spanning tree based on the distance in the physical topology measured in hops. The clients are connected to the nearest broker.

On top of this static setting a concrete scenario is described by a number of active subscriptions $\mathcal{N}_s$ in the system with an exponentially distributed expected lifetime of $60s$, a notification rate ω and a locality measure L described below. The service times for processing messages at the brokers are equally distributed and have an exponential distribution independent of the message type. Each subscription and notification is assigned to one of 500 different filter classes. Regarding locality, each class has a randomly chosen hot spot domain in which a fraction L of its notifications are produced and a fraction L of its subscriptions originate from clients within that domain. The remaining subscriptions/notifications are uniformly distributed among the other four domains.

Capacity Planning.

Given such a scenario description defined by a physical network topology, the possible broker positions, and a workload specification, capacity planning aims at answering the question where to best place how many brokers using which routing algorithm in order to minimize the average notification delay. Regarding our company example, there are ten possible broker positions (two in each of the five domains) leading to $2^{10} - 1 = 1023$ different broker placements for which each of the seven individual routing algorithms has to be investigated. Thus, even this small-sized example finally leads to 7161 cases to be considered for just a single workload specification.

Up to now, discrete event simulation was the only viable approach to evaluate these scenarios and derive estimations

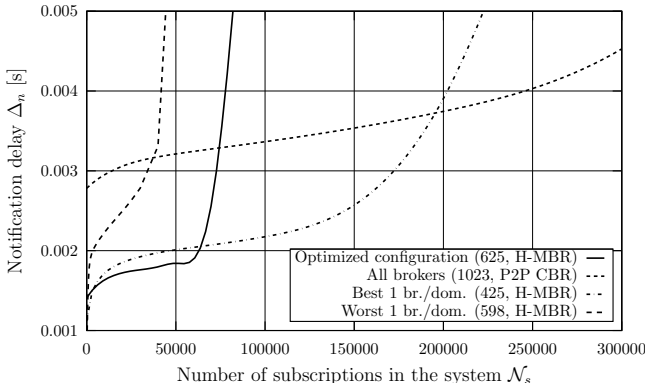

Figure 4: Expected worst-path notification delay of different broker placements & routing configurations

for the traffic rates, message delays, and routing table sizes in advance. However, as simulation is costly in terms of computation time and memory requirements, one usually restricts the number of considered cases drastically by, e.g., adding further placement constraints and/or preselecting the routing algorithm. Thus, a manageable, conventional simulation strategy would be to place exactly one broker per domain (maybe evaluating all 32 permutations) and to compare the obtained results to a complete placement with all ten brokers.

Leveraging the stochastic analysis derived in this paper, however, we are able to cover a much larger parameter space. In fact, it becomes possible to solve this optimization problem by conducting an extensive search with justifiable resources in terms of computing power and time. For the company scenario with $50,000$ active subscriptions and a locality of 0.8, we determined the optimal broker placement and best routing configuration within minutes. The results are shown in Fig. 4. The best solution leads to an expected notification delay of 1.8 ms, employs hierarchical merging-based routing and uses five brokers which are not equally distributed over the domains: one domain has two, another has none, while the others have one. In reality, this means that some company offices have a bad connectivity within their domain while offices in neighbor domains are better connected. In the restricted simulation setup described above, this case would not be evaluated at all. Furthermore, if considering all 32 configurations with exactly one broker per domain the results vary between 2 ms and 7 ms for the best (topology 425) and the worst case (topology 598). Thus, getting a good candidate randomly is quite unlikely.

Scalability.

Usually, parameters such as load and locality are estimated based on previous experience or expectations and are therefore subject to errors. This leads to systems facing workloads beyond the workload intensities for which they have been optimized. Therefore, it is necessary to investigate how scalable the derived configurations are. To illustrate this problem, in Fig. 4, we varied the load from zero to $300,000$ active subscriptions to study the impact on the different configurations including the optimal one. Obviously, the optimal configuration does not scale well and saturates for higher loads. On the other hand, configuration 425 is

less sensitive for an increasing number of subscriptions and should be preferred although the delay for the optimization point is marginally higher. However, the best scalability is provided by a complete 10 broker topology, but with a significantly higher average delay. This example shows the need for a comprehensive system analysis that is hard to achieve with simulation-based approaches.

Contrary to discrete event simulation, our analytical approach is especially efficient for higher subscription and notification rates making such scalability evaluations very cheap. This is due to the fact that our approach does not need to deal with each message separately and its computational time does not depend on the message load. Fig. 5 shows the relative execution times of our analysis compared to the runtime of a corresponding discrete event simulation in relation to the number of subscriptions. One can see that for approximately $60,000$ active subscriptions the complete evaluation of all 1023 possible broker placements is as costly as one simulation run for one placement.

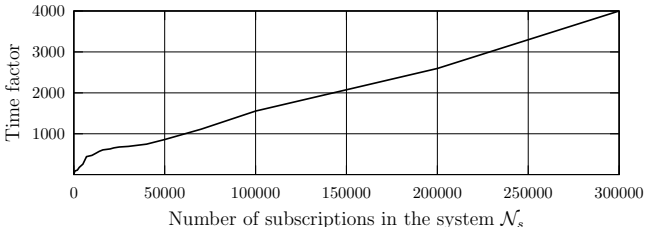

Figure 5: Runtime: simulation vs. analysis

We compared the results to those gathered by a discrete event simulation. Since we did not vary any model assumptions and only used the $M/M/1$ model for links and brokers in the case study, all simulation results eventually converge to the analytical results gained using our model with an increasing number of simulation runs and a longer simulation time, respectively.

Advanced Evaluations.

Beside capacity planning for concrete workload scenarios, our approach allows to gain further insights into the behavior of the publish/subscribe system without expensive simulation runs. In the considered setting, we were able to make several important observations described in the following.

First, we considered the selection of an optimal routing algorithm for a given subscription load. We derived the optimal (with respect to minimizing the maximum expected notification delay) routing algorithm for each of the 1023 broker placements, four different localities (0.2, 0.4, 0.6 and 0.8) and a subscription load varying from 10 to $700,000$ subscriptions in the system. Fig. 6 shows the distribution of the four routing algorithms hierarchical covering-based (H-CBR), hierarchical merging-based (H-MBR), peer-to-peer covering-based (P2P-CBR) and peer-to-peer merging-based (P2P-MBR) in a way that the numbers represent the fraction of cases where the appropriate algorithm is optimal. It is easy to see that peer-to-peer merging-based routing is optimal for low load, hierarchical covering-based routing for medium load and hierarchical merging-based routing for high load. For the given setting, this insight might help to optimize the system behavior in case of only vague knowledge on expected load.

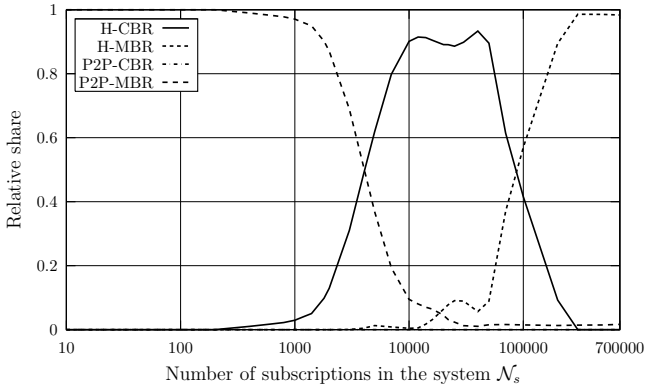

Figure 6: Optimal routing algorithms and load

Second, we analyzed the impact of locality. Fig. 7 shows the distribution of optimal routing algorithms for different localities based on the experiment described above (with load and locality exchanged so that each data points represents results from evaluating 1023 broker placements with varying load). Here, we see that both types of merging-based routing profit from high locality while for hierarchical covering-based routing the opposite effect can be seen. Furthermore, in our setting locality seems to influence the selection of the optimal routing algorithm much less than it is influenced by the number of subscriptions.

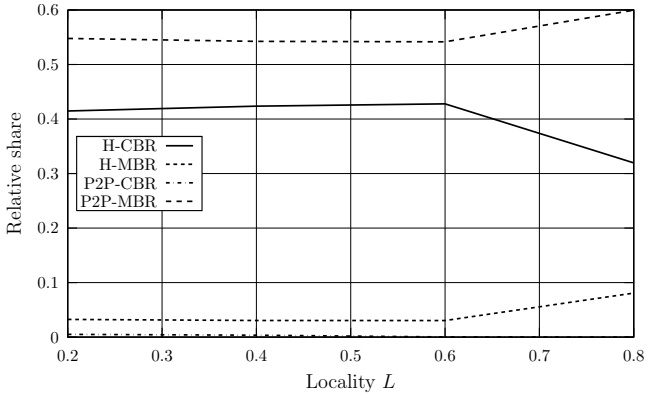

Figure 7: Optimal routing algorithms and locality

The described insights are just two examples of how the proposed stochastic modeling approach allows to analyze the impact of design decisions on the system performance without the need for expensive and time consuming simulations.

5. FUTURE DIRECTIONS

As part of our future work, we plan to conduct case studies with real-world applications and workloads in order to evaluate the effectiveness of our modeling approach in real-life settings. The validity of the assumptions underlying the proposed models will be evaluated as well as the applicability of the models in cases where the assumptions are violated. To improve the representativeness of the models, we intend to refine and extend them along several different directions.

One major extension would be to introduce a more powerful client behavior model. The presented model can be extended to support notifications with several attributes as well as subscriptions constraining several notification attributes conjunctively using different operators. Also non-uniform value distributions and other value ranges are possible. The necessary changes only affect the derivation of the forwarding probabilities of notifications and subscriptions which must be adapted appropriately. We also intend to extend the broker behavior model. More specifically, we plan to derive the service time for notifications depending on the routing tables size and the applied matching algorithm (e.g., brute force, predicate counting). Similarly, the service time for control messages will be calculated depending on the routing tables sizes and the applied routing algorithm.

We also plan to consider correlated client behavior, where the arrival processes of subscriptions and notifications are no longer assumed to be independent. Currently, we can capture correlated behavior only partially by considering equivalent quasi-stationary settings. For example, if notifications A and B are expected to be published once an hour and 600 times an hour, but all occurrences of B directly follow that of A within ten minutes, we can capture this situation by setting the publications rate of B to $1s^{-1}$ instead of $(6s)^{-1}$.

Furthermore, we will investigate different variants of notification and subscription delays and their potential approximations in addition to those presented in Sects. 3.6 and 3.7. Our goal is to find an approximation which is rather easily computable but still allows to predict the behavior of the system precisely enough.

Finally, the models will be extended to support more sophisticated routing algorithms, like hybrid routing [22] or algorithms based on advertisements. While hybrid routing allows to apply different routing algorithms for different overlay links in the same publish/subscribe topology, advertisements allow the propagation of subscriptions to be restricted to those parts of the overlay network where matching notifications can potentially be produced.

6. RELATED WORK

Castelli et al. [8] present an analytical model of publish/subscribe systems. The authors provide closed form analytical expressions for the overall network traffic required to disseminate subscriptions and propagate notifications, as well as for the message forwarding load on individual system nodes. However, the model, at least in its presented form, assumes that subscribers and publishers are uniformly distributed over a balanced tree in which each inner node has the same number of children.

Mühl et al. [17] present an approach to stochastic analysis of publish/subscribe systems employing identity-based hierarchical routing. While this work has a similar direction, it only considers routing table sizes and message rates as metrics and does not consider peer-to-peer routing. Moreover, it does not deal with the more advanced routing algorithms such as covering-based and merging-based routing, and the proposed models require more restrictive assumptions such as identical processing costs for all notification and control messages. Finally, the physical network and its effect on the traffic generated by the overlay network is not captured.

Bricconi et al. [5] present a model of the JEDI publish/subscribe system that is mainly used to calculate the number of notifications received by each broker using a uniform distribution of subscriptions. To model the multicast communication, the authors introduce a spreading coefficient which models the probability that a broker at a given hop-distance from the publishing broker receives a published notification.

Baldoni et al. [3, 4] propose an analytical model of distributed computation based on a publish/subscribe system. The system is abstracted through two delays, namely the subscription/unsubscription delay and the diffusion delay which are assumed to be known. However, the proposed model is only used to calculate the number of notifications missed by subscribers due to high network delays. Performance metrics such as notification delays and broker utilization are not considered.

A basic high-level cost model of publish/subscribe systems in mobile Grid environments is presented by Oh et al. [18]. This model, however, does not provide much insight into the behavior of the system since it is based on the assumption that the publish/subscribe cost and time delay per notification are known. He et al. [11] use probabilistic model checking techniques and stochastic models to analyze publish/subscribe systems. The communication infrastructure (i.e., the transmission channels and the publish/subscribe middleware) are modeled by means of probabilistic timed automata. The analysis considers the probability of message loss, the average time taken to complete a task and the optimal message buffer sizes. However, distributed broker topologies are not considered.

Kounev et al. [14] present a methodology for workload characterization and performance modeling of distributed event-based systems. A workload model of a generic system is developed and analytical analysis techniques are used to characterize the system traffic and to estimate the mean notification delivery latency. For more accurate performance prediction queuing Petri net models are used. While this technique is applicable to a wide range of systems, it relies on monitoring data obtained from the system and it is therefore only applicable if the system is available for testing. Furthermore, for systems of realistic size and complexity, the queuing Petri net models would not be analytically tractable and thus one would have to resort to simulation.

7. CONCLUSIONS

In this paper, we presented a set of analytical models for stochastic performance analysis of publish/subscribe systems. The work presented here extends existing work on modeling of publish/subscribe systems in several important aspects: (a) all major performance-relevant metrics of publish/subscribe systems are considered, (b) a large variety of different peer-to-peer and hierarchical content-based routing algorithms is supported, (c) the proposed models cover a wider range of systems and can be used for comprehensive analysis. We presented a case study of a representative publish/subscribe system demonstrating the effectiveness and practicality of the modeling approach. The proposed models can be used to identify and eliminate performance bottlenecks before a system is deployed as well as for sizing and capacity planning. Moreover, the ability to predict the system performance is important to ensure that applications meet their Quality of Service requirements. Finally, performance prediction helps to optimize a publish/subscribe system and find an optimal broker overlay topology.

8. REFERENCES

[1] J. Abbott, K. B. Manrodt, and P. Moore. From Visibility to Action: Year 2004 Report on Trends and Issues in Logistics and Transportation. Technical report, March 2005.

[2] J. Bacon, A. Beresford, D. Evans, D. Ingram, N. Trigoni, A. Guitton, and A. Skordylis. TIME: An open platform for capturing, processing and delivering transport-related data. In *Proc. of the 5th IEEE Consumer Comm. and Netw. Conf. (CCNC)*, pages 687–691, 2008.

[3] R. Baldoni, R. Beraldi, S. T. Piergiovanni, and A. Virgillito. Measuring notification loss in publish/subscribe communication systems. In *Proc. of 10th IEEE Pacific Rim International Symposium on Dependable Computing*, pages 84–93, 2004.

[4] R. Baldoni, R. Beraldi, S. T. Piergiovanni, and A. Virgillito. On the modelling of publish/subscribe communication systems. *Concur. and Comput.: Pract. and Exper.*, 17(12):1471–1495, Oct. 2005.

[5] G. Bricconi, E. D. Nitto, and E. Tracanella. Issues in analyzing the behavior of event dispatching systems. In *Proc. of 10th Intl. Workshop on Software Specification and Design*, pages 95–103, 2000.

[6] I. Burcea and H.-A. Jacobsen. L-ToPSS – Push-oriented location-based services. In *Proc. of the 4th VLDB Workshop on Techn. for E-Services*, pages 131–142, 2003.

[7] A. Carzaniga. *Architectures for an Event Notification Service Scalable to Wide-area Networks*. PhD thesis, Politecnico di Milano, Milano, Italy, Dec. 1998.

[8] S. Castelli, P. Costa, and G. P. Picco. Modeling the communication costs of content-based routing: The case of subscription forwarding. In *Proc. of the Inaugural Conference on Distributed Event-Based Systems (DEBS'07)*, pages 38–49, June 2007.

[9] G. Cugola and J. E. M. de Cote. On introducing location awareness in publish-subscribe middleware. In *25th IEEE Intl. Conf. on Distributed Computing Systems Workshops*, pages 377–382, 2005.

[10] P. Eugster, P. Felber, R. Guerraoui, and A.-M. Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys*, 35(2):114–131, 2003.

[11] F. He, L. Baresi, C. Ghezzi, and P. Spoletini. Formal analysis of publish-subscribe systems by probabilistic timed automata. In *27th IFIP WG 6.1 Intl. Conf. on Formal Techniques for Networked and Distributed Systems*, volume 4574 of *LNCS*, pages 247–262, 2007.

[12] A. Hinze, K. Sachs, and A. Buchmann. Event-based applications and enabling technologies (keynote). In *Proc. of DEBS 2009*, July 2009.

[13] L. Kleinrock. *Queuing Systems; Theory*, volume 1. John Wiley and Sons, New York, 1975.

[14] S. Kounev, K. Sachs, J. Bacon, and A. Buchmann. A methodology for performance modeling of distributed event-based systems. In *Proc. of the 11th IEEE Intl. Symposium on Object/Component/Service-oriented Real-time Distributed Computing*, pages 13–22, May 2008.

[15] A. Medina, A. Lakhina, I. Matta, and J. Byers. BRITE: An approach to universal topology generation. In *Proceedings of the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS'01)*, pages 346–353. IEEE Computer Society, Aug. 2001.

[16] G. Mühl, L. Fiege, and P. Pietzuch. *Distributed Event-Based Systems*. Springer-Verlag, July 2006.

[17] G. Mühl, A. Schröter, H. Parzyjegla, S. Kounev, and J. Richling. Stochastic analysis of hierarchical publish/subscribe systems. In *Proceedings of the 15th European Conference on Parallel Processing (Euro-Par 2009)*, LNCS. Springer Verlag, Aug. 2009.

[18] S. Oh, S. L. Pallickara, S. H. Ko, J. H. Kim, and G. Fox. Cost model and adaptive scheme for publish/subscribe systems on mobile grid environments. In *5th Intl. Conf. on Computational Science*, volume 3516 of *LNCS*, pages 275–278, 2005.

[19] B. Oki, M. Pfluegl, A. Siegel, and D. Skeen. The information bus: an architecture for extensible distributed systems. In *SOSP '93: Proceedings of the fourteenth ACM symposium on Operating systems principles*, pages 58–68, New York, NY, USA, 1993. ACM.

[20] P. R. Pietzuch. *Hermes: A Scalable Event-Based Middleware*. PhD thesis, Computer Laboratory, University of Cambridge, February 2004.

[21] K. Sachs, S. Kounev, J. Bacon, and A. Buchmann. Performance evaluation of message-oriented middleware using the SPECjms2007 benchmark. *Performance Evaluation*, 66:410–434, Aug. 2009.

[22] A. Schröter, D. Graff, G. Mühl, J. Richling, and H. Parzyjegla. Self-optimizing hybrid routing in publish/subscribe systems. In C. Bartolini and L. P. Gaspary, editors, *20th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM 2009)*, volume 5841 of *LNCS*, pages 111–122, Berlin, Germany, Oct. 2009. Springer-Verlag.

[23] T. Sivaharan, G. S. Blair, and G. Coulson. GREEN: A Configurable and Re-configurable Publish-Subscribe Middleware for Pervasive Computing. In *OTM Conf. (1)*, volume 3760 of *LNCS*, pages 732–749, 2005.