Dimension-Based Subscription Pruning for Publish/Subscribe Systems

Sven Bittner & Annika Hinze
Department of Computer Science
University of Waikato, Hamilton, New Zealand
{s.bittner, a.hinze}@cs.waikato.ac.nz

Abstract

Subscription pruning has been proven as valuable routing optimization for Boolean subscriptions in publish/subscribe systems. It aims at optimizing subscriptions independently of each other and is thus applicable for all kinds of subscriptions regardless of their individual and collective structures. The original subscription pruning approach tries to optimize the event routing process based on the expected increase in network load. However, a closer look at pruning-based routing reveals its further applicability to optimizations in respect to other dimensions.

In this paper, we introduce and investigate subscription pruning based on three dimensions of optimization: network load, memory usage, and system throughput. We present the algorithms to perform prunings based on these dimensions and discuss the results of a series of practical experiments. Our analysis reveals the advantages and disadvantages of the different dimensions of optimization and allows conclusions about the suitability of dimension-based pruning for different application requirements.

1. Introduction

One of the major challenges for distributed publish/subscribe (p/s) systems is to improve event routing. To enhance the routing process, various optimizations have been proposed. They range from subscription covering [13] and subscription merging [12], which are suitable for conjunctive subscriptions, to subscription pruning [4], having an optimization potential for all kinds of Boolean subscriptions. Whereas the former two optimizations make strong assumptions on subscriptions, e.g., the existence of similarities or other, more general relationships [5], subscription pruning optimizes subscriptions independently of each other. Thus, subscription pruning is a valuable routing optimization applicable for general-purpose p/s systems [5].

Generally, we can optimize the routing load in p/s systems based on different dimensions (parameters). According to the chosen dimension, the event routing process is improved in respect to this parameter. In this paper, we consider optimizations on three important parameters affecting distributed p/s systems: (i) network load created by event routing, (ii) memory usage required for routing tables, and (iii) event throughput achieved by the overall system. These parameters determine scalability and efficiency of p/s systems, two of their major implementation concerns [10].

The original subscription pruning approach [4] bases its optimization on the expected influence of optimized event routing on network load (cf. Sect. 2.2). This is a suitable solution to the optimization problem because the number of additionally forwarded events (due to optimizing) affects both network usage (and thus scalability) and throughput (filtering on more messages). But we can conversely base pruning on the other two dimensions (memory usage and throughput). Each of these options targets different application requirements and results in a distinctive system behavior: Compared to the other optimizations, network-based pruning causes the least increase in network load, memory-based pruning shows the strongest reduction in routing table sizes, and throughput-based pruning results in the most efficient filtering. Thus, according to application requirements we can choose the most suitable optimization. We are also able to dynamically adjust our optimization based on current system parameters, e.g., if the number of subscriptions increases strongly, we use memory-based pruning; bandwidth limitations suggest to apply network-based pruning.

In this paper, we present an investigation of subscription pruning based on the three dimensions network load, memory usage, and system throughput. We start by giving background information on the utilized p/s model, the general subscription pruning approach, and related optimizations in Sect. 2. Section 3 introduces our three dimensions of optimization, theoretically analyzes their effects on routing, and proposes algorithms and heuristics to quantify these influences. Our practical experiments as well as their evaluation is presented in Sect. 4. Finally, we conclude this paper and present ideas for future work in Sect. 5.

2. Publish/subscribe and subscription pruning

In this section, we give background information on the p/s model assumed in this paper (Sect. 2.1) and the general subscription pruning approach (Sect. 2.2). We also analyze related optimization approaches in Sect. 2.3.

2.1. Background on publish/subscribe

For this paper, we assume the attribute-value pair p/s model: Each event message consists of a set of attribute-value pairs presenting its content. A subscription s_x is an arbitrary Boolean filter expression; variables of this expression are referred to as predicates and specify conditions on event messages in form of attribute-operator-value triples. Subscriptions can be represented by a tree structure [2].

A p/s system filters all incoming event messages. Whenever the Boolean filter expression of a subscription s_x evaluates to *true* on an event message (i.e., s_x is fulfilled by the message/the event is matching s_x), the respective subscriber is notified. The filtering task is obtained by filtering algorithms, e.g., [2] to directly filter on Boolean subscriptions.

A distributed p/s system consists of a network of brokers [1]. Subscribers register subscriptions with one of these brokers (referred to as local client and broker). We assume acyclic broker connections; extensions to dynamic reconfigurations and other topologies exist in literature [7, 14]. Brokers exchange information about subscriptions to allow for the selective routing of events to relevant neighbors (subscription forwarding [6]). To improve event routing, we can apply routing optimizations. These optimizations modify routing entries to improve the routing process.

2.2. Background on subscription pruning

Subscription pruning [4] works for arbitrary Boolean subscriptions. It optimizes the event routing by considering subscriptions independently of each other. The basic idea of pruning is to generalize subscriptions s_x, i.e., the pruned subscription is fulfilled by a superset of the events fulfilling s_x. This generalization is obtained by removing parts of the subscription tree. To ensure correct routing, pruning is only applied to subscriptions from non-local clients. These subscriptions are pruned to achieve an optimization; the pruning order is determined by heuristics (Sect. 3), deciding on the next pruning. Unsubscriptions do not require a specialized handling compared to un-optimized routing.

Pruning a subscription influences various aspects of event routing: Firstly, subscription trees become smaller and thus routing entries require less memory. Secondly, subscriptions become less complex resulting in more efficient event filtering. Finally, a pruned subscription is fulfilled by more event messages. Hence, the network load increases and, in a distributed setting, neighbor brokers need to filter on more messages (post-filtering). As long as the number of additionally forwarded event messages remains reasonably small, the overall throughput of a p/s system increases. Thus, subscription pruning improves the efficiency of event routing, decreases the sizes of routing tables, and only slightly increases the network load for event routing.

Because subscription pruning affects all of the mentioned parameters, we are able to utilize this optimization to influence these three dimensions to different degrees. We will show how to achieve this behavior in Sect. 3.

2.3. Other routing optimizations

The main drawback of other optimizations than pruning is their restriction to conjunctive subscriptions[1]. Furthermore, these optimizations rely on relationships among subscriptions. Thus, they are only applicable for restricted applications and lack the support for general-purpose systems.

Covering is an optimization exploiting the fact that some subscriptions are more general than others, i.e., there exists a subset relationship among their sets of fulfilling messages. The covering optimization has been applied in various p/s systems, e.g., PADRES [11], REBECA [13], SIENA [6], and XROUTE [8]. They only support conjunctive subscriptions. Systems computing coverings for Boolean subscriptions are not described in literature. Because covering is directly aligned with the mentioned subset relationship, it does not allow for optimizations based on different dimensions. We can use pruning as an extension of covering.

The other existing routing optimization is subscription *merging* [9, 11, 13, 15], which tries to summarize several subscriptions. Again, this optimization is restricted to conjunctive subscriptions. Finding a merger satisfying given conditions has been proven as NP-hard problem [9]. The general challenges one has to face when applying merging are when, what, and how to merge [11]. The answers to these problems influence the optimization potential of subscription merging, i.e., differing solutions might optimize the routing load according to different dimensions. We can use subscription pruning to solve the merging problem [5].

In the next section, we present how to utilize subscription pruning for optimizations based on different dimensions.

3. Dimension-based subscription pruning

According to the requirements of applications, we should optimize the routing load based on different dimensions. In this paper, we consider optimizations based on network load (Sect. 3.1), memory usage (Sect. 3.2), and system throughput (Sect. 3.3). We show how to use our heuristics for practical pruning in Sect. 3.4.

[1]Subscriptions in DNF, e.g., [7], do not eliminate this disadvantage.

3.1. Network load: selectivity

Network-based subscription pruning aims at minimizing the amount of additionally routed events. Thus, to decide on the next pruning at any system time, we need to calculate its effects on the network load. We generally choose to prune the subscription that is supporting a pruning operation whose pruning minimally increases the network load.

To quantify the influence on network load, we utilize the selectivity of subscriptions s_x. Using an estimation $sel^{\approx}(s_x)$ instead of the actual selectivity $sel(s_x)$ allows its time and space efficient calculation [4]. A suitable estimation contains three values, the average $sel^{avg}(s_x)$, the minimal $sel^{min}(s_x)$, and the maximal possible $sel^{max}(s_x)$ selectivity. To heuristically estimate the effects of a pruning of subscription s_x leading to s_y, we calculate the maximum of the differences of all three components of our estimation:

$$\begin{aligned}\Delta^{\approx}_{sel}(s_x, s_y) \quad = \max(\quad & sel^{min}(s_y) - sel^{min}(s_x), \\ & sel^{avg}(s_y) - sel^{avg}(s_x), \\ & sel^{max}(s_y) - sel^{max}(s_x))\end{aligned}$$

We call the result the *estimated selectivity degradation* $\Delta^{\approx}_{sel}(s_x, s_y)$; it estimates the decrease in selectivity of a pruned subscription. The actual degradation $\Delta_{sel}(s_x, s_y)$ is situated between 0 and $sel^{max}(s_y) - sel^{min}(s_x)$.

In practice, s_x in $\Delta^{\approx}_{sel}(s_x, s_y)$ refers to the originally registered subscription. Comparing to the unpruned subscription s_x, even after several prunings, is advantageous because the overall selectivity degradation is incorporated in our calculations. Otherwise, several small degradation values as the result of subsequently pruning s_x might appear as a reasonable choice; however, adding them up reveals the total degradation of s_y. Our approach of choosing the unpruned subscription s_x for calculations avoids this problem.

3.2. Memory usage: subscription size

Memory-based subscription pruning targets at decreasing the sizes of routing tables as effect of the optimization. Thus, we should always perform the pruning operation resulting in the biggest reduction in memory usage.

Again, we use an estimation describing the change in the memory requirements $mem(s_x)$ of a subscription s_x. Our memory estimation $mem^{\approx}(s_x)$ only considers the sizes of subscriptions themselves. We heuristically estimate the improvement in respect to memory usage when pruning s_x to s_y as the difference in the sizes of their subscription trees:

$$\Delta^{\approx}_{mem}(s_x, s_y) \quad = \quad mem^{\approx}(s_x) - mem^{\approx}(s_y)$$

We refer to the result as the *estimated memory improvement* $\Delta^{\approx}_{mem}(s_x, s_y)$; it directly describes the reduction in memory (in bytes) to store the pruned subscription tree. The actual decrease in memory usage $\Delta_{mem}(s_x, s_y)$ is always greater than our estimation because the sizes of all indexing structures are reduced by pruning operations.

In contrast to network-based pruning, in $\Delta^{\approx}_{mem}(s_x, s_y)$, s_x refers to the subscription before it is pruned to s_y. This allows us to quantify the direct influence of each pruning on routing tables. Using the unpruned subscription would mean to consider prunings as worthwhile even if a strong decrease in memory is only the result of an early pruning. Incorporating the optimization of each step avoids this.

We are aware of the fact that we always experience the strongest reduction in memory usage if we prune the largest subtree of a subscription. Thus, we additionally restrict our pruning algorithm (other restrictions can be found in [4]) to consider a pruning of node n as valid if and only if there exists no valid pruning in the subtree rooted in n.

3.3. Throughput: computation complexity

The purpose of throughput-based pruning is to increase the filter efficiency when using this optimization. Thus, throughput-based pruning has to consider the filtering algorithm used within the system. This allows for the execution of prunings based on their influence on filter efficiency.

Considering the non-canonical algorithm presented in [2], we can heuristically derive filter efficiency by the minimal number of fulfilled predicates $p_{min}(s_x)$ [2]. This parameter describes the number of predicates that is required to lead to a fulfilled subscription s_x. If it holds $p_{min}(s_x) > p_{min}(s_y)$, s_x needs to be evaluated less frequent than s_y, assuming they share the same predicates.

So, we can heuristically estimate the improvement in respect to throughput when pruning s_x leading to s_y as:

$$\Delta^{\approx}_{eff}(s_x, s_y) \quad = \quad p_{min}(s_y) - p_{min}(s_x)$$

We name $\Delta^{\approx}_{eff}(s_x, s_y)$ the *estimated throughput improvement*. Because we do not introduce new predicates when pruning (we remove some of them), $\Delta^{\approx}_{eff}(s_x, s_y) > 0$ means that we need to evaluate the pruned subscription tree less frequent. Generally, the greater $\Delta^{\approx}_{eff}(s_x, s_y)$, the rarer the pruned subscription is evaluated. Hence, filtering becomes more efficient, which means a higher system throughput.

As in network-based pruning, s_x in $\Delta^{\approx}_{eff}(s_x, s_y)$ refers to the originally registered subscription. This allows us to incorporate the influence of all performed prunings into our heuristic. Otherwise, a strong decrease in $p_{min}(s_x)$ would only be regarded in one pruning operation; our heuristic, conversely, considers this case in all subsequent prunings.

3.4. Practical dimension-based pruning

When optimizing by pruning, we always perform the most effective pruning operation based on the currently used dimension of optimization. Comparing the heuristic ratings of two choices of pruning might not allow us to decide on the more effective one. This happens if both choices result in the same degradation/improvement in respect to the applied optimization. To circumvent these cases, we should refer to another dimension to decide on the most effective pruning. The order of dimensions depends on current application requirements. For our experiments, we use the following orders of heuristics for our pruning options:

Network-based pruning: $\Delta^{\approx}_{sel}, \Delta^{\approx}_{\mathit{eff}}, \Delta^{\approx}_{mem}$

Memory-based pruning: $\Delta^{\approx}_{mem}, \Delta^{\approx}_{sel}, \Delta^{\approx}_{\mathit{eff}}$

Throughput-based pruning: $\Delta^{\approx}_{\mathit{eff}}, \Delta^{\approx}_{sel}, \Delta^{\approx}_{mem}$

This procedure should allow for the determination of the most effective of several pruning options. If all heuristics show the same value, the chosen option is arbitrary.

We can determine the most effective pruning by utilizing a priority queue: Initially, we insert the most effective pruning of each subscription into our queue. Using the described orders of dimensions places the most effective pruning on top: $\Delta^{\approx}_{sel}$ is ordered in a descending manner, $\Delta^{\approx}_{\mathit{eff}}$ and $\Delta^{\approx}_{mem}$ in an ascending one. After performing the overall most effective pruning, the new most effective pruning of the just optimized subscription is reinserted into the queue.

By using this scheme, we can apply routing optimizations step by step. Either we perform a specified number of optimizations, or we optimize until a given degradation/improvement is reached. In our experiments, presented in the next section, we investigate the influence of performing different amounts of pruning operations in batch.

4. Experiments and evaluation

We now present an evaluation of a series of experiments undertaken to analyze and compare subscription pruning based on the three dimensions. In Sect. 4.1, we show our results and describe their illustration within this paper. Their detailed discussion and analysis is presented in Sect. 4.2.

As application area, we have chosen an online-auction scenario. Subscriptions conform to three classes typical for online book auctions [4]. Event messages also follow the characteristic distributions for these auctions [3]. We register 200,000 subscriptions in our experiments. Our results represent the average values for publishing 100,000 events.

We analyze a centralized and a distributed setting. The former allows for conclusions about the effects of dimension-based pruning in general. The latter setting shows the effects on the distributed system; it involves five brokers connected as a line. We analyze network load, memory usage, and event throughput. In our experiments, we use machines equipped with a 2 GHz processor and 512 MB of main memory, and connected by a 10 Mbps network.

4.1. Experimental results

We present our experimental results in Fig. 1. Abscissae show the proportional number of performed prunings and range from 0, describing the un-optimized situation, to 1, i.e., any other pruning removes a complete subscription. We always show the behaviors of our three heuristics, described by the indexes of the labels of the shown curves.

Figures 1(a) and 1(d) show our results regarding time efficiency. The behavior in the centralized setting is depicted in Fig. 1(a); Fig. 1(d) represents the distributed system. Ordinates show the average filtering time per event in seconds.

The expected network load (centralized) is illustrated in Fig. 1(b). The ordinate describes the proportional number of matching events. The actual load (distributed) is shown in Fig. 1(e). There, the ordinate represents the proportional increase in routed events compared to the un-optimized situation, i.e., 1.0 means a doubled number of event messages.

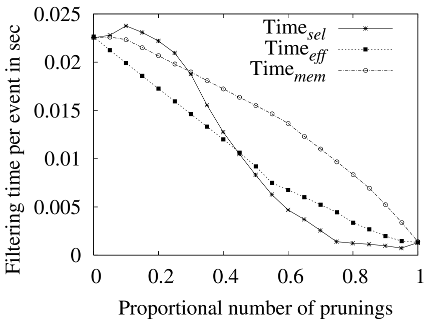
Figure 1(c) (central) and Fig. 1(f) (distributed) show the behavior of our last parameter, memory usage. Ordinates present the proportional decrease in predicate/subscription associations; Fig. 1(c) considers all subscriptions, Fig. 1(f) only those from non-local subscribers, e.g., 0.5 states that the number of registered predicates is reduced by 50%.

4.2. Discussion of results

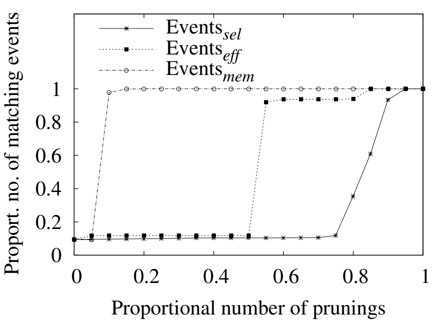
The intention of our heuristics is to achieve optimizations in respect to different parameters. We now want to investigate whether our heuristics influence the quantitatively measured parameters to different degrees and whether these behaviors align with our expectations (Sect. 1).

Analyzing the throughput in a single broker (Fig. 1(a)), throughput-based pruning results in the most time efficient filtering for up to 43% of all optimizations. Then, network-based pruning shows a higher throughput, i.e., smaller filtering times. Worst time efficiency is achieved by memory-based pruning. These results align with our expectations except of network-based pruning emerging as the most efficient solution. This behavior results out of our throughput-based heuristic: We consider $p_{min}(s_x)$ as the only factor influencing throughput. But, the selectivity of predicates does also have an effect: Pruning general predicates results in less fulfilled predicates per subscription. This, in turn, yields to less subscription evaluations. This pruning of little selective predicates is obtained by network-based pruning. After a certain amount of prunings this influence of selectivities becomes more worthwhile to consider than $p_{min}(s_x)$.

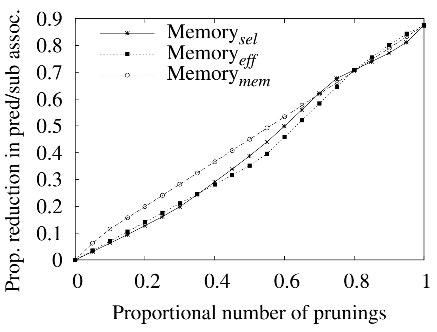
For the expected network load in the centralized setting (Fig. 1(b)), the best results (least routed messages) are achieved by the network-based followed by the throughput and memory-based heuristics. In network-based pruning, up to 75% of all prunings, the amount of messages increases



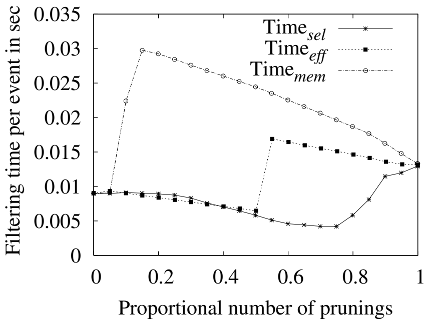
(a) Time efficiency (centralized).



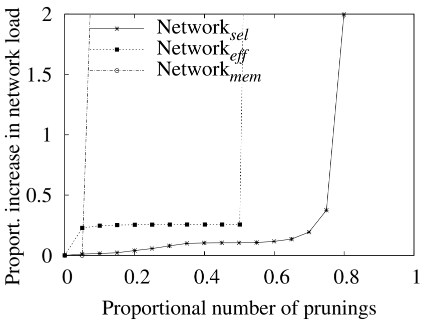
(b) Expected network load (centralized).



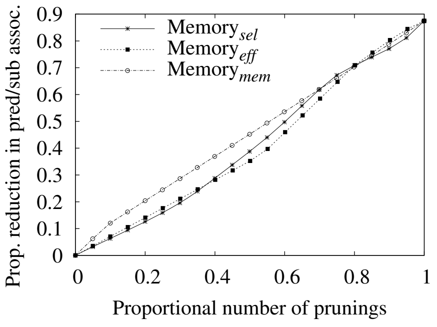
(c) Memory usage (centralized).



(d) Time efficiency (distributed).



(e) Actual network load (distributed).



(f) Memory usage (distributed).

Figure 1. Time efficiency, network load, and memory usage in both settings.

only slightly. For throughput-based pruning, the sharp bend in the curve occurs at 50% of prunings. The memory-based optimization does not consider network load (approx. 5%).

For our last parameter, memory usage, memory-based pruning shows the best results (cf. Fig. 1(c) and 1(f)). The other two heuristics reveal nearly the same influence; network is slightly more effective than throughput-based pruning. However, the benefit of our memory-based optimization is not that significant: Memory is reduced by at most 10% more than for the other heuristics. After 70% of prunings, the reduction in memory using any heuristic is similar. Though, memory-based pruning is advantageously to utilize for unpruned subscriptions if the overall pruning goal is to reduce memory usage by little pruning operations. The influence of all heuristics on memory usage is similar in the centralized (Fig. 1(c)) and the distributed (Fig. 1(f)) setting.

The throughput in the distributed system is not only influenced by the registered subscriptions but also by additionally routed and post-filtered messages. This is reflected in our time efficiency measurements (Fig. 1(d)): The overall best filter efficiency is achieved by our network-based heuristic (4.2 ms per message). For throughput-based pruning, it is 6.5 ms (network-based filters 35% faster). Looking at the overall behavior of these heuristics, we realize a similar developing as in the centralized case: Initially, throughput-based pruning results in the most efficient filtering. After several prunings, network-based pruning shows the highest throughput. The reason is again the inaccuracy of our throughput-based heuristic. The difference between network and throughput-based pruning is not as significant as in the central setting because additionally routed events need to be sent, received, and filtered in several brokers. Memory-based pruning shows no throughput improvement.

The network load in the distributed setting (cf. Fig. 1(e)) performs as predicted: Network-based pruning only slightly increases additionally routed messages. At 75% of all prunings (sharp bend in the curve), the network load is increased by 37%. For throughput-based pruning, the sharp bend occurs at 50%. There, network utilization has grown by 26%; for network-based pruning, at this point the load increased by 10%. Memory-based pruning does not consider the network utilization: The sharp bend already occurs at 5%.

Altogether, our experiments prove that all three heuristics optimize according to their target dimension: Network-based pruning affects the network load to the smallest extend. Additionally, it supports the largest number of pruning operations (75% of all possible) without strongly increasing the network load (increase by 37%). Memory-based pruning results in the strongest decrease in routing table sizes. It is especially useful for unpruned subscriptions; sizes of routing tables are up to 10% smaller than for the other optimizations after the same amount of prunings. Throughput-based pruning results in the highest filter efficiency by requiring the smallest number of prunings, when applied to unpruned subscriptions (18% higher after 35% of prunings).

However, for the distributed setting network-based pruning is the best dimension of optimization. This is due to the strong impact of additionally routed events on system throughput: Even if filtering in a single broker is more efficient, the overall throughput might degrade. Thus, network-based pruning achieves the best overall result regarding efficiency with an improvement of 53% compared to un-optimized routing. At the same time, network load increases by only 37% and memory usage is reduced by 67%.

The reduction in space usage when applying memory-based pruning is not as significant as expected. Due to the strong decrease in throughput, using more space (and thus less time) efficient algorithms might become a better choice.

5. Conclusion and future work

In this paper, we have investigated the behavior of event routing in p/s systems when applying different approaches of routing optimizations. For our analysis, we have identified three dimensions of parameters characterizing p/s systems: (i) network load created by event routing, (ii) memory usage required for routing tables, and (iii) event throughput achieved by the overall system. We have determined subscription pruning as a routing optimization allowing for an optimization based on these three dimensions.

To perform dimension-based pruning, we have developed heuristics estimating the effects of prunings on the respective parameter. By applying these heuristics, we are able to decide on the most effective pruning operation from all registered subscriptions. This allows us to perform prunings according to their influence on the three parameters.

We ran a series of experiments to analyze and compare subscription pruning based on the three dimensions. We have been able to show that network-based pruning causes the least increase in network load, memory-based pruning shows the strongest reduction in routing table sizes, and throughput-based pruning results in the most efficient filtering. However, the advantages of memory-based pruning are not as significant as expected; memory usage is reduced by at most 10% more than in our other optimizations. This effect is only achieved when optimizing on unpruned subscriptions. Also throughput-based pruning performs only slightly better than our network-based optimization; this effect only holds up to around 35% of all possible prunings.

Network-based pruning achieves the overall best results: Compared to un-optimized routing, it improves filter efficiency by 53%, and, at the same time, it reduces the memory requirements for routing tables by 67%. These beneficial behavior increases the network load by only 37%. Hence, when aiming at general-purpose p/s systems, network load is the favorable dimension of optimization. Specialized system, although, might optimize based on one of the other dimensions to achieve their specific optimization goals.

In the future, we want to further improve our heuristic for the network-based optimization. We also plan to investigate the question of how to dynamically determine the number of pruning operations leading to the best overall optimization.

References

[1] G. Banavar, T. Chandra, B. Mukherjee, J. Nagarajarao, R. E. Strom, and D. C. Sturman. An Efficient Multicast Protocol for Content-based Publish-Subscribe Systems. In *Proceedings of ICDCS '99*, pages 262–272, USA, June 1999.

[2] S. Bittner and A. Hinze. A Detailed Investigation of Memory Requirements for Publish/Subscribe Filtering Algorithms. In *Proceedings of CoopIS 2005*, pages 148–165, Agia Napa, Cyprus, October 31–November 4 2005.

[3] S. Bittner and A. Hinze. Event Distributions in Online Book Auctions. Technical Report 03/2006, Comp. Sc. Department, Waikato University, February 2006.

[4] S. Bittner and A. Hinze. Pruning Subscriptions in Distributed Publish/Subscribe Systems. In *Proceedings of ACSC 2006*, Hobart, Australia, January 16–19 2006.

[5] S. Bittner and A. Hinze. Subscription Tree Pruning: A Structure-Independent Routing Optimization for General-Purpose Pub/Sub Systems. Technical Report 01/2006, Comp. Sc. Department, Waikato University, January 2006.

[6] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Design and Evaluation of a Wide-Area Event Notification Service. *ACM Transactions on Computer Systems*, 19(3):332–383, 2001.

[7] A. Carzaniga, M. J. Rutherford, and A. L. Wolf. A Routing Scheme for Content-Based Networking. In *Proceedings of INFOCOM 2004*, Hong Kong, China, March 7–11 2004.

[8] R. Chand and P. A. Felber. A Scalable Protocol for Content-Based Routing in Overlay Networks. In *Proceedings of NCA 2003*, pages 123–130, USA, April 16–18 2003.

[9] A. Crespo, O. Buyukkokten, and H. Garcia-Molina. Query Merging: Improving Query Subscription Processing in a Multicast Environment. *IEEE TKDE*, 15(1):174–191, 2003.

[10] F. Fabret, A. Jacobsen, F. Llirbat, J. Pereira, K. Ross, and D. Shasha. Filtering Algorithms and Implementation for Very Fast Publish/Subscribe Systems. In *Proceedings of SIGMOD 2001*, pages 115–126, USA, May 2001.

[11] G. Li, S. Hou, and H.-A. Jacobsen. A Unified Approach to Routing, Covering and Merging in Publish/Subscribe Systems based on Modified Binary Decision Diagrams. In *Proceedings of ICDCS '05*, pages 447–457, USA, June 2005.

[12] G. Mühl. *Large-Scale Content-Based Publish/Subscribe Systems*. PhD thesis, TU Darmstadt, September 2002.

[13] G. Mühl and L. Fiege. Supporting Covering and Merging in Content-Based Publish/Subscribe Systems: Beyond Name/Value Pairs. *IEEE DSOnline*, 2(7), 2001.

[14] G. P. Picco, G. Cugola, and A. L. Murphy. Efficient Content-Based Event Dispatching in the Presence of Topological Reconfiguration. In *Proceedings of ICDCS '03*, pages 234–243, Rhode Island, USA, May 19–22 2003.

[15] Y.-M. Wang, L. Qiu, C. Verbowski, D. Achlioptas, G. Das, and P. Larson. Summary-based Routing for Content-based Event Distribution Networks. *ACM SIGCOMM Computer Communication Review*, 34(5):59–74, 2004.