Modelling Performance Optimizations for Content-based Publish/Subscribe

(Work-In-Progress Paper)

Alex Wun†
wun@eecg.utoronto.ca

Hans-Arno Jacobsen†‡
jacobsen@eecg.utoronto.ca

† Department of Electrical and Computer Engineering
‡ Department of Computer Science
University of Toronto

ABSTRACT

Content-based Publish/Subscribe (CPS) systems can efficiently deliver messages to large numbers of subscribers with diverse interests and consequently, have often been considered an appropriate technology for large-scale, event-based applications. In fact, a significant amount of existing research addresses the issue of providing scalable CPS services [3, 8, 7, 11]. In these approaches, scalability and high performance matching is often achieved by taking advantage of similarities between subscriptions. However, even though such optimization techniques are widely used, no model has been developed yet to capture them. Such an abstraction would allow CPS matching algorithms to be studied, analyzed, and optimized at a more fundamental and formal level. In this work-in-progress paper, we present the initial results of our work towards modelling and analyzing matching optimizations frequently used by CPS systems. Using our proposed model, we find that probabilistically optimal CPS matching is possible in certain types of subscription sets and that there is also a non-obvious upper bound on the expected cost of some subscription sets. We also provide experimental results that support the model proposed and studied in this paper.

Categories and Subject Descriptors

H.4.m [**Information Systems Applications**]: Miscellaneous

General Terms

Performance, Theory

Keywords

Publish/Subscribe, Performance analysis

DEBS '07 , June 20-22, 2007 Toronto, Ontario, Canada

1. INTRODUCTION

The importance of Content-based Publish/Subscribe (CPS) is becoming apparent as CPS and related technologies continue to surface in industry [1, 2]. In these large-scale, event-based applications, it is often necessary to notify large numbers of subscribers with diverse interests efficiently. In CPS systems, message delivery occurs based on the results of *matching* algorithms that compute subsets of interested subscribers from all possible subscribers. As such, scalability is a key concern for CPS systems. Fortunately, existing CPS research has been able to provide scalable content-based messaging using a variety of different approaches and optimizations [3, 8, 7, 11]. In these approaches, scalability and high performance matching is often achieved by taking advantage of similarities that occur between subscriptions in a set. However, there is little existing work attempting to model or abstract these subscription set semantics even though such work would allow CPS matching to be studied, analyzed, and optimized at a more fundamental level. This work is important because it contributes to formalizing the domain of CPS but is also difficult because the languages and features that define CPS have not been standardized. In this paper, we present work-in-progress that attempts to abstract subscription set semantics and analyze the resulting model independent of specific CPS implementations and data structures. More specifically, we study CPS matching in terms of expected matching costs and find that probabilistically optimal matching is possible in certain subscription sets. Furthermore, we present experimental results to support the accuracy of the model and potential applications of the model.

The contributions of this paper are: (1) a summary of computational complexities for probabilistically optimal CPS matching, (2) an analysis of the expected cost behaviour of CPS matching, and (3) experimental results supporting the model presented in this paper.

The remainder of the paper is organized as follows: Section 2 presents the details of the model we study, Section 3 presents the computational complexities of probabilistically optimal CPS matching, Section 4 presents the expected performance costs of CPS matching, Section 5 presents the results of experiments run using our own CPS implementation, Section 6 discusses related work, and Section 7 concludes with future work.

$\bar{\psi_1} \rightarrow \bar{S_1} \wedge \cdots \wedge \bar{S_m}$

$\cdots$

$\bar{\psi_k} \rightarrow \bar{S_1} \wedge \cdots \wedge \bar{S_m}$ $\quad\quad$ $S_1 \vee \cdots \vee S_m \rightarrow \psi_L$

(a) Cluster topology (b) Link-group topology

Figure 1: Subscription topology expressions

2. SUBSCRIPTION SETS

Our studies are based on a subscription language that is widely used in most existing CPS systems [3, 4, 8, 7, 11]. In this language, publications are sets of attribute/value pairs (tuples), while subscriptions are conjunctions of predicates specifying constraints over attributes in publications. More formally, a publication P is defined as a set $P = \{\alpha_1 \ldots \alpha_n\}$, where each tuple $\alpha_i = (a_i, v_i)$ is an attribute a_i and its associated value v_i. A subscription S is then defined as $S \equiv \phi_1 \wedge \cdots \wedge \phi_n$, where each predicate $\phi_i = f_i(\alpha)$ is a Boolean-valued function on some tuple and $\wedge$ is the logical *and* operator. A publication P *matches* a subscription S iff $\forall \phi_i \in S, \exists \alpha_j \in P | \phi_i(\alpha_j) = true$. Clearly, a subscription S is equivalent to a conjunctive Boolean expression $S(P)$ where each predicate in S maps to a variable in $S(P)$. Therefore, the process of matching a publication P against S by evaluating each predicate is equivalent to determining the truth value of $S(P)$ by evaluating each Boolean variable[1]. Consequently, the number of predicate evaluations needed to compute a matching result depends on the order of predicate evaluations. We refer to the order in which predicates are evaluated as a *matching plan*. An *optimal matching plan* is then a matching plan that evaluates only the minimum number of predicates needed to determine a matching result. An optimal matching plan for a single subscription evaluates false predicates as soon as possible, using the well-known lazy evaluation approach to evaluating Boolean expressions. However, this optimization only applies to matching publications against a single subscription. In practice, publications are matched against *sets* of subscriptions in order to take advantage of similarities between subscriptions and possibly leverage lazy evaluation at the subscription set level[2]. We broadly refer to these similar characteristics as *commonalities* and focus on studying how they affect the performance of CPS matching algorithms.

Note that although we start with a language in which subscriptions are only conjunctive expressions, our subsequent *commonality model* is amenable to capturing subscriptions in DNF form as well.

2.1 Commonality Model

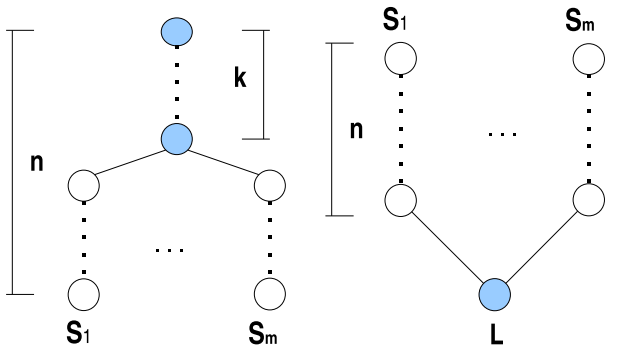
In this section, we describe how commonalities in a subscription set are modelled and also use example optimizations found in existing CPS matching algorithms to support the relevance of the model. In order to capture the performance characteristics of subscription sets, we must necessarily model such commonalities because they represent a fundamental technique used to achieve scalability in existing CPS systems [3, 11, 8, 4, 7]. We express commonalities as necessary conditions for a publication to match a set of subscriptions. That is, given a subscription set $\Phi = \{S_1 \ldots S_m\}$ that share a commonality C, we can say:

$$S_1 \vee \cdots \vee S_m \rightarrow C \quad (1)$$

or equivalently:

$$\bar{C} \rightarrow \bar{S_1} \wedge \cdots \wedge \bar{S_m} \quad (2)$$

$\bar{X}$ is the logical negation of X, $\rightarrow$ is logical implication, and $\vee$ and $\wedge$ are the logical *or* and *and* operators, respectively. The expressions above are referred to as *commonality expressions*. When C appears on the right side of an expression as in (1), we call it a *disjunctive commonality*. When C appears on the left side of an expression as in (2), we call it a *conjunctive commonality*. The commonality itself is equivalent to a Boolean function on a publication P, so $C(P)$ evaluates to *true* or *false* just as $S_i(P)$ evaluates to *true* or *false*. The subscription set Φ and its commonality characteristics can be described with a *subscription topology*, which is simply a set of commonality expressions that involve all subscriptions in Φ. Figure 1 shows two example subscription topologies, in which commonalities are represented by ψ_i. Figure 2 visualizes the same two example subscription topologies as graphs in which each node represents a predicate and shaded nodes are commonalities. Although the two types of commonality expressions are logically equivalent, the distinction is important because the two types of expressions capture two fundamental ways in which a common characteristic between subscriptions of Φ are used to optimize matching algorithms.



(a) Cluster topology (b) Link-group topology

Figure 2: Subscription topology graphs

Disjunctive commonalities appear on the right side of the expression because $C(P)$ is not directly evaluated by a matching algorithm, but determined by evaluating $S_i(P)$ in Φ. This captures optimizations that allow a matching algorithm to skip evaluation of remaining subscriptions in a set

[1]From here on, we will consider single subscriptions to be synonymous with Boolean expressions and predicates to be synonymous with Boolean variables.

[2]For example, tree-based matching engines [3, 11] benefit from similar lazy evaluation style optimizations by "propagating" publications down a subscription tree. In this way, matching can terminate early for a subscription when a false predicate is encountered.

based on the matching results of previous subscriptions. An example of a disjunctive commonality used in practice is per-link matching (as opposed to per-subscription matching) [4, 7]. In CPS systems, a single entity will often issue multiple subscriptions over a single overlay link. However, no matter how many of those subscriptions a given publication matches, only one instance of the publication is generally forwarded back over the link. Therefore, the matching process can terminate as soon as a "forward" or "don't forward" decision can be made. In this example, the commonality between subscriptions in Φ is having originated from the same link. More formally, a subscription set $\Phi = \{S_1 \dots S_m\}$ originating from the same link L has a disjunctive commonality $C(P) = \{$*true* if publication P is to be forwarded over L, *false* otherwise$\}$. Furthermore, subscriptions in DNF form can also be modelled using disjunctive commonalities. Each conjunctive clause becomes a "sub-subscription" while the overall DNF subscription becomes a commonality $C(P) = \{$*true* if $S_i(P) =$ *true* for some i, *false* otherwise$\}$. For disjunctive commonalities in general, matching optimizations are based on determining the value of $C(P)$ with as few predicate evaluations as possible.

In contrast, conjunctive commonalities appear on the left side of the expression because $C(P)$ can be directly evaluated by a matching algorithm. This captures optimizations that allow a matching algorithm to eliminate subscriptions as potential matching candidates, or conversely, select only subscriptions that are potential candidates for matching. Examples of conjunctive commonalities include the use of messages types [13], access predicates [8], and determinants [7]. These algorithms group subscriptions using common characteristics and perform a "pre-matching" phase to eliminate irrelevant or select only relevant subscriptions for matching. Example conjunctive commonalities $C(P)$ in these algorithms are, respectively:

> $C(P) = \{$*true* if publication P is of type T, *false* otherwise$\}$
> $C(P) = \{$*true* if publication P has more than n tuples, *false* otherwise$\}$
> $C(P) = \{$*true* if publication P contains attribute A, *false* otherwise$\}$

Furthermore, conjunctive commonalities also capture optimizations based on re-using matching results between subscriptions since each commonality is only represented once. Many existing CPS matching algorithms try to express only unique predicates in a subscription set so that the cost of evaluating unique predicates is only incurred once [11, 7]. In this case, conjunctive commonalities would be $C(P) = \phi(P)$, where $\phi(P)$ is some common predicate that appears in all subscriptions of the set. It is important to emphasize that conjunctive commonalities can both be actual predicates that explicitly appear in subscriptions as well as subscription characteristics that are not explicitly expressed as predicates (such as the number of predicates, subscription lifetime, etc.).

2.2 Commonalities and Selectivities

Based on the commonality model, we can see that the topology of a subscription set (i.e., existing commonalities and their relationship to subscriptions in the set) determines available opportunities for optimizing the performance of matching. However, recall that the performance cost of any given matching plan for a publication P against a subscription set also depends on the actual truth values of subscriptions and predicates when evaluated on P. For example, suppose we have the subscription set in which all subscriptions originate from the same overlay link and that this is the only commonality. We will refer to this topology as the "link-group" topology from here on. Figure 2(b) visualizes the link-group topology as a graph. Recall that matching terminates for a given publication P once the truth value of the disjunctive commonality $C(P)$ is known. If there is a subscription in the set for which $S(P) =$ *true*, then it is better to evaluate that subscription first since it necessarily determines that $C(P) =$ *true*. Evaluating any predicate not in S is redundant in this case. However, if every subscription has at least one predicate ϕ for which $\phi_i(P) =$ *false*, then it is better to evaluate each of these predicates first to obtain $C(P) =$ *false*. The optimal matching plan changes depending on the truth values of the predicates. In Section 4, we will show how the link-group topology affects the expected cost of matching.

To further illustrate the relationship between commonalities and truth probabilities, suppose we have a subscription set Φ in which a single predicate appears in all subscriptions of Φ. That is, Φ has a conjunctive commonality C common to all subscriptions. In this set, matching terminates for a publication P once the truth values of all subscriptions $S_i(P)$ are known. We will refer to this topology as the "cluster" topology from here on. Figure 2(a) visualizes the cluster topology as a graph. If $C(P) =$ *false*, then it is better to evaluate C first since $S_i(P) =$ *false* $\forall i$ will be the result and matching is complete. However, if $C(P) =$ *true* but each subscription has at least one predicate ϕ for which $\phi(P) =$ *false*, then it is better to evaluate the predicates rather than the commonality first since any other predicate evaluation is redundant in this case. Again, the optimal matching plan is different depending on the truth values of the predicates. Of course, we cannot know the truth values of subscriptions and predicates without being given P. So rather than analyzing exact matching costs with respect to specific publication instances, we look at the probabilistic matching costs of subscription sets when given selectivities for each predicate. The selectivity of a predicate ϕ is simply the probability that $\phi(P) =$ *true* on any given publication P. In practice, selectivities can be estimated based on attribute ranges in subscriptions and attribute distributions in publications. In the domain of databases, selectivity estimation is often done using statistical histograms to track data distribution [10]. A similar notion for publications applies here as well.

3. ON OPTIMAL MATCHING PLANS

Since the performance predictions for a given subscription topology depend on the matching plan used, we first look at whether it is possible to determine a probabilistically optimal matching plan. If so, then this optimal matching plan[3] can be used for performance predictions. As it turns out, finding an optimal matching plan is difficult even for relatively simple subscription topologies. In this section, we begin by presenting topologies for which probabilistically optimal matching plans are known and then proceed to present

[3]For brevity, optimal matching will imply *probabilistically* optimal matching from here on.

topologies for which optimal matching plans remain an open research problem.

First, for a single subscription (or any conjunction of predicates), the optimal matching plan is fairly straight-forward. If each predicate ϕ_i in a subscription is associated with a cost c_i of evaluation and selectivity p_i, then an optimal matching plan evaluates ϕ_i before ϕ_j if $(1-p_i)/c_i \geq (1-p_j)/c_j$. The proof of this was presented in [14] under the context of probabilistic Boolean expressions. Their work was in the context of problem-solving searches and artificial intelligence. In the special case of equal costs, predicates are evaluated in non-decreasing selectivity order, which is consistent with our intuition that false predicates be evaluated earlier.

Next, we consider the link-group topology since it is widely applicable across CPS systems and easily modelled. Because the link-group topology has a single commonality C for which matching terminates once $C(P)$ is known, this particular topology can be written as a DNF Boolean expression. More specifically, $C(P) = S_1(P) \vee \cdots \vee S_m(P) = (\phi_{11} \wedge \cdots \wedge \phi_{1n_1}) \vee \cdots \vee (\phi_{m1} \wedge \cdots \wedge \phi_{mn_m})$. Determining the matching result of this subscription set is then equivalent to evaluating the Boolean expression $C(P)$ where each predicate is a variable with an associated evaluation cost and truth probability. Clearly, matching against a link-group topology parallels the problem of finding an optimal strategy for evaluating probabilistic Boolean expressions. A depth-first algorithm for finding an optimal strategy for DNF expressions in which each variable appears only once was presented in [9] and has complexity $O(n \ln(n))$, where n is the total number of variables (predicates) in the expression. Therefore, we find that an algorithm exists for determining an optimal matching plan for subscription sets in the link-group topology when there are no shared predicates. As we can see, determining an optimal matching plan even for this relatively simple subscription topology is super-linear in the number of predicates. Unfortunately, this depth-first algorithm no longer applies once we consider shared predicates (i.e., conjunctive commonalities) because shared predicates map to variables that appear more than once in the DNF Boolean expression. Therefore, the assumption that makes the depth-first algorithm applicable no longer holds.

We first consider conjunctive commonalities by combining the link-group and cluster subscription topologies and refer to it as the "link-cluster" topology. In this topology, subscriptions all originate from the same link and are additionally allowed to share common predicates. Note that since there is only a single cluster, there may be multiple common predicates, but each common predicate must appear in all subscriptions. Although this topology can still be written as a DNF expression $C(P) = (\psi \wedge \phi_{11} \wedge \cdots \wedge \phi_{1n_1}) \vee \cdots \vee (\psi \wedge \phi_{m1} \wedge \cdots \wedge \phi_{mn_m})$, it has repeated occurrences of the common predicate ψ, so the depth-first algorithm mentioned earlier no longer applies. However, $C(P)$ can be written with the common predicates factored out. Although the resulting Boolean expression $C(P) = \psi \wedge ((\phi_{11} \wedge \ldots \phi_{1n_1}) \vee \cdots \vee (\phi_{m1} \wedge \cdots \wedge \phi_{mn_m}))$ is no longer in DNF, it can be represented as a tree structure where internal nodes are *and*/*or* operators and each leaf is a unique variable (predicate). Given these conditions, the authors in [9] present a dynamic programming algorithm for determining an optimal evaluation strategy in $O(d^2(r+1)^d)$, where d is the number of leaf-parents in the tree and r is the largest number of leaf-siblings in the tree. In the CPS domain, $d = m + 1$ and $r = \max(\max(n), k)$, where m is the number of subscriptions in the subscription set, $\max(n)$ is the maximum number of predicates appearing in any subscription, and k is the number of common predicates. If we assume that $k \leq \max(n)$[4], then $r = max(n)$. Therefore, we find that an algorithm exists for determining an optimal matching plan for a subscription set in the link-cluster topology in $O(m(\max(n)+1)^m)$. However, the algorithm is exponential in the number of subscriptions.

The dynamic programming algorithm also applies to a "multi-cluster-link" subscription topology. In this topology, subscriptions all originate from the same link and are allowed to share common predicates as with a link-group topology. However, common predicates do not necessarily have to appear in all subscriptions of the set. That is, there may be multiple conjunctive commonality clusters in a multicluster-link subscription topology. Using the same argument presented for link-cluster topologies, a Multicluster-Link topology can also be represented as a tree structure. However, we find that the number of leaf parents d in the tree is now dependent on both the number of subscriptions and the number of clusters. That is, $d = m + K$, where K is the number of clusters. Therefore, the algorithm becomes exponential in the *sum* of the number of subscriptions and the number of clusters.

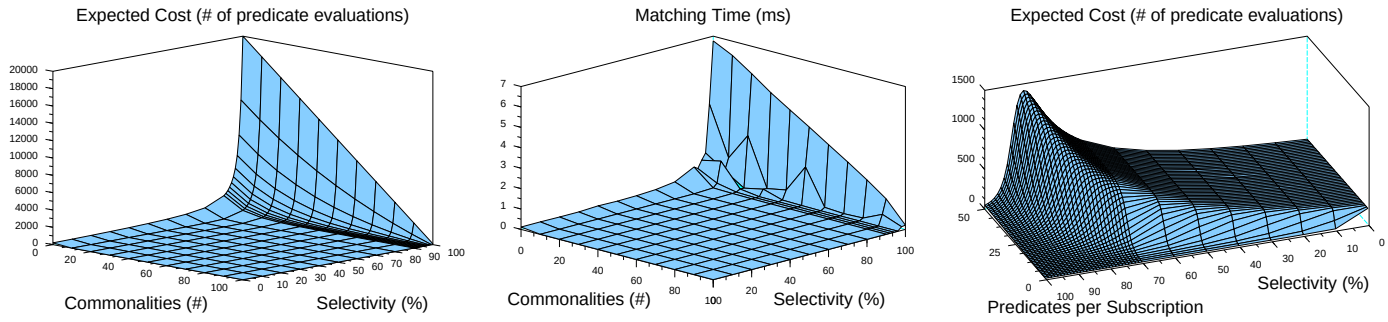
It may be tempting to believe that a subscription set can always be represented as a single Boolean expression. However, this is not generally the case. In particular, the cluster topology (as shown in Figure 2(a)) cannot be expressed as a single Boolean expression. Recall that in a cluster topology, common predicates must appear in all subscriptions, but subscriptions do not necessarily originate from the same overlay link. Therefore, the result of matching is no longer limited to a single Boolean value and is instead, a list of matched subscriptions. The complexity of finding an optimal matching plan in this type of topology is currently unknown and remains an open research problem.

4. MATCHING PERFORMANCE

In Section 3, we summarized the known complexities of determining optimal matching plans for various subscription topologies. We now quantify the actual expected costs of matching against the cluster and link-group topologies. We start with the cluster topology because the link-group topology expected cost is more complex and builds on concepts presented for the cluster topology.

In order to determine these optimal matching plans, subscription topologies had to be transformed into Boolean expressions. Unfortunately, there exist topologies (such as the cluster topology shown in Figure 2(a)) that cannot be transformed into Boolean expressions since the output of matching is a set of matching subscriptions rather than a single Boolean value. Consequently, optimal matching plans are currently unknown for these topologies. However, even though the optimal matching plan is unknown, we can still attempt to predict the performance of these topologies under the assumption that a reasonable heuristic is used. Specifically, we analyze the cluster topology shown in Figure 2(a) based on the optimal matching plan for a conjunction of

[4]If common predicates are the only conjunctive commonality recognized by the matching algorithm, then this is the case.



(a) Cluster (simulation) (b) Cluster (experimental) (c) Link-group (simulation)

Figure 3: Performance costs of CPS matching in different subscription topologies.

predicates.

For a conjunction of n predicates where predicate ϕ_i has cost c_i and selectivity p_i, the expected cost $E(n)$ of evaluating the conjunction is:

$$E(n) = c_1 + p_1c_2 + p_1p_2c_3 + \cdots + \Pi_{i=1}^{n-1} p_i c_n \quad (3)$$

If $c_i = 1, p_i = p, \forall i$, then:

$$E(n,p) = \frac{p^n - 1}{p - 1} \quad (4)$$

Recall that $(1-p_i)/c_i \geq (1-p_{i+1})/c_{i+1}$ holds if an optimal matching plan is used. Based on Equation 3, the expected cost of a cluster subscription topology with k commonalities, m subscriptions, and n predicates[5] in each subscription is:

$$E_{cluster}(k,m,n) = E(k) + \Pi_{i=1}^{k} p_i(mE(n-k)) \quad (5)$$

p_i is the selectivity of the i^{th} commonality. If $c_i = 1, p_i = p, \forall i$, then:

$$E_{cluster}(k,m,n,p) = \frac{p^k - 1}{p - 1} + p^k(m\frac{p^{n-k} - 1}{p - 1}) \quad (6)$$

In Figure 3(a), we plot Equation 6 for $m = 200, n = 100$ and allow k to vary from 0 to n while p varies from 0 to 1.0. We observe that introducing conjunctive commonalities to "high-selectivity" clusters and to clusters with a low number of existing commonalities results in the greatest matching performance gains. That is, conjunctive commonalities significantly improve the performance of matching in clusters where successful matches occur frequently and in clusters where subscriptions do not already share common predicates. More importantly, it is clear that commonalities contribute significantly to matching performance optimizations at all selectivities. Since commonalities are derived from relationships between subscriptions in a set, they represent a key scalability optimization that is unique to CPS systems.

[5]Note that the subscriptions are not restricted to having the same number of predicates, we only do so to simplify presentation of the equation. In the general case, $mE(n-k)$ is replaced with $\Sigma_{i=1}^{m} E(n_i - k)$ in Equation 5, where subscription S_i has n_i predicates.

In Section 3, we observed that a depth-first algorithm [9] exists to determine the optimal matching plan for a link-group subscription topology. As its name implies, the algorithm models a DNF probabilistic Boolean expression as a tree and determines the optimal matching plan by recursively applying the depth-first algorithm on sub-trees. In the context of CPS matching, each subtree is a subscription and we have already described the optimal matching plan for a single subscription when treated as a conjunction of predicates. Therefore, using an optimal matching plan, the expected cost of a link-group subscription topology with m subscriptions $S_1 \ldots S_m$, and n predicates[6] in each subscription is:

$$E_{link}(m,n) = c'_1 + (1-p'_1)c'_2 + \cdots + \Pi_{i=1}^{m-1}(1-p'_i)c'_m \quad (7)$$

c'_i and p'_i are the expected cost and truth probability of S_i, respectively. That is, $c'_i = E(n)$ and $p'_i = \Pi_{j=1}^{n} p_{ij}$, where p_{ij} is the selectivity of the j^{th} predicate in S_i. If $c_i = 1, p_i = p, \forall i$, then:

$$E_{link}(m,n,p) = \frac{p^n - 1}{p - 1}\left\{\frac{(1-p^n)^m - 1}{-p^n}\right\} \quad (8)$$

Notice that Equation 7 is similar to Equation 3 where each subscription is treated as a "mega-predicate". This is a characteristic of the depth-first algorithm. Also recall that optimal matching implies that $p'_i/c'_i \geq (p'_{i+1})/c'_{i+1}, \forall i$. For unit costs, the optimal matching plan evaluates subscriptions in the order of most to least likely to be *true*, but evaluates predicates within each subscription from most to least likely to be *false*. An important observation is that the optimal matching plan for link-groups evaluates all predicates within a subscription before moving on to the next subscription. The same is not necessarily true for link-clusters and multicluster-groups since the depth-first algorithm no longer applies.

In Figure 3(c), we plot Equation 8 for $m = 200$ and allow n to vary from 1 to 50 while p varies from 0 to 1.0. Notice that there is an apex in the expected cost graph that illustrates the observation discussed in Section 2.2 regarding compet-

[6]Again, the n predicates restriction is only used to simplify presentation of the equation. Otherwise, each subscription S_i has n_i predicates and $c'_i = E(n_i)$.

ing optimization strategies for link-group topologies. When $p = 1$, this strategy evaluates only a single *true* subscription, so the expected cost is n. If $p < 1$, then there is a chance that the subscription most likely to be *true* in fact evaluates to *false*, thereby causing additional predicate evaluations in other subscriptions and increasing the overall expected cost. As p decreases towards 0, the expected cost is pulled back down by the likelihood of predicates evaluating to *false*, allowing the matching algorithm to quickly match each subscription. Therefore, the apex shown in Figure 3(c) represents a non-obvious upper bound on the expected cost of an optimal matching plan for link-group subscription topologies.

Note that fixing the values of m and n in our graphs does not affect the observed trend, which is dependent on the proportion of commonalities to unique predicates. The values of m and n only affect the absolute magnitudes of the matching costs (i.e., the scale of the graphs).

5. EXPERIMENTS

In this section, we present experiments performed in support of our model and simulations. The following experiments were performed using PADRES[7], a Java-based CPS middleware platform implemented by our research group and ran on an Intel Pentium 4 Linux system with 1GB of memory.

5.1 Model Verification

To verify our model, we isolated the core subscription matching module from our CPS middleware platform and measured the performance times of matching a publication against various subscription sets. The same parameters used to generate Figure 3(a) were used here as well (i.e., 200 subscriptions with an average of 100 predicates each). Publications were generated by randomly selecting values for each attribute from a uniform distribution, which also allowed us to generate predicates for subscriptions with certain selectivities. Each matching time measurement was obtained from an average of 200 test runs. In each test run, we randomly generated a publication as described earlier and averaged the matching times of 100 independent matches against the subscription set. The results of these experiments are shown in Figure 3(b). We can see that the general trend of the actual matching times closely resembles the expected costs described by Equation 6.

Again, changing the number of subscriptions and predicates (e.g., more subscriptions with fewer predicates) scales the matching time magnitudes but does not affect the observed trend.

5.2 Application

In previous sections, we discussed how commonalities are widely used by existing CPS systems to achieve scalability and good matching performance [3, 4, 8, 7, 11]. This implies that a "commonality-aware" CPS system may be able to manage and provide high-availability services more effectively. For instance, consider the problem of Denial of Service (DoS) attacks in which a malicious adversary directs large volumes of subscriptions at a broker. Not only does such an attack consume resources by causing the accumulation of subscription state, but overall matching performance is adversely affected as well due to useless predicate evaluations and subscription state maintenance. In this section, we present a proof-of-concept DoS resilience scheme based on our initial study of commonality effects.

Figure 4(a) shows the effects of a subscription flooding DoS attack on a CPS broker[8]. At the end of the time span shown, the broker is no longer able to function properly since all available memory is exhausted by the subscription flood. We can also see in Figure 5(a) that response times grow as the broker becomes more loaded and eventually, notification deliveries stop altogether once the broker has run out of memory.

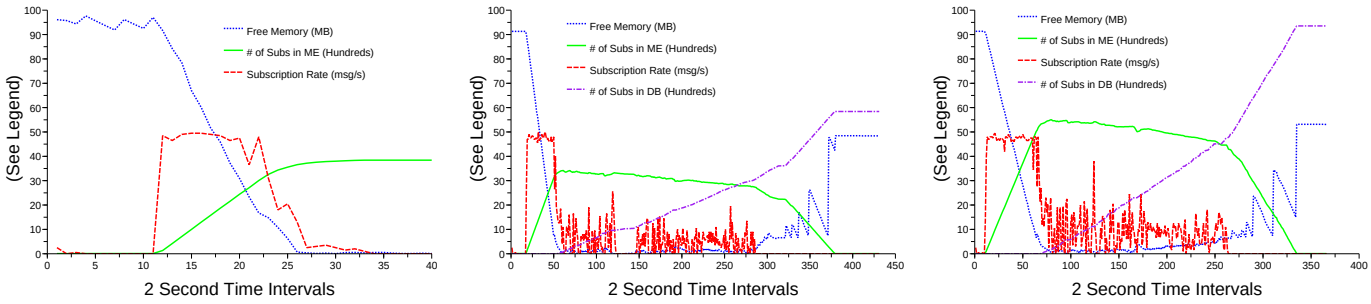
In our scheme, we implement a two-level priority-based subscription matching module. High priority subscriptions are stored and matched in a fast tree-based in-memory matching module that takes advantage of subscription commonalities, while low priority subscriptions are stored and matched in a much slower relational database-implemented matching module[9]. During operation, the broker maintains information about subscription clusters that exist and the degree to which predicates are "shared" within each cluster. In other words, we maintain a measure of conjunctive commonality for each cluster. Recall that conjunctive commonalities greatly improve matching performance by avoiding unnecessary predicate evaluations. Because of this, clusters with a high degree of conjunctive commonality will be greatly optimized by the in-memory matching algorithm. On the other hand, clusters with a low degree of conjunctive commonality will not benefit from these optimizations. Consequently, subscription clusters with a lot of shared predicates are considered efficient and are assigned higher priorities than clusters with little predicate sharing. This priority information is used if the broker becomes overloaded and runs low on memory, at which point, the lowest priority subscription clusters are migrated to the database matching module. Future messages are redirected accordingly depending on whether they are classed as high or low priority. Figure 4(b) shows how the broker can free memory to stay running by migrating subscription clusters while under a continuous subscription flooding DoS attack. Figure 5(b) shows the response times for two publication streams during the attack. One stream is classed as high priority and continues delivering notifications in a timely manner, while the other publication stream is classed as low priority and suffers from poor response times. Although some publication streams will suffer poor response times, high priority streams can continue with degraded, but much more acceptable performance.

Figure 4(c) shows the effects of the same experiment with subscription migration enabled, but the generated subscription workload contains many more common predicates so that the resulting subscription clusters exhibit a higher overall degree of predicate sharing. The in-memory matching module is able to maintain more subscriptions before running low on memory because of the increased commonality in subscription clusters. Figure 5(c) shows the response times of the same two publication streams used to generate Figure 5(b). In this experiment however, both streams ex-

[7]http://padres.msrg.utoronto.ca

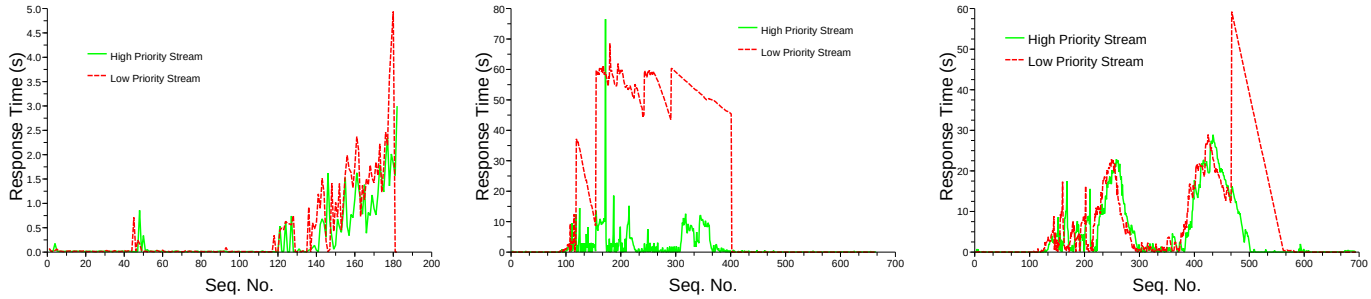
[8]We have limited the amount of memory used by the broker to more quickly observe the effects of the attack. However, the characteristic effects are the same when more memory is allocated, the plots are just stretched out over a longer time span.

[9]Implemented using PostgreSQL.



(a) No subscription migration (b) Subscription migration (c) Subscription migration and high commonality clusters

Figure 4: Effects of CPS broker under subscription DoS flooding attack.



(a) No subscription migration (b) Subscription migration (c) Subscription migration and high commonality clusters

Figure 5: Response time of publication streams

perience similar response times because the increased commonality has affected the order in which subscription clusters are migrated. That is, the subscription cluster to which the second publication stream belongs had a relatively high degree of sharing and was migrated much later in the attack (not until around the 500th notification was delivered). This shows how the commonality characteristics of a cluster can be used to ensure that high-performance streams remain high-performance, while low-performance streams are assigned lower priorities.

Figure 6 shows the degree of commonality in each migrated subscription cluster calculated as the ratio of the number of unique predicates represented in the cluster to the total number of predicates in the cluster disregarding any sharing. We also show the number of represented predicates in each migrated cluster along with the number of predicates weighted by the commonality ratio. The weighted predicate count is used to determine the priorities of subscription clusters.

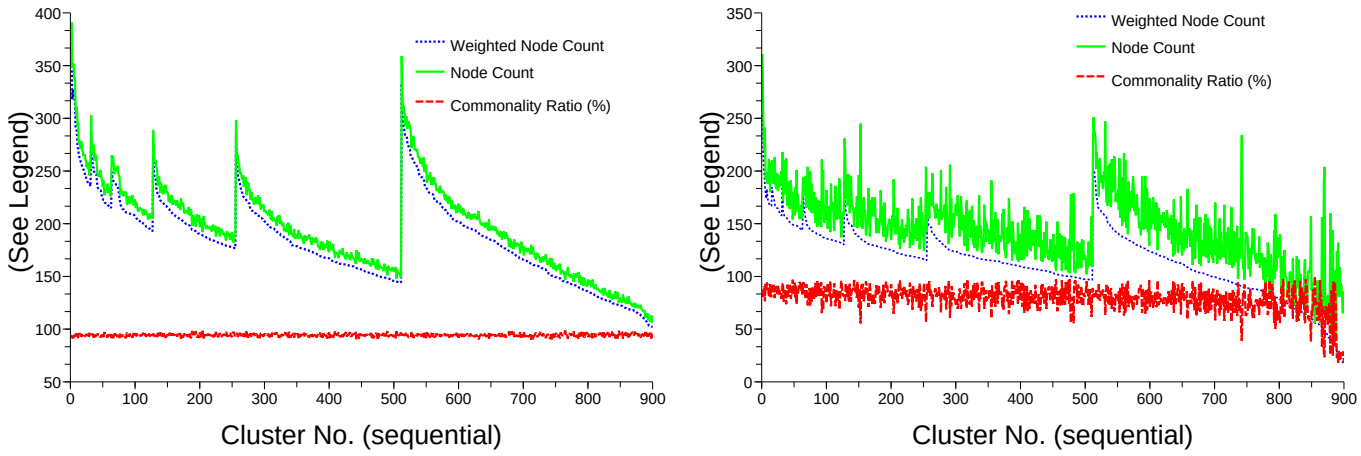
Although this is far from a complete DoS resilience system, the technique we present here allows a CPS broker to degrade services gracefully in the face of subscription flooding DoS attacks. The degree of commonality metric we use to prioritize subscriptions clusters in this scheme can potentially be one of several metrics in a full CPS intrusion detection system.

6. RELATED WORK

In the domain of relational databases, query planning is a well-established optimization concept. Since there are typically many different ways for a database to execute a query, the database can evaluate several alternatives and attempt to choose an execution plan with good performance. Our concept of a matching plan is similar in the respect that there are many alternatives for performing a match, some with better performance than others. However, the similarity ends there as matching plans deal with predicate evaluation order rather than database row retrieval and can also leverage subscription semantics for optimization.

In [6], Carzaniga *et al.* present a formal notation for expressing CPS matching semantics. Their notation centers around the concept of *covering* and takes both advertisements and subscriptions into consideration. However, their notation is only used to describe matching semantics and does not extend to any theoretical evaluation of performance, which is the focus of our work.

In [12], Mühl formalizes CPS routing using a syntax based on linear temporal logic. The formal specification focuses



(a) Low Predicate Sharing

(b) High Predicate Sharing

Figure 6: Degree of commonality in subscription clusters.

in-depth on reasoning about the correctness of routing algorithms at a distributed system level. In contrast, our work is an effort to abstract matching algorithms, which exists at a lower level and complements routing algorithm formalizations.

Both Li *et al.* [11] and Campailla *et al.* [5] present CPS matching algorithms based on Binary Decision Diagrams (BDD), which also map predicates to Boolean expression variables. However, they do not discuss probabilistically optimal matching plans nor do they investigate the performance implications of subscription set commonalities as we do. Additionally, the variable ordering heuristics used by BDDs are specific to that data structure, while we are presenting a more abstract view of the subscription set semantics widely found in CPS systems.

7. CONCLUSION

In this paper, we presented our initial efforts at modelling common optimizations used in practice by existing Content-based Publish/Subscribe (CPS) systems to improve the scalability of matching. Our model and analysis focuses on the observation that matching algorithms frequently take advantage of similar characteristics between subscriptions in a set. We abstracted these similar characteristics as *commonalities* and used them as the basis for modelling two types of subscription sets frequently found in existing CPS systems: link-groups and clusters. We found that link-groups facilitate achieving probabilistically optimal matching performance, while optimal matching in clusters remains an open research problem. We further provided expected matching cost equations for these two subscription set types and found that: (1) commonalities are a significant factor in matching performance across all predicate selectivities and (2) there exists a non-obvious upper bound on the expected matching cost of link-groups. Finally, we presented experimental results based on our own CPS middleware implementation (PADRES) that support our model.

For future work, there are still many open research problems that need to be addressed. First, we intend to investigate how the concepts of covering [6] and correlation can be expressed in our model and analyzed. Briefly, our approach is based on the observation that if subscription S_1 covers subscription S_2, then $S_2 \rightarrow S_1$ (or equivalently, $\bar{S_1} \rightarrow \bar{S_2}$) in our commonality model. That is, if $S_1(P) = \mathit{false}$ for some publication P, then evaluation of $S_2(P)$ can be avoided due to the covering relationship. Furthermore, we intend to extend our quantitative analysis of matching complexity and performance to include more general types of subscription sets as well as provide deeper insight into the types of subscription sets already presented in this paper. More specifically, the optimal matching characteristics of subscription sets in clusters is currently less well-understood than those of subscription sets in link-groups and requires deeper study since the technique of mapping a subscription set to a probabilistic Boolean expression is not always applicable. For clusters and other arbitrary subscription sets containing arbitrary commonalities, we are still investigating techniques for unified performance and complexity analysis.

Finally, we are investigating metrics that can be used to capture the "compression" of a subscription set due to the existence of commonalities. The commonality ratio we used in Section 5.2 is a step in this direction.

Acknowledgements

This research was funded in part by OCE, NSERC, CA and Sun. We would also like to thank Vinod Muthusamy and the entire Middleware Systems Research Group at the University of Toronto for their valuable feedback regarding this work.

8. REFERENCES

[1] http://www.cisco.com/en/US/products/ps6480/index.html.

[2] http://www.solacesystems.com/products/Tech_leadership.asp.

[3] M. K. Aguilera, R. E. Strom, D. C. Sturman, M. Astley, and T. D. Chandra. Matching Events in a Content-based Subscription System. In *Principles of Distributed Computing*, 1999.

[4] 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 *ICDCS*, 1999.

[5] A. Campailla, S. Chaki, E. M. Clarke, S. Jha, and H. Veith. Efficient Filtering in Publish-Subscribe Systems Using Binary Decision. In *International Conference on Software Engineering*, pages 443–452, 2001.

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

[7] A. Carzaniga and A. Wolf. Forwarding in a Content-Based Network. In *Proceedings of ACM SIGCOMM*, 2003.

[8] F. Fabret, H. A. Jacobsen, F. Llirbat, J. Pereira, K. A. Ross, and D. Shasha. Filtering Algorithms and Implementation for Very Fast Publish/Subscribe Systems. In *Sigmod*, 2001.

[9] R. Greiner, R. Hayward, M. Jankowska, and M. Molloy. Finding optimal satisficing strategies for and-or trees. *Artif. Intell.*, 170:19–58, 2006.

[10] Y. Ioannidis. The History of Histograms (abridged). In *VLDB*, 2003.

[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 the 25th IEEE International Conference on Distributed Computing Systems*, 2005.

[12] G. Mühl. *Large-Scale Content-Based Publish/Subscribe Systems*. PhD thesis, Vom Fachbereich Informatik der Technischen Universitat Darmstadt, 2002.

[13] P. R. Pietzuch and J. M. Bacon. Hermes: A Distributed Event-Based Middleware Architecture. In *DEBS*, 2002.

[14] H. A. Simon and J. B. Kadane. Optimal Problem-Solving Search: All-or-None Solutions. *Artificial Intelligence*, 6:235–247, 1975.