Practical High-Throughput Content-Based Routing Using Unicast State and Probabilistic Encodings

Antonio Carzaniga[†] Cyrus Hall[†] Giovanni Toffetti Carughi[†] Alexander L. Wolf[‡]

[†]Faculty of Informatics
University of Lugano
Lugano, Switzerland

[‡]Dept. of Computing
Imperial College London
London, UK

University of Lugano
Faculty of Informatics
Technical Report 2009/06
August 2009

Abstract

We address the problem that existing publish/subscribe messaging systems, including such commonly used ones as Apache's ActiveMQ and IBM's WebSphere MQ, exhibit degraded end-to-end throughput performance in a wide-area network setting. We contend that the cause of this problem is the lack of an appropriate routing protocol. Building on the idea of a content-based network, we introduce a protocol called B-DRP that can demonstrably improve the situation. A content-based network is a content-based publish/subscribe system architected as a datagram network: a message is forwarded hop-by-hop and delivered to any and all hosts that have expressed interest in the message content. This fits well with the character of a wide-area messaging system. B-DRP is based on two main techniques: a message delivery mechanism that utilizes and exploits unicast forwarding state, which can be easily maintained using standard protocols, and a probabilistic data structure to efficiently represent and evaluate receiver interests. We present the design of B-DRP and the results of an experimental evaluation that demonstrates its support for improved throughput in a wide-area setting.

1 Introduction

Publish/subscribe messaging is a central feature of modern enterprise computing platforms. They are typically based on high-performance implementations of the Java Message Service (JMS).[1] Examples include IBM's WebSphere MQ,[2] Fiorano's FioranoMQ,[3] and the Apache Foundation's ActiveMQ.[4] The central element of these systems is the so-called "broker" (or dispatcher), which is responsible for dispatching messages from publishers to subscribers. In some situations, such as balancing load or increasing reliability, multiple brokers are configured into a "federation" (or cluster, or network), using some protocol for exchanging messages so that they can cooperatively realize an end-to-end message service.

These systems work extremely well in a local-area setting, having been engineered primarily for use in large data centers. However, they encounter severe performance problems when used in a wide-area setting with multiple brokers positioned at different places in the network. (As indicated in the documentation for these systems, distributed brokers would be deployed in a wide-area network when, for example, they are administered by different organizations and/or serve end points in different administrative domains.) The high end-to-end throughput they exhibit in small, fast, local-area networks is lost in large, slow, wide-area networks. In fact, our experiments demonstrate that the protocol they use to route messages in the wide-area network substantially dictates their end-to-end messaging performance, overshadowing the impressive performance capabilities of the individual brokers. In other words, *existing high-performance, publish/subscribe messaging systems do not currently embody a routing protocol appropriate for use in a wide-area network.*

In this paper we present and experimentally evaluate a new routing protocol, called B-DRP, that can significantly improve the performance of messaging systems deployed in a wide-area network. The protocol derives from ideas developed in the fields of distributed publish/subscribe systems, content-addressable networks, and content-based networking, which have resulted in proposals for a number of interesting systems and routing schemes [3, 5, 6, 8, 9, 17, 24].

Most relevant here is content-based networking. A content-based network offers a distributed, content-based, publish/subscribe communication service architected as a datagram (overlay) network: a message is presented to the network with no specific destination, and then forwarded hop-by-hop to any and all hosts that have expressed an interest matching the content of the message. This fits well with the character of a wide-area messaging system.

B-DRP is designed to be practical, flexible, and admit to a straightforward implementation that can utilize and leverage existing unicast protocols. The two essential ideas contained in the protocol are (1) to evaluate the content of a message against the interests of all receivers only once, when the message enters the network, and (2) to use the resulting set of interested receivers to direct an efficient, unicast-based forwarding process for the message.

Matching message content at its source requires that all receiver interests be broadcast to, and maintained by, all routers in the network, since any one of them could have attached to it a host that is the source of a message of interest to some receiver. This leads to an obvious scalability problem in large, general-purpose networks. To mitigate the problem, we encode the predicates that represent receiver interests with a compact data structure based on Bloom filters. This probabilistic, bit-vector encoding reduces the storage and processing requirements for routing information, admits to a fast matching algorithm, and may open new possibilities for defining a privacy-preserving content-based routing scheme.

The protocol requires that each router in the network maintain receiver predicates as a bit matrix, together with traditional unicast forwarding state. Our current implementation uses a distance-vector protocol to maintain unicast forwarding state. However, it is completely independent of this, and so another protocol (e.g., a link-state protocol) would work just as well.

Assuming that the unicast forwarding state is built from minimal paths, the forwarding process directs a message along a minimum-path spanning tree toward the set of receivers interested in the message. The message is replicated only when reaching a point in the tree where there are interested receivers down different branches. (Notice the difference here from a multicast delivery tree, where there would be no such decision process and packets are replicated at all branch points.) Thus, a given message can be thought of conceptually as a bundle of messages whose elements recursively "peel off" into sub-bundles only at appropriate branch points.

To effect this behavior, the protocol defines a packet format that provides for multiple unicast destination addresses. The set of addresses is used to forward the packet hop-by-hop along a shared path, with the set

[1] http://java.sun.com/products/jms/

[2] http://www.ibm.com/software/integration/wmqfamily/

[3] http://www.fiorano.com/products/fmq/

[4] http://activemq.apache.org/

progressively partitioned as the packet replicates along two or more forwarding paths. In the limit, the set consists of a single destination address. We refer to this process as *dynamic receiver partitioning (DRP)*. We originally developed DRP to support content-based routing in sensor networks [13]. However, the same idea, referred to as "dynamic multicast," was also developed by Cao and Singh for their MEDYM routing scheme [6].

Our protocol is called B-DRP (pronounced "bee drip") to capture the two main techniques of Bloom filter encoding and dynamic receiver partitioning.

We have conducted an extensive experimental evaluation of B-DRP and its prototype implementation. Beyond the most basic validation (does the protocol work at all?) we wanted to thoroughly evaluate the effectiveness of B-DRP in terms of its support for an end-to-end communication service. The primary metrics we focus on are therefore the effective global throughput and the number of false negatives (lost packets) that are due to congestion in routers. At a high level, our intuition is that the scalability of a multi-broker, publish/subscribe message service deployed in a wide-area network depends crucially on *routing*, more than on the speed of local matching and delivery. The results of our experiments clearly confirm this hypothesis.

In practice, we compared our implementation of B-DRP for the Siena publish/subscribe system[5] against ActiveMQ, which reportedly provides the best available JMS-based message service [14, 19] and IBM's WebSphere MQ. We did this on a network testbed by applying a battery of workloads intended to represent a mix of distributed applications. Having obtained low message throughput with WebSphere MQ, substantially confirming the findings of another study [15], in this paper we omit our results for Websphere MQ.

While ActiveMQ achieves higher throughput and lower loss rates than Siena/B-DRP in high-bandwidth scenarios with few clients and subscriptions, Siena/B-DRP out performs ActiveMQ in all other cases, showing a progressively greater advantage as the number of clients, subscriptions, and brokers grows.

Previous studies confirm that a growing number of subscriptions or brokers in a network have a negative effect on total system throughput for ActiveMQ [15, 22]. In our evaluation we were careful to establish that this was due purely to the choice of routing protocol and not to, say, the fact that Siena/B-DRP exhibits less functionality than ActiveMQ. Furthermore, the workloads we used in the evaluation were chosen in such a way as to favor the behavior of ActiveMQ brokers.

It is important to note that our intention here is to propose and demonstrate a routing protocol that can improve the performance of messaging systems in practical scenarios, not to argue that the Siena/B-DRP implementation should be used in place of messaging systems such as ActiveMQ. In fact, we are currently developing a version of ActiveMQ that incorporates B-DRP in the hopes of influencing the future evolution of that system.

2 B-DRP Routing

We start by defining the basic elements of a content-based networking model, and then proceed to present the B-DRP routing protocol in those terms.

2.1 Preliminary Models and Assumptions

For the purpose of this paper, a content-based network [10] is an interconnection of content-based routers, each one serving a potentially large set of locally connected hosts and their applications. Routers and hosts have unique identifiers and their interconnections are stable point-to-point links.

Applications express their interests and send messages through their hosts. In this paper we do not concern ourselves with the specifics of application interfaces within hosts, and simply assume that each host collects and maintains the interests of its applications, and similarly that it handles the publication of messages on behalf of its applications. Therefore, we assume that a router exports a minimalistic interface to the hosts in its local subnet. This interface consists of two primitives: a *set-predicate*(*host-id*, *p*) function, which defines the interests of the given host, and a *send-message*(*m*) function to publish a message.

For our purposes, a message is equivalent to the representation of a record in a database. It consists of a set of attributes (the columns in the database) and their values. Attributes are identified by name and have a type chosen from among a set of common predefined data types (integers, strings, etc.). For example, a message representing, say, a news item in a news-alert application might consist of the following attributes: [agency="AP", date='29/08/2008', title="Italian prime minister resigns", keywords="politics Italy", priority=−1, url="http:..."]. Predicates are Boolean expressions equivalent to the WHERE clause of an SQL expression. More

[5] http://www.inf.usi.ch/carzaniga/siena/

specifically, they are a disjunction of conjunctions of elementary constraints on the values of attributes. For example, the predicate (agency="AP" *and* priority=2 *or* "Italy" ∈ keywords) would match the message above.

2.2 B-DRP Routing Scheme

B-DRP uses three types of forwarding state. First, each router collects the predicates of its locally connected hosts in a table called LOCAL-PREDICATES, which represents a map *host-id* → *predicate*. Second, each router computes the combination (logical disjunction) of all its local predicates, encodes this potentially large predicate using the algorithm sketched in Section 2.3, and then advertises the resulting encoded predicate. Advertisements are received and stored by each router in a table called PREDICATES, which represents a map *router-id* → *encoded-predicate*. The third type of state is the unicast forwarding information needed to reach every other router in the network. We refer to this state through a table called UNICAST that represents a map *router-id* → *neighbor-link*.

Notice that LOCAL-PREDICATES contains only local information. Furthermore, both PREDICATES and UNICAST can be compiled and maintained using any one of several well-known protocols (e.g., OSPF or IS-IS), so we do not consider this aspect of B-DRP in this description and instead focus on its forwarding algorithm.

B-DRP uses a two-level forwarding process. The router that initially receives the *send-message*(*m*) command uses LOCAL-PREDICATES to find the set of local hosts interested in *m*, and then proceeds to forward a copy of *m* to those local hosts. We refer to this process as *local delivery*.

After local delivery, the same router proceeds with remote delivery. The router uses PREDICATES to find all other routers that advertise a predicate matched by *m*. We denote this set of destinations as *D*, so formally $D := \{id \mid m \text{ matches } \textsc{predicates}(id)\}$. Notice that router-advertised predicates are encoded, so this latter matching operation uses a different algorithm than local delivery.

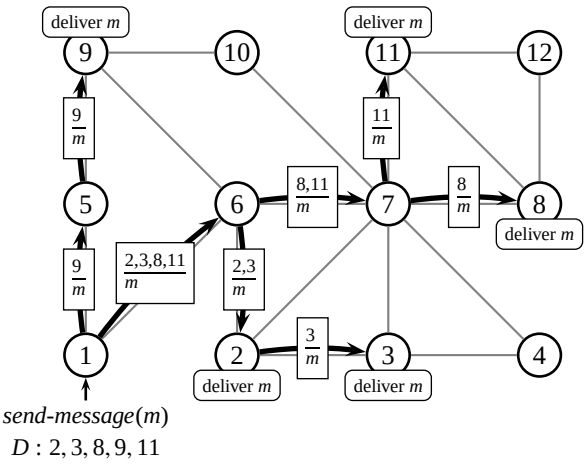

Figure 1: Forwarding By Dynamic Receiver Partitioning

The router then proceeds to forward *m* towards all destinations in *D* using a process called *dynamic receiver partitioning*. An example of this process is illustrated in Figure 1. Using UNICAST, the router groups the set of destinations by their next hop. That is, the router partitions *D* into disjoint subsets of destinations $D_1, D_2, \ldots$ where all destinations in D_i share the same next-hop neighbor link. Then, for each D_i, the server sends a packet containing the pair $\langle D_i, m \rangle$ through the corresponding neighbor link *i*. When a router receives a packet with destination set *D* and message *m* from one of its neighbors, it first checks whether its own identifier is in *D*. If so, it processes *m* for local delivery. The router then removes its identifier from *D* and forwards *m* toward all remaining destinations in *D*, using the same partitioning algorithm.

2.3 Scalability Challenges

There are two aspects of B-DRP that might immediately be seen to limit its scalability.

The first aspect is that B-DRP requires every router to store the predicates advertised by all routers in the network, each one aggregating the interests of the applications connected through that router. In other words, the table PREDICATES of each router amounts to a representation of the interests of *all* applications running throughout

the network. This problem is mitigated by the fact that aggregate predicates are likely to admit more compact representations than the sum of their parts [7]. However, the reduction factors we observed in practical cases, while good, are still limited and certainly exhibit a high variance. Therefore, we address this scalability problem by using a compact encoding of predicates based on Bloom filters, which we describe in detail in Section 3.

The second potential scalability issue affecting B-DRP is that the destination sets attached to messages may be very large. By "large" we mean in the number of elements, not necessarily in its representation, which can take the form of a bit vector, where each position in the bit vector corresponds to a router according to some agreed-upon assignment. The mechanism that we propose to deal with large destination sets establishes a maximum size called *broadcast threshold*. This threshold defines a reasonable bound on when to apply dynamic receiver partitioning—in particular, when the destination set approaches the full set of routers. If the first destination set (i.e., the one computed by the remote delivery process executed at the source of the message) exceeds the broadcast threshold, then the message is treated simply as a broadcast packet and forwarded according to a reverse-path forwarding algorithm using the available unicast forwarding state. For purposes of this paper our results reflect experiments that do not exceed a broadcast threshold.

3 Message and Predicate Encoding

B-DRP uses two methods to cope with the scalability of the table PREDICATES, which stores non-local predicates, and its associated maintenance protocols. First, routers *aggregate* predicates from the hosts and applications in their subnet. Second, routers *encode* predicates using a compact data structure based on Bloom filters. The first technique amounts to rewriting a logical disjunction of predicates in a compact form by performing logical predicate simplification. We discuss an application of this technique and its reduction of required forwarding state in another paper [7]. Here we describe the encoding of predicates using Bloom filters together with a matching algorithm that operates on encoded predicates.

Briefly recalling their structure and properties [4], a Bloom filter is a bit vector of a given length M that represents a set S and provides a probabilistic membership test. The 0^M vector represents the empty set. An element x is added to a Bloom filter B by setting $B[h_1(x)] \leftarrow 1, B[h_2(x)] \leftarrow 1, \ldots, B[h_k(x)] \leftarrow 1$, where $h_1, h_2, \ldots, h_k$ are k independent hash functions that map elements into bit-vector positions $\{1, \ldots, M\}$. The membership test follows intuitively: given a Bloom filter B and an element y, the test is positive if $B[h_1(y)] = 1 \wedge B[h_2(y)] = 1 \wedge \ldots \wedge B[h_k(y)] = 1$. This test is only a necessary condition, and yields a false positive with a probability that depends on M, k, and the size of the represented set $|S|$. We do not further discuss these design parameters or other information-theoretic properties of Bloom filters, since those are well understood [20]. However, we note that the membership test can also be easily extended to implement a subset test: given two sets S_1 and S_2, and their representations as Bloom filters B_1 and B_2, testing that $S_1 \supseteq S_2$ amounts to testing that $\forall i \in \{1, \ldots, M\} : B_2[i] = 1 \Rightarrow B_1[i] = 1$. For brevity, interpreting bit vectors themselves as sets in $\{1, \ldots, M\}$, we denote this test between bit vectors as $B_1 \supseteq B_2$.

We propose a two-phase encoding scheme for messages and predicates. The first phase amounts to a *categorization* of messages and predicates. Let $\mathcal{M}$ and $\mathcal{P}$ represent the set of allowable messages and predicates, respectively ($\mathcal{M}$ and $\mathcal{P}$ are defined informally in Section 2.1), and let C be a set of categories. Categories can be seen as so-called *tags* for data and predicates. Below, we make this definition concrete through a specific categorization algorithm.

A message categorization function $\mu : \mathcal{M} \rightarrow \mathbb{P}(C)$ transforms a message m into a set of categories S_m ($\mathbb{P}$ denotes the power set). A predicate categorization function $\pi : \mathcal{P} \rightarrow \mathbb{P}(C)^*$ transforms a predicate p into one or more sets of categories $S_p^1, S_p^2, \ldots$ The functions μ and π are such that for all predicates p and messages m

$$p(m) \Rightarrow (\exists S_p^i \in \{S_p^1, S_p^2, \ldots\} : S_m \supseteq S_p^i)$$

In words, if a message m matches a predicate p, then the set of categories associated with m must contain at least one of the sets associated with p. An analogy with tags can help explain the idea behind this definition. A message will have a set of tags S_m, and a predicate p will map to a set of sets $S_p^1, S_p^2, \ldots$ of desired tags. The mapping is such that a message matches a predicate when the message has all the tags of at least one set of tags associated with the predicate.

In practice, the categorization we chose for messages is based on the categorization of its attributes, and the categorization of predicates is based on its constraints. Specifically, each attribute a in a message maps to a set of categories $\alpha(a)$, and the categorization of a message $\mu(m)$ is simply the union of the categorizations of its attributes: $\mu(m) = \bigcup_{a \in m} \alpha(a)$. Similarly, the categorization of predicates is based on a categorization of constraints.

In the second phase of the encoding, we simply encode sets of categories as Bloom filters through the function $\beta : \mathbb{P}(C) \to \{0,1\}^M$. In summary, a message m is encoded as a bit vector $B_m \in \{0,1\}^M$ with the transformations

$$m \xrightarrow{\mu} S_m \xrightarrow{\beta} B_m$$

while a predicate p is encoded as one or more bit vectors $B_p^1, B_p^2, \ldots, B_p^N$ with the transformations

$$p \xrightarrow{\pi} \{S_p^1, S_p^2, \ldots, S_p^N\} \xrightarrow{\beta} \{B_p^1, B_p^2, \ldots, B_p^N\}$$

Notice that since the second phase of the encoding preserves the subset relations introduced in the first phase (i.e., $S_m \supseteq S_p^i \Rightarrow B_m \supseteq B_p^i$) the matching condition required on the categorization functions translates directly into a similar relation on the Bloom filters. Specifically,

$$p(m) \Rightarrow (\exists B_p^i \in \{B_p^1, B_p^2, \ldots\} : B_m \supseteq B_p^i) \tag{1}$$

which means that the existence of a B_p^i in the encoded predicate, such that $B_m \supseteq B_p^i$, is a necessary condition for matching between the original m and p.

The concrete encoding scheme we use is based on the model of messages and predicates outlined in Section 2.1.[6] In particular, an attribute a has a name $name(a)$ and a value $value(a)$ belonging to a set of atomic data types, including numbers (integers and floating-point) and byte strings. Attribute names are also byte strings. A constraint c consists of an attribute name $name(c)$, a type-specific binary relation $op(c)$, and a comparison value $value(c)$. Relations over numeric types include equality and order relations; relations over strings include equality, lexicographical order, prefix, suffix, and substring. For example, the constraint $[x < 100]$ represents the subset of numeric values less than 100, while the constraint $[s * \text{“abc”}]$ represents all string values containing “abc” as a substring.

In order to specialize the encoding, we first define a set of categories C, which we conveniently represent as strings. Categories can be of two types:

- *Existence* categories are strings obtained by concatenating the existence symbol ‘∃’, a type identifier (e.g., ‘#’ for numbers and ‘$’ for strings), and an attribute name.
- *Equality* categories are strings obtained by concatenating a type identifier, an attribute name, the equality symbol ‘=’, and a value. (Values are encoded as strings in the obvious way. Similarly, special characters can be encoded to avoid ambiguities. Below we use the *dot* operator ‘·’ to represent string concatenation.)

We then specialize the categorization functions for attributes and constraints. An attribute a defined by name $n = name(a)$ and value $v = value(a)$ is categorized with an existence category plus an equality category:

$$\alpha(a) = \{\text{‘∃’} \cdot type(v) \cdot n, type(v) \cdot n \cdot \text{‘=’} \cdot v\}$$

A constraint c defined by name n, binary relation op, and comparison value v is categorized with either an existence or an equality category, depending on the type of the binary relation op:

$$\gamma(c) = \begin{cases} \{type(v) \cdot n \cdot \text{‘=’} \cdot v\} & \text{if } op = equality; \\ \{\text{‘∃’} \cdot type(v) \cdot n\}, & \text{otherwise.} \end{cases}$$

For example, a message containing the following attributes [stock=“XYZ”, price=50, quantity=100] would map to the following categories: “∃*$stock*”, “*$stock=XYZ*”, “∃*#price*”, “*#price=50*”, “∃*#quantity*”, and “*#quantity=100*”. A predicate [stock=“XYZ”∧price>20] would map to the categories “*$stock=XYZ*” and “∃*#price*”.

It is easy to see that this categorization satisfies the matching requirements, since $\alpha(a) \supseteq \gamma(c)$ for every attribute a and every matching constraint c. It is also clear that this categorization is quite coarse grained, and therefore will induce false positives, because singleton (equality) constraints map to the equality category and everything else (constraints with two or more allowed values) maps to the generic existence category for that attribute and type. Nonetheless, this categorization in practice strikes a good balance between accuracy and efficiency, introducing false positives, but still allowing routers to filter out the vast majority of messages. We use this scheme as part of a hierarchical routing protocol where false positives are considered a loss of accuracy rather than an incorrect behavior: they are filtered out by local routers applying the precise predicates requested by receivers before being forwarded to a client.

[6] This data model and the encoding are implemented within the Siena Fast Forwarding (SFF) software package, which is available for download at http://www.inf.usi.ch/carzaniga/cbn/forwarding/.

4 Implementation

The basis for our implementation of B-DRP was the Siena publish/subscribe prototype.[7] Siena was designed for acyclic networks of publish/subscribe dispatchers. More specifically, the version of Siena we started with, implements a hierarchical routing scheme whereby each router (except a distinguished root) is assigned a "master" router to which all notifications and some subscriptions must be forwarded. Therefore, in order to implement B-DRP, we had to extend Siena and its internal structures to allow multiple router-to-router connections.

Beyond these modifications, we also re-engineered some parts of the Siena system in order to improve its performance as a message router. These changes were necessary to make Siena at least minimally competitive with modern publish/subscribe systems in terms of throughout. Two changes are worth mentioning here because they had a significant impact on the performance of the new Siena/B-DRP router. First, we introduced proper queuing and thread-pooling to interconnect input links, output links, and processing modules. Second, we replaced Siena's original matching algorithm, which is based on the same subscriptions poset used for routing, with an implementation of a fast matching algorithm based on a specialized index [11].

There were other aspects of the implementation of the Siena system that we considered in our re-engineering effort. Notably, its on-the-wire protocol, which uses a textual "human-readable" representation intended to facilitate debugging and extensions. This format requires the server to parse an entire packet and build an internal representation of the packet before processing can even start. Of course, such operations are inappropriate for a high-throughput router, and indeed we observed many cases in which processing time becomes the bottleneck. However, we decided to keep the same format to avoid the potentially high cost of testing and validation for a radically new format, and because ultimately we were not interested in pushing throughput to a maximum.

5 Experimental Evaluation

A routing protocol can be evaluated from different and complementary perspectives. One is to evaluate the cost of the routing protocol *per se*. This amounts to measuring the overhead of the control traffic necessary to install and maintain the protocol's forwarding state. A second perspective is to measure the quality of the delivery service resulting from the forwarding state. In this paper we take the latter perspective.

Our motivation for focusing on end-to-end delivery, rather than on route maintenance, is that B-DRP is not in and of itself innovative from the perspective of routing-state maintenance. In fact, this is a very explicit choice: we designed B-DRP to be generic with respect to the maintenance protocol. Thus, B-DRP uses forwarding information that can be easily assembled and maintained through standard protocols. For example, a classic link-state protocol would suffice. Furthermore, the dynamics of such protocols and the corresponding controls have been studied extensively in the development and engineering of Internet protocols such as OSPF and IS-IS, but also for equivalent protocols in overlay networks [1].

Another consideration is that, while content-based routing has been studied in simulation, it has rarely if ever been evaluated on real implementations. In fact, we are not aware of any systematic comparative study of content-based routing in real deployments, which is precisely one of the goals of this paper. In this context, end-to-end delivery is the most significant metric that can be measured meaningfully and uniformly across different systems.

Our evaluation is intended to provide answers to the following questions:

1. *How does B-DRP perform compared to state-of-the-art messaging systems?* We want to assess B-DRP through a comparative analysis with the best existing systems.

2. *How well does B-DRP support content-based communication on a wide-area network?* We want to see whether B-DRP is capable of sustaining a high, end-to-end, global throughput when deployed on networks of increasing size, and when the connectivity is characteristic of wide-area networks.

3. *How does B-DRP handle increasing message traffic?* Is B-DRP prone to congestion? Notice that the issue of congestion *control*, although clearly important in the context of content-based communication, is outside the scope of this paper. What we want to find out here is how B-DRP handles traffic beyond its peak throughput. Does the throughput remain constant, does it decline gracefully, or does it fall abruptly due to some destructive congestion effect?

[7] http://www.inf.usi.ch/carzaniga/siena

5.1 Experimental Methods and Setup

Here we describe the methods we have chosen to conduct our experiments and the particular configuration and behavioral parameters we employ.

5.1.1 Touchstone System

The system we selected for our comparative study is ActiveMQ, a JMS implementation developed by the Apache Software Foundation. ActiveMQ is advertised as "the most popular and powerful open source Message Broker" and is also reportedly the fastest JMS implementation, according to a recent study [15]. ActiveMQ supports full content-based subscriptions, as well as a distributed deployment through a "network of brokers" and, therefore, is an ideal candidate for our study. The specific version of ActiveMQ we used was 5.2, patched with change set 745558.

5.1.2 Testbed

We ran most of our experiments on our local, 43-machine network testbed. We also ran some experiments on the Emulab testbed [27], which validated our experimental environment. (Unfortunately, we were unable to run large-scale experiments on Emulab due to the limited availability of nodes.) Our testbed is set up with one machine dedicated to control the experiment, and the remaining 42 machines dedicated to running the components of the experiments. The machines each have 4GBs of system memory and two dual-core AMD Opteron 2212 processors (2000MHz clock frequency), each with a 1024KB cache. The operating system is Ubuntu Feisty, with Linux kernel version 2.6.20-16-lowlatency. Java applications run on the Java SE Runtime Environment (build 1.6.0-b105) with the HotSpot 64-Bit Server VM (build 1.6.0-b105, mixed mode).

The physical connectivity between the testbed machines is assured by an isolated gigabit Ethernet. In order to emulate a wide-area network deployment on top of our physical testbed, we use the Linux Traffic Control tools,[8] which allow us to shape traffic between running components, limiting the bandwidth as necessary.

5.1.3 JMS Configuration

The JMS API supports two message delivery modes. The *persistent* delivery mode, which is the default, instructs the JMS implementation to take extra care to ensure that a message is not lost in transit due to a failure or communication problem. Therefore, a message sent with this delivery mode is always logged to stable storage before being sent. The *non-persistent* delivery mode does not require the JMS implementation to store the message or otherwise guarantee its delivery.

The JMS API also distinguishes between *durable* and *non-durable* subscriptions. A durable subscription amounts to a persistent mailbox. If the subscriber is active, the JMS implementation delivers all matching messages to it. Otherwise, the implementation holds all messages in a persistent store until the subscriber becomes active again, or until they expire. Conversely, a non-durable subscription instructs the implementation to deliver all matching messages to the subscriber only if it is active. So, when the subscriber is not active, matching messages are simply discarded.

We compare the maximum achievable output throughput of ActiveMQ and Siena/B-DRP using the fastest possible configuration: non-persistent messages and non-durable subscriptions.

In addition to deploying brokers,[9] we deploy programs that serve as surrogates for client applications to drive the experiments. Each client consists of a simple sequential program running as multiple threads within one Java VM. Client programs do nothing other than subscribe, send, and receive messages.

5.1.4 Experiment Parameters and Workloads

The formulation of workloads is clearly a crucial part of the evaluation of any routing protocol. The main difficulty is that workloads are characterized by several parameters. Despite some recent progress, there does not seem to be a commonly accepted workload, nor a common set of parameters for these kinds of protocols. A JMS benchmark has recently been developed.[10] However, this benchmark is based on a single application that, even

[8] http://lartc.org/

[9] For ease of exposition, we consistently use the term "broker" to refer to both the ActiveMQ broker and the Siena/B-DRP router.

[10] http://www.spec.org/jms2007/

though distributed among sites, is not necessarily representative of the range of traffic seen on a wide-area network hosting several applications and several more individual users.

We generate workloads based on the following general experimental program:

1. select a network configuration and configure the traffic shapers of all links accordingly;
2. deploy and connect the brokers;
3. deploy all client programs;
4. have all clients set their predicates;
5. pause to allow the forwarding state to propagate and stabilize;
6. have all clients start sending messages for a set amount of time, at either a constant rate or at a constantly growing (linear) rate;
7. shut down all brokers and clients, collect the necessary data, and clean up.

This program constrains the behavior of the clients, and therefore the workloads. Clients set their predicates only once, at the beginning of the experiment, and only after that do they start sending messages. Therefore, each experiment is characterized by (1) a network configuration, (2) an assignment of predicates to clients, (3) a set of messages, and (4) a fixed sending rate or a fixed growth speed for the sending rate.

Network configurations are determined by the specific experimental scenario, with either 1, 2, 4, 8, or 42 testbed machines hosting brokers. Clients are then distributed uniformly onto the entire network. Clients act as Poisson processes when sending messages, so their behavior is fully determined by the output rate.

The most difficult parts of the experiments to define are their content-based aspects, namely the nature of predicates and messages. As we have done in past experiments [9], we first define predicates and messages according to what we believe is a realistic generative model. We do this because we want to test the matching functions using realistic messages and predicates. We then abstract the content-based components of an experiment and simply characterize the experiment by the distribution of the matching messages over predicates.

Messages are composed of 10 attributes. 60% of those attributes are strings taken from a set of 200 frequent terms extracted from a large corpus of news reports [18]. 40% of the values are integers between 0 and 100. Predicates are composed of 20 conjunctions, each one consisting of between one and five constraints. Each constraint is defined by the same dictionary of terms and the same distributions of values used for messages. In terms of constraint operators, 70% of string constraints are equalities, and 30% are equally divided among prefix, suffix, and substring, respectively. Integer constraints use equality, less than, and greater than, with probabilities 80%, 10%, and 10%, respectively.

In our comparative analysis we chose to generate simple predicates with only one constraint per conjunction. This is intended to increase the matching probability so as to induce higher volumes of forwarding traffic. Notice that this choice gives ActiveMQ an advantage over Siena/B-DRP. This is because all JMS implementations show their best performance with smaller conjunctions of constraints [15], whereas the algorithm used by Siena/B-DRP to match Bloom-filter-encoded predicates is not sensitive to the size of the sets represented by each Bloom filter.

Figure 2 shows the distribution of matching messages over predicates (where one predicate corresponds to one client) and brokers in experiments of Figures 4 and 5 with 80 clients and 8 brokers. Each histogram shows a normalized count, and therefore a probability distribution. In particular, the Y-axis shows the probability that a message matches a given number of predicates or brokers. The most important parameters that characterize the experiment are the centers of gravity of those two probability masses. In particular, the *client replication factor* is the expected number of predicates matched by a message, and the *broker replication factor* is the expected number of brokers to which a message must be delivered. Since we distribute clients uniformly across the network, the two replication factors are related by the density of clients per broker.

In summary, we characterize each experiment by these high-level parameters:

- *network:* size and type of topology;
- *links:* bandwidth limits set on each link;
- *replication factor:* expected number of destinations for a message; and
- *input rate:* total publication rate (messages per second).

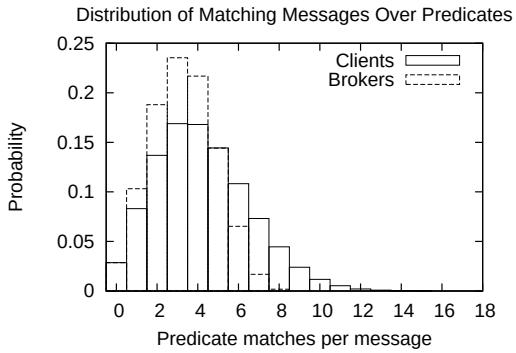

Figure 2: Distribution of messages per number of matches, client replication factor is 4.2, broker replication factor is 3.3

5.1.5 Measurements and Analysis

We focus on end-to-end delivery. Therefore, the primary metric of interest is the *output throughput*, measured as the total number of messages delivered across the network per unit of time. In our analysis we present two types of measurements. In some cases we focus on a single experiment and show the output throughput over time, subjecting the network to an increasing input rate and plotting the output throughput as a function of time. In other cases we consider a series of experiments in which we keep a constant input rate and measure the output throughput rate at steady state.

5.2 Results

5.2.1 Comparative analysis

We start our comparative evaluation with a scenario in which a single broker is used with a simplistic workload. The workload consists of 50 clients with four subscriptions, each one consisting of a single constraint. We also use a small number of attribute names (for constraints and messages) so as to obtain a high replication factor. The result is that each published message is delivered on average to 27 clients.

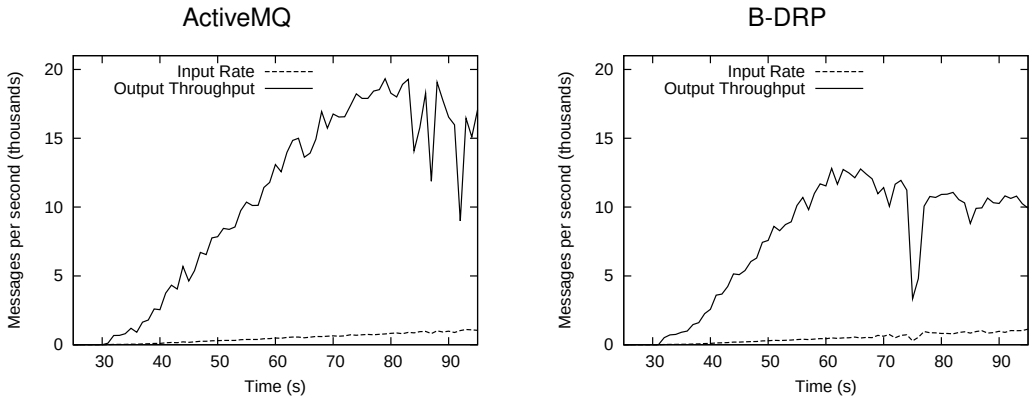

Figure 3: ActiveMQ and Siena/B-DRP, single-router throughput comparison

Figure 3 shows the measured input rate and output throughput for this scenario under a constantly increasing input rate. ActiveMQ (left side) outperforms Siena/B-DRP. In this case, routing is not a factor (there is a single broker/router) and the matching function is not the bottleneck (there are only a few and simple predicates), so we conclude that ActiveMQ's delivery functions and its infrastructure in general are better engineered than Siena's. The quantitative results for ActiveMQ obtained here are consistent with the most recent results obtained by others [22].

Having established that ActiveMQ performs well with simple workloads and with a single broker, we move to a more realistic and therefore complex workload. In this case, we use eighty clients, uniformly distributed across 8 sites connected through 1MBps links. Each client has twenty, single-constraint subscriptions, with a resulting replication factor of 5%. Once again, we test the two system by subjecting them to a ramp of incoming traffic: after subscribing, each client starts publishing at a rate of one message per second, and growing linearly to a final rate of 100 messages per second.

Given this fixed workload, we perform a series of experiments. We first deploy one broker, but notice that a single broker is unable to deliver at full throughput. Therefore we scale the network of brokers to 2, 4, and then 8 brokers. The 4-broker network has the topology of a square plus one diagonal link; the 8-broker topology has a topology of diameter 3 generated by the BRITE generator.

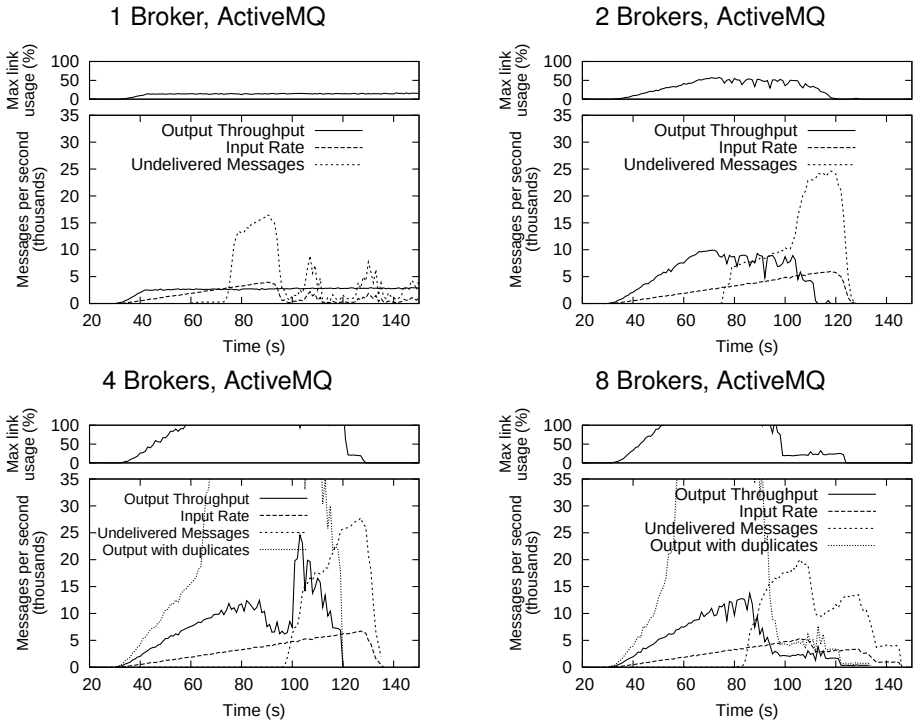

Figure 4: ActiveMQ in 1, 2, 4, and 8-broker networks, 5% client replication

Figure 4 shows the results of these experiments for ActiveMQ. In each of the four graphs we plot the recorded input rate and output throughput, plus the rate of undelivered messages. Also, separately on top of each graph, we plot the percent link usage of the most heavily loaded link in the network.

The results are interesting. We first note that a single broker does not saturate the network, and therefore conclude that the low throughput is due to the heavy CPU load at the broker. When we move to the 2-broker network, we notice an initial improvement. In particular, two brokers manage to deliver a bit more than twice the number of messages during the initial ramp. However, they also exhibit an unstable behavior beyond a certain input rate. Still, the two-broker network does not saturate the network. As we progress to networks of 4 and 8 brokers, we see a slight improvement in the maximum output throughput, but also a more evident instability and congestion. Moreover, we see that these two configurations become at least partially network bound, as they saturate some of the links. The behavior we observe in this experiment is consistent with a recent analysis that shows that the output throughput decreases with a high number of subscriptions [15].

Furthermore, scaling the same workload to 4 and 8 brokers reveals a more fundamental problem in ActiveMQ's routing algorithm. Our further analysis conducted on smaller, ad-hoc scenarios, and corroborated by developers' comments, shows that ActiveMQ suffers network congestion because it uses a simplistic routing protocol based on controlled flooding.

Turning now to Siena/B-DRP, we replicated the same experiment described above with 1, 2, 4, and 8 Siena/B-DRP brokers. The results of these experiments, displayed in Figure 5, show that Siena/B-DRP scales gracefully up to almost the full throughput in the 1, 2, and 4 brokers configurations, and to the full throughput (no undelivered

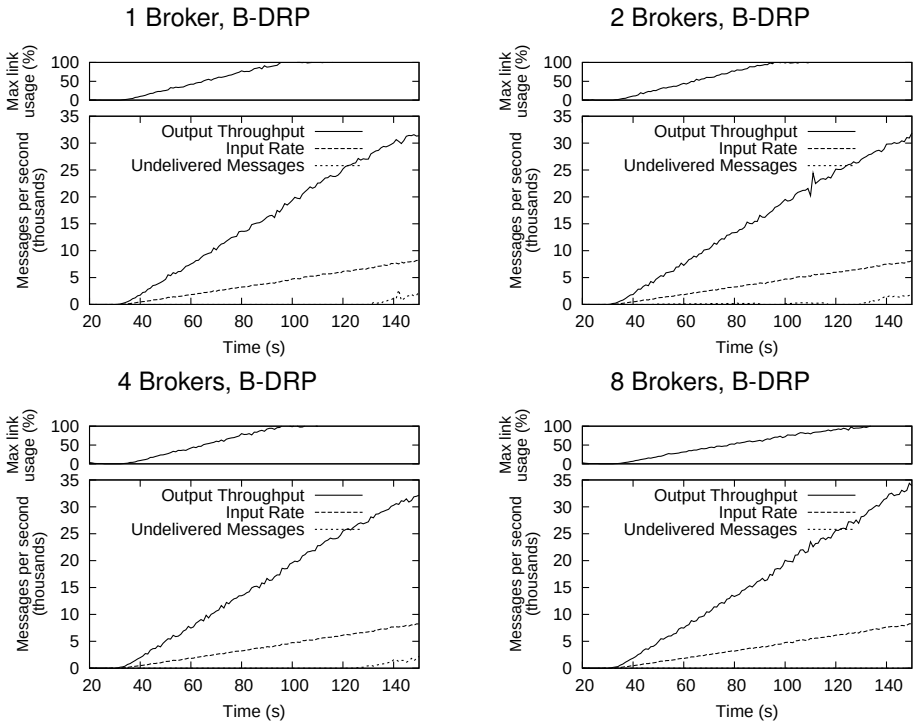

Figure 5: Siena/B-DRP with 1, 2, 4, and 8 broker networks, 5% client replication

messages) in the 8-broker network. Notice how in all cases, the maximum link utilization grows linearly with the input-rate.

To validate the results of the comparative experiments conducted with variable-rate input, and to further highlight the congestion effects incurred by ActiveMQ, we have replicated the above experiments at four input rates. The results are shown in Figure 6 (notice the different Y-axis scales). The plots clearly show how there is a clear advantage in using two brokers instead of one. However, it is also clear that the advantage of broker replication is overshadowed by instability at higher rates and by network traffic in the presence of more brokers.

5.2.2 Scalability of Siena/B-DRP

In our comparative analysis of ActiveMQ and Siena/B-DRP, Siena/B-DRP was able to sustain the highest input rates we tested. We now move away from ActiveMQ and focus on Siena/B-DRP. In particular, our goal is to test Siena/B-DRP's ability to scale to large networks of brokers, and thereby to sustain a high throughput in a wide range of scenarios.

To do that, we set up a series of experiments over networks of 42 brokers. We interconnect the brokers according to a topology generated by the BRITE generator. In this case, the diameter of the topology is eleven. Over this network, we initially distribute 420 clients, with an average of 10 clients per broker and construct a workload that maintains the same replication factor (5%) as in the previous experiments. Then, on the basis of this deployment, we construct a series of workloads by varying a number of workload parameters. In particular, we explore the parameter space, and therefore the scalability of Siena/B-DRP, in the following dimensions:

- *number of clients:* we increase the number of clients, obtaining in effect a higher replication factor;
- *number of filters:* we increase the number of subscriptions per client, therefore also increasing the replication factor;
- *filter complexity:* we increase the (average) number of constraints per filter, thereby lowering the replication factor;
- *link capacity:* we vary the maximum link capacity (uniform across network links) so as to vary the maximum achievable throughput.

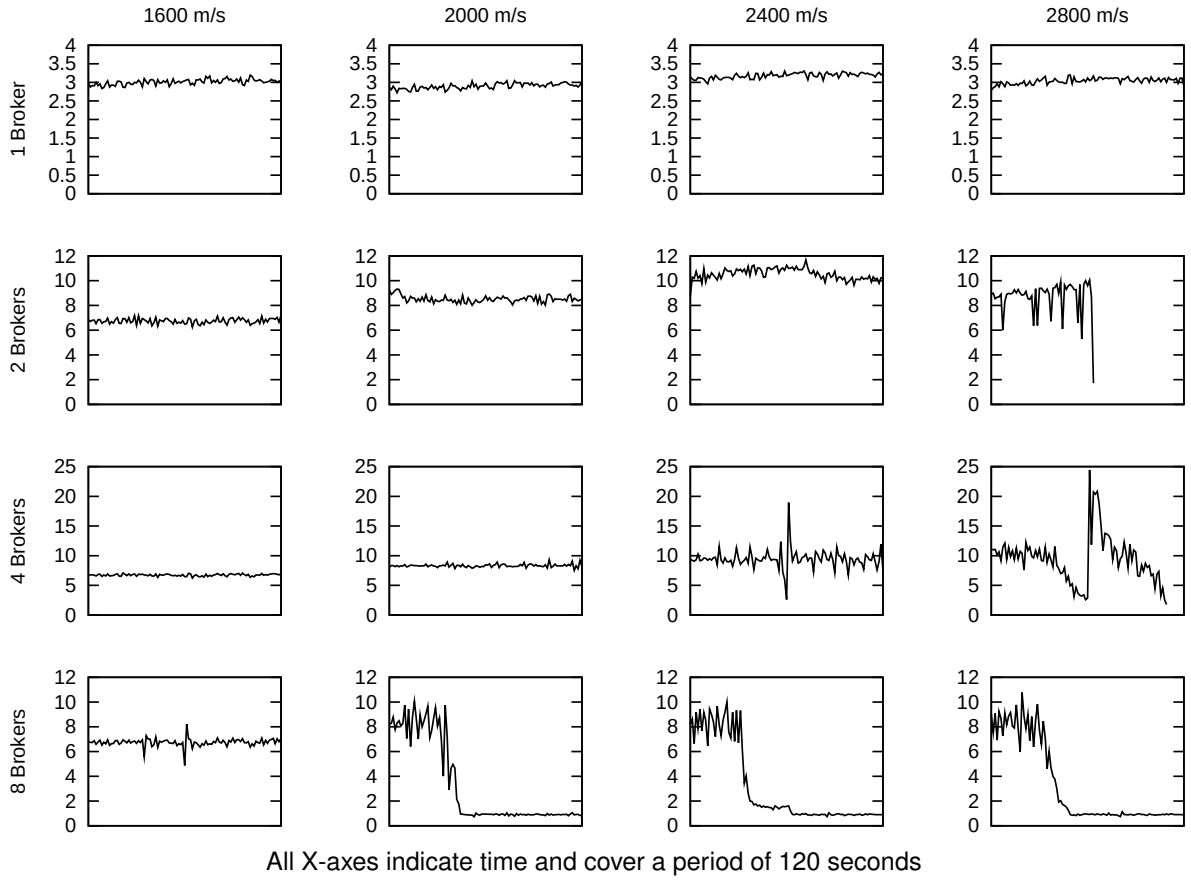

Figure 6: ActiveMQ's output throughput in thousands of messages per second at constant input rates of of 1600, 2000, 2400, and 2800 messages per second (columns) for networks of 1, 2, 4, and 8 nodes (rows)

The results of these experiments are shown in Figure 7. As in the experiments of figures 4 and 5, we have each client send message as a Poisson process at a linearly increasing rate. We then plot the output throughput, with one line for each parameter value. In addition to the observer output throughput, we plot the *ideal* output throughput. That is, the output throughput computed by multiplying the input rate by the replication factor. Notice that in some of the experiments, the replication factor varies with the chosen parameters of the experiment. In those cases, we plot one ideal-throughput line for each value of the parameters. We use these lines for comparison with the measured throughput.

At a high-level, the results of these experiments are very good. Siena/B-DRP is capable of sustaining a high throughput in almost all cases, with a graceful degradation as the input rates ramp up. There are two notable exceptions, in which Siena/B-DRP performs in a way that is markedly worse than ideal. One is the case where we increase the number of clients up to an average of 50 for each of the 42 brokers. In this case, Siena/B-DRP exhibits relatively low and unstable throughput. The other case in which Siena/B-DRP fails to scale up to a high throughput is the case in which network links are limited to 100KBps. However, this latter case is justifiable, since the limitations links capacity poses obvious limits on the throughput of any protocol.

6 Related Work

B-DRP is related to distributed publish/subscribe systems in the sense that they implement what amount to content-based routing protocols. It is also related to other protocols either designed specifically for content-based networks or for other, conceptually similar network services.

Despite the number of studies of content-based routing to be found in the literature, few protocols have been developed to a point beyond a simulation where an actual implementation could be evaluated in an extensive,

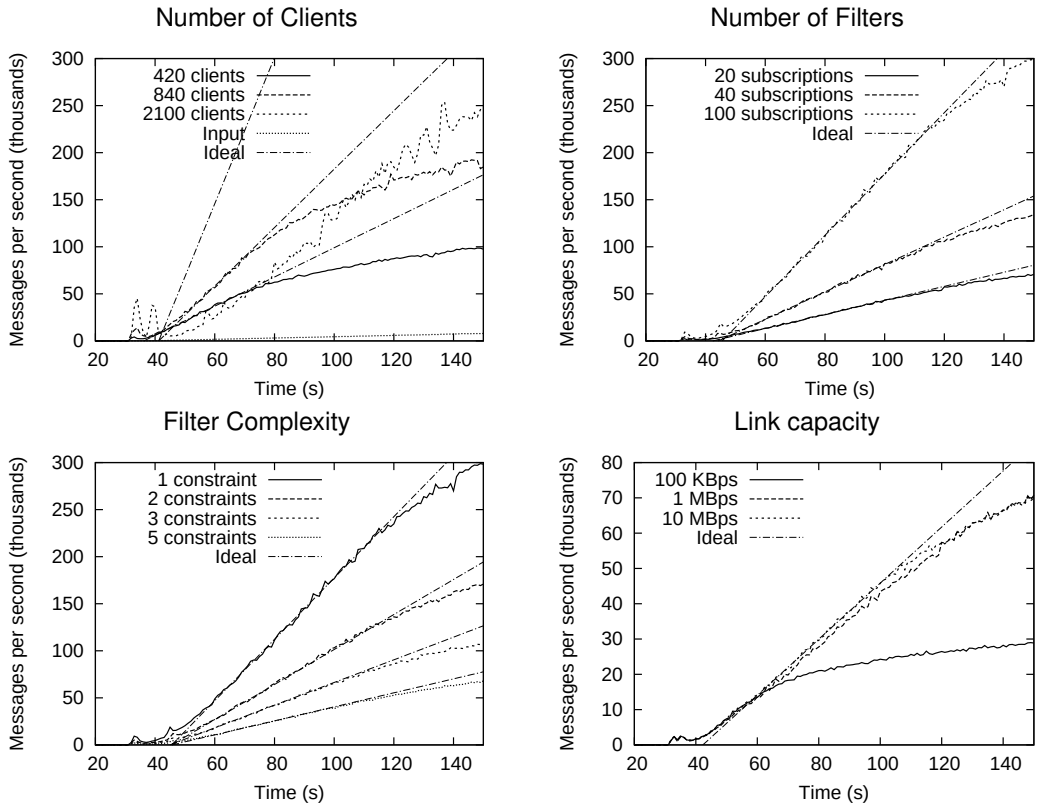

Figure 7: Siena/B-DRP Throughput varying number of clients, filters, filter complexity, link capacity

comparative experiment. In fact, we are not aware of any comparable study to the one presented here, and therefore consider this as a novel contribution of the work described in this paper.

The protocol that is most closely related to B-DRP is MEDYM by Cao and Singh [6]. MEDYM uses a routing scheme called "dynamic multicast," which is essentially identical to the dynamic receiver partitioning used in B-DRP. The main difference between B-DRP and MEDYM is that MEDYM uses ad-hoc routing algorithms to reduce the general transmission overhead and/or the overhead of destination sets, while B-DRP uses a broadcast threshold when destination sets become very large, and may use a compact representation of those sets [13]. This difference, but also the similarities between B-DRP and MEDYM should be considered in relation to the different objectives of B-DRP and also in relation to its development history. We originally developed dynamic receiver partitioning in the context of sensor networks, including a compressed representation of destination sets suitable for that application domain [13]. Then, in designing B-DRP for general-purpose wide-area networks, we preferred to focus on simplicity and practicality (and therefore modularity) of routing, and end-to-end throughput (and therefore efficiency of in-network processing). Reducing communication overhead was only a secondary goal insofar as it affects throughput.

Beyond MEDYM, it is useful to classify and compare the protocols related to B-DRP in terms of their structure and their assumptions on the underlying network. Two main classes of techniques have been explored in the literature.

The first uses explicitly defined topologies. For example, early publish/subscribe systems make this assumption [3, 8]. In these cases, they also assume an acyclic topology and have further limitations on the structure of their routing protocols. These designs were surpassed by more recent (derivative) protocols that are usable on any given network topology. Yet even these new designs make significant assumptions, either on the availability of services such as IP multicast, a broadcast protocol, or a mesh architecture [5, 9, 17, 24], or on the structure of predicates and their mapping onto corresponding network structures, for example by requiring an *a priori* partitioning of the attribute space [12, 21]. We believe that these assumptions complicate the routing protocol and certainly reduce their applicability. By contrast, B-DRP makes very reasonable assumptions on the form of predicates, and uses commonly available routing information and protocols.

The second common scheme for content-based routing amounts to routing from rendezvous points, typically implemented using a distributed hash table (DHT). This common scheme is found in various forms in systems such as Corona [23], Ferry [28], and the work of Baldoni et al. [2]. The main idea is to map constraints and attributes into the identifier space of the underlying DHT in such a way that matching constraints and attributes intersect at the same node. Message publication involves hashing each attribute in a message, and sending the message to the appropriate node (or nodes) in the DHT, from which the message is then forwarded to matching subscribers. These routing schemes are not applicable in the general content-based model we adopt, where the network topology is given, and may not allow for the full connectivity needed by a DHT.

Turning now to predicate encoding, B-DRP uses such a technique primarily for the purpose of compressing the predicates. Triantafillou and Economides developed this same idea into what they call *subscription summaries* [26]. However, although the structure of their subscriptions is almost identical to that of predicates in B-DRP, subscription summaries are quite different from B-DRP's encoded predicates. The fundamental difference is that their summaries are based on the semantic properties of the constraints and admit to an exact matching algorithm. So, for example, a summary may contain a compact representation of "arithmetic" constraints, which amounts to a list of ranges of interesting values for numeric attributes. Therefore, summaries might achieve compression through a space-efficient representation of these ranges. Interestingly, Triantafllou and Economides also propose to use Bloom filters to compress subscriptions in another paper [25]. However, they use them in a limited way within another data structure, only for representing attribute names and string values.

Jerzak and Fetzer have recently proposed the use of Bloom filters for content-based routing [16]. However, their use of Bloom filters differs significantly from B-DRP in scope as well as in essential technical aspects. In particular, Jerzak and Fetzer define a data structure to represent predicates together with a corresponding matching algorithm. This structure is used internally for the purpose of forwarding and has no relation to the process of routing.[11] Also, this structure uses Bloom filters to represent sets of individual constraints that must be used in combination with an explicit representation of the constraints as well as with an explicit representation of the attributes in a message. In practice, this means that the Bloom filters defined by Jerzak and Fetzer are not (and can not be) used to represent and compress predicates.

7 Conclusion and Future Work

We have presented a novel routing protocol, called B-DRP, and evaluated its end-to-end performance through an extensive set of experiments on a network testbed. Our goal was to design a protocol that is easy to implement and yet capable of supporting content-based publish/subscribe communication at a high throughput over a wide-area network. Our experiments show that B-DRP achieves these goals. In particular, comparing our implementation of B-DRP with ActiveMQ, which is purported to be the fastest distributed JMS implementation today, we show that it is capable of sustaining a higher throughput of messages even on relatively small networks, and that its advantage grows on larger networks. Moreover, in contrast to ActiveMQ, B-DRP exhibits a more stable behavior even in the presence of congestion. The general conclusion we draw from our experience with ActiveMQ on networks of non-trivial size is that a proper routing protocol is essential for a wide-area deployment of a publish/subscribe message system.

The development of B-DRP is part of our general efforts to develop and deploy a content-based network. Within this long-term project, and more or less related to this paper, we are pursuing several promising lines of research. First, we are continuing to explore the design space of content-based routing. On the one hand, in an attempt to make B-DRP scale far beyond its intended scope of hundreds or thousands of routers, we are developing a generalization of it for a two-level or even multi-level hierarchical protocol coupled with a corresponding clustering algorithm. On the other hand, we are studying completely different routing schemes. One such scheme is based on the use of per-source trees.

Another area of research that is related to B-DRP, and also fundamental to content-based networking in general, is privacy. In particular, we are interested in developing a privacy-preserving routing scheme—that is, a routing scheme that would allow routers to carry out their filtering and delivery function at a high throughput, but without learning anything about the interests of clients. This problem is particularly difficult, since it is by definition antithetic to the very idea of content-based networking. However, the idea of encoding predicates developed in B-DRP seems to offer a good platform for such a routing scheme.

[11] We use the term "routing" strictly to refer to the distributed algorithm that establishes paths across routers. By contrast, we use the terms "matching" and "forwarding" to refer to the local algorithm that processes and forwards messages at each router. Jerzak and Fetzer use the term "routing" more liberally referring to both algorithms.

Yet another related and important area of research in content-based networking is congestion control. Many traditional end-to-end techniques for congestion control do not seem applicable in this context, since the essential notion of a connection or session is hard to define in content-based networking. Nevertheless, we are considering ways to incorporate rate limitations into predicates, and to combine them to implement per-link congestion control within the network.

Finally, returning to the original theme of this paper, we are nearing completion of a modified version of ActiveMQ that incorporates B-DRP.

References

[1] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. Resilient overlay networks. In *18th ACM Symposium on Operating Systems Principles (SOSP '01)*, pages 131–145, Banff, Canada, Oct. 2001.

[2] R. Baldoni, C. Marchetti, A. Virgillito, and R. Vitenberg. Content-based publish-subscribe over structured overlay networks. In *25th IEEE International Conference on Distributed Computing Systems (ICDCS '05)*, pages 437–446. IEEE Computer Society, 2005.

[3] G. Banavar, T. D. Chandra, B. Mukherjee, J. Nagarajarao, R. E. Strom, and D. C. Sturman. An efficient multicast protocol for content-based publish-subscribe systems. In *19th IEEE International Conference on Distributed Computing Systems (ICDCS '99)*, pages 262–272, Austin, Texas, May 1999.

[4] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, July 1970.

[5] F. Cao and J. P. Singh. Efficient event routing in content-based publish-subscribe service networks. In *IEEE Conference on Computer Communications (INFOCOM '04)*, pages 929–940, Hong Kong, China, Mar. 2004.

[6] F. Cao and J. P. Singh. Medym: match-early with dynamic multicast for content-based publish-subscribe networks. In *Middleware '05: Proceedings of the ACM/IFIP/USENIX 2005 International Conference on Middleware*, pages 292–313, New York, NY, USA, 2005. Springer-Verlag New York, Inc.

[7] A. Carzaniga, A. J. Rembert, and A. L. Wolf. Understanding content-based routing schemes. Technical Report 2006/05, Faculty of Informatics, University of Lugano, Sept. 2006.

[8] 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, Aug. 2001.

[9] A. Carzaniga, M. J. Rutherford, and A. L. Wolf. A routing scheme for content-based networking. In *IEEE Conference on Computer Communications (INFOCOM '04)*, pages 918–928, Hong Kong, China, Mar. 2004.

[10] A. Carzaniga and A. L. Wolf. Content-based networking: A new communication infrastructure. In *NSF Workshop on an Infrastructure for Mobile and Wireless Systems*, number 2538 in Lecture Notes in Computer Science, pages 59–68, Scottsdale, Arizona, Oct. 2001. Springer-Verlag.

[11] A. Carzaniga and A. L. Wolf. Forwarding in a content-based network. In *Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM '03)*, pages 163–174, Karlsruhe, Germany, Aug. 2003.

[12] S. Castelli, P. Costa, and G. P. Picco. HyperCBR: Large-scale content-based routing in a multidimensional space. In *IEEE Conference on Computer Communications (INFOCOM '08)*, pages 1714–1722. IEEE Computer Society, 2008.

[13] C. P. Hall, A. Carzaniga, J. Rose, and A. L. Wolf. A content-based networking protocol for sensor networks. Technical Report CU-CS-979-04, Department of Computer Science, University of Colorado, Aug. 2004.

[14] R. Henjes, M. Menth, , and V. Himmler. Throughput performance of the ActiveMQ JMS server. In *ITG/GI Symposium on Communication in Distributed Systems (KiVS)*, Bern, Switzerland, Feb. 2007.

[15] R. Henjes, M. Menth, and V. Himmler. Impact of complex filters on the message throughput of the ActiveMQ JMS server. *Lecture Notes In Computer Science*, 4516:192–203, 2007.

[16] Z. Jerzak and C. Fetzer. Bloom filter based routing for content-based publish/subscribe. In *DEBS '08: Proceedings of the Second International Conference on Distributed Event-Based Systems*, pages 71–81, Rome, Italy, July 2008.

[17] P. Ji, Z. Ge, J. Kurose, and D. Towsley. Matchmaker: Signaling for dynamic publish/subscribe applications. In *11th IEEE International Conference on Network Protocols (ICNP '03)*, pages 222–233, Nov. 2003.

[18] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, Apr. 2004.

[19] M. Menth, R. Henjes, C. Zepfel, and S. Gehrsitz. Throughput performance of popular JMS servers. *ACM SIGMETRICS Performance Evaluation Review*, 34(1):367–368, June 2006.

[20] M. Mitzenmacher. Compressed Bloom filters. In *Proceedings of the Twentieth Annual ACM Symposium on Principles of Distributed Computing (PODC '01)*, pages 144–150, New York, NY, USA, Aug. 2001. ACM.

[21] L. Poutievski, K. L. Calvert, and J. N. Griffioen. Speccast. In *IEEE Conference on Computer Communications (INFOCOM '04)*, pages 2263–2274, Hong Kong, China, Mar. 2004.

[22] Progress Software. Performance evaluation and recommendations for ActiveMQ/FUSE message broker, Sept. 2008.

[23] V. Ramasubramanian, R. Peterson, and E. G. Sirer. Corona: A high performance publish-subscribe system for the world wide web. In *3rd Conference on Networked Systems Design & Implementation (NSDI '06)*, pages 15–28, San Jose, California, 2006. USENIX Association.

[24] A. C. Snoeren, K. Conley, and D. K. Gifford. Mesh-based content routing using XML. In *Eighteenth ACM Symposium on Operating Systems Principles (SOSP '01)*, pages 160–173. ACM Press, Oct. 2001.

[25] P. Triantafillou and A. Economides. Subscription summaries for scalability and efficiency in publish/subscribe systems. In *International Workshop on Distributed Event-Based Systems (DEBS '02)*, pages 619–624, Vienna, Austria, July 2002.

[26] P. Triantafillou and A. Economides. Subscription summarization: A new paradigm for efficient publish/subscribe systems. In *24th International Conference on Distributed Computing Systems (ICDCS '04)*, pages 562–571, Washington, DC, 2004. IEEE Computer Society.

[27] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An integrated experimental environment for distributed systems and networks. In *5th Symposium on Operating Systems Design and Implementation (OSDI '02)*, pages 255–270, Boston, Massachusetts, Dec. 2002. USENIX.

[28] Y. Zhu and Y. Hu. Ferry: A P2P-based architecture for content-based publish/subscribe services. *IEEE Transactions on Parallel and Distributed Systems*, 18(5):672–685, May 2007.