Fast and Flexible Forwarding for Internet Subscription Systems

Joanna Kulik
Advanced Network Architectures Group
MIT Laboratory for Computer Science
jokulik@lcs.mit.edu

ABSTRACT

This paper analyzes mechanisms for addressing and forwarding notifications in Internet subscription systems. The first section of the paper focuses on existing subscription systems, including single-identifier and content-based multicast approaches. It evaluates these systems on issues of subscription complexity, application-level flexibility, and efficiency and finds that both systems run into problems in these areas. The second section of the paper introduces an alternative to current approaches, called match-structure forwarding. In this approach, routers forward each message via a structure contained in the header of each message. These structures are specifically to reflect a variety of complex applications as well as for efficient forwarding by network routers. This paper proposes two alternative header formats for match-structure forwarding, called content lists and content graphs. The third section of the paper presents results from experiments that compared the performance of content lists and content graphs with other subscription systems. Results show that the match-structure forwarding approaches outperform other approaches in applications with large numbers of subscribers and overlapping subscription categories. Overall, these results suggest that match-structure forwarding systems are a promising development in the area of Internet Subscription Systems.

1. INTRODUCTION

The Internet is, among other things, the greatest library of information ever built. But information is not meant just for storage on library shelves. To be useful, information must get into people's hands and heads, and it must often get there quickly. Today, therefore, researchers are working to expand Internet functions to include automatic and immediate routing of incoming Internet information to those who need it.

The mechanisms that are now being developed can be grouped together under the heading of Internet Subscription Systems. Put simply, Internet Subscription Systems are distributed mechanisms for notifying subscribers as quickly as possible to the arrival of relevant information on the Internet. Subscribers first sign up for notifications on topics that are important to them. As soon as relevant information on a topic arrives on the Internet, it is routed to the appropriate subscribers. This information can include news alerts, traffic announcements, location-tracking information, weather reports, and many other notices. Subscription systems can also support large-scale interactive games and transmission of on-line entertainment events.

This paper addresses the question of how forwarding mechanisms for subscription systems should be designed. A subscription system's addressing scheme is simply the format of its subscription and notification messages. A subscription system's forwarding scheme comprises a set of rules for matching subscription and notification addresses. When IP multicast is used to disseminate notifications to subscribers, for example, every subscription and notification carries a single address, which is a fixed-length, numeric identifier. The IP multicast forwarding scheme dictates that a subscription matches a notification if and only if they both carry the same identifier.

Existing approaches to addressing and forwarding in distributed Internet Subscription Systems can be divided into two broad categories: (a) single-identifier multicast systems, which send messages to discrete message channels that subscribers with identical interests may subscribe to; and (b) content-based multicast systems, which forward messages based on the text content of the messages. No one now knows which of these systems will best meet the needs of Internet users.

The first major purpose of this paper is to evaluate the strengths and weaknesses of these two approaches. It evaluates them in terms of both ease of use and network cost. Questions arise on all sides. In terms of ease of use, can they handle subscription requests involving complex, evolving, and overlapping categories of subscriber interest? In terms of network cost, are there inherent limitations on the speed of forwarding messages? How heavy a load do these systems place on network resources?

I conclude that neither of the current approaches handles all of these issues well. Single-identifier multicast systems can quickly distribute notifications to large numbers of subscribers, but they have trouble handling complex subscription categories. Content-based systems are scalable, and they work with complex subscription categories. Unfortunately, content-based systems are also expensive to mount, modify, and run.

The second major purpose of this paper is to present an alternative approach to addressing and forwarding in Internet Subscription Systems, called match-structure forwarding. In this approach, all subscription and notification messages are first processed by special match-structure processors before being submitted to the subscription system. Match-structure processors identify those features in each message's content that are relevant to subscription forwarding. They then store this information in a data-structure that (a) ensures that each subscription will match the correct notifications and (b) enables routers to match these messages efficiently. These match-structures are then attached to the headers of each message and serve as the sole basis for subscription forwarding.

Like single-identifier multicast systems, match-structure forwarding systems separate the message used by applications from the the message format used by routers. These systems can therefore support a variety of complex application formats at the edge of the network using a single, efficient message format within the subscription system itself. The difference between match-structure forwarding and single-identifier multicast systems, however, is that match-structures are specifically designed to capture the complexity of real-world subscriptions and notifications. In this way, match-structure systems resemble content-based multicast systems as well.

This paper also proposes two alternative approaches to match-structure forwarding systems, called content lists and content graphs. A content list is simply a list of numeric identifiers that indicate the subscription topics that a particular notification covers. Using content lists, subscription systems can disseminate overlapping subscription categories without duplicating messages. Content graphs are partially ordered digraphs that represent relationships among topic identifiers. By storing content graphs, routers can infer relationships between subscriptions and notifications, further improving their efficiency.

The third purpose of this paper is to evaluate the performance of subscription systems that use match-structure forwarding to disseminate notifications. The paper presents results gathered from experiments performed on both a simulated network as well as an actual router. In each of these experiments, notification sources used content lists, content graphs, and single-identifiier multicast forwarding to disseminate two types of announcements: sports scores and traffic alerts. Results show that both content-list and content-graph systems outperform single-identifier multicast systems in applications with large numbers of subscribers and overlapping subscription categories. Overall, these results suggest that match-structure forwarding systems are a promising development in the area of Internet Subscription Systems.

The rest of this paper describes match-structure forwarding systems in further detail. Section 2 describes the central problem addressed by this paper, the design of addressing mechanisms for Internet Subscription Systems, and analyzes previous approaches to this problem. Section 3 describes a novel alternative to these mechanisms that uses effcient match-structures, carried on the headers of messages, to disseminate notifications. It proposes two specific formats for such structures, called content lists and content graphs. Section 4 compares these two approaches to single-identifier multicast systems through experimental results. Finally, Section 5 summarizes findings and discusses future directions for research.

2. ANALYSIS OF CURRENT APPROACHES

The purpose of this section is to evaluate the strengths and weaknesses of two existing approaches to forwarding messages in Internet Subscription Systems: single-identifier and content-based multicast systems. It examines how well each approach handles three major challenges:

- Complexity of subscription requests. Subscriptions should reflect the requirements of real-world applications.
- Application-level flexibility. The system should evolve with changes in applications.
- Forwarding efficiency. Routers should be able to direct messages efficiently from incoming links to outgoing links. Note that forwarding efficiency is separate from routing efficiency, which refers to a system's ability to create efficient topologies for disseminating messages.

Although subscription systems can function without meeting all three of these challenges, an ideal subscription system would have each of these characteristics.

2.1 Single-Identifier Multicast Systems

IP multicast is well-suited to real-time, group communication because it can deliver publications to large groups of subscribers using the minimum number of messages. Theoretically, an IP multicast system should be able to send only a single copy of a publication over any given link in the network while disseminating the publication to hundreds and millions of users. Since its original proposal, several Internet Subscription Systems have been proposed that build on the IP multicast model of communication. Examples of such systems include Orbixtalk, TIBCO, early Gryphon [8], Herald [5], and Scribe [12]. Though these systems differ in many respects, they have one trait in common. In these systems, each subscription and notification request carries a single, numeric identifier, and routers use these identifiers to disseminate messages. For this reason, I refer to this general category of subscription system as *single-identifier multicast* systems.

Single-identifier multicast systems meet two of the three criteria for Internet subscription systems. First, because these systems do not process application-level information, developers can change application message formats without any modification to routers in the network. Second, the cost of forwarding a message in such systems is comparable to the cost of forwarding a message in a unicast system. In order to retrieve the list of interfaces to which a notification should be forwarded, a multicast router simply needs to look up a fixed-length integer address in a table.

It is in the area of complex subscriptions that single-identifier multicast systems run into problems. This problem is often referred to as the IP multicast channelization problem, and occurs when subscribers request information from overlapping subscription categories, as illustrated in Figure 1. For example, three subscription categories currently available from the New York Times News Tracker are: "Baseball," "New York," and "Boston." If the News Tracker were to disseminate notifications for these categories using

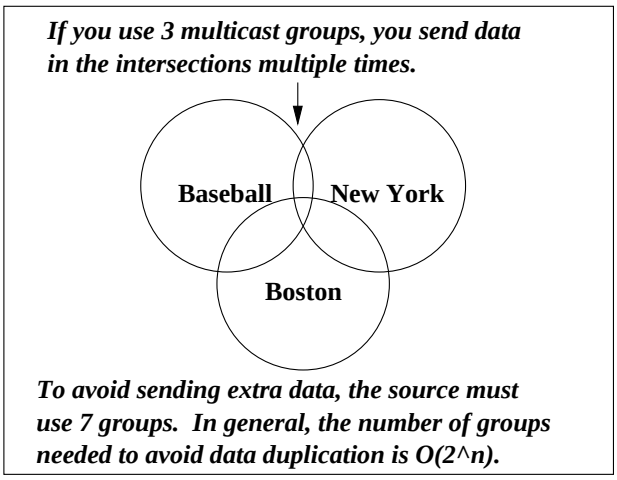


Figure 1: The IP Multicast channelization problem.

only three multicast identifiers, subscribers with interests in two or more of the categories would receive multiple notices when a news story was relevant to all three categories. The News Tracker could solve this problem by assigning a separate multicast identifier to each of the 7, disjoint subcategories covered by the three larger categories. In the general case, however, applications would need $O(2^n)$ multicast identifiers to disseminate n categories of notifications without any duplication. The News Tracker could try to optimize its use of multicast identifiers by assigning identifiers to only the most active news categories. Unfortunately, this optimization problem has been shown to be NP-hard [1].

2.2 Content-Based Multicast Systems

Content-based systems resemble traditional single-identifier multicast systems in many respects. They use the same approaches to topology-formation and routing as single-multicast systems do, for example. The difference between the two kinds of systems is that content-based multicast systems are designed specifically to disseminate text information to large numbers of subscribers. They therefore forward messages based on the application-level, text content of the message, rather than a single identifier contained in the header of the message. Current content-based systems include SIENA [7], the mesh-based XML routing system [13], Gryphon [8], Hermes [10], and Le Subscribe [11].

Subscribers in content-based multicast systems first describe the notifications of interest to them in subscription requests. When a content-based router receives a notification, it then matches the notification's contents against the subscriptions it has received and forwards the notification to the appropriate subscribers. Content-based multicast systems usually require that applications format their messages using a pre-defined, structured format, such as attribute-value pairs, as illustrated in Figure 2.

Content-based multicast systems therefore get high marks on one of the three criteria laid out in the beginning of this section. By imposing structure on messages, content-based systems make it easier for routers to match complex subscriptions with notifications. They are therefore able to handle such subscriptions while avoiding the IP multicast channelization problem.

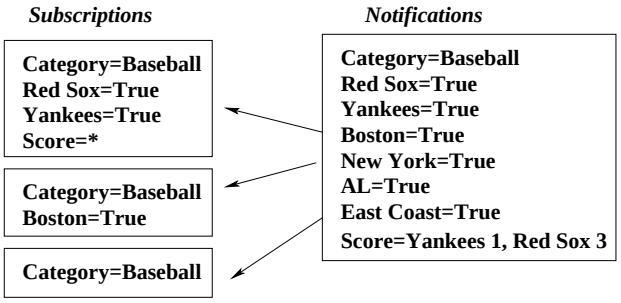


Figure 2: Content-based subscriptions and notifications formatted as attribute-value pairs.

Content-based systems get lower marks when measured on the other two criteria, however. First, because content-based multicast systems forward messages based on application-level formats, changes in applications sometimes necessitate changes in routers. For example, a subscription system that matches notifications based on a standard Julian calendar might need to handle transactions on a business calendar. Augmenting a content-based system to recognize business as well as Julian dates would entail making changes to every router in the system. In general, the cost of maintaining content-based systems will be high because the machinery of content-based multicast systems must be replicated on all the routers at which forwarding decisions are made.

The second major problem with content-based multicast systems is its forwarding complexity. Whenever a content-based router receives a message, it must read the entire message. It must then parse application-level text expressions, check syntax, and possibly support other language features, such as type-checking and regular expression matching. Compared to single-identifier multicast routers, content-based routers spend a significant amount of time performing application-level tasks. In one XML-based routing system, in fact, "packet processing cost is dominated by XML parsing time" [13]. The significance of the problem is perhaps most easily measured by the number of research papers that focus exclusively on alleviating it [2, 11, 4, 9, 3].

3. MATCH-STRUCTURE FORWARDING

Though it is clear that many applications need to disseminate time-critical data to large numbers of users, it is not clear that either single-identifier or content-based multicast systems adequately achieve this goal. Single-identifier headers do not seem rich enough to express the complex relationships that exist between real-world subscription topics. Content-based multicast systems, which support application-level features directly at the router-level, are undoubtedly more expressive than single-identifier systems. However, content-based systems may sacrifice features such as flexibility and efficiency. The goal of this paper is to develop a single system that provides applications with all of these features.

The question then remains: How is it possible to design a system that supports complex, application-level features without adding those features directly to routers in a subscription system? In answer to this question, I propose a new approach to subscription forwarding, called match-

structure forwarding. In this approach, each application first submits its message to a match-structure processor. Match-structure processors are not part of the subscription system itself. They are application-specific processors and may consist of one or more distributed processors. Match-structure processors take notification and subscription messages as input and generate match-structure headers as output. Match-structures are special data-structures that indicate to routers how to match subscriptions with the correct notifications. Match-structures are also designed specifically for efficient processing by routers. They do not contain application-level information, such as string names, type definitions, and text syntax. Applications attach match-structures to the headers of messages, and these match-structures serve as the sole basis for forwarding within the subscription system.

Match-structure forwarding systems lie half-way between single-identifier and content-based multicast systems. Like single-identifier multicast systems, match-structure forwarding systems route messages based on router-level headers only, rather than application-level message content. Unlike in single-identifier system, however, match-structure headers are designed to express the complexity of real-world subscription categories. In this way, match-structure forwarding systems also resembles content-based systems. Unlike content-based systems, match-structure forwarding systems process application-level information at the edges of the network, not within the network itself. Using this hybrid approach, match-structure systems are able to achieve the flexibility and efficiency of a single-identifier multicast systems as well as the expressiveness of a content-based system.

The rest of this section presents two alternative formats for match-structure headers, content lists and content graphs, and discuss how they may benefit future subscription systems.

3.1 Content Lists

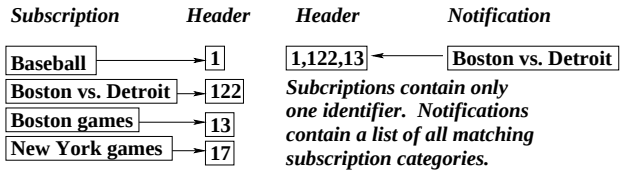


Figure 3: Baseball subscriptions and notifications and their corresponding content lists.

A *content list* is simply a list of abstract identifiers. Figure 3 illustrates baseball subscriptions and notifications and their corresponding content lists. In this figure, the match-structure processor assigns each subscription topic its own unique, integer identifier. Whenever a source sends out a notification, it then attaches a list of identifiers to the notification, specifying all of the subscription topics that match the notification. A notification matches a subscription if the subscription's identifier is listed in the notification's content list.

In practice, the only difference between a content list and a single-identifier multicast header is that a content list may contain multiple identifiers, whereas a single-identifier header may contain only one. This small difference between the two formats may have a large effect on the performance of subscription systems, however. Specifically, content lists provide more efficient support for overlapping subscription topics than single-identifier multicast headers. When a notification source generates a message matching multiple subscription topics using content lists, every link in the network needs to carry only a single copy of the message, rather than multiple copies of the message.

3.2 Content Graphs

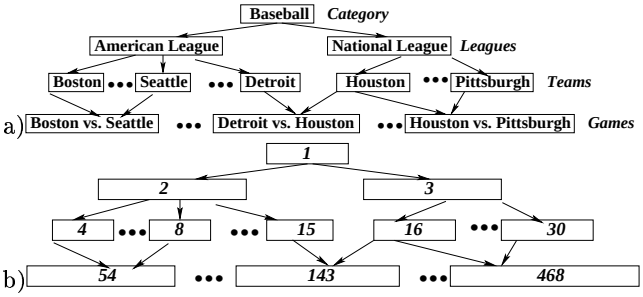
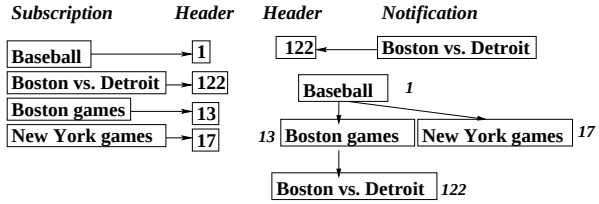


Figure 4: A partial-ordering of baseball announcement labels (a) and their corresponding content graph (b).



Using the content graph at the left, a router can determine that all of the subscriptions above match the given notification. Routers need only store graph information for the subscriptions that they have received.

Figure 5: Baseball subscriptions and notifications and their corresponding content graphs.

The main drawback of using content lists in subscription systems is that content lists may contain many identifiers, one identifier for every subscription category that matches a given message. To overcome this drawback, I propose that routers store an additional data-structure, called a *content graph*, to reduce the length of content lists. A content graph is a directed, acyclic graph, or digraph. Each node in the graph represents a set of data streams. Each edge in the graph represents the relationship between two of these sets. Specifically, these graphs maintain the following invariant: If the set of streams represented by node A is a superset of the set of streams represented by node B, then there exists a path from A's node to B's node in the content graph. Figure 4 (a) illustrates how a set of baseball topics might be arranged into a partial-ordering. Each node in the graph represents a subscription topic, such as "Baseball" or "Boston vs. Seattle". Nodes at the top of the graph represent the most general topics, and nodes at the bottom represent more specific categories. Figure 4 (b) depicts the corresponding content graph for these announcements.

Figure 5 illustrates how routers would match baseball subscriptions and notifications using content graphs. Just as in Figure 3, the match-structure processor assigns each subscription category its own, unique identifier. Each router in the system stores a content graph for the subscriptions it has received. Whenever a source sends out a notification, it attaches a list of identifiers to the notification. These notifications correspond to the lowest nodes in the content graph that match the notification's contents. Whenever a router in the system receives a notification, it looks up the nodes in the graph listed in the header of the notification. If it has received a subscription for nodes in its graph from one of its neighbors, it will forward the message to its neighbor, listing the matching nodes. Routers do not need to store graph nodes corresponding to every possible subscription category. A router need store only the nodes in the graph corresponding to its current downstream subscribers.

Content graphs have two key features. First, when routers store content graphs, they can infer that a single notification identifier matches many subscriptions. Content-graph headers may therefore be shorter, and faster to process, than their corresponding content-list headers. Second, applications can translate a variety of application-level message formats into content graphs. Most current content-based routing systems format their messages using attribute-value pairs [13, 6, 4], for example. Applications can automatically order attribute-value pairs using a simple set of rules [6] and this ordering can be used to generate content graphs. Applications can also format any hierarchical categorization of message labels, such as the news categories depicted in Figure 4 (a), as a content graph. Even applications that cannot order their messages can use content graphs, since an unordered set of messages is simply a content graph with no edges.

4. EXPERIMENTAL EVALUATION

To evaluate the performance of match-structure forwarding systems, I carried out two experiments. Each experiment compared the performance of match-structure systems to that of single-identifier multicast approaches. The experiments addressed two questions: (a) How do different Internet subscription systems affect the amount and nature of router traffic? (b) How do these systems affect processing time at routers?

4.1 Experiment I

The first experiment examined the performance of match-structure forwarding systems to other systems in a simulated network environment. The purpose of the experiment was to determine how these systems affect the quantity and characteristics of network traffic.

4.1.1 Method

The simulator used in this experiment was the `ns` simulator. Developed as a variant of the Real network simulator in 1989, `ns` has evolved into a public-domain simulator that is widely used in network research. Maintained and updated by a large user base and a small group of developers at the Information Sciences Institute at the University of Southern California, it supports simulation of TCP, routing, and multicast protocols over wide and wireless networks.

Simulation scenarios. To carry out the simulations, I created a 1200-node, transit-stub network using the GT-ITM topology generator. I set up two scenarios for this experiment. In each scenario, a subscriber at each node in the network signed up for notifications from a single source, and a network source that provided notifications relevant to the subscribers.

The first scenario involved simulation of baseball game announcements from a news source. Each subscriber signed up for announcements on either a single team (e.g., Boston Red Sox), a pair of teams (e.g., Boston Red Sox and Detroit Tigers), or for all teams. Subscribers made their choices randomly from a total of 30 teams, with 40% signing up for notices on a single team, 50% signing up for notices on a pair of team, and 10notification from the source covered a specific pair of teams, with each pair being equally likely to be subject of a notification.

The second scenario called for traffic alerts for a single highway with 20 exits. Subscribers signed up for alerts between two randomly selected exits (e.g., Exits 12 through 14), and each notification was relevant to a randomly selected stretch of highway (e.g., Congestion between Exits 5 through 12). A notification was relevant to a subscription if it covered any portion of highway specified in the subscription.

The source then sent out 1000 notifications. Routers in the network distributed these notifications using a shortest-path dissemination tree. These simulations did not contain any network losses or queuing delays. I ran each scenario 20 times using the same topology, each time with a different, random notification source.

Subscription Systems. The performance of match-structure forwarding systems was the main focus of this experiment, but baseline comparisons were needed to interpret performance data. I therefore compared match-structure systems with two single-identifier multicast methods. I did not include content-based routing systems in my simulations. This is because of the extensive computational resources required to simulate large-scale, content-based multicast systems.

In all, I compared the performance of four approaches: two multicast approaches and two match-structure forwarding approaches. These approaches were as follows:

- Single-identifier multicast with overlapping subscription categories. Subscribers sign up for notifications, and each unique subscription receives its own identifier, regardless of whether different categories overlap. For example, in the baseball simulation, a subscription for notifications of Red Sox results would receive one identifier; a subscription for notifications of both Red Sox and Tigers results would receive another identifier; a subscription for notification of all baseball results would receive still another. Notifications (e.g., a Red Sox vs. Tigers score) would carry only a single identifier. When a notification matched more than one topic, therefore, the source would have to generate multiple copies of the notification, one for each topic, as illustrated in Figure 6.

- Single-identifier multicast with unique notification categories. In this system, all notification topics are broken into disjoint sub-categories and assigned a unique identifier. When a subscriber signs up for notifications,

it must identify the composite list of all sub-categories that the desired notifications cover. Each notification then carries a single identifier, indicating the disjoint topic that the notification covers. For example, the subscriber would translate a subscription for all baseball results into the list of identifiers that covered all possible baseball games. Each notification would carry only a single identifier, corresponding to a particular game, as illustrated in Figure 7.

- Match-structure forwarding using lists. This approach is described in Section 3.1 and illustrated in Figure 3.
- Match-structure forwarding using graphs. This approach is described in Section 3.2 and illustrated in Figures 4 and 5.

Though neither of the single-identifier multicast approaches simulated is very sophisticated, in both approaches, the algorithm for assigning identifiers to the topic space is known. Recall that the general problem of selecting an optimal set of identifiers for a single-identifier multicast system is NP-complete [1].

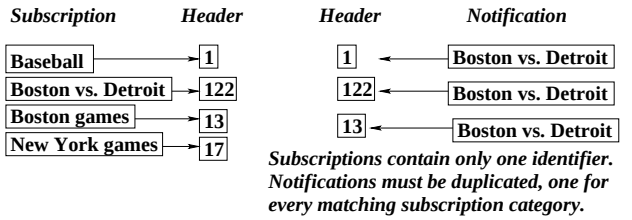


Figure 6: Baseball subscriptions and notifications and their corresponding headers using a single identifier for each overlapping subscription category.

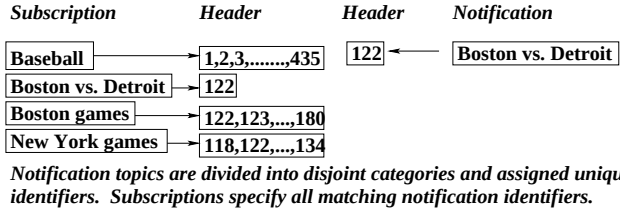


Figure 7: Baseball subscriptions and their corresponding headers using a single identifier for each disjunctive subscription category.

Performance Measures. For both simulations, I measured (a) the number of notifications sent per link, (b) the number of identifiers stored per routing table, (c) the number of identifiers contained in each subscription header, and (d) the number of identifiers contained in each notification header. I tabulated results in two ways: average result on each measure for the whole network and average result for the most heavily loaded router in the network.

4.1.2 Results

Simulation results appear in Table 1. As these results show, single-identifier systems with disjunctive notifications were economical in notification production. They did not over-produce messages as the multicast method with overlapping categories did. Single-identifier methods with disjunctive notifications did not perform well, however, on Ids per subscription. Ids per subscription were 50 to 100 times higher for multicast methods with disjunctive notifications. Results for Ids per table were similar. Single-identifier methods with disjunctive notifications produced far more Ids per table at the typical router than the other methods did. These indicators suggest that with large-scale implementations, multicast methods with disjunctive notification categories would put unnecessary pressure on the Internet namespace.

Match-structure forwarding systems that use lists perform well under most conditions, but there is one striking exception. In the traffic simulation, the list-based system produced inordinately long lists of Ids for some headers. At the most heavily loaded router, the average number of Ids per notification is approximately 87. Clearly, list-based headers can become very long in complex problems, and this puts a limit on the utility of this approach to notification systems.

The bottom line is that a graph-based, match-structure forwarding system avoids the pitfalls of the other approaches. This approach does not clog a network with unnecessary messages, as multicast systems with overlapping subscription categories do. Nor does a graph-based system put unnecessary pressure on the Internet namespace, as multicast systems do. Finally, graph-based systems do not produce headers of unnecessary length, as list-based systems do. These pitfalls may not be disabling in a network system that involves notices about 30 baseball teams or a two- or three-mile stretch of highway. But these flaws could be fatal in a global system of play-by-play announcement of sports events or a global system of traffic alerts.

4.2 Experiment II

The second experiment examined processing times for match-structure forwarding and other subscription systems on actual routers. The purpose of this experiment was to determine the cost of processing subscriptions and notifications.

4.2.1 Method

I have developed a working router implementation for each of the four subscription systems described above. Each router is implemented in C++ and runs in user space. For each experiment, I executed a single router process on a 1.4 GHz Athlon processor running the RedHat Linux 7.0 operating system.

Simulation scenarios. I used the results from the two simulations in Experiment I to generate sample workloads for this experiment. I then used these sample workloads to drive a single router for each of the subscription systems studied. Because the notifications simulated in Experiment I did not carry payloads, each notification message in this experiment was modified to carry a 64-byte, dummy payload.

Subscription Systems. For this experiment, I compared the performance of the same four subscription systems studied in Experiment I.

Format	Typical Router		Most Heavily Loaded Router	
	Baseball	Traffic	Baseball	Traffic
Notifications per link				
Overlapping-Id	312.7	1870.0	1299.8	61,423.5
Disjunctive-Id	245.5	619.5	1000.0	1000.0
Content lists	245.5	619.5	1000.0	1000.0
Content graphs	245.5	619.5	1000.0	1000.0
Ids per subscription				
Overlapping-Id	1.0	1.0	1.0	1.0
Disjunctive-Id	50.8	118.4	50.7	118.5
Content lists	1.0	1.0	1.0	1.0
Content graphs	1.0	1.0	1.0	1.0
Ids per table				
Overlapping-Id	4.5	3.9	361.2	118.0
Disjunctive-Id	103.6	145.8	465.0	210.0
Content lists	4.5	3.9	361.2	187.6
Content graphs	4.6	3.9	361.2	187.6
Ids per notification				
Overlapping-Id	1.0	1.0	1.0	1.0
Disjunctive-Id	1.0	1.0	1.0	1.0
Content lists	1.2	3.1	3.3	86.7
Content graphs	1.0	1.5	1.2	4.2

Table 1: Performance of 4 Internet Subscription Systems under Four Simulation Conditions

Performance Measures. I measured the total time that it took the router to process each sample workload. Specifically, I measured the total time for processing subscriptions and the total time for processing notifications when the router was handling the load at a typical (or average) router and when it was handling the load at the most heavily loaded router.

4.2.2 Results

The average total subscription and notification processing times appear in Table 2. Times for processing notifications are very high, especially at heavily loaded routers, for both overlapping-ID systems and list-based systems. Disjunctive-ID systems do not require so much time for handling notifications, but these systems do require large processing times for handling subscriptions. Like disjunctive-ID systems, graph-based systems require relatively little time for handling notifications but require extra time for handling subscriptions. The extra time needed for handling subscriptions is not nearly so great, however, as the extra time needed by disjunctive-ID systems.

Note that these results measure the *total* time that it took each router to process the experimental workload. The speed at which these systems processed notification messages can be inferred from these results. For example, the list-based router processed 1000 traffic announcements in 2727.2 ms, on average. Its notification processing speed for traffic announcements is therefore 2.7 ms/notification, or 370 notifications/sec. In contrast, the speed at which the graph-based router processed the same announcements is 217 μs/notification, or 4608 notifications/sec. The total throughput of each router, measured in bits per second, depends upon the size of each notification's payload. When each notification carries a 64-byte payload, as they did in these experiments, then a speed of 4608 notifications/sec corresponds to a total throughput of 2.4 Mbps. If the payload of each notification increased, then the total throughput of each router in bits/second would also increase proportionally.

Format	Typical Router		Most Heavily Loaded Router	
	Baseball	Traffic	Baseball	Traffic
Total time to process notifications (ms)				
Overlapping-Id	20.7	37.3	235.7	5753.0
Disjunctive-Id	26.0	46.6	85.5	76.9
Content lists	20.6	47.4	162.7	2727.2
Content graphs	20.1	43.0	99.1	217.6
Total time to process subscriptions (ms)				
Overlapping-Id	0.2	0.2	36.9	33.0
Disjunctive-Id	22.7	22.7	687.5	1139.2
Content lists	.2	.2	36.9	33.0
Content graphs	.9	.3	289.5	138.8

Table 2: Processing Times for 4 Internet Subscription Systems under Four Simulation Conditions

5. CONCLUSIONS

Neither single-identifier nor content-based multicast systems provide applications with all of the features that they need. The design of single-identifier multicast systems does not take into account the fact that real-world subscription categories overlap. As a consequence, single-identifier multicast systems must either use large routing tables or waste network bandwidth. Though content-based multicast systems do take application structure into account, they sacrifice other features that single-identifier multicast systems support. Routers in these systems operate at the application-level, potentially making them more complex and less efficient than conventional routers. These systems may also prove expensive to maintain. As new features are added to applications, these features must often be added to each router in the system.

Match-structure forwarding promises to bridge the gap between these two approaches. Unlike in single-identifier multicast systems, match-structure headers are designed to represent the complex relationships between subscriptions and notifications. Unlike content-based headers, these headers are also designed to represent these relationships in an efficient, router-level format. Furthermore, match-structure forwarders are able to support a variety of evolving application-level formats at the edge of the subscription system, while only supporting a single format within the subscription system itself.

Content lists are one kind of structure that can be used in match-structure forwarding systems. Content-lists are appealing because they can be implemented by making only a small change to single-identifier systems. As the results show, if multicast messages contained not one, but multiple identifiers, subscription systems could conserve addresses, efficiently support applications with overlapping categories, and conserve routing table entries. Though it takes more time for routers to process multiple addresses than it takes to process single addresses, it takes significantly less time to process multiple addresses than XML attribute-value pairs.

Any single-identifier multicast system–whether it is an IP multicast, overlay, or peer-to-peer system–may benefit from making this simple change.

Content graphs are another approach to match-structure forwarding systems. As the results show, content-list routers take a long time to process notifications that match many subscription categories, such as traffic announcements. Routers in a content-graph system overcome this problem by storing graphs which indicate the relationships between subscription categories. The results show that heavily-loaded content-graph routers process traffic notifications more than ten times faster than the equivalent content-list systems do. These savings might prove crucial if notification processing time eventually dominates the cost of such systems. There are many other choices that other systems might take toward incorporating this approach, however. Overall, these results suggest that the further exploration of match-structure forwarding holds great promise for a new class of Internet Subscription Systems.

Acknowledgments

This work is a direct result of the invaluable feedback and support of my thesis advisor, Dave Clark. Steve Bauer, Chuck Blake, and Mike Halle also contributed to this work with their frequent insights, often providing them at a moment's request. The term "match-structure forwarding" is due to Steve Bauer. Finally, Karen Sollins, Hari Balakrishnan, and John Wroclawski have provided me with frequent opportunities for sharing my ideas. Without all these people, this work would not have been possible.

This effort was sponsored by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-00-2-0553.

6. REFERENCES

[1] M. Adler, Z. Ge, J. Kurose, D. Towsley, and S. Zabele. Channelization problem in large scale data dissemination. In *Proc. of the 9th IEEE International Conference on Network Protocols (ICNP)*, Riverside, CA, USA, November 2001.

[2] M. K. Aguilera, R. E. Strom, D. C. Sturman, M. Astley, and T. D. Chandra. Matching events in a content-based subscription system. In *Proc. of the 18th ACM Symposium on Principles of Distributed Computing (PODC '99)*, May 1999.

[3] M. Altinel and M. J. Franklin. Efficient filtering of XML documents for selective dissemination of information. In *The VLDB Journal*, pages 53–64, 2000.

[4] 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 *International Conference on Distributed Computing Systems*, pages 262–272, 1999.

[5] L. F. Cabrera, M. B. Jones, and M. Theimer. Herald: Achieving a global event notification service. In *Proc. of the 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII)*, 2001.

[6] A. Carzaniga, D. Rosenblum, and A. Wolf. Content-based addressing and routing: A general model and its application. Technical Report CU-CS-902-00, Department of Computer Science, University of Colorado, January 2000.

[7] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Achieving scalability and expressiveness in an internet-scale event notification service. In *Symposium on Principles of Distributed Computing*, pages 219–227, 2000.

[8] I. T. W. R. Center. Gryphon: publish/subscribe over public networks. Technical Report.

[9] J. Pereira, F. Fabret, F. Llirbat, and D. Shasha. Efficient matching for web-based publish/subscribe systems. In O. Etzion and P. Scheuermann, editors, *Proc. of the Int. Conf. on Cooperative Information Systems (CoopIS)*, volume 1901 of *LNCS*, Eilat, Israel, 2000. Springer-Verlag.

[10] P. Pietzuch and J. Bacon. Hermes: A distributed event-based middleware architecture. In *In Proceedings of the 1st International Workshop on Distributed Event-Based Systems (DEBS'02)*, July 2002.

[11] R. Preotiuc-Pietro, J. Pereira, F. Llirbat, F. Fabret, K. Ross, and D. Shasha. Publish/subscribe on the web at extreme speed. In *Proc. of ACM SIGMOD Conf. on Management of Data*, Cairo, Egypt, 2000.

[12] A. I. T. Rowstron, A.-M. Kermarrec, M. Castro, and P. Druschel. SCRIBE: The design of a large-scale event notification infrastructure. In *Networked Group Communication*, pages 30–43, 2001.

[13] A. Snoeren, K. Conley, and D. Gifford. Mesh-based content routing using xml. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP '01)*, Chateau Lake Louise, Banff, Canada, October 2001.