A Collaborative Infrastructure for Scalable and Robust News Delivery

Werner Vogels
Cornell University
vogels@cs.cornell.edu

Chris Re
Cornell University
cmr28@cornell.edu

Robbert van Renesse
Cornell University
rvr@cs.cornell.edu

Ken Birman
Cornell University
ken@cs.cornell.edu

Abstract

In this paper we describe the model used for the NewsWire collaborative content delivery system. The system builds on the robustness and scalability of Astrolabe to weave a peer-to-peer infrastructure for real-time delivery of news items. The goal of the system is to deliver news updates to hundreds of thousands of subscribers within tens of seconds of the moment of publishing. The system significantly reduces the compute and network load at the publishers and guarantees delivery even in the face of publisher overload or denial of service attacks.

1. The Current Model for Internet Content Delivery

The traditional model of web-based publish/subscribe is poorly matched to websites that update their information very frequently. The pull-model requires subscribers to return to the publisher periodically to retrieve new information. The information they receive may be not have changed since their last access, and even if it has changed, will often contain a large redundant subset when compared to an earlier retrieval action. For example a community news site such as Slashdot.org, where the front-page summarizes recently added news articles, receives about a million hits a day on this page, many from returning consumers. It is estimated that a consumer who returns 4 times during a day receives about 70% redundant data. Consumers who return more frequently (and Slashdot.org has many) receive a much higher rate of redundant data. From an end-to-end perspective, the bandwidth necessary to support such a site is heavily underutilized, as it is primarily employed to transport redundant data. Slashdot.org has a policy that requests its consumers, and certainly the automated consumers, to not pull the site more than once per hour.

Another problem with the centralized approach is that the publishers are very sensitive to overload and denial of service attacks. As we have seen during the terrorist attacks in September 2001, Internet news sites become completely useless under overload, failing even to service a small percentage of the visitors.

One way to address these problems involves what are called RSS channels. Here an automated consumer pulls a summary of the available information in XML format, which can be used to determine whether full information needs to be retrieved. Again using Slashdot as our example, the RSS channel data would contain only headlines and URLs to full articles. A second approach in dealing with frequent pulls by subscribers is to use of a "last-modified/ if-modified-since/not-modified" http request/response sequence combined with a delta-encoding for transmitting only the changes to the information source. In both the RSS and delta-encoding case the model for accessing the summaries and the data, namely a pull-mechanism, remains unchanged.

To guarantee robust and scalable access to information, one would prefer to use a push mechanism for publishing data that changes frequently. In this approach, the consumer receives exactly the desired information without any unnecessary overhead, in a timely manner, and avoids having to frequently pull the producer. Such a service is actually offered by some news site through proprietary means (for considerable fees). Other sites offer e-mail based services to provide client with latest news updates. Many of the highest-volume news sites use a hybrid push/ pull approach to push their information to geographically distributed content delivery nodes, from which the consumer still has to pull the data.

2. The Case for Collaborative Content Delivery

Our premise is that current push solutions fail to take advantage of the collaborative power of the Internet. The solutions are often proprietary, and employ a one-to-many model where the producer is expected to deliver "personalized" content directly to each of the consumers. The approach clearly has scalability limitations. Yet despite these problems, there has been little activity by publishers of the many real-time news sites to provide a coordinated solution for this problem. We believe that the time has come for an Internet-wide infrastructure for efficient real-time content delivery.

The approach we favor is based on a scalable and robust push-based publish/subscribe system, which delivers data directly to consumers. The system would support multiple publishers (news sources), each of which would input updates into the system. A consumer would subscribe to the appropriate subjects. This position paper tackles the distribution aspects of the problem, using peer-to-peer techniques. Notice that existing off-the-shelf publish/subscribe systems are not scalable to Internet-wide use, and often require extensive manual configuration and dedicated dedicate server infrastructure.

This paper describes a peer-to-peer publish/subscribe system that, we believe, really could operate at Internet scale. Our solution is robust to disruption, rapid, offers new ways of customizing delivery to subsets of the subscribers, and needs no centralized infrastructure or dedicated servers. The system is based on a set of peer-to-peer component technologies, with which it weaves an infrastructure for the delivery of message updates through the direct use of cooperating end-nodes. Specifically, our solution is based on Astrolabe; a software system designed for ultra-scalable infrastructure and distributed systems management. Astrolabe already exists, and our new publish-subscribe technology is under development now. We expect to make a prototype available for public download in the late spring of 2002.

3. Astrolabe – Scalable Distributed Management and Control

Astrolabe is technology that is designed for the monitoring, management and data-mining of large-scale distributed systems. The technology has four principal properties:

1. Robust, through the use of epidemic techniques
2. Scalable, through the use of information aggregation and fusion
3. Secure, through pervasive use of certificates
4. Flexible, through secure mobile code.

A detailed description of Astrolabe is outside the scope of this paper and can be found in [1]. In the remained of this section we will try to give sufficient background to establish an intuitive notion of Astrolabe's capabilities.

Astrolabe is best visualized as a collection of hierarchical database tables. At the leaf table, a row is assigned to a particular process or user, which is allowed to update this row with attributes & values. These leaf tables (and there may be a great many of them, perhaps tens of thousands) are aggregated, with each leaf table contributing a read-only summary row to its parent table Each of these tables is limited to some small size (say, 64-rows); thus the hierarchy may be several levels deep. The same aggregation mechanism is used between each level. We use the term zone to denote one of these tables (DNS has a similar abstraction, the domain).

Stepping back, Astrolabe can be understood as a database distributed through the network, but not residing on any server. In contrast, the database in question is virtual: like a jigsaw puzzle, each participant stores just a part of the data structure, and the illusion of a tree of tables is constructed at runtime through a peer-to-peer protocol. The protocol also disseminates updates, which trigger re-computation of parent tables much as a spreadsheet updates dependent cells when the cells on which they depend are updated. In a similar sense, one could say that a peer-to-peer file system like PAST [2] resides in the network, but is not fully represented by any server. Details can be found in [1].

Our use of Astrolabe in this paper focuses on the data associated with each computer connected to the system. This data is organized into a single row per machine (or per user), and can be understood as containing a time-varying list of attributed exported by the machine. For our purposes, those attributes define the machine's subscription. Attributes could be classical "subjects", or could be very remote from classical subjects, containing any sort of value or descriptive information or even program-generated information associated with the machine.

Recall that parent tables contain summaries or aggregations of information in their child tables. Astrolabe computes these summaries using aggregation functions, which are expressions in SQL that take any number of attributes from the child table and produce new attributes for inclusion into the appropriate row in the parent table. These aggregation functions are recomputed whenever a row changes in a child table. The aggregations functions are thus a form of mobile code, distributed throughout system using the same epidemic techniques as are used for updates to the data in the rows themselves, and executed on the machines that run the system – the leaf nodes. The system is fully secured through the use of public key certificates.

Astrolabe's epidemic communication techniques guarantee that the state represented is eventually consistent, e.g. if one were to freeze the system, all nodes would eventually enter into consistent states. Of course, through the information hierarchy imposed by the zoning structure, each instance will only “see” the state of its neighbors and of tables between itself and the root.

4. Astrolabe as an infrastructure management service

Earlier, we commented that a weakness of off-the-shelf publish-subscribe systems is that they require extensive configuration. One of the premier applications of Astrolabe technology is in the realm of infrastructure management. The fact that Astrolabe represents state, instead of state-transitions, and provides dynamic aggregation of the individual state into information hierarchies, makes the system highly appropriate for managing a distributed infrastructure. For example, we have explored the use of Astrolabe to manage dynamic bandwidth in media applications, and for flow management and control in telecommunications settings. The robustness and scalability of the underlying epidemic technology make it a very attractive system for environments where guaranteed eventual consistency is essential to the operation of a critical infrastructure. Thus, one use for Astrolabe in a scalable publish-subscribe setting is to simply manage the publish-subscribe subsystem. In doing so, we can overcome one of the important obstacles to performing publish-subscribe at Internet scale.

Examples of infrastructure management attributes that can easily be stored in Astrolabe include the availability and configuration of local communication paths, as well as performance measurements of local networking and computing elements. The aggregation functions used in this setting would typically compute aggregated availability and performance of network, and might offer real-time guidance concerning which elements are in the min/max category, and hence represent targets for new operations.

5. Application level multicast service based on Astrolabe

A second use of Astrolabe is to support an application level multicast service. The basic primitive used here is a method SendToZone(zone,data), where the data is disseminated through all the children zones to all the leaf nodes that are in the tree under this zone. If the method is executed with the root zone as the parameter, all the nodes in the Astrolabe system will receive the data.

The information that the multicast system requires from Astrolabe is the set of multicast representatives in each zone. The representatives are selected in each zone through an aggregation function that combines the local knowledge of availability of independent network paths to a node, the load on those paths and the load on each node. This function will post the results to its entry in the parent zone; together with some basic attributes on which higher-level zone aggregation can be performed.

When a SendToZone is executed the system will visit each of the entries in zone table, each representing a child of this zone. For each of the entries the attribute with the set of multicast representatives will be retrieved and the data will be forwarded to one of the representatives based on a set of local criteria such as where there currently are open connections to one of the representatives. At the arrival of the data at the representative, the process is repeated recursively for all the children in the zone it represents, until the data arrives at the leaf nodes, where it will be delivered to the application Astrolabe is part of. In effect, multicast is performed as a kind of recursive computation on the aggregation in the zone.

For reasons of brevity, we omit additional details of this mechanism. The actual implementation includes a number of optimizations that accelerate multicasts in cases where the same aggregation function is used by a series of SendToZone operations. Although work remains to be done, the protocol thus obtained should have many of the properties of Bimodal Multicast, a peer-to-peer reliable multicast protocol developed by our group several years ago.

6. Publish/Subscribe based on Astrolabe

Jointly, these mechanisms permit us to define a publish/subscribe architecture based on Astrolabe. Basically, the solution extends the Astrolabe-based application-level multicast with a selective forwarding mechanism. To support this leaf nodes publish attributes that represent the subscriptions in which the associated machine is interested. Aggregation functions than perform a simple summary that posts, in the parent zone, a list of the child zones in which there are nodes interested in this subscription. Eventually (within tens of seconds) the root zone will have all the information on whether there are leaf nodes in the system that have subscribed to particular publications.

Publishing data into the system is similar to the multicast summarized above, except for that the decision to forward the data to a child zone now is conditional on a leaf below that child zone to have subscribed to the publication. At the forwarding node this is a simple test that inspects the subscription attribute in the aggregated information for that child zone. If the attribute is present and it shows active subscribers, the data will be forwarded to a child zone representative.

Having an attribute for each possible subscription would be poorly scalable because the work done for purposes of filtering would be at least linear in the number of subscriptions. Accordingly, our system replaces the attributes with a Bloom filter that represents all the subscriptions in the system. A Bloom filter is a probabilistic mechanism for rapidly testing membership in a large set using multiple hash functions into a single array of bits. In the pub/sub system we can use a large single bit array in the order of a thousand bits or more. At a leaf node a subscription is hashed to a single bit in the array, and the subscription arrays are aggregated into parent zones trough a simple binary-or operation on the child arrays. At the publishing node an attribute is added to the data representing the bit position in the subscription array this publication corresponds to. This information is then used at each of the forwarding nodes to test whether the particular bit position in the subscription array is set, and whether the data should thus be forwarded.

The use of Bloom filters is not perfect, insofar as multiple subscriptions can hash to the same bit in the array. Accordingly, a final test is needed at the leaf node whether the data that arrives at the node truly matches a subscription. However, the accuracy can be made as good as desired by varying the size of the bit array, and we believe that a relatively small array will be more than adequate for the target domain of our effort: Internet news services.

7. News item subscriptions

Our publish/subscribe system is, among other things, a research vehicle for understanding the trade-offs in different mappings between news article meta-data and subscription expressions. The news articles are published in the ICE, NITF and NewsML formats [3], which are all XML standards used in the news industry, which not only deal with the description of the content but also provide a mechanism for the standard description of the news-item meta-data that is used in the construction of subscriptions.

An early internal prototype of the system, built as a proof of concept, uses the simpler NITF format, with the subscriptions expressed as a set of interest areas on a per-publisher basis. Each available publisher is represented as an attribute in Astrolabe, where the value of the attribute is a small bit mask that corresponds to a specific set of news categories this publisher provides. The bit masks are aggregated in the same way as the Bloom filters, as described in the previous section. This prototype has limited scalability in the selection of publishers and is not flexible in term of the expressiveness of subscriptions. However, we expect to do much more as we move towards NewsML and begin to enrich the subscription "space" within which our Bloom filters operate.

8. A Collaborative Delivery Infrastructure for News Items

Our publish-subscribe system is intended as a single application that people can download and use to insert themselves into the Collaborative Content Delivery Network. Users would subscribe to a set of publishers and provide more complex selection criteria based on the meta-data associated with the news-items, in the form of an SQL query. By inserting themselves into the network they possibly provide forwarding services to other nodes in the network depending on the criteria used to construct set of zone representatives. A user will have access to a set of configuration parameters that provides input into the selection process.

The automatic configuration of application instances into zones and the location in the zone hierarchy, as well as the configuration necessary to handle firewalls, has been addressed in the context of our overall Astrolabe research effort, but is outside of the scope of this paper [1].

News producers would download and run a different application capable of publishing information according to a restrictive set of rules. These restrictions are necessary to handle the authentication of publishers, to assure the authenticity of the data they publish, and to perform flow control. The infrastructure necessary to support these functions is still a research topic.

Under the covers of the publisher is an application identical to the subscriber application core, insofar as it is just another Astrolabe leaf node, using its local aggregation zone tables to drive the dissemination of its data. The selection and filtering mechanisms used in each forwarding component protect the system from flooding by publishers.

A publisher is able to restrict the scope of the dissemination of the data by selecting another zone than the root zone to publish data into. This for example allows the publisher to disseminate localized news items in Asia. A future feature planned for the system is to allow the publisher more control over the dissemination by adding a predicates to the metadata that needs to be evaluated using the attribute values of a child zone before it can be forwarded to that zone. This would allow the publisher to select the set of subscribers to which an item will be delivered. For example, a publisher could send some item only to "premium" subscribers, or onto to subscribers which have previously shown an interest in certain products.

9. The Forwarding & End System Components

Our system seeks to deliver news items to the subscribers in the order of tens of seconds, even if tens or hundreds of thousands of subscribers are active. Each forwarding component maintains a log file and a set of forwarding queues, one for each of the representatives at a child zone. The best strategy to fill queues is still under research. We are experimenting with weighted round-robin strategies, as well as some more aggressive techniques. In this context we are also looking at what information the representative can post into their tables to aid the selection process.

News items are uniquely identified by the publisher as part of the news item meta-data; this can be used to remove duplicates, when (in the manner of the MIT scalable publish-subscribe work [4]) we use multiple representatives to forward a new item, to increase the robustness of the delivery.

At the end system the news items are delivered to a message cache, which is feeds the applications that use the news items. Automatic cache management can be configured to provide item management based on the meta-data of the news items, which includes information about item revision history. On the basis of this metadata, the news item can be garbage collected, or fused or aggregated into a more compact form.

The same cache is used for assisting in achieving end-to-end reliability in the case of forwarding node failures, and for a limited state transfer to participants that are joining the system.

10. Experimentation and deployment

As described earlier in this paper we are currently working with an early prototype which is to serve as a proof of concept for the system. We are experimenting to understand issues such as the complexity of forwarding node selection, the use of redundant message publishing, node failure & automatic zone reconfiguration and the impact of those issues on end-to-end reliability, publisher authentication, and of course the overall performance of the system.

In parallel with this research we have started to build a first production version of the system that is targeted for wide-scale experimentation by actual users. We intent to make two system configurations available late in the spring of 2002; the first will be targeted towards the publishing of technical news articles by sites such as Slashdot.org, Wired, The Register, SilliconValley.com, News.com, etc. The second configuration will be targeted towards the general news distribution with publishing by Reuters, Associated Press, the New York Times, etc. We are working to get the collaboration of the major news sites, but we have already developed some agents that are capable of transforming the current RSS/HTML information from some publishers into message streams for the system to bootstrap it.

The interface to the system is currently still under development but it will be a full user control application in the same style as many of the current file sharing applications, with an additional web interface for access. We are also looking for integration into popular content aggregation systems such as Radio Userland [5] using XML-RPC mechanisms

11.References

[1] Robbert van Renesse and Kenneth Birman, Astrolabe, A Robust and Scalable Technology for Distributed Monitoring, Management and Data Mining, submitted for publication, 2002.

[2] Rowstron and P. Druschel, Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility, Proceedings of the 18th ACM Symposium on Operating Systems Principles, pp. 160-173, Banff, Canada, October 2001

[3] International Press and Telecommunication Counsil, http://www.iptc.org.

[4] Alex C. Snoeren, Kenneth Conley, and David K. Gifford, Mesh-Based Content Routing using XML, Proceedings of the 18th ACM Symposium on Operating Systems Principles, pp. 160-173, Banff, Canada, October 2001.