Channel Islands in a Reflective Ocean: Large Scale Event Distribution in Heterogeneous Networks

Jon Crowcroft, Jean Bacon, Peter Pietzuch,
Computer Laboratory
and
George Coulouris, Hani Naguib,
Laboratory for Communications Engineering

University of Cambridge
William Gates Building
J J Thomson Avenue
Cambridge
CB3 0FD

{jac22|jmb|prp22|gfc22|han21}@cam.ac.uk

July 2, 2002

Abstract

We discuss the design of a multicast event distribution service intended to support extremely large scale event distribution.

To date, event notification services have been limited in their scope due to limitations of the infrastructure. At the same time, Internet network and transport layer multicast services have seen limited deployment due to lack of user demand (with the exception more recently of streaming services, e.g. on Sprint's US core network, and in the Internet II). Recent research in active networks and reflective middleware suggests a way to resolve these two problems at one go.

The goal of this paper is to describe a reflective middleware system that integrates the network, transport and distributed middleware services into a seamless whole.

The system integrates this ’low-level’ technology into an event middleware system, suitable for telemetry, novel mobile network services, and other as yet unforeseen applications.

1 Introduction

In this paper we discuss requirements and propose a design for a multicast service that can distribute event messages to subscribers throughout the Internet on a scale comparable to todays transport-level services. We can envisage a world in which pervasive computing devices generate 10,000,000,000 events per second. We can foresee a time when there are thousands of millions of event subscribers all over the planet, with publishers having popularities as low as no or only a single subscriber, or as high as the entire world.

Event-driven and messaging infrastructures are emerging as the most flexible and feasible solution for enabling rapid and dynamic integration of legacy and monolithic software applications into distributed systems. Event infrastructures also support deployment and evolution of traditionally difficult-to-build active systems such as large-scale collaborative environments and mobility aware architectures[6].

Event notification is concerned with propagation of state changes in objects in the form of events. A crucial aspect of events is that they occur asynchronously. Event consumers have no control over when events are triggered. On the other hand, event suppliers do not generally know what entities might be interested in the events they provide. These two aspects clearly define event notification as a model of asynchronous and de-coupled communication, where entities communicate in order to exchange information, but do not directly control each other.

The architecture of an event distribution overlay layer is illustrated in figure 1. In this we can see that a publisher creates a sequence of events, which carry attributes with given values. A consumer subscribes to a publisher, and may express content based filters to the publisher. In our system, these filter expressions can be distributed up-stream from the consumer towards the publisher. As they pass through Application-level event notification distributors, they can be evaluated and compared, and possibly combined with other subscription filters. Notifications of interest are passed up-stream all the way to the publisher, or to the application-level event notification distributer nearest the publisher, which can then compute a set of fixed tags for data; it can also, by consulting with the IP and GRA routers, through the reflective multicast routing service, compute a set of IP multicast groups over which to distribute the data. This will create the most efficient trade-off between source and network load, and receiver load, as well as tag and filter evaluation, as the events are carried downstream from the publisher, over the IP multicast, GRA, and application-level event notification nodes.

Devising and evaluating the detailed performance of the algorithms to carry out these tasks form the core of the requirements for future work.

2 Background

The IETF is just finishing specifying a family of reliable multicast transport protocols, for most of which there are pilot implementations. Key amongst these for the purposes of this research is the exposure to end systems of router filter functionality in a programmable way, known as *Generic Router Assist*. This is an inherent part of the Pragmatic General Multicast service, implemented by Reuters, Tibco and Cisco in their products, although it has not been widely known or used outside of the *TIBNET* products until very recently.

The last decade has seen great leaps in the maturity of distributed systems middleware, and in one particular area - event notification systems, in support of a wide variety of novel applications, Current work on event notification middleware [11, 12], has concentrated on providing the infrastructure necessary to enable content-based addressing of event notifications. These solutions promote a publish-subscribe-match model by which event sources publish the metadata of the events they generate, event consumers register for their events of interest passing event filter specifications, and the underlying event notification middleware undertakes the event filtering and routing process. Solutions differ usually on whether they undertake the filtering process at the source or at an intermediary mediator or channel. The trade-off lies in whether to increase the computational load of sources and decrease the network bandwidth consumption, or minimise the extra computational load on the sources and outsource the event filtering and routing task to a mediator component (hopefully located close to the source). All of these solutions do not leverage on the potential benefits that event multicasting to consumers, which requires the same type of events, and which applies very sim-

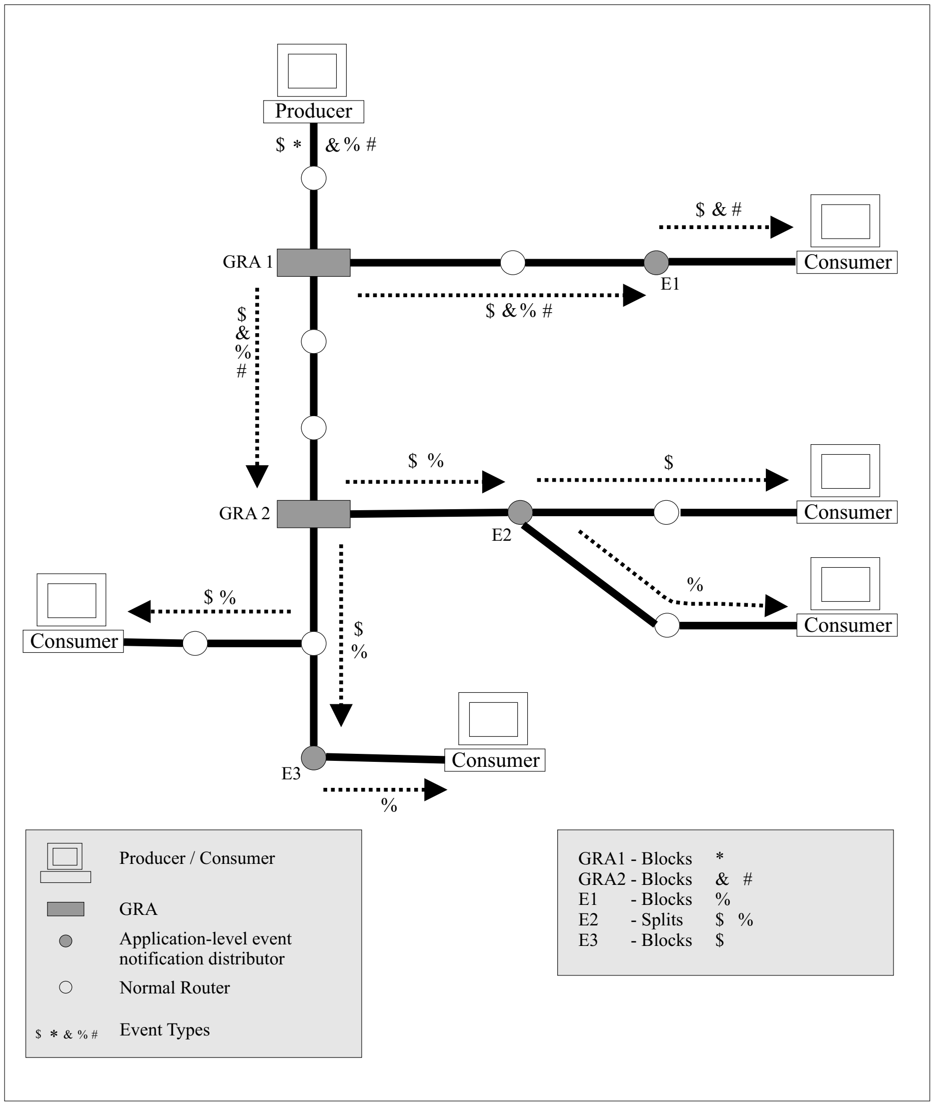

Figure 1: Channel Islands Event Distribution Overlay Architecture

ilar filters, could bring. They usually require an individual unicast communication per event transmitted.

At the same time, the underlying network has become very widespread. New services such as IP multicast are finally seeing widespread deployment, especially in core networks and in intranets.

The combination of these two technologies, event services and multicast, originates historically with Tibco [7], a subsidiary of Reuters. However, their approach is somewhat limited as it takes a strictly layered approach.

At the highest level, there is a publish/subscribe system, which in *TIBNET* uses *Subject Based Addressing* and *Content Based Addressing*. Receivers subscribe to subjects. The Subject is used to hash to a multicast group. Receivers subscribe to a subject but can express interest by declaring filters on content. The *TIBNET* system is then hybrid. In the wide area, IP multicast is used to distribute all content on a given subject topic to a set of site proxy servers. The site proxy servers then act on behalf of subscribers at a site and filter appropriate content out of each subject stream and deliver the remains to each subscriber.

Between the notification layer and the IP layer there is a transport layer, called Pragmatic General Multicast. To provide semi-reliable, in-order delivery, the subject messages are mapped onto PGM [2] messages, which are then multicast in IP packets. PGM provides a novel retransmission facility which takes advantage of router level functionality for "nack aggregation", preventing message implosion towards the event source, and to provide filtering [3] of retransmissions, so that only receivers missing a given message sequence number, receive it. The PGM protocol is essentially a light weight signaling protocol which allows receivers to install and remove filters on parts of the message stream. The mechanism is implemented in Cisco and other routers that run IP multicast. The end system part of the protocol is available in all common operating systems.

Almost all other event notification systems have taken the view that IP multicast was rarely deployed[1], and that the overheads in the group management protocols were too high for the rate of change of interest/subscription typical in many applications usage patterns [14].

Instead, they have typically taken an alternative approach of building a server level overlay for event message distribution. Recent years have seen many such overlay attempts [9]. These have met with varying degrees of success. One of the main problems of application layer service location and routing is that the

[1]Ironically, this view was fuelled partly by a report by Sprint [8], when in fact the entire Sprint IP service supports multicast and they have at least 3500 commercial customers streaming content.

placement of servers does not often match the underlying true topology of the physical network, and is therefore unable to gain accurate matching between a distribution tree and the actual link throughput or latencies. Nor is the system able to estimate accurately the actual available capacity or delay. Even massive scale deployments such as Akamai, for example, do not do very well.

Secondly, the delays through application level systems are massively higher than those through routers and switches (which are after all designed for packet forwarding, rather than server or client computation or storage resource sharing). The message is that overlays and measurement are both hard to optimise, and inefficient. It is frequently the case that in the long term, business migrates into the infrastructure. (c.f. voice, IP, etc). We expect many overlay services to do this. We believe that this process will accelerate due to the use of state of the art network middleware and software engineering approaches. However, this process will not stop - there is an endless stream of new services being introduced “at the top”, and making their way down to the bottom, to emerge as part of the critical information infrastructure.

3 Proposed Approach

We see a number of advantages in continuing forward from where Tibco left off in integrating efficient network delivery through multicast, with an event notification service including:

scale We obviate the need to deploy special proxy servers to aid the distribution.

throughput The system is able therefore to distribute many more events per second.

latency Event distribution latency will approximate the packet level distribution delay, and will avoid the problems of high latency and jitter incurred when forwarding through application level processes on intermediaries.

There are two ideas we draw from in moving forward. Firstly we exploit advances in the network support for multicast, such as Generic Router Assist service in the PGM router element in IP multicast. Secondly, we distribute an open interface to the multicast tree computation that IP routers implement. The way we do this is through reflection.

Reflection is becoming commonplace in middleware [10], but has not been applied between application level systems and network level entities to our knowledge. The choice here is to offer a common API to both the multicast service, and the filtering service, so that the event notification module implementor need not be aware which layer is implementing a function. We envisage an extremely simple API, viz:

```
Create(Subject)
Subscribe/Join(Subject)
Publish/Send(Subject, Content)
Receive(Subject, Content Filter Expression)
```

In most current event services objects and filters are specified using a string hierarchy. An XML[4] or SOAP-based hierarchy would offer stronger typing. The router level creates both a real distribution tree for subjects, and a sub-tree for each filter or merged filter set. This is done with regard to the location (and density) of receivers. We can use a multicast tunnel or multicast address translation service to provide further levels of aggregation within the network. This requires the routers to perform approximate tree matching algorithms.

We are building a piece of reflective middleware that will add a thin layer between an existing event notification service and the reflective routing and filter service. This involves extending the PGM *signaling* protocol that installs and activates the filters, via IP router alerts. We are investigating efficient hashes for subject to group and content to sequence number mapping [13].

We intend to evaluate our approach by applying it to a large-scale event driven (sentient) application, such as novel context-aware applications for the emerging UMTS mobile telephony standard or large-scale location tracking applications. For example, there is the possibility of developing location tracking (people, vehicles and baggage) for large new airport terminals. There has been a surge in interest in mobile event sinks, as we can see in recent published work[15][16][17].

4 Discussion

One of the goals of this work is to explore the way that the multicast trees and the filtering system evolve in large heterogeneous application environments. Another goal is to see how multicast routing can be “laid open” as a service to be used to build distribution trees for other layers. Finally, we believe that the three levels we have may not be enough, and that as the system grows larger still, other services may emerge.

What we have designed is effectively a two-tier system, building on previous work [13], which entails multicast trees and, within these, filters. To these, we have added a third, overlay layer.

The purpose of the overlay is to accommodate a form of qualitative heterogeneity discussed below, whereas the lower two layers of multicast and filtering target the area of quantitative performance differences.

The overlay layer is required, firstly because current event distribution systems are built without any notion of a multicast filter-capable transport. Thus we must have an overlay of event distribution servers. These can, where the lower services are available, be programmed to take advantage of them, *amongst themselves*, thus providing a seamless mechanism to deploy the new service transparently to publisher and subscriber systems.

Secondly, we believe that there are *inherent* structural reasons why such an application layer overlay is needed. These include:

Policies: Different regions of the network will have different policies about which events may be published and which not.

Security: There may be firewall or other security mechanisms which impede the distribution via lower level protocols.

Evolution: We would like to accommodate local evolution of multicast routing mechanisms (in the same way that inter-domain routing protocols such as BGP allow intra-domain routing to evolve).

Interworking: We would like to support a variety of event distribution middleware systems. We have some initial results in this area[4][5].

Others: There are other such “impedance mismatches” which we may encounter as the system scales up.

A novel aspect of our approach is that the overlay system does not itself construct a distribution tree. Instead, a set of *virtual* members are added to the lower level distribution system which then uses its normal multicast routing algorithms to construct a distribution tree amongst a set of event notification servers separated in islands of multicast capable networks. These servers then use an open interface to query the routers as to the computed tree, and use this as their own distribution topology. In this way the overlay can take advantage of detailed metric information that the router layer has access to (such as delay, throughput and current load on links) instead of measuring a poor shadow of that data which would lead to an inaccurate and out of date set of parameters with which to build the overlay. In some senses, what we are doing here is a form of multicast traffic engineering for the existing network.

We believe that our proposed system can provide a number of engineering and performance enhancements over previous event notification architectures. Future work will evaluate these, which includes:

1. System performance - improvement in scalability, including reduction in join/leave publish/subscribe latency, increase in event throughput, etc.

2. Network impact - impact on router load of filter processing, group join, leave and multicast packet forwarding.

3. Expressiveness and seamlessness of API - we plan to try it with a variety of event notification systems and to export a portable implementation via public CVS to see what the open source community does with it.

4. Mobility - the dynamic nature of the location of an event sink in a mobile system has attracted recent research[6]. Our system design has the inherent ability to incorporate dynamicity in the location of event receivers. We will evaluate this in the presence of real world mobility statistics.

References

[1] A. Mankin, A. Romanow, S. Bradner and V. Paxson, "IETF Criteria for Evaluating Reliable Multicast Transport and Application Protocols" RFC2357, June 1998.

[2] Pragmatic Generalised Multicast Tony Speakman, et al, Work in Progress, `http://search.ietf.org/internet-drafts/draft-speakman-pgm-spec-07.txt`

[3] GMTS "Generic Multicast Transport Services" B. Cain, D. Towsley, in Proc. Networking 2000, Paris, France May 2000. `http://www.east.isi.edu/RMRG/cain-towsley3/`

[4] 'Reconciling event taxonomies across administrative domains", A Hombrecher, et al Work in Progress (submitted for publication).

[5] “Federating heterogeneous event systems”, A Hombrecher, J Bacon, K Moody,. Tenth International Conference on Cooperative Information Systems Oct 30 - Nov 1, University of California, Irvine.

[6] “Integrating Real-World and Computer-Supported Collaboration in the Presence of Mobility”, J. Bates, M. Spiteri, J. Bacon and D.Halls, IEEE WET-ICE (Workshop on Emerging Technologies), 1998. Best paper award.

[7] TIBCO `http://www.tibco.com`

[8] “Deployment Issues for the IP Multicast Service and Architecture”, C. Diot, B. N. Levine, B. Lyles, H. Kassem, D. Balensiefen. IEEE Network magazine special issue on Multicasting. January/February 2000.

[9] “A Case For End System Multicast”, Y. Chu, S. Rao, H. Zhang, Proceedings of ACM SIGMETRICS , Santa Clara,CA, June 2000, pp 1-12.

[10] “Integrating Meta-Information Management and Reflection in Middleware”, Fabio Costa and Gordon Blair 2nd International Symposium on Distributed Objects & Applications pp. 133-143, Antwerp, Belgium, Sept. 21-23, 2000. Internal report number MPG-00-20

[11] “A Survey of Event Systems”, A. Rifkin and R. Khare. `http://www.cs.caltech.edu/ adam/isen/event-systems.html`

[12] “Design and evaluation of a wide-area event notification service”, Carzaniga A., Rosenblum D. S. and Wolf A. L. ACM Transactions on Computer Systems, Volume 19, no. 3, pp. 332-383, 2001

[13] “Hermes: A Distributed Event-Based Middleware Architecture”, Peter Pietzuch and Jean Bacon, Proceedings of the 1st International Workshop on Distributed Event-Based Systems (DEBS’02), Vienna, Austria, July 2002.

[14] “Exploiting IP Multicast in Content-Based Publish-Subscribe Systems”, Lukasz Opyrchal, Mark Astley, Joshua Auerbach, Guruduth Banavar, Robert E. Strom and Daniel Sturman, Lecture Notes in Computer Science 1795, 185-207, Heidelberg, Germany, 2000

[15] “Herald: Achieving a Global Event Notification Service”, Luis Felipe Cabrera and Michael B. Jones and Marvin Theimer, Proc. of the 8th Workshop on Hot Topics in OS (HotOS-VIII), year = 2001

[16] “Content-Based Dispatching in a Mobile Environment”, Gianpaolo Cugola and Elisabetta Di Nitto and Gian Pietro Picco, year = 2001

[17] “Using a Publish/Subscribe Middleware to Support Mobile Computing”, Gianpaolo Cugola and Elisabetta Di Nitto, Middleware for Mobile Computing Workshop, year = 2001,