

# InfoSpaces

## A Large-Scale Content Classification and Dissemination Network

Satyam Vaghani, Michael Michael, Jairam Ranganathan,  
Sujata Kodgire, Armando Fox  
{svaghani, michmike, jairam, sujk, fox} @ cs.Stanford.edu

Stanford University

April 2001

### Abstract

Public information routing and delivery is fraught with difficulty. The functionality of large-scale publish/subscribe systems is often restricted by the choice of delivery mechanism and delivery end-point. Scheduling predicates do not hold globally over concurrent content delivery streams to diverse end-points that share resources to disseminate information. We have implemented an alternative information dissemination model that sheds off a few characteristics of personalized information delivery in order to provide ubiquitous, cost-effective and demographically targeted information. We try to characterize some important abstractions to achieve these goals and discuss a scheduler that allows the resources of a set of end-points to be grouped to display information of common interest.

### 1 Introduction

Most publish/subscribe systems [9] must deal with the issue of categorizing and disseminating large amounts of information to a group of heterogeneous end-points in a cost-effective and scalable manner. Previous work in such systems has focused on efficient multicasting of information to the end-points and content-based information routing to interested subscribers [6]. These systems assume a common application-level communication substrate (in most of the cases, the Internet). We designed an end-to-end solution, called InfoSpaces, for large-scale classification and dissemination of information. InfoSpaces is designed to retrieve/accept information from various ac-

tive and passive sources, classify it and route it to diverse physical devices. These devices, which we refer to as end-points, have different display capabilities and modes of connectivity. InfoSpaces handles a large amount of information that contends for a limited number of display slots on end-points. This required us to design a scheduler that makes in-time delivery guarantees for high-priority content, preserves the priority order of large information sets, minimizes persistent state maintenance, and does not impose too much overhead to the system.

In subsequent sections, we will discuss the architecture of our system and the implementation details. We learnt that the design of the content delivery scheduler was critical to the overall performance of the system in the presence of a large amount of information waiting to be scheduled. We modified the “memory market” scheduling model proposed by Harty and Cheriton [1] to suit our goals of making a scalable and cost-effective scheduler. The main challenges posed by large data sets on the scheduling process are highlighted in the later section on content scheduling. We have tested the algorithm to perform satisfactorily under heavy workloads, a common characteristic of public information systems.

Transforming information for delivery to a heterogeneous group of devices is a well-researched problem [4, 5, 14, 15]. In order to support such diversity, InfoSpaces makes use of existing work on client adaptation. Device specific proxies are defined for classes of devices (e.g. Web TVs, Electronic billboards, Tickers, etc) that perform information transformations on behalf of the end-

points. In order for the scheduler to work correctly, these proxies can characterize devices by “information capacity” which defines the amount of information that a device can process in one display cycle. The proxy thus presents an abstraction of the device to the scheduler so that it can deliver a non-formatted (raw) data-set to the proxy for subsequent transformation and delivery to the end-point.

A prototype of the system has been implemented to validate the feasibility of this design and prove the correctness of the modified scheduling algorithm. The prototype manages and schedules information for display on a restricted class of devices. In our prototype, we have implemented primitive ways to classify and tag content because classification of content based on its semantic analysis does not belong to our problem-domain. On a similar note, InfoSpaces is not a real-time information delivery network.

In the next sections we will describe the InfoSpaces architecture, as modeled in our prototype, including the content manager, our scheduling algorithms, and the content delivery mechanism. A brief commentary on the information providers, end-points and administrative features of the system is included in the later part of the paper. We discuss some related work before concluding with our insights for future work. Some screen-shots of the system are included in Appendix A.

## 2 Terminology

We will first define the terminology that will be used throughout the paper.

A *data-item* is a piece of information that is entered in the system so that it can be displayed. Each data-item in the system has a

lifetime, which begins when the data-item needs to be scheduled for display, and ends when the current time is past the user-defined expiry time for the item. The user can also choose a *priority* for the data-item from a three-point scale. The user is also responsible for classifying data-items based on their semantic properties into different *channels*. Channels represent a topical division of the information that allows information to be routed to the most appropriate end-points. We will first describe the system assuming only a single, global administrative domain. Later, we will relax this restriction to show how distinct administrative domains, called *subnets*, share end-point capacity while maintaining jurisdiction/autonomy over their own content and the capabilities afforded to their users. The subnet access hierarchies and topological organization of users will also be discussed in later sections.

## 3 Architecture

InfoSpaces has identified three abstractions, which interact with each other through well-demarcated interfaces, and are used to describe the core of our system. These are:

- Content Creation
- Content Management
- Content Scheduling and Delivery

We have implemented the architecture presented in Figure 1 using a Windows 2000 Advanced Server running MS-Internet Information Server 5.0 and the ColdFusion application server. The system stored its data in a database managed by SQL Server 2000. Security contexts, like the Security Sandbox, were constructed using the ColdFusion server manager and content adaptation proxies were implemented as server extensions.

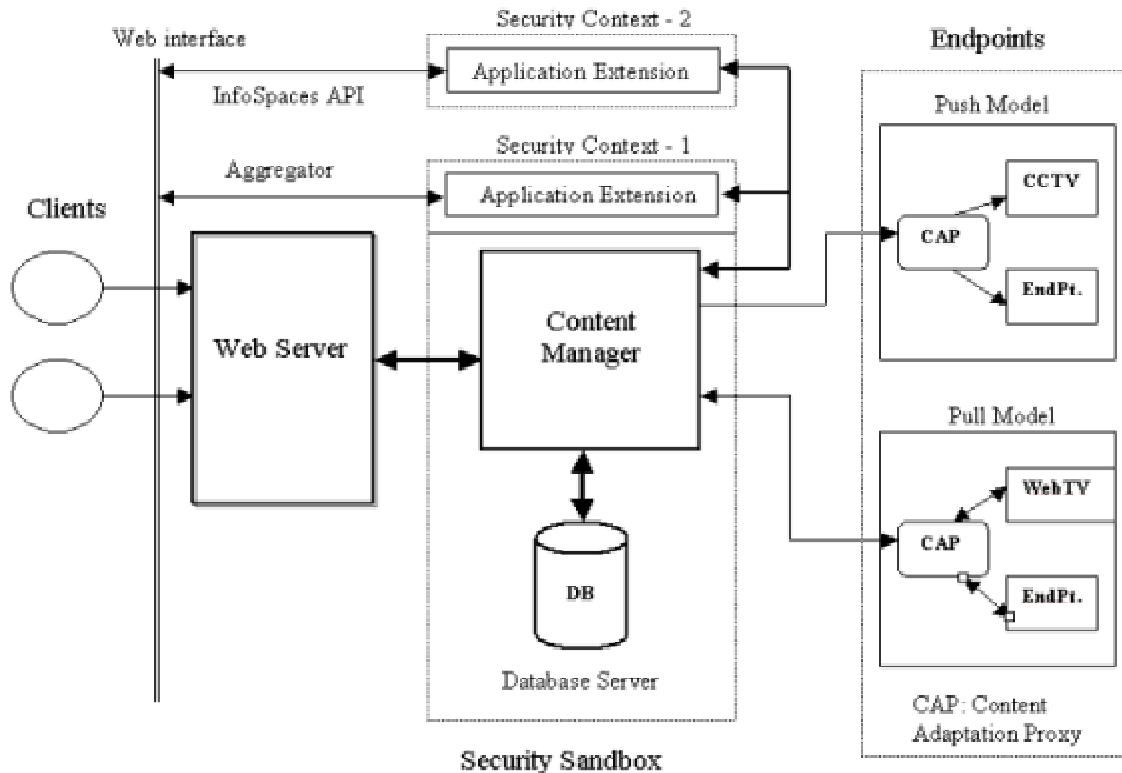


Figure 1: The InfoSpaces Architecture

### 3.1 Content Creation

In order for InfoSpaces to be adaptable to diverse content types, content is introduced into the system through several input streams. We have implemented a web-based form interface that allows InfoSpaces users to enter information directly into the system. An InfoSpaces API has also been defined for programmatic content input. We have additionally implemented information aggregators that use pull mechanisms to extract content from the Internet. These extraction modules were implemented using Compaq's Web Language [3]. The objective is to allow content providers to write their own extraction modules that can be plugged into the system. Content Creation is further discussed in the I/O section.

### 3.2 Content Management

The Content Manager is responsible for maintaining a consistent state in the content repository. When new information arrives in the system, the Content Manager classifies it and queues it up for approval, by more privileged users, if needed. It maintains the "active data item set", which is the content that is ready for presentation and can be fed to the scheduler. When a data item is due to be scheduled, the Content Manager moves it into the active set. Once the lifetime of a data item is over, the Content Manager is responsible for removing it from the active set so that it can no longer be scheduled for presentation.

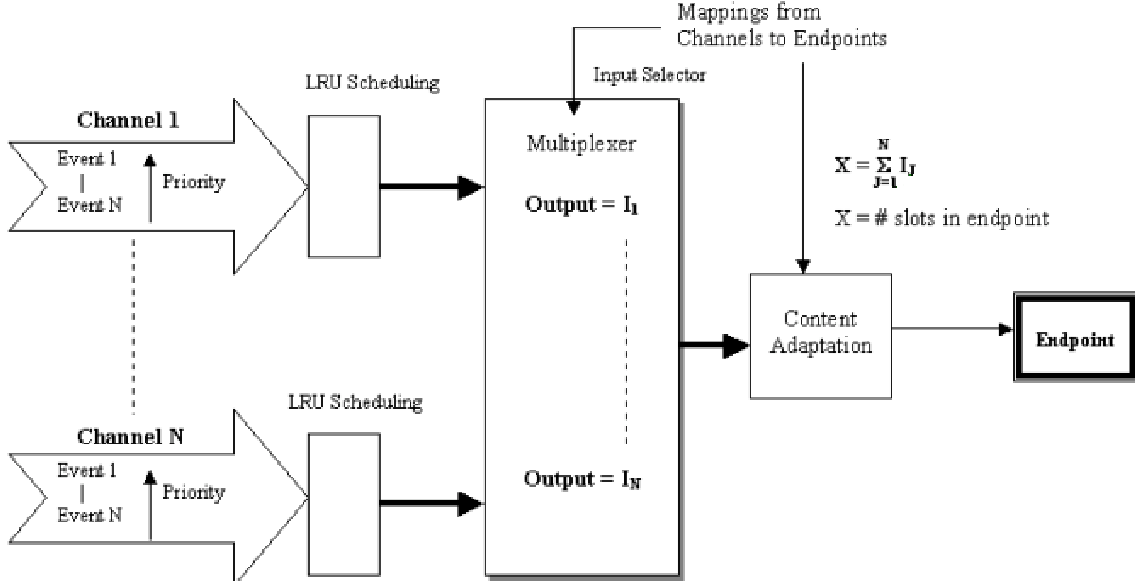


Figure 2. Interaction between entities in the process of forming a valid schedule

In our prototype, we have implemented the Content Manager with helper threads that are executed at regular intervals. A helper thread queries the content repository for data items that are due to be scheduled and moves these to the active set. When a data item is past its expiration time, it is removed from the active set by another thread.

### 3.3 Content Scheduling

Once in the active set, data items compete to ‘earn’ a display slot on the end-points. An active data item cycles between the ‘active’ state and the ‘online’ state until it expires and is removed from the system by a helper thread. We shall now describe the ‘modified memory market’ scheduler that we have implemented.

#### 3.3.1 The Scheduling Algorithm

Many ideas used in designing the scheduler are borrowed from the ‘memory market’ model proposed by Harty and Cheriton [1]. Consider a set,  $S$ , of active data items in a particular channel. A data item earns ‘money’ in proportion to its priority,  $P$ , while in the active queue. At a given point of time, an active data item might have a

balance,  $B$ , with which it can purchase additional display slots. The scheduler charges a fixed amount,  $A$ , (equivalent to the least count of salary distribution) for each display slot that can be rented out to data items. Salary is disbursed at the end of a pay-period,  $C$ .

Consider an active data item set

$$S1 = \{d1, d2\}$$

where  $d1$  and  $d2$  are data items with priorities  $P(d1) = 1$  and  $P(d2) = 3$ . Taking  $A$  to be 1, we can define the correct behavior of a priority-based scheduler for this system over a least-count period of 4 service cycles to be the allocation:

$$\{\#d1, \#d2\} = \{1, 3\}$$

where the numbers represent actual number of times each data item was displayed. To generalize:

$$C = \sum_{i \in S} P_i$$

Data items are scheduled in order of their current balance  $B$ . At the beginning of a pay period,

$$\forall i \in S, B_i = P_i$$

The scheduler uses LRU semantics to break contention between equally wealthy data items. Let the time elapsed in the current

cycle be  $E$ . At the beginning of a new pay period,

$$E = 0$$

We will represent the state of a channel by the ordered pair  $\langle E, C \rangle$  and the state of an active data item by  $\langle P, B \rangle$ .

We maintain an invariant across all state changes, namely,

$$\sum_{i \in S} B_i + E = C$$

Four kinds of data-items affect these states:

1. Scheduling of data items in display slots.
2. Insertion of a new data item into the active data-item queue.
3. Deletion of a data item from the active queue.
4. Update of a data item's priority or expiry data while the data item is in the active queue.

Finally, it is possible that towards the end of  $C$ , the total residual balance remaining with all the data items eligible for a particular end-point is lesser than the number of display slots on the end-point. In this case, we 'tax' the data items that have outstanding balances. This effectively makes  $E=C$  and we can start a new cycle with a bigger eligible data item set.

### 3.3.2 Design challenges

In this section we shall document the design challenges for our scheduler and explain how our scheduling model successfully addresses those. One might observe that the problems discussed below are applicable to large-scale information dissemination systems in general.

The scheduler needs to ensure that each data-item receives its fair share in the schedule of active data-items across devices interested in displaying the same information. Given the fact that the active data-item queue is accessed by multiple devices with a high frequency and hard state maintenance

across requests is prohibitively expensive, the problem is to calculate a channel specific predicate which holds over a one-to-many channel-end-point mapping. According to our model, every good (data item) figures on a unique market (channel). Inter-market transactions are not allowed and they are not required since the goods are semantically different. The consumer (end-point) can, however, buy goods from different markets. Thus we have the following predicate: Transactions in a particular market should take it from one consistent state to another; and these transactions should not use data from another market. This predicate localizes consistency requirements and since it is independent of the affiliations of a customer to other markets.

In order to schedule data-items for a particular channel, we need to select data-items in proportion to their priorities, and their last display time. This would require a history list of schedules on a per-channel and per-end-point basis. However, we cannot maintain a history of schedules for each channel-end-point mapping for three reasons:

1. The length of such a history list would be dependent on the refresh rate of each end-point and the arrival rate of 'active' data-items in each channel.
2. Our system does not put an upper bound on the number of end-points or the number of channels. Clearly, storage and retrieval of a possibly huge number of schedules per unit time is inefficient.
3. The channel-end-point mapping is  $M \times N$ . Even if we were to optimize the storage by storing the schedules either on a per-channel basis or a per-end-point basis, algorithms to create history lists on a per-channel basis from per-channel list, and vice-versa, are  $O(m \log n)$  at best. It is impractical to run these algorithms on every incoming request (for e.g., a MS-Web TV device requests data every 10 seconds in our

prototype and there are a number of such devices)

We maintain a cumulative history for each channel by using two data item specific variables: B and P and two channel specific variables: C and elapsed time, E. An algorithm to retrieve the per-channel history is now constant-time and one to retrieve the cumulative history for a particular end-point is now linear in the number of channels to which it subscribes.

Another observation is that highly persistent data items tend to occupy a large proportion of display cycles, thus causing a temporally localized increase in the degree of contention. This phenomenon is accentuated by the presence of high priority, highly persistent data items. This problem arises because some of the data items become eligible to enter the active set a long time before they are semantically relevant. For example, an over-publicized data-item may prevent a low priority current news story from going online. The solution is to limit the earnings of long-display-cycle data items in the early stages of their activation; in other words, limit the earnings of these data items during their childhood phase. As a corollary, short display-cycle data-items do not have a childhood phase.

Schedules have a service cycle during which the queue of data items waiting to be scheduled is assumed to be stable. Consider a conventional FCFS single-CPU scheduler. It maintains an M/M/1 queue where the customer (a process) can depart only after being serviced by the server (CPU). This scenario has an upper bound on the waiting time and the length of the queue. The InfoSpaces content streams present a much more dynamic queue of data items waiting to be scheduled. A striking difference arises due to absence of guarantees on life cycles of data items. This translates to the fact that with the InfoSpaces scheduler, customers would be free to leave the waiting queue at will without informing the scheduler, thus bypassing the server. This implies that if the scheduler were to maintain local state, it would be cor-

rupt for the purposes of scheduling in the next service cycle. To counter the violation of the strict queuing discipline in the InfoSpaces scheduler, we take care to see that the scheduling within a channel is reversible. In other words, if a data-item leaves an active set without going through the server, the effects of its inclusion can be undone without disturbing the  $\langle P, B \rangle$  tuples for other active data-items in the same channel.

## 4 Execution Example

We will now walk through an example scenario to show how the concepts presented so far orchestrate the flow of information from content providers to content consumers. A system installed at Stanford University is used to disseminate information to the student body at Stanford. Thus, Stanford University would represent an administrative body in our domain. Within Stanford, each department provides content for students of that department. Thus, there is a ‘Computer Science’ channel, a ‘Law’ channel, and a ‘Student Affairs’ channel, to name a few. Consider a Stanford user who logs in to the system and enters information about a seminar in computer science to be held a week from now. Content input by authors with fewer privileges is subject to approval by a user higher up in the access rights hierarchy. On approval, the content manager receives the information. It is responsible for maintaining and classifying the content according to channels and subnets. This content would be classified as part of the Stanford subnet and belonging to the Computer Science channel. The next section explains the concept of subnets in greater detail. For the purpose of this example, it is sufficient to know that a subnet is a high-level administrative domain. When this content becomes “active”, that is, needs to be displayed, the content manager moves this data item into a set of active data items. At this point, the scheduling engine starts using this new data item while scheduling as well. When an endpoint (e.g. a large-screen TV at the Com-

puter Science building) needs to have new content sent to it, the end-point's proxy makes a request for new content to the scheduler based on the display capacity of the end-point, demographic characteristics of the end-point and the channels that the end-point is interested in displaying. The scheduler chooses the optimal set of data-items and returns this to the proxy. The proxy performs the necessary transformations and device specific formatting for the content and sends it to the end-point for delivery. Data items remain active until their expiration time. This means that the content could be displayed many times in its active period. Once the content expires, the content manager is responsible for removing the information completely from the system.

## 5 I/O and other machinery

Having explained the core of the system, we will now move out to the periphery and examine the information input and output (delivery) interfaces.

### 5.1 Content Providers

Since data sources are varied, we need to support multiple input methods. Our prototype supports the following content input streams:

*Web-based active agents:* These can be web extraction modules, web crawlers, or any other Information Retrieval (IR) system. For the purposes of the prototype we used Compaq's Web Language to build our HTML information extraction modules. An active agent is run within a security context, the privileges of which depend on its author.

*InfoSpaces API:* We can input information in the form of data feeds or any custom application module. For security purposes we have built in authentication as well as a session-specific key exchange.

*Web-based interface:* This interface is managed through the InfoSpaces web-based au-

thentication scheme and connects to the core via a web server.

### 5.2 Content Consumers

InfoSpaces outputs information onto an end-point. An end-point represents an abstraction of the actual physical devices used for displaying content (e.g. MS-Web TV, Closed-Circuit TV, Interactive Kiosks, etc.). End-points are classified based on their characteristics with the most metric being the amount of information that can be displayed. Others include the physical dimensions, media capabilities, processing power, local storage available, and type of connectivity to the content adaptation proxy (CAP).

The end-point either requests data-items from the CAP or receives data-items from the CAP depending on whether it is a pull or push compatible device. Based on the characteristics of the end-point, the CAP will request the appropriate information from the content scheduler. The scheduler applies the algorithm described in the scheduling section and will return a generic XML data stream of the content to be displayed. The CAP then applies a series of transformations based on device-specific characteristics and outputs the content in the desirable format to the end-point.

### 5.3 Administrative domains

We now introduce the notion of a subnet. Subnets are the logical representation of an organization or a community. Each subnet forms a common ground that associates people with shared interests, defines a pool of devices and associates a concept of ownership with them, and organizes people into groups that have different access rights/roles within the organization. In addition, information is classified into channels, which are a characteristic of that particular subnet, according to semantic content. In our prototype, we have used subnets as a means for demarcating administrative domains.

## 5.4 User Management

InfoSpaces supports access controls for controlling the content that is published to the various end-points. Currently, the system provides four different levels of access. These have been implemented as groups with a common set of privileges and users are assigned to these groups.

*Administrators:* This is the InfoSpaces administrator group. It is a globally unique group that controls the entire system and has administrative privileges over all Subnets.

*Subnet Moderators:* Members of this group can manage the various subnet operations like creation of channels, approval of new content providers and content managers in the subnet.

*Content Editors:* Members of this group can approve content submitted by the content providers. They can disapprove inappropriate content from being published by providers/users. This additional group allows us to open up the next category (content providers) to everyone who wants to participate in the information network. This is coherent with the design goal of making InfoSpaces accessible to all willing participants in an organization.

*Content Providers:* These are the InfoSpaces users who are the primary sources of information. Content provided by them is subject to approval by a group higher up in the InfoSpaces hierarchy.

Privileges given to the user are checked upon authentication, and access to the menu of functions is restricted accordingly on a per subnet basis. However, different users can have different privileges across subnets (since they can belong to different groups in different subnets).

## 6 Evaluation

In this section we present our observations when we tested the InfoSpaces prototype against our design goals, the main ones being:

1. Scalability in terms of number of subnets that can be supported
2. The number of data-items that can be supported for each subnet and each endpoint; and for the entire system
3. The performance of the scheduler in the presence of heavy load conditions
4. Support for high display refresh rates on the end-points (and consequently higher scheduling activity)
5. Ability to support diverse device types and number of configuration changes required to support a new device class
6. The ease of use of the system and the overall cost of deployment of the system.

The system does not put an upper bound on 1 and 2, consequently their values are constrained only by the capability of the database server. Since we are using an industrial strength RDBMS, 1 and 2 cannot become bottlenecks to scalability.

The worst-case performance of the scheduling algorithm is of the order of  $\log n$ , where  $n$  is the number of active data-items in a particular channel. We have tested our system with approximately 10000 data-items per channel, thousands of channels and several subnets. We gathered the data-items by using a web extraction module written in WebL [3]. At the same time, we had several clients accessing and modifying the content database, thus checking the performance of the scheduling algorithm under rapidly changing load. The system performed as expected without any noticeable degradation of performance. Since Quality of Service and fairness were important issues in our design, in addition to making sure that the



system performed well, we verified that each end-point receives the content in the right priority order. In addition, across all end-points subscribed to the same channel, content was disseminated in a fair and timely manner, thus guaranteeing our scheduling predicates mentioned in section 3.3.1.

A new class of device requires its own content adaptation proxy which can receive the output of the scheduler and format it according to the device properties. The cost of implementing such device-class specific content adaptation proxy is amortized across all the devices of that particular class.

The interface and the APIs are very simple. We simplified the user management process and did not implement finer grain access rights and event priorities to make the system easy to use by a normal user.

## 7 Related Work

While InfoSpaces differs from other publish/subscribe systems, such systems are the foundations for the project in many ways. The “Information Bus” project headed by Dale Skeen, now at Vitria, laid some of the foundations for such work. The Gryphon Project at IBM TJ Watson Research Center has also produced interesting results in this area with a focus on efficient *content-based* systems that route information based on the semantics of the information.

The Infospaces work borrowed heavily from concepts outside the traditional publish/subscribe world for issues such as scheduling and content adaptation. Harty and Cheriton’s work in memory management, while clearly not belonging to this domain, was highly useful to us in our design.

Content adaptation proxies have been studied in detail and are a well-understood subject. Several companies [Pumatech, Broadvision] have created products that use content adaptation at a proxy to improve avail-

ability of data to clients. Research work in this area is extensive and well documented and several pointers to papers in this area are included in the references.

## 8 Future Work

Currently, our system has static channels in each subnet with data-items being assigned to the channels statically. The publishers publish the data-items to specific channels and the end-points subscribe to the channels of interest. We intend to extend this to provide interest based dynamic channels with temporal subscription semantics.

We are also debating on the possibility of ‘short-circuiting’ the information flow path from the content-provider to the content-consumer (end-point) to route information directly onto the end-point and bypass the scheduling mechanism. Although this approach might give an extra degree of freedom to the content-provider, it decentralizes device control.

## 9 Conclusions

The InfoSpaces implementation demonstrates how a centralized public information network can be built to cater to the information aggregation and dissemination needs of large organizations. We have successfully designed and implemented one such network.

Our network aims to deliver data that is relevant to the majority of the people who happen to see it. We have achieved this goal by implementing demographically targeted information delivery.

In addition, the problem of managing and scheduling large data sets led us to modify a scheduling model with successful results.

We removed the dependency of the scheduler on device specific parameters by introducing a level of indirection through the

content adaptation proxy. This ensures that our system scales well to diverse device types and different device capabilities.

We anticipate our work to form a substrate on which future work towards public information delivery systems can be carried out. Our platform closes the gap between research focused on information extraction from diverse providers and research on designing new delivery end-points.

## Acknowledgements

We would like to thank Andy Huang and Emre Kiciman for reviewing this paper. Andy helped us with the logistics for the prototype. We have also benefited from discussing the prototype with Kathy Richardson.

## References

- [1] Kieran Harty, David R. Cheriton. Application-Controlled Physical Memory using External Page-Cache Management. Proceedings of the Fifth International Conference on Architectural Support for Programming Languages and Operating Systems, Boston MA, Vol. 34, No. 5, October 1992.
- [2] IBM T.J. Watson Research Center. Gryphon: Publish/Subscribe over public networks. <http://www.research.ibm.com/gryphon>
- [3] Compaq Web Language (WebL) Specification 3.0h <http://www.research.compaq.com/SRC/WebL>
- [4] W3C. Extensible Markup Language (XML) Specification 1.0 <http://www.w3.org/TR/2000/REC-xml-20001006>
- [5] Laurence Melloul, Trevor Pering, and Armando Fox. WeSCoS: A First Step Towards Web Programming. Unpublished draft. <http://swig.stanford.edu/pub/papers/webc/we-scoss.pdf>
- [6] Banavar et. al., An Efficient Multicast Protocol for Content-based Publish-Subscribe Systems. IBM T.J. Watson Research Center.
- [7] James Cutler, Charles Fraleigh, Devendra Jaisinghani, Dora Karali, and Emre Kiciman. Luddide: Location and User Dependent Information Delivery. Project Report. Available at: <http://www.stanford.edu/~emrek/class/luddide/report.html>
- [8] Udi Manber, Ash Patel, and John Robison. Experience With Personalization of Yahoo! CACM 43(8), August 2000.
- [9] IBM Gryphon project documentation, An introduction to Content-based publish/subscribe Systems. [http://www.research.ibm.com/gryphon/Contentbased\\_Publish\\_Subscrib/content-based\\_publish\\_subscrib.html](http://www.research.ibm.com/gryphon/Contentbased_Publish_Subscrib/content-based_publish_subscrib.html)
- [10] Eric A. Brewer. Lessons from Giant-Scale Services. Submitted for publication.
- [11] Ralph B. Case, Stéphane H. Maes and T. V. Raman. Position paper for the W3C/WAP Workshop on the Web Device Independent Authoring. <http://www.w3.org/2000/10/DIAWorkshop/case.htm>
- [12] Dale Skeen. Vitria's Publish/Subscribe Architecture: Publish/Subscribe Overview. Available at: <http://www.vitria.com>
- [13] Brian Oki, Manfred Pfluegl, Alex Siegel, Dale Skeen. "The Information Bus – An Architecture for Extensible Distributed Systems", Operating Systems Review, Vol.27, No.5, Dec.1993
- [14] Armando Fox, Steven D. Gribble, Yatin Chawathe, Eric A. Brewer. Adapting to Network and Client Variation Using Active Proxies: Lessons and Perspectives. IEEE Personal Communications (invited submission), August 1998.
- [15] B.Zenel, D.Duchamp, A General Purpose Proxy Filtering Mechanism Applied to the Mobile Environment, MobiCom 1997.
- [16] Davies et. al. Caches in the air: Disseminating tourist information in the GUIDE sys-

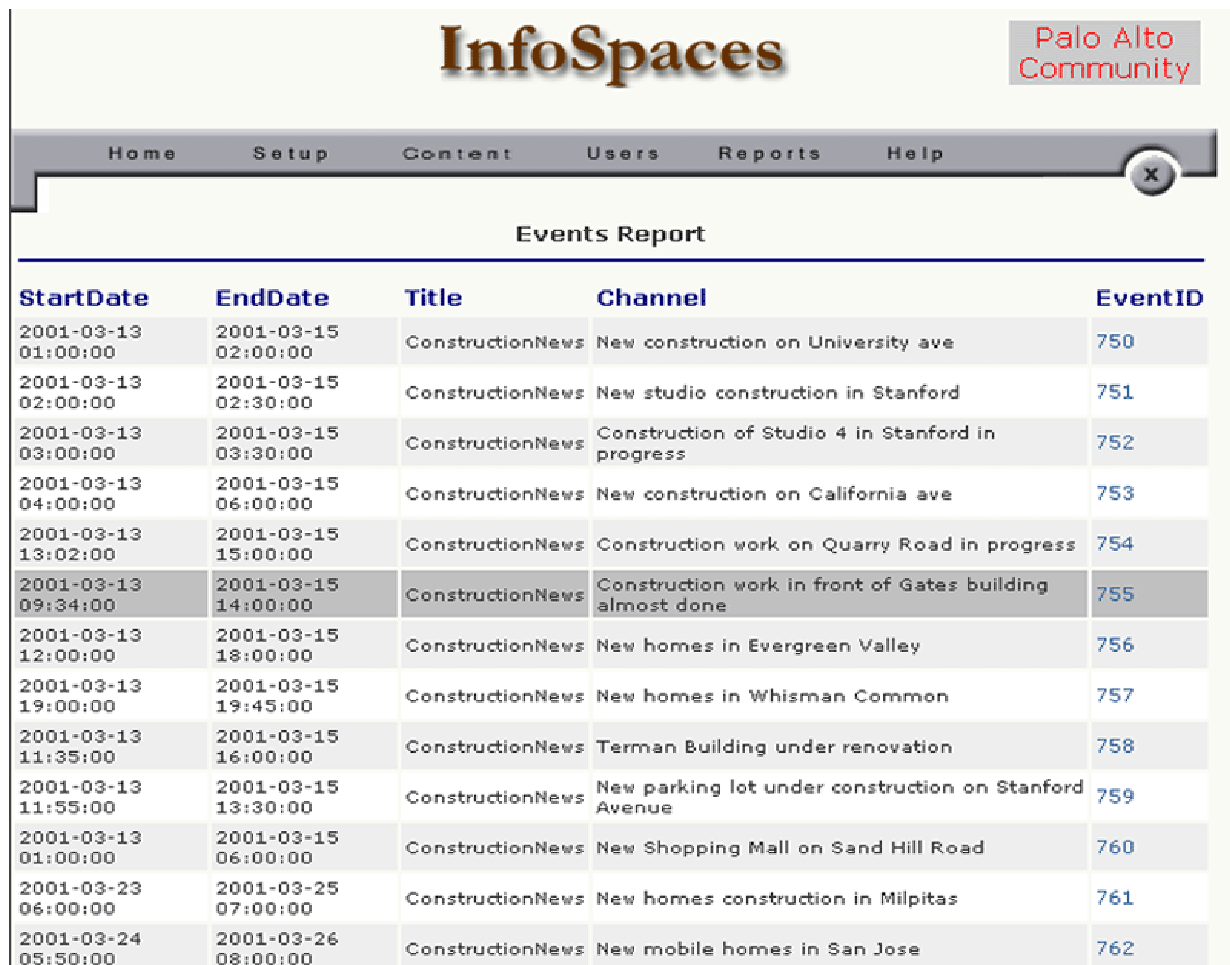
tem. In Proceedings of Second IEEE Workshop on Mobile Computing Systems and Applications, New Orleans, Louisiana, February 1999. IEEE Computer Society Press.

[17] S. Acharya, M. Franklin, and S. Zdonik, Dissemination-based data delivery using broadcast disks. IEEE Personal Communication, pp. 50--60, Dec. 1995

[18] N. Vaidya and S. Hameed. Improved Algorithms for Scheduling Data Broadcast. Technical Report 96029, Dept. of Computer Science, Texas A&M University, 1996.

## Appendix A

We present some screenshots of both the web interface of InfoSpaces as well as two different kinds of end-points to illustrate how content is displayed on them. Figure 3 shows the administrative user interface and one see the listing of all the current content for the “Palo Alto Community” subnet. Each data-item (event) is listed with its start date, end date, and a brief description. In Figure 4, one can see the content as displayed on a ‘Ticker’ class end-point that scrolls data from left to right. Only a part of the data-item is visible in the snapshot. Figure 5 illustrates Microsoft Web-TV formatted content. This end-point has the ability of displaying multiple events at a time, with a refresh rate customizable to the number of events currently being displayed.



The screenshot shows the InfoSpaces administrative web interface for the Palo Alto Community. The page has a navigation menu with links for Home, Setup, Content, Users, Reports, and Help. Below the menu is a table titled "Events Report" with columns for StartDate, EndDate, Title, Channel, and EventID. The table lists 16 construction-related events, each with a start and end date and a brief description.

StartDate	EndDate	Title	Channel	EventID
2001-03-13 01:00:00	2001-03-15 02:00:00	ConstructionNews	New construction on University ave	750
2001-03-13 02:00:00	2001-03-15 02:30:00	ConstructionNews	New studio construction in Stanford	751
2001-03-13 03:00:00	2001-03-15 03:30:00	ConstructionNews	Construction of Studio 4 in Stanford in progress	752
2001-03-13 04:00:00	2001-03-15 06:00:00	ConstructionNews	New construction on California ave	753
2001-03-13 13:02:00	2001-03-15 15:00:00	ConstructionNews	Construction work on Quarry Road in progress	754
2001-03-13 09:34:00	2001-03-15 14:00:00	ConstructionNews	Construction work in front of Gates building almost done	755
2001-03-13 12:00:00	2001-03-15 18:00:00	ConstructionNews	New homes in Evergreen Valley	756
2001-03-13 19:00:00	2001-03-15 19:45:00	ConstructionNews	New homes in Whisman Common	757
2001-03-13 11:35:00	2001-03-15 16:00:00	ConstructionNews	Terman Building under renovation	758
2001-03-13 11:55:00	2001-03-15 13:30:00	ConstructionNews	New parking lot under construction on Stanford Avenue	759
2001-03-13 01:00:00	2001-03-15 06:00:00	ConstructionNews	New Shopping Mall on Sand Hill Road	760
2001-03-23 06:00:00	2001-03-25 07:00:00	ConstructionNews	New homes construction in Milpitas	761
2001-03-24 05:50:00	2001-03-26 08:00:00	ConstructionNews	New mobile homes in San Jose	762

Figure 3: The InfoSpaces administrative web interface



The screenshot shows a black background with yellow and white text. The text reads "ray Architecture - Mar 22 1".

Figure 4: Content as displayed on a ticker-type end-point

**InfoSpaces : 19-Apr-01 09:04 AM**

**Database Seminar, Winter 2001**

Mar 13 12:00 AM to Mar 15 12:00 AM  
Gates B01

**Compiler Seminar**

Mar 13 12:00 AM to Mar 15 12:00 AM  
Terman Building

**Advanced Topics in Operating System**

Mar 16 12:00 AM to Mar 18 12:00 AM  
TCSeq 201

**X Window System Programming**

Mar 15 12:00 AM to Mar 17 12:00 AM  
Gates B03

Figure 5: Content as displayed on Microsoft Web-TV