Replication and Notification Management in a Knowledge Delivery Network

Nikos Anerousis

Voicemate, Inc.
29 W. 34th Street, 10th Floor
New York, NY 10001
nikos@voicemate.com

Abstract. Knowledge delivery networks are slowly becoming the foundation behind the day-to-day operations in information-intensive industries. This paper attempts to highlight some central management problems facing these networks, especially for voice/telephony-based interaction. We present the concept of a knowledge delivery network and identify the data structures and its basic architecture. Using an overlay technique, we implement the knowledge network over an existing data network infrastructure. The knowledge network moves (replicates) information using XML-formatted packets that are processed in every node, drawing from principles encountered in active networks. We describe the management problems encountered in such networks, and focus in particular on replication strategies, and notification delivery.

1 Introduction

Today, information-intensive industries generate and consume large amounts of information in the form of facts, opinions, recommendations etc. The information can be of different forms such as text, audio or video. We are interested in a particular subset of this information that has the following characteristics:

- It is of small to medium length (it does not take a human more that a few seconds or minutes to absorb it)
- It is self-contained
- It contains sufficient meta-information that allows it to be indexed in a database.

Example: An analyst in the financial industry has just completed the quarterly earnings report meeting with the management of the company that he/she covers. There is new information regarding the company's revenue guidance for the next quarter. This information needs to be communicated to several clients that the analyst is supporting. In addition, it needs to be stored and properly indexed in a database for future reference, and also perhaps communicated to the media. The typical process of creating such a report would require the analyst to return to his office and create a document that is then submitted to the research database/repository and then sent to all interested parties.

The publish/subscribe model that we just described is today a well understood process and there are several commercial products to implement this functionality today. In early 2000 we began to experiment with alternate methods of capturing and delivering information with the ultimate objective to significantly shorten the lifecycle from the moment it became available to the moment it was delivered to its recipients. We were strongly motivated by work in the area of pervasive computing [ABO01], and examined devices and modes of interaction that made this possible in mobile settings. The ubiquity of the cellular telephone and advances in speech recognition technology attracted our attention and motivated us to explore further the possibilities of capturing and organizing information in audio form. Especially in mobile settings, the ability to continuously capture one's voice as opposed to interacting with a small screen and keyboard has significant implications in the overall time needed to publish information. Capturing voice as opposed to text has also other advantages such as maintaining the speaker's texture, speed and tone [CHA91].

In our previous work [ANE02a] we have described an integrated environment for the capture, organization and delivery of semi-structured voice (audio) knowledge with the above characteristics. We refer to this information as "knowledge" because it always carries a semantic structure that allows its classification in a knowledge database. Typically, it corresponds to short messages or notes that reflect the author's knowledge of a topic. The system that we developed was capable of supporting the authoring process via a telephone, guiding the author through the mandatory specification of semantic structures (e.g., keywords) that allow a subsequent classification in the database, hosting and delivering this knowledge from a centralized location. We demonstrated that, by using voice rather than text, it is possible to significantly shorten the knowledge production lifecycle while at the same time increasing the pervasiveness of this process. Assuming a typical length of 2-3 minutes for a new story, we demonstrated the feasibility of delivering breaking news and other similar urgent messages in less than 5 minutes from the time the information became available to a large number of recipients. Note that we are not arguing that these timescales are not feasible with text/GUI-based interaction, but rather that the overall experience, convenience and increase in productivity were superior using the voice-based system, based on user trials that we conducted.

One of the main issues that we observed with the original system was its centralized nature. In an era when knowledge drives important decision-making processes and workforces are becoming increasingly mobile and information-dependent, we found several problems with the centralized approach. For example:

- Telephony access due to its very nature is more cost effective and reliable when placing calls to a nearby location compared to a (possibly remote) centralized facility.
- The centralized system proved cumbersome in terms of administration when multiple user groups with distinct localization parameters and access privileges had to be configured on the same node.
- The centralized system constituted a single point of failure for the service.

In this paper we are interested in connecting such (previously centralized) knowledge repositories through a data network such as the Internet. We used the term "knowledge network" to describe our work since the objective was to create an environment where knowledge could move seamlessly from its point of origin to other locations where it would be utilized. A knowledge network is not a fundamentally new data transmission network. Rather, it relies on well-understood networking techniques to implement transport and notification functionalities. It bears some similarity with previous work in content distribution networks [VER02], but focuses exclusively on the following fundamental requirements:

- Allow new stories created in one node of the network to become available in remote nodes. This is accomplished through a replication mechanism.
- New stories falling in a user's particular area of interest must be able to be delivered through an alerting function to the user, typically a telephone callout or a text message.

Data replication mechanisms are an old problem in computer science literature with applications ranging from simple file replication (mirroring) [HAC88, RAB01] to memory management in multiprocessor computer architectures. In the networked knowledge environment that we describe, the replication task has the following additional attributes:

- The unit of replication (a story) is a more complex structure that consists of meta-data and digital media (files). It involves both file system and database replication operations.
- Replication is not statically configured; rather, replication parameters are evaluated dynamically as knowledge "flows" through the network, in a very similar way as data packets are routed through the Internet. In fact, determining a knowledge replication policy is quite similar to setting up a multicast tree.
- Notifications are an integral part of the replication process, sharing the basic triggering mechanisms.

To deal with the complexity involved in setting up a knowledge replication and notification service we introduce a mechanism for routing knowledge between the nodes that relies on parsing XML-formatted messages (packets). In contrast with traditional packet headers in data networks that are optimized for hardware operations, a packet with an XML header requires a sophisticated processing stack implemented in software; it allows however for implementing very flexible packet processing policies, since it is now possible to extend the basic format with arbitrary extensions that are not subject to backward compatibility constraints. Our work is very related to active network technologies [TEN96] that rely on additional information encapsulated in the packets to perform additional control operations as packets travel through the network.

This paper is organized as follows: Section 2 presents the architectural details on the basic data structures and mechanisms that compose the knowledge network. Section 3 presents the management aspects encountered in knowledge networks. Section 4 concludes with our findings and directions for further study.

2 Architecture

Knowledge networks are formed by the combination of numerous components and data representation schemes. We begin our description with the fundamental data structures used to describe knowledge:

2.1 Information Model

The information model defines the fundamental metaphors and data structures required to characterize knowledge. Several models have been used in the past in similar knowledge representation applications: The IEEE learning object model [LOM], the Dublin core metadata initiative [DCMI], etc. model is The unit of knowledge is a "*story*". A story consists of a set of media components, referred to as *story items*, and meta-data. Story items are multimedia files, such as audio and video clips, textual documents, graphics etc. Currently, we require that at least one of these items (the "body") be in audio format. Each audio story item can exist in various formats to support applications that require delivery to devices other than telephones (e.g., PCs and portable audio players). We support 4 different audio formats: G.711 (for telephone delivery), Windows Media, Real Audio and MP3. Story items can also include references to external content (e.g., URLs) for integration purposes.

Table 1. Example of story attributes

Story ID	01JX2002032508SL			
Title	Key take-aways from the Application Software Group			
Author	Research Analyst			
Date	03/25/02 01:08 PM			
State	Online			
Story Items	Name	Length	Format	Length
	Body	2:11	G711	125s
	Body	2:11	WMA	125s
	TextReport	41KB	DOC	12KB
Keywords	Ticker	BVSN		
	Secon-dary	EPNY, ITWO, ARBA, INFA, MSFT		
	Industry	Application Software		
	Subject	Change of Opinion		
	Region	North America		
	Tags	intellectual property, law, strike price, expense stock options, Application Software, Inc., Ariba, Broadvision, Informatica, accounting rules, stock options, software industry, Proposed Changes In Stock Options, Microsoft, I2 Technologies, current price, E.piphany, knowledge based industry		

Story meta-data is textual and consists of a set of fixed attributes, such as story title, author, creation date, physical location, current state (online, offline, sent for approval, under revision, deleted, etc.), expiration date, priority, language, length, external references, etc., and a set of keywords in the form of attribute name-value pairs. The names of these keywords are determined from the context in which a story is created. The content administrator determines beforehand a set of keyword categories that can be used as keyword names (e.g. company name, region, subject, etc.). Keyword values can be selected from an enumerated set (e.g., region can be North America, South America, etc.) or from an unrestricted set. Enumerated sets can easily be entered using speech recognition since they translate into finite vocabulary lists that can be supplied to a speech recognizer. Unrestricted sets, however, are not compatible with a voice user interface that does not support natural language recognition; they are compatible however with Web access. The following table is an example of a story according to the above information model.

For convenience, stories are grouped into *channels*, which represent semantic grouping structures. For example, an "Automotive Industry" channel will contain news and research on issues related to the automotive industry. Channels can then be organized in a tree-like structure of categories and subcategories, or for that matter, any hierarchical structure that conveys an intuitive classification of knowledge. The latter is determined by the content administrator and should be designed with the objective to facilitate navigating and location of stories both for PC (Web) and telephony access.

2.2 Knowledge Nodes

A knowledge network facilitates the access and distribution of knowledge to its participants. As every network, it consists of a set of interconnected nodes. However, unlike traditional data networks where nodes implement simple packet processing and routing tasks, a knowledge network node is a collection of components that support the entire functionality required to create, organize and deliver knowledge. Details about the internals of a knowledge node and client connectivity can be found in [ANE02a]. At a minimum, a knowledge node (K-Node) contains the following:

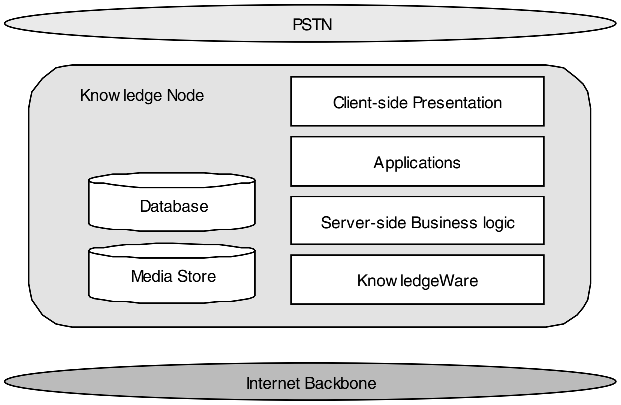

Fig. 1. Example of a Knowledge Node

- A media storage system (usually a traditional file system)
- A relational database
- Connectivity to a data network (the Internet) and the telephone network
- A middleware layer (the KnowledgeWare) that facilitates the development of applications by exposing an API that controls most knowledge creation and management functions
- Applications that allow users to create and retrieve knowledge using PCs or telephones and handle all related administration tasks
- A presentation layer that generates the client-facing user interfaces (web pages or VXML-based speech interaction).

Users typically connect to this network from their home node, which contains the master copy of their personal profile. There are two modes of interaction:

a) A “pull” mode, where the user connects to the node (typically via a phone call) to create a new story or browse the local knowledge database.
b) A “push” mode, where knowledge is pushed to the user via a telephone call-back or a text-based message (SMS, email).

The methods used to author or navigate knowledge using a telephone and voice commands were presented in [ANE02]. The problem that we need to solve here is how to allow users connected to their home node to access knowledge created remotely; and how to make users aware of the existence of such knowledge through an alerting function. In the browsing or “pull” mode, users always interact with the database at their home node and they cannot be aware of the existence of a story that has been created remotely, unless its metadata has been copied locally. To solve the problem for the pull access mode we need to introduce a replication mechanism that copies the story metadata and media files to the local mode (note that copying of the actual media files is always necessary to ensure smooth telephone access).

Delivery of the story to a recipient in the push mode requires that either the story is first replicated to the home node, where the user has registered its notification preferences, or the notification preferences are propagated to other locations in the network that are capable of monitoring all new stories that are being created. These nodes are capable of initiating callouts to the appropriate users.

2.3 Network Architecture

One way to implement the knowledge network is to use an overlay structure on top of the Internet. The Internet is used as the primary transport mechanism and the knowledge nodes act as intelligent routing points. The traffic flowing through the knowledge network is of two types: a) story metadata and b) digital media (the actual contents of the story).

The story meta-data is encapsulated in packets in the XML format. We named these packets *K-Grams* (for “Knowledge-Gram”). In addition to the story meta-data, the K-Gram has additional fields that specify the handling of the packet in other nodes of the knowledge network. It is possible for example to incorporate in the packet a source routing policy, a replication policy, explicit alerting information, etc. In essence, the knowledge network behaves as an active network, with K-Grams in the role of active packets that receive special processing in every node. The digital media components of the story do not participate in active network functions. Rather, K-Gram processing may trigger the transfer (replication) of the digital media components.

The resulting architecture consists of two network planes: The transport plane, which is a conventional data network, and the knowledge plane, which consists of knowledge nodes. Fig. 2 demonstrates this concept.

Nodes use the facilities of the transport network to exchange K-Grams. K-Grams are exchanged using a SOAP interface, i.e., they have procedure call semantics. When a K-Gram is received at a node, its contents are passed to a packet processing function, which determines the next steps. One or more of the following are possible:

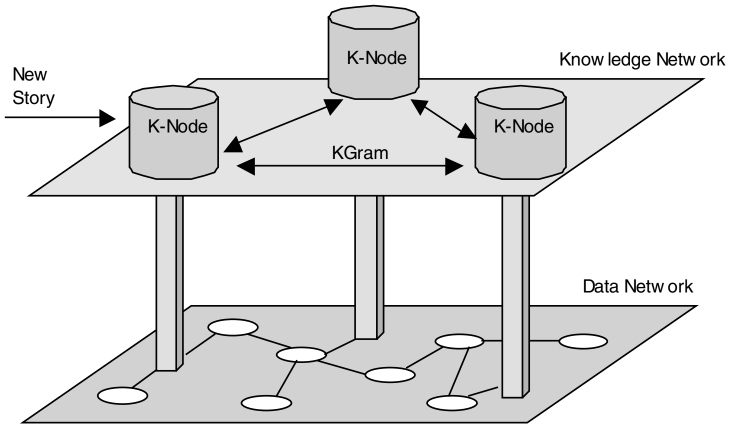

Fig. 2. Network architecture

- The K-Gram is forwarded to one or more other nodes
- The digital media components are copied (replicated) to the media storage of the current node
- The K-Gram is further processed locally to generate notifications to users.

The forwarding function is implemented by the knowledge routing component. The latter will determine the subsequent steps of routing the packet:

- Replicate the story in the current node: The metadata from the K-Gram is used to create a new entry for the story in the database. Then, the media components are replicated to the local media storage. Replication is necessary because interactive knowledge delivery applications such as the ones that enable voice/telephony access require low-latency access to the digital media. We use the HTTP protocol to retrieve the media components from the URL that is carried in the K-Gram.
- Drop the packet (no further processing required)
- Forward: the packet is relayed to other peer nodes. Rewriting of several K-Gram fields may be necessary (e.g. to replace to replace the URL for the media components with the one of the current node).

2.4 Fundamental Data Structures

There are two fundamental data structures used in our system: The K-Gram is used as a “knowledge packet” between nodes of the knowledge network. The knowledge filter is used to trigger replication and notification actions.

2.5 K-Gram (Knowledge Packet)

The K-Gram is an XML-formatted data structure used to communicate all information related to a story. It carries all the story meta-data and in addition, a set of pointers (URLs) to the media components.

The XML format [XML] has many advantages including easy consistency checking using DTDs, compatibility with remote procedure call semantics (SOAP). The purpose of K-Grams is 2-fold: One is to serve as the basic messaging unit with which stories can be replicated across the knowledge network. The other is to deliver a story to a number of recipients.

The K-Gram is composed of 3 main blocks (see Figure 3): The header is denoted by the <head> tag and contains routing information. There is always an <origin> field representing the node where the K-Gram was originally created (e.g. the node where a story was published). There is an optional <route> tag which is used in source-routing applications. The <action> tag specifies the type of action that will be taken upon receipt. The second block contains information about the story carried in the message. There are several XML tags representing the story metadata and the media components. For the latter, a URL points to a location where the actual media resides. The third block is used in notifications and includes the users and their contact information. The node that receives a notification K-Gram goes through the list of users and initiates the appropriate alerting application.

```
<xml>
<kgram>

<head>
  <origin node ="A"/>
  <route>B, C, D</route>
  <action>replicate</action>
  <action>notify</action>
</head>

<story id=01JX2002032508SL/>
  <title>Key take-aways from the
          Application Software Group</title>
  <author id=98321/>
  <creation_date>03/25/02 01:08 PM </creation_date>
  <state>Online</state>
  <media>
    <item name=body format=G711 url=U1 />
    <item name=body format=WMA url=U2 />
  </media>

  <keywords>
    <key name=Subject value="Change of Opinion" />
    <key name=TICKER value="BVSN" />
    <key name=Industry value="Application Software" />
    <key name=Region value="North America" />
  </keywords>

</story>

<recipients>
  <user id="8823003" telno="93849434"/>
  <user id="8823004" telno="92149412"/>
</recipients>

</kgram>
```

Fig. 3. Example of a K-Gram

2.6 Knowledge Filters

The creation and routing of a K-Gram is often the result of a new story matching a particular set of criteria. The latter are expressed using a "knowledge filter". A knowledge filter is defined as a Boolean assertion on the story meta-data. For example, a filter may match if the author is named "John Q. Public" and one of the ticker symbols associated with the story contains "AOL" and "MSFT".

```
Story.author = "John Q. Public" AND Story.ticker CONTAINS ("AOL", "MSFT")
```

Filters are stored in a separate table in the database of the knowledge node. Every time a story is created (either as the result of an authoring action or through replication), its meta-data is matched against the stored filters in the local node. When a filter matches, the action associated with the filter is executed. The following actions are possible:

- Notification: the story is delivered to a particular user in one of the supported interaction modes (SMS, email, telephone call, etc.).
- Replication: The story is replicated to a remote node. Note that replication of a story in another node may trigger itself the evaluation of filters in the remote node. This may result in new replication or notification actions, propagating the story even further.

Typically, the filters containing replication actions are programmed by an administrator, through a management function responsible for controlling the replication policy across the entire network.

3 Management

Both the replication and notification functions need to be optimized across the entire network. The algorithms that perform these optimizations work from a centralized location, and therefore a centralized management architecture was a natural fit.

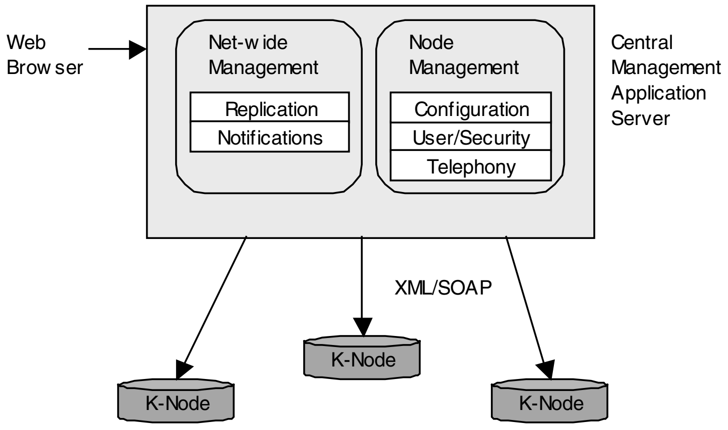

Fig. 4. Management Architecture

The management console provides a web front end driven by a Java application server, which hosts all the management applications. A management agent resides at every knowledge node. Since our system uses XML extensively, we decided to use an XML-based management protocol. Recently, XML is increasingly being used in network management applications [ENS01] because of its flexibility to describe complex data structures and interactions. We chose to implement a SOAP layer on top of the application server resident in every node to handle all management activity between the central location and the node [SOAP]. Management applications range in complexity from simple node configuration and user management functions, to the more complex replication and notification management functions and are classified in two categories: Node-specific, and network-wide.

The node-specific management functions include node connectivity configuration, user and security management, content organization, database connectivity, media storage control, telephony resource and user interface component management. The network-wide management functions include node group management, replication and notifications (see Fig. 4).

The detailed description of the entire management architecture and services is beyond the scope of this paper. The remainder of this section focuses exclusively on replication and notification management.

3.1 Replication

Using knowledge filters and K-Grams the administrator can implement many replication policies to satisfy latency, topology or bandwidth constraints in delivering stories to end users. We found through experiments that the configuration that balances administrative overhead and implementation cost is a hierarchical structure at the *country* level. The reason for this is fairly straightforward: serving users within their country is much cheaper than serving them from other international locations, especially when telephone access is a priority. There is at least one K-Node in every country domain that participates in the top-level domain. In addition, there may be 0 or more nodes within a country organized in a subordinate domain. Several levels can be defined in this hierarchy, however, at the scales with which the system has been deployed so far, we have not found any advantages of using more than 2 levels (the top level and one level of subordinate nodes).

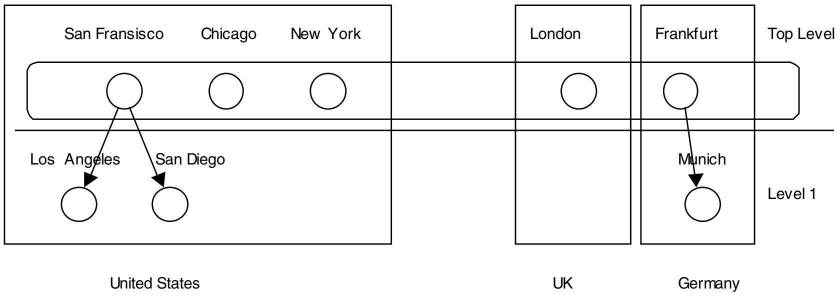

Fig. 5. Example of a 2-level hierarchy

Every node maintains a communication link with the peers at the same level. An additional link is maintained with the parent (except of the nodes at the top level). K-Grams flow only from the node to its peers, to the parent (for a top-level node) or to the node's subordinates (if any).

In order for a story to be replicated across the knowledge network, the administrator needs to install the appropriate knowledge filters in a number of nodes. This task is very similar to controlling routing tables in a packet network. We have developed a dedicated tool to accomplish this task. The tool takes as input the topology of the knowledge network and allows the installation of a number of predefined replication policies. It is also possible to manually edit a replication policy, however, this is still an arduous task. The predefined replication policies are the following:

Complete replication: nodes are configured so that if a newly created story matches a filter at the origin node, it will be replicated across all the nodes in the network. This is accomplished in the following way:

The filter in the origin node sends the K-Gram to its subordinates, and its parent(s). If the receiving node is at the top level of the hierarchy, there are no parents, and the K-Gram is forwarded to the peer nodes at the top level. The iteration continues recursively until there are no more nodes to forward the message. At every step, several fields of the K-Gram are updated to reflect the progress of the replication (i.e., content URLs are modified to point to the newly replicated copies of the media files). Note that in all iteration steps, a node is not required to send the K-Gram to its peers, since they will be receiving it from their common parent.

Subordinate replication: The story is sent only to the subordinate nodes (and then recursively to all subordinates if there are more than 2 levels in the hierarchy).

Parent/peer replication: The story is sent to the parent, and from there, to the peers of the origin node.

Single top node: The story is replicated under a particular top node

Country only: the story is sent to the top nodes of a particular country, and from there to their entire subtrees.

The process to install a replication policy works as follows: The manager starts the replication configuration tool. He/she specifies the filter criteria that trigger a replication and the nodes in which the filter will be installed. He/she then selects a replication policy from the list above. There is a graphical verification tool that depicts the nodes at which replication will be triggered and the path that K-Grams will follow to their destination. The graphical tool makes it substantially easier to catch configuration errors. The above process is repeated for all necessary replication filters. Once the entire policy is completed, a policy optimizer is invoked to remove duplicates and determine the filter action parameters with which the filters will be installed at the various nodes. The final step is the installation of the policy in the actual nodes.

3.2 Notifications

In addition to being replicated, a story can be sent immediately upon its publication to users that have expressed interest in receiving it. User interest is expressed by the creation of a knowledge filter at the home node of a user. The filter is similar to a replication filter (it contains an assertion on the meta-data associated with the story) but in addition contains a set of delivery preferences: a) the alerting mode (SMS, email or telephone), and, b) the delivery time window (e.g. business hours only or anytime).

Privacy is a very important design parameter of the service, and the above features have been engineered to implement exactly that: let the user control the intrusiveness of the service he/she receives. It is possible for example to use intrusive telephone notifications during the business day and email for all other times. Telephone notifications are useful since most of the knowledge collected in our system is in the form of short audio notes.

We are mostly interested in the telephone-based notification service, where the story is delivered in its (native) audio form to the user alongside with other interactive dialogues (e.g. to confirm receipt, leave a feedback to the author, etc.). Implementing a high-performance and cost-effective telephone-based notification service requires some additional considerations. First, nodes have typically limited port capacity to handle many simultaneous outgoing calls since they have been engineered to handle mostly the incoming traffic. Second, the great variability in pricing telephone calls has created some paradoxical cases where, for example, it is cheaper to call from New York to Ireland than from London. Assume for a moment that nodes exist both in London and New York and that the rates to call London from Ireland are low, but the reverse (from the UK to Ireland) is prohibitively expensive. Even if the London node serves the users in Ireland (because of low rates from Ireland to the UK), it may be cheaper to initiate notifications from New York (because of the high rates from the UK to Ireland. This paradox, which occurs quite often in international calling, had some interesting implications in the design of the notification service.

To handle the cases such as the one described above, we have chosen to study 3 different notification implementation policies:

3.3 Decentralized with Local Delivery

In this case the filter triggering the notification exists only on the home node of a user. The notification is triggered when the new story has been created in the local node, or when it has been replicated from a remote node. The outgoing calls are made from the local system and are limited by the local port capacity.

This scheme is the easier to implement since it does not involve processing filters outside of the local node. Its main disadvantage however is that it does not propagate the author's interest to remote nodes. If for example the administrator has not enabled replication of stories on a particular UK company in London, a user in New York will not be able to receive the story even if he expresses interest. Other disadvantages are the increased delay, especially for remote stories, before notifications can be generated (since there is a replication step), and the possibly suboptimal cost for delivering the notifications to the user.

3.4 Centralized with Centralized Delivery

By centralizing the evaluation of notification filters and the alerting process it is not necessary to replicate stories across the entire network. Rather, the administrator only needs to make certain that stories are replicated to the node that performs filter evaluation. Also, moving the alerting process to a single location can also result in better toll savings since the delivery cost can be controlled more closely. In addition, callouts can be prioritized based on criteria such as class-of-service, etc.

The main disadvantage of the centralized processing is that it consists a single point of failure. In addition, user filters need to be transferred from the user's home node to the filter evaluation node. Further, the installed port capacity in the local nodes is not used for notifications, something that generates significant capacity requirements for the node that handles notifications.

3.5 Centralized with Decentralized Delivery

We devised a scheme that compromises between the centralized and the decentralized approach. According to this scheme, filters are evaluated in a central site (so new stories still need to be replicated to the central site) but the delivery is happening in a decentralized fashion, after solving an optimization problem that minimizes the delivery cost and the maximum delay in delivering a notification to the end user. The problem is formulated as a linear programming that is executed at the central (coordination) node after all filters have been evaluated for a new story.

The Boolean variables x_{uN} denote whether user u will be notified from node N. Only one node will be initiating the notification to a particular user.

$$\min \sum_u \sum_N x_{uN} C_{uN} l \qquad x_{uN} \in \{0,1\} \quad \sum_N x_{uN} = 1$$

$$b_N + \left\lceil \frac{\sum_N x_{uN}}{P_N} \right\rceil \cdot l < d_{\max} \qquad N = 1 \ldots n$$

where

C_{uN} is the unit cost to deliver to user u from node N

b_N is the backlog (in minutes) at node N

l is the length of the story (in minutes)

P_N is the port capacity of node N

$d_{\max}$ is the maximum delay for outbound calls

Once the notification schedule has been determined, the coordination node uses K-Grams to replicate the story to the nodes where callouts will be initiated. In addition to the story metadata, the K-Gram carries a list of users to be notified (along with the phone numbers and other contact information). The receiving node then uses the K-Gram to create a local replica of the story and initiate the notifications.

4 Conclusions and Future Work

Knowledge delivery networks are slowly becoming the foundation behind the day-to-day operations in information-intensive industries. This paper attempted to highlight some central management problems facing these networks, especially for voice/telephony-based interaction. We defined the function of a knowledge delivery network and identified the data structures and its basic architecture. Using an overlay network technique, we were able to implement the knowledge network over an existing data network infrastructure. The knowledge network moves (replicates) information using XML-formatted packets that are processed in every node, drawing from principles encountered in active networks. To the best of our knowledge, the use of XML to represent a packet header is new and presents some significant opportunities in active network design. We described the management problems encountered in such networks, and focused in particular on replication strategies, and notification delivery.

Although most of the concepts described here have been already implemented and available as a commercial service (especially the knowledge node architecture), we are only beginning to acquire experiences from a large-scale deployment of knowledge nodes that form a knowledge network, including measurements of scalability and performance. Our future work in this area will be able to include useful experimental data and insights from the management problems we described here.

References

[ABO01] K.N. Truong, G.D. Abowd and J.A. Brotherton, “Who, What, When, Where, How: Design Issues of Capture and Access Applications”, in *Proceedings of Ubicomp 2001*, Atlanta, GA, pp. 209–224.

[ANE02a] N. Anerousis and E. Panagos, “Making Voice Knowledge Pervasive”, IEEE Pervasive Computing, Vol. 1, No 2, June 2002.

[CHA91] B. L. Chalfonte, R, S. Fish, and R. E. Kraut. “Expressive richness: A comparison of speech and text as media for revision”. In Proceedings of the Conference on Computer Human Interaction, pages 21–26. ACM, April 1991.

[DCMI] The Dublin Core Metadata Initiative, URL: http://dublincore.org/

[ENS01] C. Ensel and A. Keller, "Managing Application Service Dependencies with XML and the Resource Description Framework," in Proceedings of the 7th International IFIP/IEEE Symposium on Integrated Management (IM 2001), G. Pavlou, N. Anerousis, and A. Liotta, Eds., Seattle, Washington, USA, May 2001, IEEE Publishing.

[HAC88] Anna Hac, Xiaowei Jin, Jo-Han Soo: Algorithms for File Replication in a Distributed System. SIGMETRICS 1988: 271.

[LOM] IEEE, “Draft Standard for Learning Object Metadata”, 18 April 2001.

[RAB01] Michael Rabinovich, Oliver Spatscheck: Web Caching and Replication. Addison-Wesley 2001

[SOAP] W3C, “Simple Object Access Protocol (SOAP) 1.1”, URL: http://www.w3.org/TR/SOAP/.

[TEN96] David L. Tennenhouse: Active Networks. OSDI 1996: 89

[VER02] D. Verma, “Content Distribution Networks: an Engineering Approach”, John Wiley & Sons, 2002.

[XML] Elliotte Rusty Harold. XML: Extensible Markup Language. Structuring Complex Content for the Web. Foster City/Chicago/New York: IDG Books Worldwide, 1998.