Information Delivery with Quality of Service

Calton Pu
College of Computing
Georgia Institute of Technology
calton.pu@cc.gatech.edu

Abstract

We outline information-driven applications, particularly new applications that provide fresh information such as update monitoring and videoconferencing. Delivery of fresh information requires systems software support for quality of service. We summarize the Infosphere project, where the Infopipe abstraction is being developed and software implemented to support information flow with quality of service.

1 Introduction and Background

1.1 Information-Driven Applications

Several important emerging classes of distributed applications are inherently information-driven. Instead of occasionally dispatching remote computations, such information-driven systems tend to transfer and process streams of information continuously. Member of this class range from applications that primarily transfer information over the wires such as digital libraries, teleconferencing and video on demand, to applications that require information-intensive processing and manipulation, such as distributed multimedia, Web search and cache engines. Other applications such as electronic commerce combine heavy-duty information processing (e.g., during the discovery and shopping phase, querying a large amount of data from a variety of data sources) with occasional remote computation (e.g., buying and updating credit card accounts as well as inventory databases).

We are particularly interested in *fresh* information that changes the way we interact with our environment. For example, weather is considered an inevitably uncertain element of nature. Given fresh sensor information at suitably fine granularity (both in space and in time), accurate weather reporting and forecasting for small areas can be very valuable for everyone. For a birthday party in the park, an accurate weather forecast could mean happiness instead of disappointment. For farmers facing a sudden freeze, it may spell the difference between saving the crop or bankruptcy. For a military commander leading a rescue mission in a hostile country, it is a matter of life or death. Similarly, having an accurate picture of the highway conditions may mean arriving at work on time, beating the deadline for an important delivery, or reaching the hospital before the patient dies. It is well known that specialized agencies such as CIA and NSA have a long latency in their information collection and analysis. In contrast, CNN has shown us the value of up-to-date information about current affairs. We envision the fresh information providing more details about the current state of our physical world than ever imaginable, impartially to all human beings.

The delivery of fresh information over a network requires assured quality in the information flow. The success of guaranteed delivery services such as FedEx illustrates the importance of timely delivery in the physical world. Physical goods that are perishable usually have easily distinguishable criteria, such as sell-by dates, to separate the fresh from the decayed. Labeling goods in this way not only allows users to make informed decisions on whether or not to buy them, it also enables delivery mechanisms to make informed decisions about how to transport them. In general, many different properties are involved in such decisions. For example, the rate of decay for some produce is dependent not only on the amount of time it takes to reach the store, but also on transportation conditions such as temperature and humidity. Analogously, transportation conditions, such as constrained bandwidth, delay and noise, affect the concrete delivery properties of information flows which themselves span many dimensions including resource level concerns such as bandwidth, latency, and jitter as well as higher level concerns such as freshness, reliability, trustworthiness, security and survivability. Our challenge is to develop the systems that manage these concrete delivery properties for fresh information flows from information producers all the way to information consumers.

Complex information systems depend on a number of fundamental components working together in a critical path. For example, the Internet infrastructure and hypertext were fully developed several years before the World Wide Web standards (HTTP and HTML) were approved. Similarly, the Web was deployed a couple of years before the Mosaic browser was released. The importance of the Web browser is that it was the missing link in a long chain of breakthroughs that made the Web and the Internet finally take off. In the technology chain leading towards ubiquitous computing, we divide the concrete information flow into the producer end, the consumer end, and the missing link in the middle. On the information producer end, the network technology push and the resulting massive content generation (e.g., the Web) provide the supply of information. On the information consumer end, many dot.com and traditional companies are working on information appliances and application software that provide many different ways to access information. The missing link is the systems software that links networks to applications, including operating system, middleware, and data management layers.

1.2 Ubiquitous Information Vision

We call our vision *ubiquitous information*, in contrast to ubiquitous computing, since our focus is on the delivery of fresh information. Ubiquitous information goes beyond just gathering, storing, and retrieving increasing amounts of information. Living in a ubiquitous information environment means being in constant contact with both the physical world and the information civilization. For example, no one will ever get lost or become helpless, whether hiking in remote mountains or walking in the dark alleys of an inner city area. Help is always and immediately available through two-way low latency information flow to and from proper authorities, friends and family, or cyber-neighbors and cyber-Samaritans. Similar difficult situations arise from violent phenomena in our world such as flash floods, tidal waves, wild fires, volcanic eruptions, hurricanes, tornadoes, riots, wars, and terrorism. By providing detailed and up-to-date information about the actual situation, humans can handle these problems from a much higher vantage point. Even if we are still some years away from controlling these violent phenomena, the world will be a much more civilized place with ubiquitous information.

The Infosphere project (Section 4) is building the systems software support towards ubiquitous information, focusing on the proper treatment, transmission and delivery of information. While networking and communications researchers have been working on data transmission and delivery for many years, traditional computer science disciplines such as operating systems and programming languages have focused primarily on computation. By shifting our attention to proper information treatment and propagation, new research questions arise. Examples include delivery property management and adaptive resource management.

2 Fresh Information Application Scenarios

The deluge of Internet information shows the human limitations of browsing in finding the right information. Consequently, the discovery of fresh information is best achieved through automated monitoring. In traditional database applications, update monitoring has not included freshness or other delivery properties, since these properties are "outside the database". The only guarantee that traditional update monitoring systems provide is to deliver information updates when they reach some specified conditions of interest. In contrast, fresh information delivery in the ubiquitous information requires two levels of filtering: the conditions of interest and the freshness requirement. Conditions of interest then trigger active push information delivery mechanisms. Furthermore, fresh information delivery needs to keep track of not only the arrival of fresh information (conditions of interest) but also the decay rate of fresh information.

Application specific update monitoring is already in widespread use in many practical applications. For example, electronic trading companies such as E*TRADE send stock market updates, and airlines send weekly special ticket sale prices. Our goal is to provide this capability in a generic way to many information-driven applications such digital libraries, electronic commerce, personal guidance, and distributed multimedia systems such as videoconferencing.

An example of personal guidance systems is the Deep Map project being developed at the European Media Lab. Deep Map will guide tourists visiting Heidelberg, by integrating information from the stored geographical information systems, from fresh information updates, and real-time interactions with the user. For example, in a scenario describing a Japanese family in Heidelberg, the Deep Map leads Mr. Yakama and his kids through different tours, and then brings them back together for lunch at a dynamically chosen restaurant. This scenario requires the integration of a comprehensive understanding of the town, current knowledge of the roads, traffic, and restaurant bookings, and real-time interactions.

Videoconference systems require a variety of multimedia stream properties that can be represented as QoS properties. Examples include frame-rate, frame jitter, end-to-end latency, frame/data drop ratios and synchronization among audio and video streams. Despite many attempts, current videoconferencing software packages are still limited in performance and quality when running over shared environments such as the Internet. For example, frame-rate is a domain-specific notion of the temporal resolution of the video stream and can be used together with other information (such as spatial resolution and encoding format details) to calculate the bandwidth requirements of the stream. In general, not much support is provided by the underlying systems software when implementing a videoconferencing system.

3 Discussion of Distributed Transaction Processing

3.1 Distributed Transaction Processing

The research on distributed databases and distributed transaction processing followed closely the development of centralized transaction processing in the late 70's. However, the deployment of distributed transaction support has not been as wide spread as expected. In 3-tier client/server distributed applications, for example, typically the backend database server is a centralized database. While there may be specific reasons for the client/server model (to be discussed in the following subsection), we conjecture that perhaps there are some fundamental limitations in the concept of distributed transaction processing itself.

First, the performance problem of two-phase commit protocol is well known. The simplest and fastest of distributed agreement protocols has inherent scalability problems when the number of participants increases moderately. To put it simply, the protocol has to wait for the slowest link to respond. Since most distributed applications do not require strict atomicity, commit protocols became less used.

Second, OLTP monitors such as Encina, TopEnd, and Tuxedo (all with new product names now) have had relative success. They did not build into a large market for distributed transaction processing. Their success was due to their usefulness in bridging heterogeneous transaction processing systems.

Third, one of the most important tools in building reliable distributed applications is the reliable messaging facility such as IBM's MQ Series. Instead of forcing a synchronous agreement protocol such as two-phase commit, reliable messaging supports asynchronous distributed processing by providing reliability in information flow. This indicates the importance of system support for information flow. However, the level of programming abstraction provided by reliable messaging is limited (think TCP), even compared to RPC.

3.2 Comparison with Client/Server

Remote procedure call (RPC) is a well-established mechanism for constructing distributed systems and applications, and a considerable amount of distributed systems research has centered on it. RPC is based on the procedure call abstraction which raises the level of abstraction for distributed systems programming beyond raw message passing and naturally supports a request-response style of interaction that is common in many applications. The widespread use and acceptance of RPC has led to the development of higher-level architectural models for distributed system construction. For example, it is a cornerstone for models such as client/server, DCOM, and CORBA. The client/server model is widely considered to be a good choice for building practical distributed applications, particularly those using computation or database backend servers.

On the other hand, while these models have proven successful in the construction of many distributed systems, RPC and message passing libraries offer limited support for information-driven applications. Concretely, when information flows are subject to real-world timing constraints certain elements of distribution transparency – an often-cited advantage of RPC – can cause more problems than they solve. For example, restrictions on the available bandwidth or latency over a network link between two components of a media-streaming application are a serious concern and should not be hidden by the programming abstraction. Similarly, the reliability and security-related characteristics of a connection may be significant to applications that are streaming critical or sensitive information. We refer to these characteristics as the QoS properties of an information flow.[1]

We argue that an appropriate programming paradigm for information-driven applications should embrace information flow as a core abstraction and offer the following advantages over RPC. First, data parallelism among flows should be naturally supported. Second, the specification and preservation of QoS properties should be included. And third, the implementation should scale with the increasing size, complexity and heterogeneity of information-driven applications. We emphasize that such a new abstraction offers an alternative that complements RPC, *not* to replace it. In client/server applications, RPC is clearly the natural solution.

4 Infosphere Project Overview

The Infosphere Project [5] is a collaboration between Georgia Institute of Technology and Oregon Graduate Institute. In addition to Calton, Pu, who is the Principal Investigator (PI), we have several co-PIs at Georgia Institute of Technology (Mustaque Ahamad, Ling Liu, Karsten Schwan, Yannis Smaragdakis) and Oregon Graduate Institute (Jonathan Walpole).

We propose the *Infopipe* as an abstraction for capturing and reasoning about information flow in information-driven applications [5]. Intuitively, an Infopipe is the information dual of an RPC and a conceptual simplification of reliable messaging. Like RPCs, Infopipes raise the level of abstraction for distributed systems programming and offer certain kinds of distribution transparency. Beyond RPCs, Infopipes have attached QoS properties that allow control over the quality, consistency, reliability, security and timeliness of the information flowing through them. Furthermore, the Infopipe concept has inherent data parallelism and is concerned with a high level abstraction that embraces content semantics and user requirements in order to control information flows and optimize resource consumption. This distinction becomes particularly significant when considering QoS properties such as the quality or consistency of a flow of information.

A simple Infopipe has two ends – a consumer (input) end and a producer (output) end – and implements a unidirectional information flow from a single producer to a single consumer. The processing, buffering, and filtering of information happen in the middle of the Infopipe, between the two ends. As mentioned before, an Infopipe links information producers to consumers. The information producer exports an explicitly defined information flow, which goes to the input end of the Infopipe. After appropriate transportation, storage, and processing, the information flows through the output end to the information consumer.

[1] We use the term quality of service and QoS in a broad sense, including many systemic properties such as performance, availability, and security. This includes the initial definition of guaranteed QoS through reserved resources (e.g., in network bandwidth) as a special case.

In the same way that RPCs form the foundation of the message-oriented client/server architecture, we envision Infopipes to be the basic building blocks for information-driven distributed applications. While message-oriented middleware software such as the MQ Series are very useful and powerful, we see the need for higher level abstractions that subsume the messaging interface.

The development of Infopipe software is proceeding concurrently in three levels: kernel, middleware, and applications. At the Kernel Layer, current research is divided into two areas. The first area is the research on Real-Rate Infopipes. We have defined a "QoS-adaptive real-rate network service" which will form the basis of real-rate infopipes. Some of research challenges have been summarized in a paper [6] using environmental observation and forecasting as a concrete applic ation. The second area is the study of formal system properties when feedback is used. For example, we have been developing a feedback-based model of TCP-friendly congestion control – basically, using feedback control ideas to understand the fine-grain dynamic behavior of TCP-like congestion control so that we can produce other real-rate transmission protocols that are truly TCP-friendly [7]. In addition to simulation studies, we have modified the Linux kernel to let users adjust any TCP flow's AIMD increment and decrement parameters, allowing them to produce TCP flows with various levels of aggressiveness.

The Middleware layer work is both conceptual and practical. Conceptually, we have been describing and decomposing information-driven applications such as Operational Information Systems motivated by Delta's aircraft and flight management system [4]. Practically, we have been building efficient Infopipe support based on the ECho and JECho, a publish/subscribe communications facility developed previously. The work on the middleware level Infopipe consists of three parts. First, the definition of the middleware Infopipe model, interfaces, and a specification language for the definition of middleware Infopipes. Second, the implementation of a prototype middleware Infopipe for evaluation and validation. Third, the interactions between the middleware Infopipe with the other levels (e.g., the kernel level Infopipe below and the applic ation level Infopipe above) in terms of interfaces and integration.

At the Application Layer, we have been developing software tools for the automation of Infopipe creation and execution. The Infopipe Stub Generator, for example, creates the executable code for parsing and interpreting data flowing through the Infopipe, including the XML and PBIO (an efficient binary data interchange format) formats. Another area of research at this layer consists of the methods and software to extract and filter information automatically from the Web [1, 2, 3]. The automation of the information extraction process (usually in the form of wrapper generators) is crucial to the gradual importation of non-Infopipe data, since all information originated from the real world (e.g., sensors and the WWW) probably will require some kind of wrapping and cleansing.

In summary, we see the scalability and correctness property challenges in large-scale distributed information flows as analogous to the scalability and correctness property challenges in traditional TP systems operating on databases (information reservoirs). In response to these challenges, we anticipate the area of information flow to produce innovative and good systems in the future, the same way innovative and good TP systems have been created in response to the same challenges.

5 References

Space constraints prevent us from listing all the relevant references in the literature and on the web. We include some papers for illustrative purposes.

1. David Buttler, Ling Liu, and Calton Pu, ``A Fully Automated Object Extraction System for the World Wide Web", to appear in the *Proceedings of the 2001 International Conference on Distributed Computing Systems (ICDCS'01)*, May 2001, Phoenix, Arizona.
2. Ling Liu, Calton Pu, Karsten Schwan and Jonathan Walpole, "InfoFilter: Supporting Quality of Service for Fresh Information Delivery", *New Generation Computing Journal* (Ohmsha, Ltd. and Springer-Verlag), Special issue on Advanced Multimedia Content Processing, Vol.18, No.4, August 2000.

3. L. Liu, C. Pu, and W. Han, "An XML-Enabled Data Extraction Tool for Web Sources". To appear in the special issue on Data Extraction, Cleaning, and Reconciliation of *Information Systems: An International Journal*, Kluwer Academic, 2001.
4. Van Oleson, Karsten Schwan, Greg Eisenhower, Beth Plale, Calton Pu, and Dick Amin, "Operational Information Systems: An Example from the Airline Industry", *Proceedings of the First Workshop on Industrial Experiences with Systems Software*, San Diego, California, October 2000.
5. Calton Pu, Karsten Schwan, Jonathan Walpole, "Infosphere Project: System Support for Information Flow Applications", *ACM SIGMOD Record*, Vol. 30, No. 1, March 2001.
6. David Steere, Antonio Baptista, Dylan McNamee, Calton Pu, and Jonathan Walpole, "Research Challenges in Environmental Observation and Forecasting Systems", *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MobiCom'00)*, Boston, August 2000.
7. D. Steere, A. Goel, J. Gruenberg, D. McNamee, C. Pu, and J. Walpole, "A Feedback-Driven Proportion Allocator for Real-Rate Scheduling", *Proceedings of the Third Symposium on Operating System Design and Implementation (OSDI'99)*, New Orleans, February 1999.
8. Mark Weiser, "Some Computer Science Problems in Ubiquitous Computing," *Communications of the ACM*, July 1993.