Network Awareness in Communication Path for Efficient Support to Network Services

Rakesh Kumar Singh
Scientist-C (Information Technology)
G.B. Pant Institute of Himalayan Environment & Development
Kosi-Katarmal, Almora – 263 643, Uttarakhand, India

ABSTRACT
A promising approach to access efficient network services in dynamic network environment is to provide network awareness in communication paths. Network services across a wide area network still remains a challenging task and the difficulty mainly comes from the heterogeneous and constantly changing network environment, which usually causes undesirable user experience for network-oblivious applications. A promising approach to address this is to provide network awareness in communication paths. Many challenging problems remain, in particular: how to automatically create effective network paths whose performance is optimized for encountered network conditions; how to dynamically reconfigure such paths when network conditions change; and how to manage and distribute network resources among different paths and between different network regions. This paper describes solutions for these problems, built into a programmable network infrastructure called Switching Network Services (SNS). The SNS infrastructure provides applications with network-aware communication paths that are automatically created and dynamically modified.

Keywords
Communication Path, Data Communication, Network Services, Bandwidth, Protocol, etc.

1. INTRODUCTION
Internet has undergone a transition from simply being a data repository to one providing access to a large set of sophisticated network accessible. A typical communication path between a client application and the visited server, one can observe that the path usually involves multiple links. These links can have very different bandwidth, delay, and error characteristics, ranging from serial links to wireless to broadband to fiber link. In a network links, the nodes along the path can also have very different capabilities. Complicating service access is the fact that the load on the network resources along a communication path may change continually. When running in such heterogeneous and constantly changing environments, applications require quality guarantees in data communication for delivering satisfactory user experiences.

TCP provides applications with the abstraction of an end-to-end reliable byte stream, and it also contains mechanisms for handling flow control and a few exceptional network conditions. However, TCP does not allow application to specify how to cope with the condition when the bandwidth of an individual link drops to some level, which causes a decreased throughput at the receiving end. Such changes require very different handling between banking applications and media streaming applications. The combination of these factors: heterogeneous and dynamic changing network environment and the lack of application specific control over data communication across the network can cause poor performance or unsatisfactory user experiences for network-oblivious applications.

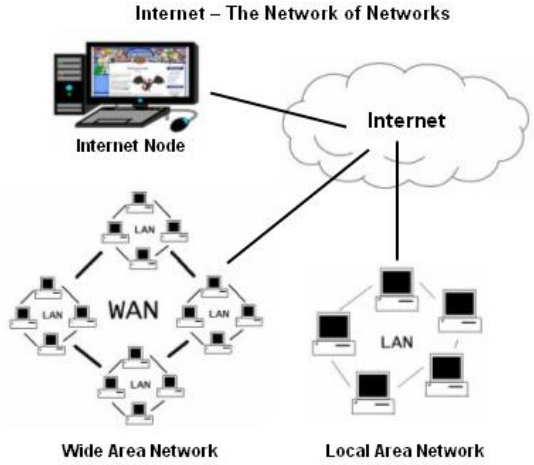

Fig.1. Network Communication Paths between Clients and Internet Services.

1.1 Data Communication and Network Awareness
Data communication in dynamic network environment should be aware of underlying network conditions, which may change dynamically. Data communication should also have the knowledge of application performance requirements, which are directly related to the way in which data is interpreted and used by the application. Combining these two together, a network-aware communication path should be able to match application performance requirements with the underlying network resource availability, and further continually adapt to dynamic changes in the network. Traditional data communication path that provides high-level abstractions such as reliable byte streams, a network-aware communication path understands application specific performance requirements and can accordingly change its behavior under different network conditions. Without the support for such network awareness, either applications themselves have to cope with the problems or the user will end up with an unsatisfactory experience.

2. OBJECTIVES OF THE RESEARCH

This research presents a path-based framework with a complete set of solutions. The main objectives of this research are:

1. To automatically create effective network communication paths whose performance is optimized for encountered network conditions?
2. To dynamically reconfigure such communication paths when network conditions change.
3. To manage and distribute network resources among different communication paths and between different network regions.

3. RESEARCH APPROACH AND GOAL

The approach has realized in a general adaptive network infrastructure that provides network-aware paths for applications whose performance is related to the quality of underlying data communication. Network-aware paths are automatically created by the underlying infrastructure, requiring only high-level input from applications. Automatically generated paths provide optimized performance to applications by customizing their behaviors to the network conditions encountered at run time. When underlying network conditions change, such paths, both globally and at the level of individual segments, can continually modify their behaviors according to the performance requirements of the application. Both path creation and reconfiguration are handled by the underlying infrastructure; therefore, regular applications can easily be augmented with network awareness without requiring onerous effort from application developers. This network infrastructure achieves the following goals:

1. It allows custom control over communication paths using various application specific components. The selection of components requires only high-level information from the application. Unlike conventional abstractions, these components understand data in transmission, thus can process it in accordance with application performance requirements.
2. It separates application business logic from what is used for creating and controlling such augmented paths. Once high-level objectives are specified by the application, the logic for creating and controlling paths is application-neutral and can be handled by the infrastructure.

4. METHODOLOGY

To find out the performance of those adaptation approaches under different network conditions, we adopt a simulation-based methodology. Using a detailed simulator modeling a typical large-scale network where multiple concurrently-active clients download media content from server sites, we characterize the performance of the three approaches: end-point, proxy-based, and path-based. We provide an overview of our simulation scenario and performance metrics of interest below, deferring a detailed description of the specific parameters. The network modeled in our simulation is depicted in the following figure. The network contains multiple ISP regions, each of which is modeled as a centralized gateway/proxy node providing a connection to the Internet backbone. The server and client nodes in the network are attached to one of these ISP nodes using various connectivity.

The simulation models users connecting to server nodes from client nodes to download and display streaming media content. The connection will be released once the download session is completed (which can happen either after the content is completely downloaded, or when the download task is cancelled by the user). To display the received content appropriately, the throughput of a download path is required to be in some specific range (i.e., a certain frame rate). When the available bandwidth is insufficient to meet the requirement, several components can be used to reduce bandwidth consumption.

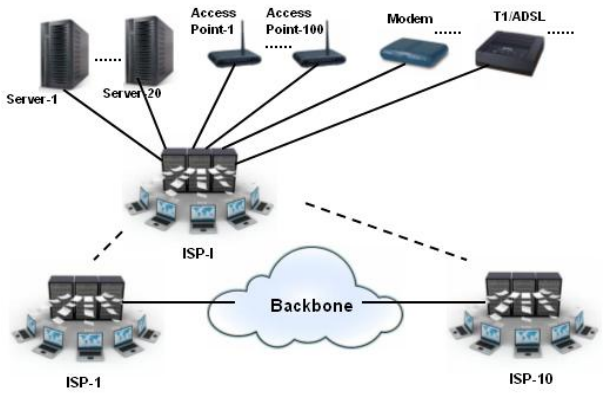

Fig.2. Experimental Network Topology

5. PROBLEM DEFINITION

The active networking emerged, which proposed general mechanisms for extending the functionality of network nodes to support execution of code embedded in network packets: in addition to passive data, a packet in an active network could contain some executable code. The network nodes (routers) in an active network are required to execute the accompanying code upon receiving an incoming packet. The code, not limited to just route packets, could perform arbitrary computation on the packets, including modification of the packet itself. An important anticipated use of active networking was for deploying new network protocols over the network. Such an approach can certainly be used to bring applications more control over the data communication in the network, but it entails significant modification of the existing infrastructure. Realizing the difficulty in modifying the existing infrastructure, overlay networks try to bring in additional functionality on top of the existing infrastructure.

A Resilient Overlay Network (RON)) is an application- layer overlay on top of the existing Internet routing substrate where each overlay node monitors the functioning and quality of the Internet paths between itself and other overlay nodes. RON can be used by distributed applications to detect and recover from path failure, often much faster than TCP/IP. Moreover, it can also improve performance of data communication, i.e. loss rate, latency, or throughput perceived by applications. Internet exhibits increasingly complex behaviors and the diversity of applications increases, the need for custom functionality in the network also grows. From the perspective of applications, this means more control over data communication for applications to obtain better performance; in other words, data

communication should be aware of network conditions and application requirements.

An augmented communication path, $P = (c_1,\ldots,c_n)$, is a sequence of type compatible components, in which c_i's output is sent to the input of c_{i+1}. A route, $R = \{N_1,N_2, \ldots ,N_p\}$, is a sequence of nodes separated by links . Each node N_i is modeled in terms of its computation capacity, $comp(N_i)$ with operations per second, and a link between two nodes, $L_i = (N_i,N_{i+1})$, is modeled in terms of its bandwidth, $B(L_i)$. Both $comp(N_i)$ and $B(L_i)$ are defined in terms of the shares of resources along the route available for a particular path. A mapping, $M : P \rightarrow R$, associates components on augmented communication path P with nodes in route R. We are only interested in mappings that satisfy the following restriction: $(M(c_i) = N_u) \Lambda (M(c_{i+1}) = N_q) \Rightarrow u \leq q$, i.e., components are mapped to nodes in path sequence order. The intuition behind this is that sending data back and forth between nodes along a route usually results in poor performance and wastes resources. Our path creation strategies exploit the type compatibility to identify valid composition patterns. The relation between types and components is depicted using a type graph G_t : a vertex in the graph represents a type, and an edge represents a component that can transform data from the source type to the sink type. The path creation problem can now be formally stated as the following: given a route R (with the resource shares allocated to the path), a type graph G_t, a source data type t_s, a destination data type t_d, select an augmented communication path P that transforms t_s to t_d and can be mapped to R so as to satisfy the following requirements: (i) Type compatibility between adjacent components; and (ii) Optimal performance. Performance can mean different things, for example, maximum throughput, minimal latency etc.

This investigation will describe solutions for above said problems, built into a programmable network infrastructure called Switching Network Services (SNS). The SNS infrastructure provides applications with network-aware communication paths that are automatically created and dynamically modified. SNS highlights four key mechanisms:

1. A high-level integrated type-based specification of components and network resources.
2. Automatic path creation strategies.
3. System support for low overhead path reconfiguration.
4. Distributed strategies for managing and allocating network resources.

We shall evaluate these mechanisms using experiments with typical applications running in the SNS infrastructure, and extensive simulation of a large-scale network topology to compare with other alternatives.

6. EXTENT OF THE SOLUTION

This research explores how to provide network-oblivious applications with network awareness in data communication using a path-based approach. These path creation strategies automatically select and map a type-compatible component sequence to underlying network resources. In addition to satisfying type requirements, the strategies respect constraints imposed by node and link characteristics and optimize some overall path metric such as response time, data quality, or throughput. We first describe a base version of the algorithm, based on dynamic programming, in which a single performance metric needs to be optimized. We then present an extension for applications that require the value of some performance metric to be in an acceptable range. For such applications, only after that range has been met does the application worry about other preferences. For example, most media streaming applications usually demand a suitable data transmission rate so that received data can be rendered appropriately at display devices; once the transmission rate is kept in that range, other factors such as data quality become the concern.

We use the terms *range metrics* and *performance metrics* to refer to the two types of preferences. Lastly, we describe a "local" scheme that can be used for a portion of the communication path. Using local planning, disjoint segments of a communication path can adjust their behaviors independently and concurrently while maintaining some overall performance guarantee. Such a local scheme can improve adaptation agility in that any portion of a communication path can be modified to respond to local changes in its network segment. More importantly, such schemes are indispensable for deploying path-based infrastructures in the situations where a communication path needs to span multiple network domains, for which fine-grained coordination across different network domains is either prohibitively expensive or infeasible due to administration policies.

7. ANALYSIS

The contributions of this research include the following:

1. A *high-level integrated specification* of components and network resources to model behaviors of both components and network resources. This specification allows late binding of components to paths, which is essential for flexibility of dynamic compositions.
2. *Automatic path creation strategies* for constructing network-aware access paths for applications. The generated network paths provide optimized performance in accordance with application performance requirements and underlying network conditions.
3. System support for *low-overhead dynamic path reconfiguration*. Path reconfiguration in our infrastructure provides semantic continuity guarantees for data transmission, and is carried out without requiring involvement from applications. Our reconfiguration strategies can be used to modify the entire communication path as well as disjoint portions of the path concurrently and independently.
4. *Distributed resource management strategies*, which can be used to manage resources among multiple paths and different network regions so as to improve performance of both individual paths and the whole network.
5. Adaptive network architecture called Switching Network Services (SNS). SNS is built from the ground up to embody our approach. A series of experiments have been conducted on SNS with different types of applications, the results validate the effectiveness of our approach.
6. Extensive performance comparison among end-point, proxy-based, and path based approaches by simulating their behaviors in a large network topology. Our simulation results will show that the path-based approach provides the best and the most stable performance under different network configurations.

8. LOGICAL VIEW OF DYNAMIC NETWORK ARCHITECTURE

This framework takes a general view that the network consists of applications, services, and communication paths connecting the two. The notion of the communication path is extended from one traditionally limited to data transmission between end points to include application-specific functionality dynamically injected by end services, applications, or the underlying infrastructure. Such functionality takes the form of components, which are self-contained pieces of code that can perform a particular activity. Components are connected with each other at run time and operate on data streams to provide network awareness in data communication by matching application requirements with physical characteristics of the underlying network and properties of end devices.

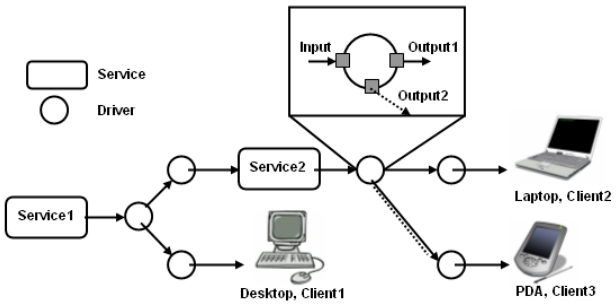

Fig.3. Logical view of a Dynamic Network Architecture.

This framework is realized in Execution Environments, an instance of which runs on all infrastructure-enabled nodes. Augmented paths are deployed to these nodes. The execution environment provides interfaces for applications to create and manage paths, and an environment for component execution, basically serving as the underlying "operating system" of our infrastructure.

9. COMMUNICATION PATHS IN DYNAMIC NETWORK

An augmented path in this framework contains functionality to process data in an application-specific fashion. Introducing such functionality into communication paths can bring application two major benefits. First, they can be used to match application requirements with the underlying network conditions. For example, compression functionality can be used for addressing the problem of low bandwidth in a network link; encryption functionality can be applied to address problems caused by network links that do not provide sufficient guarantees on data privacy and integrity. Second, by allowing computation in communication paths, functionality of an application can be extended with what exists in the network. For example, for a small device that can only display WML pages but needs to access an Internet service where only HTML format is supported, the augmented communication path can handle the conversion from HTML to WML by orchestrating functionality in the network, so that the browser running on the devices can display the contents appropriately. To construct network-aware communication paths, we need a way to orchestrate various kinds of functionality together. Instead of using a monolithic implementation, our approach adopts a much more extensible approach where communication paths are constructed by dynamically composing different components. To construct such augmented paths, only high-level information will be required, which includes services properties, application requirements, and characteristics of the underlying platform.

10. NETWORK SYSTEM SUPPORT FOR EFFICIENT PATH RECONFIGURATION

To cope with dynamic changes in the network, a network-aware communication path needs to reconfigure itself when the current configuration can no longer meet its performance requirements. Our solution of low-overhead reconfiguration has two parts: a set of simple rules placing slight restrictions on component behavior; and a reconfiguration protocol that leverages these restrictions. We observe that there are two major challenges in dynamically modifying a communication path.

First, path reconfiguration should provide semantic continuity guarantee. Since components within an augmented communication path can transform data from one type to another, the conventional notion of continuity, i.e. in-order byte level delivery, can not be applied directly to this scenario. Instead, the continuity required by applications is at the granularity of semantic segments. A semantic segment here refers to a demarcatable application-specific unit of data in transmission, e.g., an HTML page or an MPEG frame. Conventional properties such as in-order transmission and exactly-once delivery can now be defined at the granularity of semantic segments.

Second, a path reconfiguration should avoid introducing a long interruption period in data transmission. To reduce reconfiguration overhead, mechanisms that can adapt to "local" changes in the network by modifying small portions of a whole communication path are important.

11. RESOURCE DISTRIBUTION ACROSS DYNAMIC NETWORK REGIONS

This strategy is motivated by the observation that although path-based infrastructures can in general deploy operators on any network node along a communication path, usable nodes in practice are most likely a small set of strategic nodes such as ISP and gateway nodes. Besides, there usually exist some forms of administrative agreements between a higher-level network domain (e.g., the ISP) and a lower-level one (e.g., the server). Combining these two together, one can view the computation distribution problem as one of rearranging computation resources in a hierarchical network graph. Specifically, given a fixed computation resource budget, initially assumed allocated to nodes of a lower-level domain, the problem becomes one of moving a portion of the budget to nodes in a higher-level domain so that the overall performance of the whole network can be improved. The reason that such rearrangements result in better performance of the overall network is basically because of resource sharing; after such a rearrangement, overloaded servers can take advantage of shared resources at a high-level node in the network graph, contributed by servers that have a relative light load.

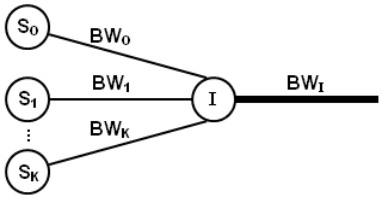

Fig.4. Hierarchical arrangement of Servers and ISP nodes

To move the maximal amount of resources to high-level network domains with the guarantee that the performance of those domains, from which the computation resources are moved out, will not be compromised after the rearrangement. This rearrangement problem is illustrated in above figure. Initially, each server s_i has a computation budget C_i (number of operations per second), and is connected to the ISP node (I) via a link with bandwidth BW_i. The ISP node in turn is connected to a higher-level network domain with a link of bandwidth BW_I. We use the terms server link and ISP link to distinguish between the two types of links. We further assume that

$$\sum_i BW_i \geq BW_I$$

The problem is to determine what portion of C_i ought to be moved from s_i to I (and what portion of the computation resources at I can be moved to the higher-level). In the description that follows, we first focus on how to distribute computation resources between these two levels. How to apply our strategy recursively within a network graph is deferred to the end of our description.

12. IMPLEMENTATION

We implement a prototype of this framework in the form of a programmable network infrastructure, called Switching Network Services (SNS). The kernel of the SNS infrastructure is the SNS Execution Environment (EE). The SNS EE serves as the runtime system for components in augmented communication paths, and provides all the infrastructural support required by these paths to realize network-aware data communication: delivering data across networks, managing resources and communication paths (i.e. path creation and reconfiguration), downloading mobile code, and providing resource availability information. A SNS network is realized by a set of SNS-Enabled nodes, each runs an instance of the SNS EE. Augmented paths are deployed on these nodes.

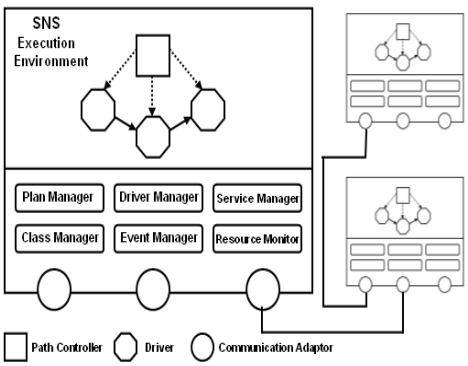

Fig.5. SNS Execution Environment

We use the Java, JVM as a programming language on windows platform. The hardware which includes the latest Intel dual Core processor with 2.8 MHz processing speed, 2 GB RAM, Two Ethernet Cards with the network speed of 10/100/1000 Mbps and other computer and networking hardware devices as switches, router, etc.

13. PROCEDURE OF PATH SETUP AND RECONFIGURATION

To set up the path, the application needs to call the Plan Manager directly or via the interception layer, with information about the server to access and its performance requirements. The next steps that follow are different depending on a centralized or distributed strategy is in use. In the centralized case, the Plan Manager first determines a network route between the server and the client application (using a shortest path algorithm). With the selected route, the plan manager constructs the component graph and the mapping using the planning algorithms. It partitions the component graph, and sends these partitions to nodes along the path. The distributed strategy works as follows: When the plan manager receives a request from the application, it routes the request towards the server. After the request arrives, the server (or the SNS node next to the server along the route) bounces back a planning request. This planning request is received by each of the nodes along the route, which calculates its portion of the communication path.

After the components graph is determined and communicated, every node along the path instantiates its components in the local EE. In addition, it also creates an instance of the path controller object for controlling this path. The path controller, running on each node along a SNS path, monitors events that reflect the performance of the path. Whenever a path controller realizes that the performance does not meet the requirements, it triggers reconfiguration using the protocol. When data transmission of a path completes, drivers and path controller of the path are removed from the EE, and any allocated resources released.

14. EXPERIMENTAL PLATFORM IN DYNAMIC NETWORK ENVIRONMENT

A typical network path between a mobile client and an Internet server as shown in following Figure has been considered in this experiment. This platform models a mobile user using a portable device (N_2) such as a laptop or a pocket PC to access an Internet service in a shared wireless environment. The communication path from the device to the visited service typically spans (at least) three hops: a wireless link (L_2) connecting the user's device to an access point, a wired link (L_1) between the wireless access point and a gateway to the general Internet, and finally a WAN link between the gateway and the host running the service. We assume that SNS components can be deployed on three sites, the mobile device (N_2), a proxy server located close to the access point (N_1), or an edge server located near the gateway (N_0). In our experiments, bandwidth on links L_1 and L_2 can change dynamically. This either results from dynamic network traffic or users joining and leaving the shared wireless network N_1. For our experiments, network configurations with different link bandwidths and computation capabilities are obtained by running SNS either

on an appropriate selected actual hardware platform, or one emulated using "sandboxing" techniques that model a range of computation capacity and link characteristics by limiting CPU consumption of applications and the rate at which applications are allowed to send and receive messages.

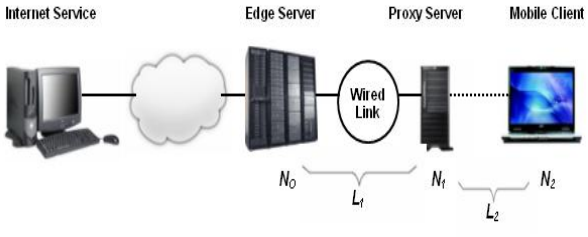

Fig.6. Network Path between a Mobile Client and an Internet Services.

The "sandbox" techniques give us the flexibility of controlling experiment parameters used in our experiments, making up for the absence of such control in current-day hardware. Additionally, the "sandbox" also provides resource availability information to SNS EEs.

15. OUTCOME OF THE RESEARCH

The main outcomes of the research are as follows:

- The end-point approach usually works well with server sites that have a large amount of computation resources and for clients that connect to the network with relatively high bandwidth links. However, servers that have limited computation capacity or clients that use weak connections may suffer from poor performance using such an approach.
- The proxy approach usually does not exhibit bias towards different types of servers or clients. The shared resource pool at proxy sites can bring better performance for small server sites or clients that have weak connectivity. However, constraining the adaptation to only occur before the last hop can cause considerable resource wastage in the network, in turn leading to early saturation as load increases.
- Support for dynamic reconfiguration is important for the performance of both individual paths and the whole network.
- The path-based approach has all the benefits of both end-point and proxy approaches. Adaptation can be conducted on upstream nodes without being limited to the node before the last hop. More importantly, the approach sets up shared resource pools across the whole network, providing the most flexibility for overloaded servers to benefit from spare computation resources elsewhere. With effective resource management strategies, this approach provides the best and the most robust performance under different network configurations.

16. REFERENCES

[1] Xiaodong Fu. *Infrastructure Support for Accessing Network Services in Dynamic Network environment*. Ph.D thesis, New York University, 2003.

[2] M. Castro, P. Druschel, A. Kermarrec, and A. Rowstron. SCRIBE: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in communications (JSAC)*, October 2002.

[3] S. D. Gribble and et al. The Ninja Architecture for Robust Internet-Scale Systems and Services. *Special Issue of IEEE Computer Networks on Pervasive Computing*, 2000.

[4] N. C. Hutchinson and L. L. Peterson. The x-Kernel: An Architecture for Implementing Network Protocols. *IEEE Transactions on Software Engineering*, 17(1):64–76, 1991.

[5] D. S. Alexander,W. A. Arbaugh, M.W. Hicks, P. Kakkar, A. D. Keromytis, J. T. Moore, C. A. Gunter, S. M. Nettles, and J. M. Smith. The switchware active network architecture. *IEEE Network Special Issue on Active and Controllable Networks*, 12(3):29– 36, 1998.

[6] B. Noble. System Support for Mobile, Adaptive Applications. *IEEE Personal Communications*, pages 44–49, February 2000.

[7] U. Varshney and R. Vetter. Emerging Mobile and Wireless Networks. *Communications of the ACM*, pages 73–81, June 2000.

[8] R. Wolski, N. T. Spring, and J. Hayes. The network weather service: a distributed resource performance forecasting service for metacomputing. *Future Generation Computer Systems*, 15(5–6):757–768, 1999.

[9] The IPSEC working group. IP security protocol (IPSec). In *Internet Draft*, April 2003.

[10] Sun Microsystems. Enterprise Java Beans(tm) specification 2.1 proposed final draft 2. Technical report, Jun 2003.