Submitted for Journal Publication

Flexible and Robust Status Dissemination Middleware for the Electric Power Grid

K. Harald Gjermundrød, Ioanna Dionysiou,
David Bakken†, Carl Hauser and Anjan Bose

Technical Report EECS-GS-003
School of Electrical Engineering and Computer Science
Washington State University
Pullman, Washington 99164-2752 USA
(kgjermun, idionysi, bakken, hauser, bose)@eecs.wsu.edu

September 25, 2003

Abstract

Electric power grids are complex interconnected systems that in North America can span more than a thousand miles. They must be operated such that the dynamic balance between supply and demand is maintained. Grids exhibit many power dynamics that are global over the entire grid, yet their control mechanisms and almost all of their operational data on the current dynamics and configuration of the grid–status data–are local in nature, typically limited to a single substation. In this paper we describe the limitations of the current communications infrastructure of the North American electric power grid and how they restrict its efficiency and its ability to respond to accidental failures and deliberate physical or cyber-attacks. We then describe status dissemination middleware, a new specialization of the publish-subscribe model that takes advantage of the semantics of status data. Next we describe the design and implementation of GridStat, a middleware framework for status dissemination across wide areas in the electric power grid and other critical infrastructures. It takes advantage of the semantics of status data to optimize delivery and to manage its subscriptions for quality of service. In doing so, GridStat allows status information, control decisions and commands to be disseminated over a wide-area.

Keywords: middleware, publish-subscribe, status dissemination, quality of service

†Contact Author

Contents

List of Figures

List of Tables

1 Introduction

This paper presents the rationale for and design of a new middleware architecture, tailored for an electric power grid, that provides services for disseminating status events in an efficient, resilient, secure and predictable manner. The electric power grids in North America are large complex interconnected systems, which operate in such a way that the dynamic balance between the demand for electricity and the amount the generators supply must be maintained at all times, under all circumstances. The dynamic nature of the power grid is subjected to regional or system-wide disturbances, and such dynamic phenomena can cause the system to become unstable. Existing automated control techniques and controllers responsible for guarding the stability of the power grid are almost all local in scope. That is, they base control decisions largely on local operational status information (for example, voltage and breaker status are delivered at most across a substation's local area network (LAN)) and they manipulate only local controls such as opening a circuit breaker.

The only automated system-wide control that exists today is the balancing of the load by adjusting generation levels (AGC) [7]. Any other form of non-local control has been achieved using dedicated communication channels between substations. The expensive hardware cost has constrained adaption of such remote control schemes to only those with very high protective value. The lack of wide-area controllers limits the stability of a power grid, because grid-wide power dynamics can only be controlled with a limited number of mechanisms on a large timescale, such as controlling generator output on an hourly basis.

These wide area controls cannot utilize the much richer set of controls available at the substation level because there is presently no mechanism to deliver information on operational status and control decisions to the appropriate parties. This limits how much power can be carried on a power grid because, in order to meet mandated stability criteria, the grid must be operated further away from the inherent thermal limits of its transmission network. This, plus the growing demand for electricity, creates an enormous demand for more transmission capacity at a time when few grids in the US seem willing to add transmission lines, due in part to the large expense and local opposition. This lack of adequate global communication and control also limits the ability of the grid to withstand both accidental failures, such as the 2003 blackout of the US's eastern grid or the 1996 blackout of the US's western grid, and malicious failures, such as a cyber-attack on the grid by terrorists, which is considered feasible by experts [17].

In addition to the operational issues, recent deregulation of the electric power grid mandates delivering status information about operational and market conditions to legitimate market participants, as an effort to promote competition within the electric market. Exchange of real-time data may take place between existing grid participants (control centers, power plants, transmission substations, distribution substations, residential customers, industrial customers, commercial customers) for carrying out operational tasks as well as between new ones, such as traders, for marketing tasks. Examples of this data include Supervisory Control and Data Acquisition (SCADA) status, negotiation of generation and transmission of schedules, available transfer capacity, forecast reports, and meter readings. Figure 1 illustrates current and proposed interactions between the various grid entities.

Data availability is critical for wide-area control; being able to check and approve transactions (such as the contracted output of a given generator) more often than current practise (hourly) can make more efficient use of the power supply. The current communication infrastructure of electric

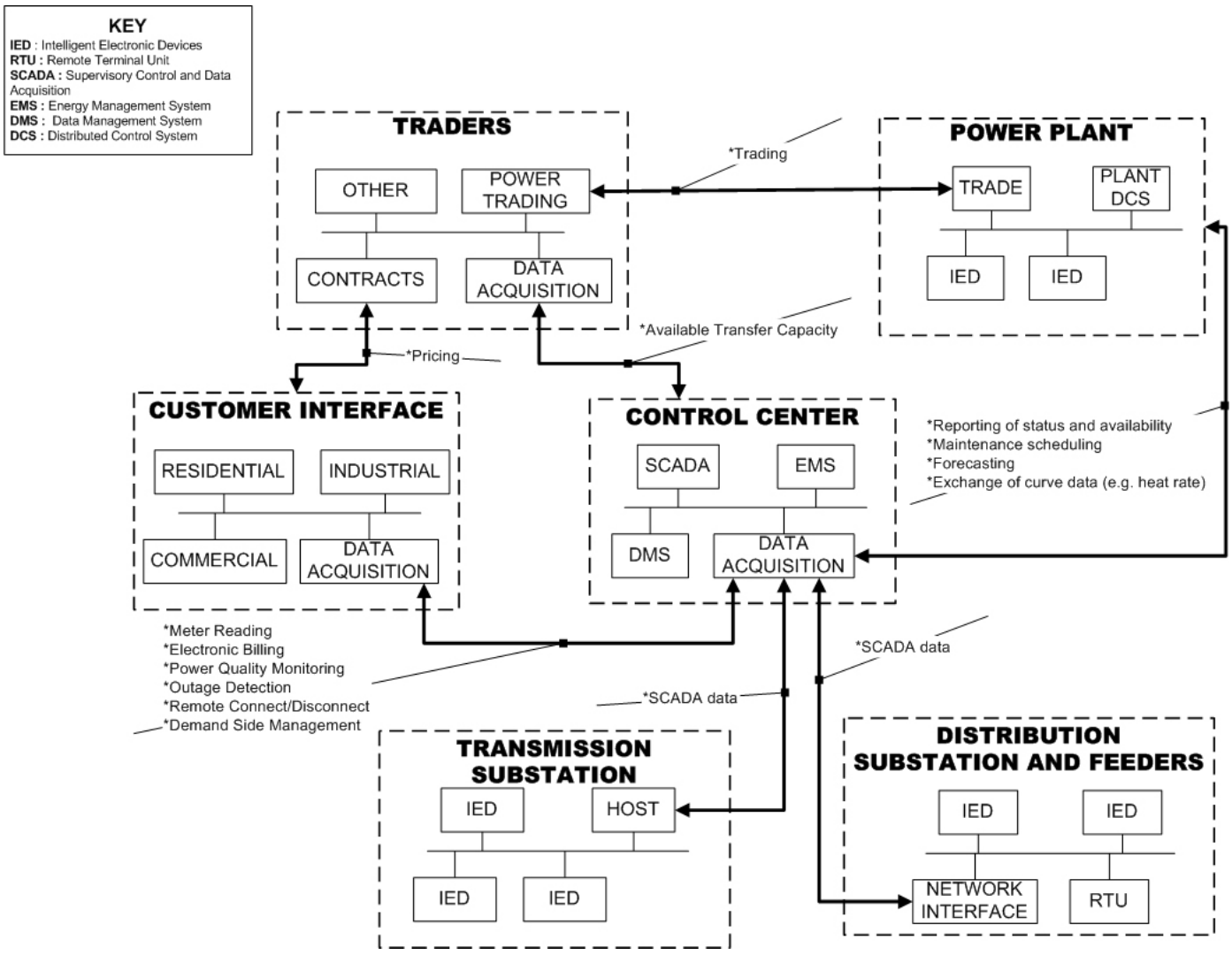

Figure 1: Deregulated Electric Power Market

power grids is not capable of disseminating *status information*, operational and market data, with the necessary flexibility, robustness and timeliness. A new communication infrastructure is thus needed that will meet these requirements.

There are a number of research efforts within the computer science and power communication communities that are attempting to address the need to disseminate information to multiple independent entities. However, none takes advantage of the specific characteristics of the status information to provide optimized delivery and management to better meet the needs of the power grids. In addition, they do not provide the necessary flexibility, timeliness, security and fault tolerance or, as they are collectively known, quality of service (QoS) guarantees [24][8].

In this paper we present a new middleware architecture supporting QoS, tailored for the electric power gird. The contributions of this paper are:

- An analysis of why an electric power grid needs more flexible, robust and timely delivery of status information than its current communications system provide (or could conceivably be extended to provide).
- The design and implementation of GridStat, a status dissemination middleware framework tailored to the power grid (but also applicable for other critical infrastructures).
- A preliminary experimental evaluation of a GridStat prototype.

The remainder of this document is organized as follows: Section 2 explains in more detail the limitations in power grid operations due to the lack of a wide area communication infrastructure. Section 3 provides an overview of status dissemination middleware and the architecture of GridStat, a status dissemination middleware framework. The status dissemination mechanisms implemented in GridStat are presented in Section 4. Section 5 presents a preliminary experimental evaluation of the framework's performance. Current implementation status is given in Section 6. Future work is outlined in Section 7. Section 8 describes related computer science research, and Section 9 concludes.

2 Rationale for Wide-Area Status Dissemination Service for the Electric Power Grid

Maintaining frequency of 50 Hz or 60 Hz throughout an entire power grid is a critical task. Frequency is monitored at two levels: at the local level, i.e. at each generator, and at the system level by the central controller. The latter control is known as Automatic Generation Control (AGC). AGC is the only automatic feedback control function in power grids today that operates system-wide.

However, the power grid exhibits various other dynamic characteristics, mostly controlled locally, that affect regional or even system-wide performance [7]. Voltage control, for example, is performed locally, but changes at one generator can affect neighboring systems. Currently, there are hardwired dedicated communication connections between substations and central controllers to relay some status information, but little communications capability between substations, mainly due to the cost of implementing such networks. In short, the communications architecture of current power grids is inadequate to support optimal use of available transmission capacity.

There are several benefits that would be associated with a richer set of wide-area controls. One of them is to help increase the stability limits of transmission networks. Transmission is constrained by the thermal limit of a transmission line and the stability limit, imposed to ensure that the system will continue to operate following failure of a major resource (for example, a generator or a transmission line). Additionally, generators in a geographic region in the US (and some other places) are no longer all owned by the same organization, so the wide-area AGC has become more decentralized; given inadequate wide-area communications, it thus becomes more difficult to manage. Furthermore, the increased use of microprocessors within the substations results in gathering data at very fast rate, one that the present control systems are unable to utilize due to the lack of the necessary communications and wide-area control systems. Some of this newly-added information would be useful to other entities on a regular basis, while even more of it would be beneficial on demand, when a control center is "drilling down" to ascertain the cause and potential remedies for a power anomaly.

Until recently, in the US, the entire operations of a power grid in each geographical area was controlled by a vertically integrated utility. This company controlled all 3 fundamental roles – generation, transmission and distribution – over an area that would typically cover thousands of square miles. As part of the electric power market deregulation, the Federal Energy Regulatory Commission (FERC) [11] mandates all public utilities to interface with other entities by posting their Available Transfer Capacity (ATC) information for their commercially active transmission paths. Envisioned benefits of deregulation include allowing industrial and individual consumers to choose their electricity provider, open bidding by utility companies for extra energy in case of shortage, and the ability for producers and consumers to enter into short and long term contracts. A true competitive market, if implemented correctly, has the potential to allow for more effective load management and more effective real-time pricing. However, there are increasing concerns within the power industry as to whether a completely open electricity market will become a reality or not [16], mainly due to the impact that it would have on the stability and security of the grid itself[1]. Universally accessible data, if not properly secured, may result in compromising consumer privacy, extracting individual consumption production patterns and destabilizing the grid by malicious acts [10].

The current communications infrastructure for the electric power grid consists of a fixed, hard-wired and sequential data acquisition infrastructure, which is not able to meet the trends of the power industry. What is needed is a communications system responsible for distributing status information to legitimate parties in a timely, secure and accurate manner. As described above, an information infrastructure for gathering and disseminating electric power grid status information is essential for both the deregulated utility communication requirements (such as available transfer capacity, rate schedules, etc.) [2], and the development and deployment of wide-area controllers to receive non-local status information. Other applications can also benefit from the communication dissemination infrastructure; for example, disseminating status information to non-local sites would help overcoming one of the barriers to a wide-area early warning system for electric power disturbances [23].

Recent advances and trends in network communications make such an information infrastruc-

[1]We note that when power engineers use the term "security" they mean approximately what computer scientists call "stability". However, when computer scientists use the term, they mean protection from hackers, denial-of-service attacks, etc. To avoid confusion, we will be more explicit, using the term "grid security" when discussing power dynamics and "network security" or "computer security" for the computer science meaning of "security".

ture feasible. Bandwidth availability, lower hardware costs, increased computer speed, lower latencies, middleware technologies, as well as the ability to embed QoS properties at all levels can be appropriately utilized in developing a new communication paradigm.

2.1 Motivating Examples

There are several examples that illustrate the importance of having a communication infrastructure delivering status information to power participants. The particular information depends on the specifics of the power application, whether it deals with operational tasks or deregulated procedures.

EPRI, the Electric Power Research Institute, conducted a number of studies in California regarding customer response to electricity prices. According to the reports, if customers know the real price of electricity, they will adjust the electricity consumption when the price is high. This adjustment leads into a reduction of electricity usage at daily peak times, when a grid is most vulnerable to disturbances. Furthermore, EPRI tried out a pilot for a fully automated system at the World Financial Center in New York, where real time pricing was received by the energy management system (EMS) of the building. Decisions on where usage could be changed, given temperature readings and humidity levels from building sensors, were made. A commercial building, therefore, can benefit from real time pricing information [13].

As far as operational tasks are concerned, there are numerous advantages of receiving data from non-local systems. One example is the development of new controllers that can take advantage of remote wide-area inputs to guard against small signal instability. Currently, local controllers are responsible for recognizing when a system perturbation excites a natural oscillatory mode of the power system[7]. Another example is the possibility of decentralizing the control center so that the functions are placed in different locations, depending on where they are needed. Techniques like the one just mentioned are crucial to enable electric power grids to adapt to changing conditions, especially in the face of disasters, failures and malicious attacks.

Clark Gellings, EPRI Vice President for Power Delivery and Markets, when elaborating EPRI's vision of the future power system explained that *"the ultimate challenge in creating the power delivery system of the* 21^{st} *century is in the development of a communications infrastructure that allows for universal connectivity"* [13]. Gellings emphasized the fact that power flow needs to be secure, reliable, available and of high quality. In order to create this new power delivery system, generators, transmission, substations, consumers, distribution and delivery must be connected via a grid of nation-wide communications system enabling the exchange of time-stamped information required for computational analysis. The ultimate goal is collecting data from all the nodes across the country performing computation on them and delivering a result in a matter of seconds, not minutes [13].

2.2 Problems in Current Electric Power Communication Infrastructure

The current communication infrastructure of the power grid is not extensible in a practical way to accommodate all the new participants that have to interact with each other in order to accomplish their tasks. Its hardwired nature does not allow flexibility and that is an obstacle holding back the implementation of wide-area controllers and deregulation of the power grid. Substations are

connected to a control center and occasionally to other nearby substations using dedicated communication lines to deliver a hardwired set of status data. This technique is not widely employed due to the costs involved, and due to the use of low-level and proprietary mechanisms for the delivery of data, which limit how widely the data could usefully be disseminated in the presence of diverse hardware and software.

A new communication infrastructure that supports delivery guarantees for data is required so that the power grid will be more stable and better utilized, as well as more competitive.

2.3 On-Going Efforts From Power Community

In the last few years, the power industry has taken steps towards the development of communication networks that accommodate interactions between different parties within the electric power industry. These efforts are discussed below. None are adequate for the needs of electric power grids that we have described; in particular, they do not provide flexibility, interoperability, QoS or management infrastructure. They also do not take advantage of computer science advances over the last decade in areas such as middleware. Note that discussion of related computer science research in middleware frameworks and fault-tolerant messaging is presented later in Section 8.

2.3.1 Utility Communications Architecture (UCA)

Currently there are 152 different communications protocols for sending data within the electric utility industry and 28 different communications protocols in consumer devices such as appliances and thermostats supplied by different vendors [13]. The goal of integrating utility operations and interactions into a single framework has been pursued by power researchers during the last few years.

EPRI, the Electric Power Research Institute, is a non-profit energy research consortium that provides technological solutions for the energy industry. It currently serves more that 700 energy-related organizations, which fund EPRI's research projects related to energy and power. EPRI has created the Utility Communications Architecture (UCA)[12], which is part of its Integrated Utility Communications program. UCA offers interconnectivity between devices from different manufacturers and interoperability between databases primarily used to exchange real-time data. Utility hardware and software from control centers, transmission systems, distribution systems are interconnected using common language, semantics and standards. We now overview both UCA and its shortcomings.

Figure 2 illustrates the UCA suite of existing standards. UCA v.2, the latest version, supports the communication needs of EMS, SCADA systems, Remote Terminal Units (RTUs), Intelligent Electronic Devices (IEDs), Substation Automation, Power Plants and energy services to customer sites.

One of the requirements identified for UCA was high speed device to multi-device communication of binary state information within a LAN [1]. The Generic Object Oriented Substation Event (GOOSE) model is a combination of a "modified" publish-subscribe model and multicasting mechanism that delivers a collection of binary states of a device. A sending device publishes the user-defined state bits in the device. Any device interested in any of the publishing device's states subscribes to the publishing device's GOOSE message.

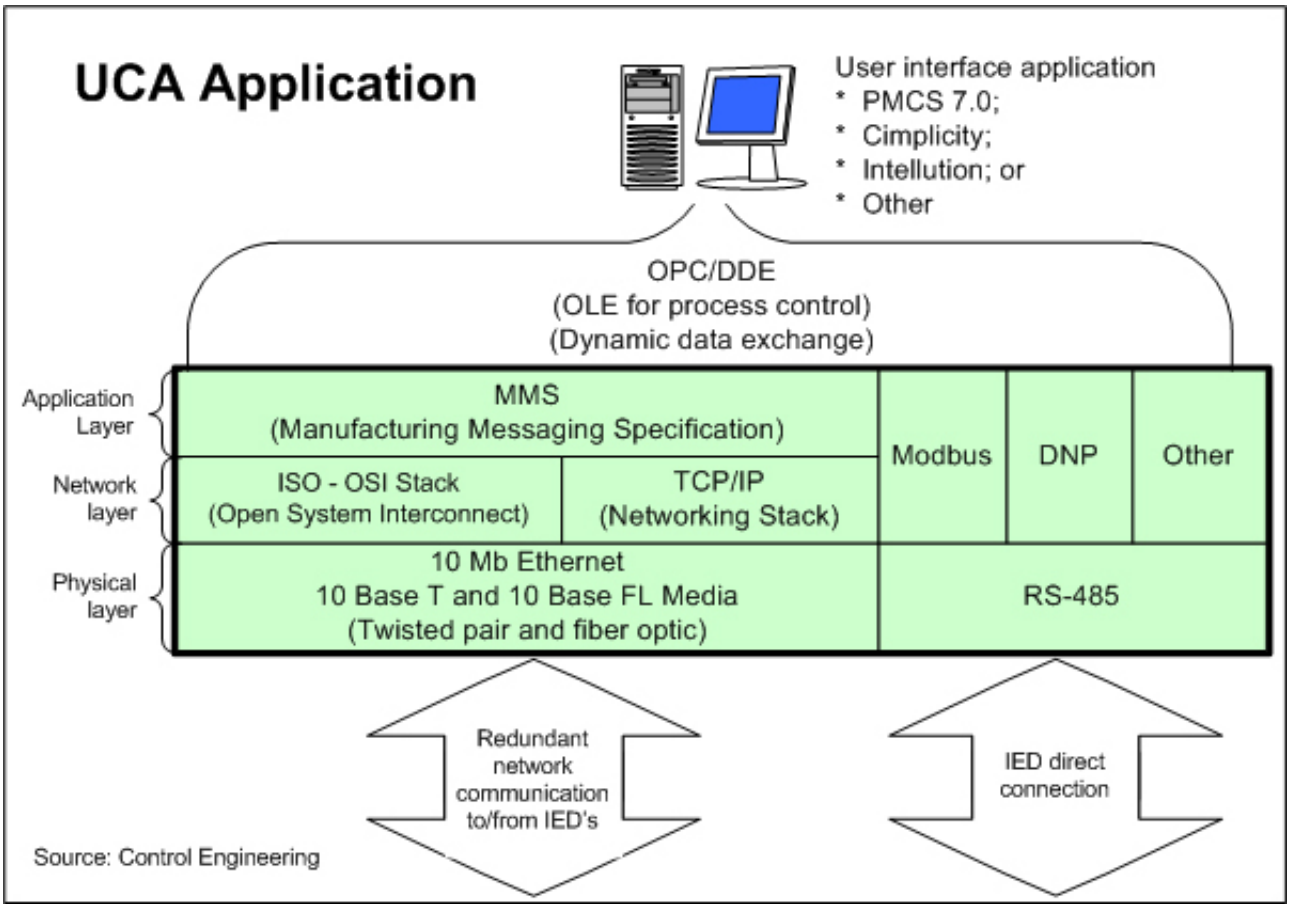

Figure 2: UCA Application

The ultimate vision is to make UCA a utility-wide standard and to seek further alliances for integrating gas and electric utilities. It has already been adopted by the International Electrotechnical Commission (IEC) to become the International Standard for Interoperability for utility communications.

As mentioned earlier, the main goal of the UCA suite of protocols was to facilitate communication between devices within a single substation (such as sensors and diagnostic equipment for transformers, switchgear, relays, recorders) regardless of manufacturer. Therefore, it is not designed to support delivery guarantees within a wide-area network. Even though UCA provides for sending and receiving commands to control the various devices, it does not take advantage of the semantics of the data for further optimizations, which can save network bandwidth and make a communications infrastructure more affordable and robust. It also lacks management capabilities for maintaining a complex network of interconnected systems that require adaptive QoS guarantees to respond not only to power anomalies but also to failures, overloading, and cyber-attacks involving the communication infrastructure itself.

2.3.2 EPRI ISI

EPRI is also involved in a two year effort (2002-2004) to evaluate the vulnerabilities of the electric power infrastructure and at the same time develop strategies for protecting and recovering from terrorists attacks. The Infrastructure Security Initiative (ISI) focuses, among other areas, on how to develop a secure private communication network (Secure Communications Project) for the electric power industry, as an alternative to Internet-based systems [14]. The secure communications project aims to determine how to use the security protocols that web-based applications are currently employing for protecting the SCADA systems and other utility systems.

It is important to note that power researchers are looking at establishing a private network rather than relying on the internet infrastructure [3]. One future plan is to evaluate “hybrid” systems that use the Internet for communicating publicly available data and a secure extranet system for communicating sensitive information. The solution will be built on e-commerce existing technologies, like the public key infrastructure (PKI) solutions, and the UCA.

ISI’s secure communications project concentrates on how to secure data flow given the diversity of security levels required for different utility applications. It is not a deployable communication framework that will provide management capabilities in order to guarantee QoS requirements.

2.3.3 NERCnet

NERC, the North American Electric Reliability Council, is a non-profit voluntary organization that ensures that the electric system in the US is reliable, secure and adequate. A comprehensive energy bill passed by the US House of Representative enables NERC to become the electric reliability organization with the authority to enforce compliance with reliability standards among all power participants.

NERCnet is a communication infrastructure developed by NERC that facilitates data communication between 17 NERC Inter-regional Security Network (ISN) nodes. Each control area is responsible for supplying real-time data to an ISN node. There is little or no information publicly available on the design and internal workings of NERCnet, but it seems to be only a private communication network with no status services, middleware, or management infrastructure.

2.4 Status Dissemination and Cross-Domain Hybrid Attacks on Critical Infrastructures

Timely, secure, and robust status dissemination that is highly scalable and managed is clearly needed over large geographic areas for electric power grids in order to better instrument and control wide-area power dynamics, as we have shown. However, this need is not limited to power dynamics.

Other critical infrastructure domains such as transportation and gas pipelines have very similar requirements for status information delivery. All have a need to instrument the operational status in their wide-area domains and to disseminate control decisions.

Further, the communication infrastructure itself, in each domain, needs status information on its own operation, to allow its management infrastructure to better respond to accidental failures and anomalies that may indicate an ongoing cyber-attack. Such communications status information can easily be delivered by status dissemination mechanisms that meet the electric power grid's requirements for status dissemination that are outlined above.

Indeed, what is required is status information gathered from both the information infrastructure and the physical dynamics of all critical infrastructures. It is considered by experts to be highly likely that a sophisticated terrorist group, such as Al Queda, that does meticulous planning would launch coordinated attacks across infrastructures. Ronald Dick, Director of the FBI's National Infrastructure Protection Center, noted "The event I fear most is a physical attack [on the electric power grid] in conjunction with a successful cyber-attack on the responders' 911 system or on the power grid" [17]. The availability of status information across domains would give a national response headquarters, such as the one that Mr. Dick directs, the situational awareness to detect and respond to sophisticated attacks. Note that this will always involve humans in the loop because the state of art does not allow for anywhere near 100% automation without a large number of false positives. However, in order to monitor potential anomalies identified by remote monitoring, the humans in the loop need status information on a regular basis as well as the ability to track new status variables when investigating a current anomaly.

2.5 Summary

The lack of an information communication infrastructure is one of the main obstacles that interfere with the practical deployment of both wide-area controllers and deregulation policies. The current communications infrastructure lacks both management capability and efficient data delivery in a secure manner.

It has been previously observed that there is a need for a better communication infrastructure for the grid [5]. Status dissemination middleware is a promising potential solution to the real-time exchange of data in a timely, robust and resilient manner. Furthermore, to ensure grid security, the status dissemination service must provide the computer and network security features such as message authentication, data integrity and confidentiality.

3 GridStat: A Status Dissemination Middleware Framework

This section first describes the concepts of status dissemination middleware (SDM). The architecture of an example SDM framework tailored to the power grid, namely GridStat, is then presented.

3.1 A Definition of Status Dissemination

Status dissemination begins with *status variables*: periodic measurements of physical quantities or control settings. Each measurement is associated with a timestamp[2]. Thus, a status variable is a periodic sequence of timestamped values. *Status dissemination* is both:

- the reliable delivery of subsequences of status variables to multiple locations, each with its own demands for delay and rate, and
- reliable delivery of aperiodic *alert* messages concerning a status variable when its value meets some particular condition, such as a threshold.

A system for status dissemination must respect the requirements of status variables' owners for which parties are allowed to see the data (data confidentiality).

3.2 Status Dissemination Middleware Concepts

A possible solution to the wide area status dissemination problem is to implement the functionality described in 3.1 in middleware. Middleware is a class of software technologies that provide common abstractions across a distributed system [4]. It lies beneath the application and above the operating system. Middleware provides a set of services that the application layer programmer can use and reuse to build distributed applications [4]. One of the design goals of middleware is to solve problems caused by heterogeneity in platforms, networks, programming languages, operating systems and vendor implementations. This is particularly important in any critical infrastructure such as the electric power grid, given the large diversity of hardware, software, and vendors thereof.

Message-Oriented Middleware (MOM) is a type of middleware that uses message exchange for distributing data. MOM provides the abstraction of a message queue that is accessible across a network. Message queueing is an indirect communication model. Messages are pulled by consumers based on the queue ordering, without direct interactions with message producers. One specialization of the message-oriented middleware framework is the publish-subscribe paradigm. The producers and consumers interact with each other through a network of intermediate servers. A powerful feature of the publish-subscribe paradigm is that it is decoupled in space, time and flow with respect to the end entities [15]. The interacting entities, publishers and subscribers, do not need to know each other (space decoupling), or actively participate in the interaction at the same time (time decoupling). Moreover, neither entity is blocked while producing or consuming events (flow decoupling).

Status dissemination middleware (SDM) is a further specialization of the publish-subscribe paradigm. A key difference between a general publish-subscribe framework and one that is specialized for status variables is that status dissemination middleware carries a strong implication

[2]This paper does not address the clock synchronization requirements for status dissemination. They can easily be met using known technology and techniques such as GPS-based clocks.

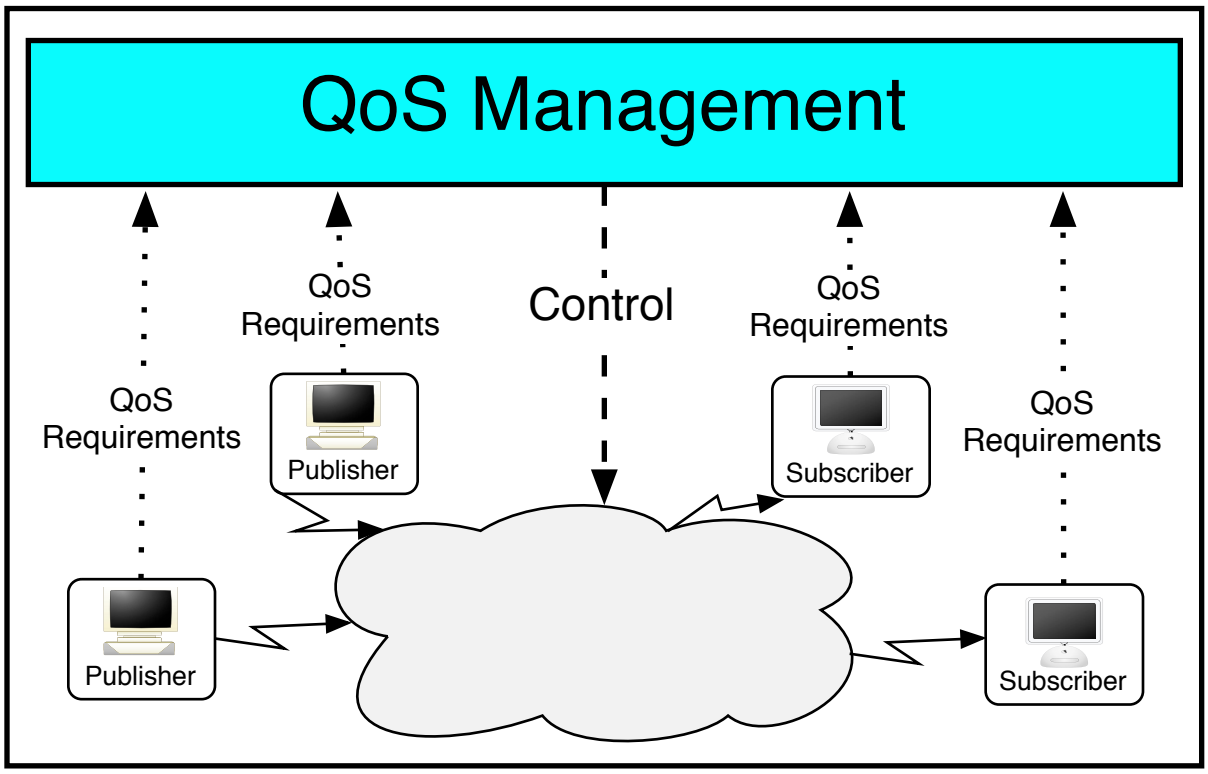

Figure 3: Basic Functionality of Status Dissemination Middleware

of real-time behavior, both on the part of the publishers, which produce at a known rate, and on the part of the middleware which must meet the real-time requirements of the subscribers. Other differences derive from the fact that each status variable is updated with a sequence of related measurements, which may be combined or filtered to meet subscribers' needs and save substantial bandwidth in doing so. In contrast, with publish-subscribe frameworks, it is not possible to filter messages in a given flow because the impact on the application is unknown. SDM captures extra semantics and QoS requirements to make this filtering possible.

A status dissemination middleware framework is designed to take advantage of the semantics of status variables subject to a number of QoS requirements. Figure 3 illustrates the basic functionality of status dissemination middleware. End entities can either take the role of a publisher, subscriber or both. The publisher and the subscriber interaction is handled by an underlying communication infrastructure that acts as a transparent forwarding agent for disseminating status events to the interested subscribers.

With appropriately-designed SDM supporting QoS, both publishers and subscribers can be fairly easy to developed by programmers who are not experts in publish-subscribe systems or QoS. This is of practical importance, because such experts are relatively rare, while many power engineers have the power expertise needed to specify QoS parameters for status subscriptions.

3.3 GridStat Architecture

GridStat is a flexible and manageable status dissemination middleware framework implementing the concepts described in Sections 3.1 and 3.2. GridStat delivers end-to-end QoS properties, such as timeliness, network security and reliability. QoS management controls all communication paths so that QoS requirements are met, while at the same time multicast mechanisms allow the QoS management to better utilize network resources. An overview of GridStat architecture appears in Figure 4.

GridStat provides a publish-subscribe model that allows publishers to post information without knowing by whom or where it will be consumed. Subscribers are able to subscribe to status variables at a given rate and with the required level of redundant delivery paths. For example, a publishing power relay that makes measurements of voltage on a monitored line will post measurement values for subscribing devices to use. Subscribers are told immediately at subscription time if the subscription requirements cannot be met, and are notified at runtime if they are later violated.

The communication infrastructure consists of a network of status routers, which delivers status events to the interested entities. These status routers are analogous to IP routers, but provide middleware level routing, aggregation, rate filtering, and other QoS management mechanisms. A hierarchy of QoS brokers manage the communication resources so that the QoS requirements of publishers and subscribers are satisfied. GridStat manages QoS in the reliability, timeliness, and network security domains as required by grid monitor and control applications.

Typically, the communication infrastructure of publish-subscribe systems is organized as a peer-to-peer network, a ring network, or a hierarchal network. GridStat takes a different approach to organizing the infrastructure for reasons of scalability and local autonomy. The status routers are organized into a set of peer-to-peer networks called *clouds*, which are managed by a broker hierarchy. In some respects, this is similar to the organization of the internet as autonomous systems, but management and data flow are separated in GridStat. The data plane consists of clouds of status routers, while the management plane consists of the hierarchy of QoS brokers.

The advantage of separating application data and management is that it allows peer-to-peer data flow while avoiding bottlenecks associated with routers organized as a hierarchy. Additionally, hierarchical management simplifies scalability and meets end-to-end QoS requirements of status dissemination, while accommodating requirements for subsystem autonomy.

For example, when a path from a status router in one cloud to a status router in another cloud is to be allocated, the management first finds a path between the clouds in the graph formed by considering each cloud as a node. Then the management plane of each cloud in the path is instructed to find a path internally using the status routers within the cloud. The administration of the system can take advantage of the hierarchy to apply divide-and-conquer techniques for the overall management of the entire system. Furthermore, different management domains with different policies can cooperate in a structured manner by using the hierarchy; for instance, an upper-level broker can override the policy of its lower-level brokers (children) since it has a broader view of the system. The separation of the data flow and administration also improves the resilience of the system. If the brokers fail or are attacked, the flow of status updates will not be affected. Similarly, if the data paths are failing, the management will not be affected, allowing diagnostic tools and recovery mechanisms to be employed to discover the failure and recover from it.

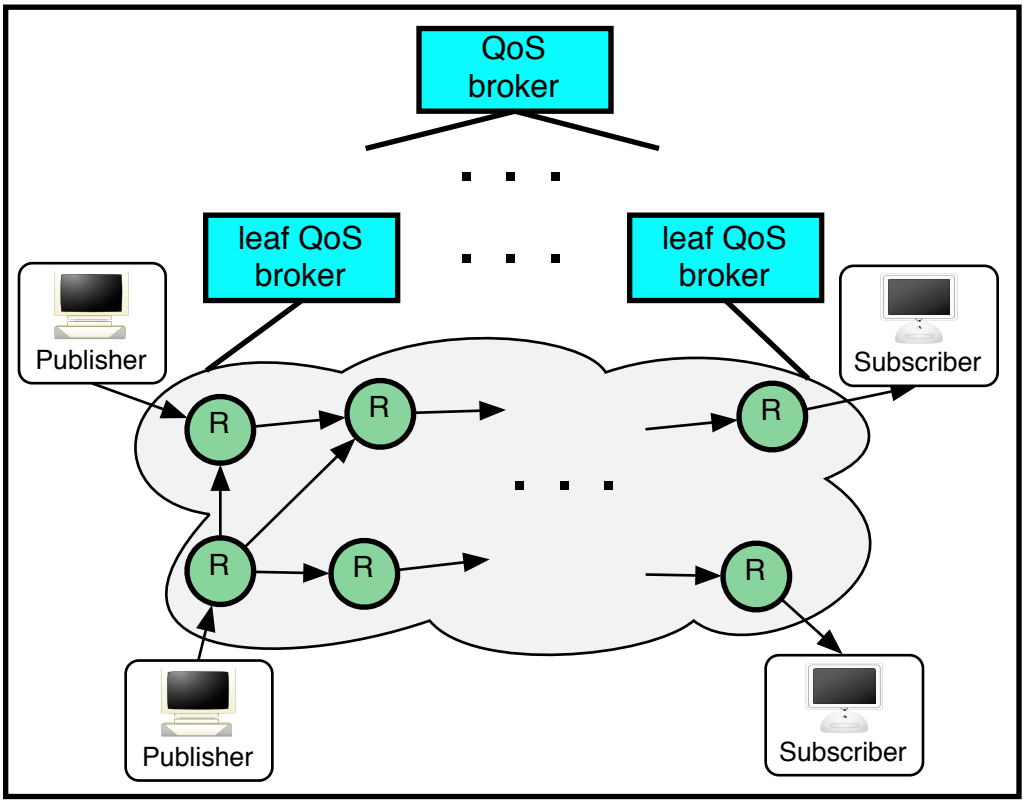

Figure 4: Overview of GridStat Architecture

3.4 GridStat Entities

GridStat relies on an underlying communication network, which can be a combination of dedicated network links and an open network (e.g., the Internet). Over paths consisting of dedicated links, GridStat management can guarantee that QoS requirements are met. In the case of an open network, GridStat status routers act as an overlay network, where it is the responsibility of the deploying organization to decide how closely it wants to map the GridStat network to the physical network. QoS properties can also be satisfied in a non-dedicated network if reservation mechanisms are provided by the underlying network. Like all middleware, GridStat masks heterogeneity in network protocols and CPU architectures.

The main passive GridStat entities are status variables and QoS policies. The main active entities are publishers, subscribers, status routers, and QoS brokers. Below, each of the entities in GridStat is explained in more detail:

A **status variable** represents some device or phenomenon that has a state that changes over time. The sequence of states of a status variable is captured in status events, which are periodically published for others to use. Status events are encapsulated into status messages that flow through the GridStat communication infrastructure from publishers to subscribers. Every status variable has static attributes such as publisher name, status name, and rate. In addition to these, a status variable has dynamic attributes consisting of its current value and derived attributes such as its moving average, rate of change, moving average of the rate of change, and minimum and maximum values over a time interval. Subscribers can subscribe to a derived attribute just as they can subscribe to a variable's value. Sometimes, subscribers consume a large set of status variables in order to calculate a derived status variable. This task can be more efficiently accomplished by user-defined *condensation functions* loaded into GridStat status routers by the QoS brokers (explained below).

Publishers are the sources of events in the publish-subscribe model. Publishers connect to the communication infrastructure through *edge status routers*. A publisher declares to its edge status router each status variable that it will publish along with its QoS properties (e.g. publishing rate). After declaring a status variable, the publisher starts publishing status events with the QoS properties that it registered. In the electric power grid, a protective relay that periodically publishes its measurements of voltage, current, and power on a monitored line is an example of a publisher.

Subscribers are the destinations of events in the publish-subscribe model. They connect to the communication infrastructure through edge status routers. A subscriber announces to its edge status router the status variables that it will consume along with its QoS requirements for these variables. The subscriber gives a reference to an object into which status event values are to be placed. In this way, subscribers have a local cache of the latest value that can be accessed when needed. It is also possible for the subscriber to register that it wants a callback if any of the QoS requirements are violated.

Status routers make up the communication infrastructure. The status routers are connected to each other through point-to-point or overlay links. They behave as smart routers, receiving and forwarding status messages. A group of status routers in one administrative domain forms a *cloud*, which is controlled by a leaf QoS broker. Different clouds are connected with links between status routers in the two clouds. Status routers acting as the connection gateways between different clouds are called *border status routers*. *Edge status routers* allow publishers and subscribers to directly connect to them, thus providing the access points to the communication infrastructure.

Table 1: QoS properties and policies

QoS Property	Policy
Delivery Guarantee	best-effort, at-most-once, at-least-once, exactly-once
Message Priority	FIFO, EDF, priority
Overflow Control	ANY, FIFO, LIFO, or message priority

QoS policies are enforced by QoS brokers. The policies affect admission control, security issues, fault tolerant strategies, etc. For example, when subscribers request subscriptions with specific QoS properties, the QoS brokers apply QoS policies to verify whether the requests can be fulfilled or not.

Leaf QoS brokers manage a domain comprising a set of status routers, publishers and subscribers. Each leaf QoS broker has information about the status routers' topology that it manages as well as information regarding the current network state (e.g link delays, link bandwidth, message processing time at the status routers, number of messages that can be processed within any interval by the different status routers, size of status events) of its domain. When a status router comes online and contacts its leaf QoS broker, the latter sends commands to the routers to set up links to the neighboring status routers. A leaf QoS broker allocates and de-allocates subscription paths subject to the QoS requirements of the publishers and subscribers within the cloud it controls. If there are conflicts in the QoS requirements or there is a conflict between the QoS requirements and the cloud's capability, the leaf QoS broker must resolve this conflict according to its QoS policy. For example, if the system is currently close to its limit, the leaf QoS broker could deny a request for subscription path that will consume a lot of resources. As an alternative, in the case that the resources are scarce, the hierarchical management, instead of denying subscription, may change the QoS requirements of existing subscriptions to ensure stable operation of the overall system. This adaptation will be done according to its QoS policies.

QoS brokers form a hierarchy of QoS management, with leaf QoS brokers at the lowest level. The upper-level QoS brokers manage multiple clouds and are responsible for allocating and de-allocating inter-cloud subscriptions.

The entire GridStat architecture is illustrated in Figure 5.

3.5 QoS Properties for Status Dissemination

The QoS properties addressed by existing publish-subscribe systems mostly concern message ordering and reliability of message delivery. Examples of such properties and associated policies appear in Table 1.

As described in Section 3.1, reliability has a different interpretation within the status dissemination middleware. Timely and reliable delivery implies that data are guaranteed to reach their destination even in the presence of failures.

One of the essential features of the GridStat architecture is the ability to provide end-to-end QoS properties such as fault tolerance, delivery rate, priority, redundancy and network security for a publish-subscribe middleware framework in general and status variables in particular. GridStat can provide them because the QoS brokers have information about the network topology and network state of the domain that they administer. Using this information, path allocation algorithms

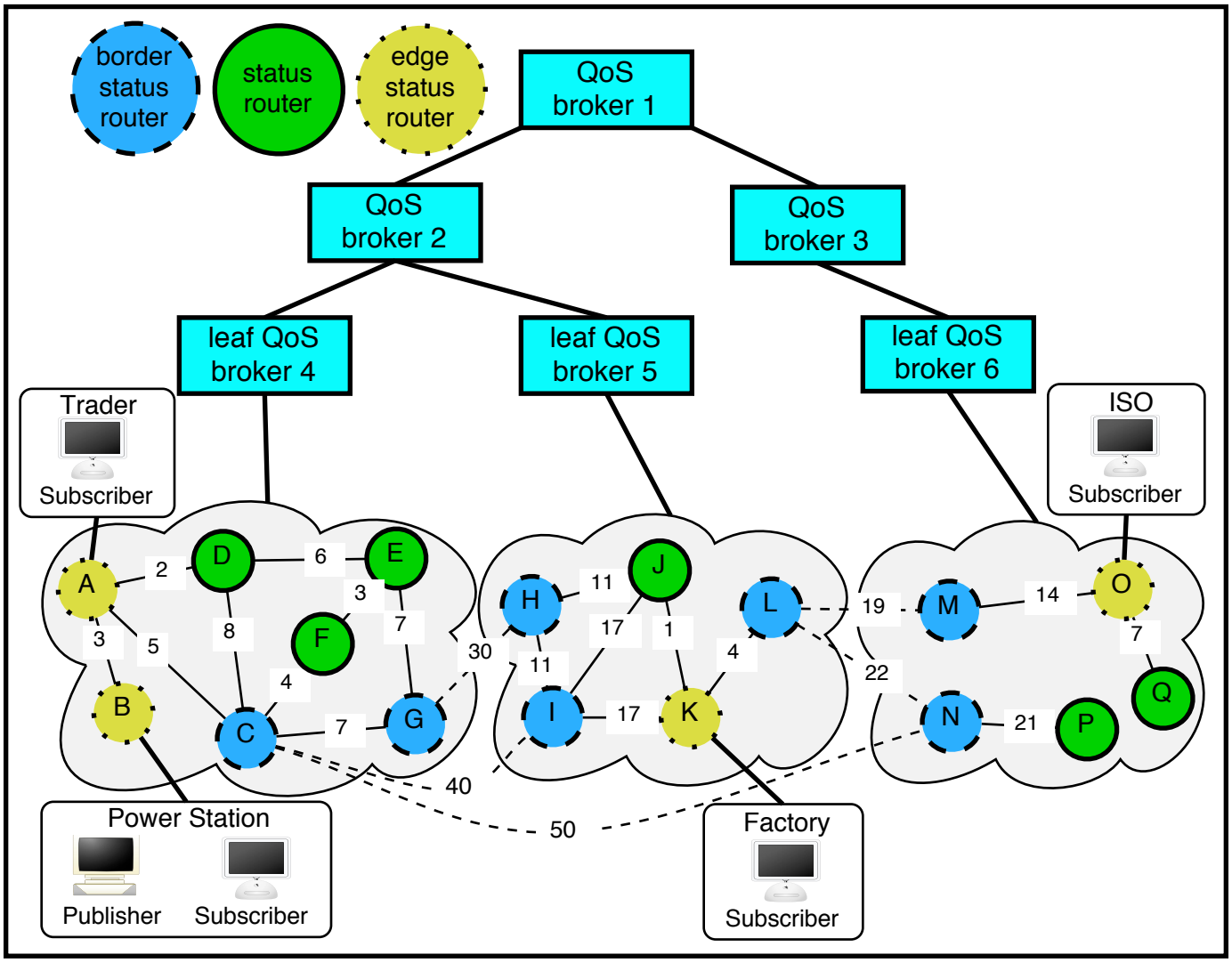

Figure 5: Detailed Architecture of GridStat

and resource reservation techniques are employed to provide end-to-end QoS guarantees.

3.6 Hierarchical Management

The hierarchical management is involved in naming, path allocation decisions, access control, and fault tolerance. Subscribers and publishers only contact their edge status routers. To subscribe to a status variable, a subscriber submits its request to its edge status router and from there the request is forwarded to the leaf QoS broker of the corresponding cloud. Similarly, publishers advertise their publications via their edge status routers. An alternative design would have the end entities directly contact their leaf QoS brokers. It was decided that the former approach better utilizes the position of the edge status routers in the network. Edge status routers perform security checks on the outgoing requests before forwarding them into the network, and they minimize the interaction of a leaf QoS broker with end entities.

3.6.1 Naming

A hierarchical naming scheme, based on the QoS hierarchy, is used for GridStat entities. The convenience of hierarchical names is apparent when computing subscription paths (explained below). QoS brokers can easily determine whether or not a publisher is in a cloud that they administer, if names are based on the QoS broker hierarchy.

3.6.2 Path Allocation Decisions

QoS routing consists of two subtasks: path selection and network state maintenance. Path selection requires finding a path satisfying a set of QoS constraints. Network state maintenance requires collecting, maintaining and distributing information regarding the network state, which is used for selecting a feasible path. Creating efficient QoS routing protocols is still an open problem [22].

Assuming that each leaf QoS broker has a consistent and accurate view of the network state, the remaining routing task is the computation of a feasible path. Determining a path that is subject to multiple additive constraints (multi-constrained path) belongs to the NP-complete problem class. An additional challenge is to compute disjoint multi-constrained paths. To provide timely delivery of messages even if network components may fail, it is necessary to establish multiple node-disjoint paths between a publisher and a subscriber.

Currently, to the best of our knowledge, there are no efficient heuristics that compute node-disjoint multi-constrained paths, although there are published algorithms for multi-constrained paths and single-metric disjoint paths. As communications infrastructures like GridStat become more prevalent, the need for a range of such algorithms with a variety of tradeoffs will become even more important.

Admission control in the GridStat environment includes accepting a request for a subscription from a legitimate entity, processing it based on the access control policies or any other management policies, determining a feasible path for the requested connection and establishing that path if possible. The admission control task is assigned to QoS brokers. Admission control processing is illustrated in Figure 6.

The functionality of the various components of Figure 6 is briefly outlined below:

Table 2: QoS broker network state representation entry

From-cloud	To-cloud	Node-Name	Node-Name	Property 1	Property N
QoS3	QoS4	F	G	X value	Y value

Network State Representation The network state that is administrated by the QoS broker is described based on network parameters, such as link latency, link availability, throughput, cost, and reliability. The leaf QoS broker has knowledge of all network parameters' values for its domain, whereas an upper-level QoS broker only has information about the parameters' values of the links between its border status routers and other clouds' border status routers. One of the tasks of QoS routing is to deliver various link properties to the entities that need them. For example, each leaf QoS broker must know exactly what the network topology of its cloud is in order to apply the routing algorithms. On the other hand, a QoS broker does not need such a detailed representation. The network state representation for an upper-level QoS broker (QOS1 in Figure 7) is shown in Table 2.

Policies Access control policies are evaluated in order to make an admission decision based on the credentials of the requester. Other policies might exist that would affect the routing decisions within a domain.

Path Selection The leaf QoS broker receives a request to establish a subscription path on behalf of one of its subscribers. It executes the routing algorithm itself (if the publisher resides in its domain) or forwards it up to its parent QoS broker. A subscription request is forwarded up the hierarchy of QoS brokers until a QoS broker that controls all the clouds necessary to establish such a subscription path is reached. That QoS broker then uses its routing algorithms to decide which clouds and inter-cloud links should be used for the subscription path. After that, it sends commands to its subordinates to establish the complete path.

Path Evaluation The sub-paths computed by lower-level QoS brokers are submitted to the upper-level QoS broker that requested them, where the concatenation of the sub-paths and the involved inter-cloud links is evaluated to check whether the resultant concatenated path satisfies the overall QoS properties as identified in the subscription request.

Path Establishment If a feasible path is found, the leaf QoS brokers are told to reserve the sub-paths that they submitted. The leaf QoS brokers send the appropriate messages to the status routers within their clouds to update their forwarding tables. If no feasible path can be found, then the involved QoS brokers and edge status routers are notified about it.

Path State When the path is successfully established, the hierarchical broker where the concatenation of paths was performed updates its own subscription table.

Path Invalidation A path can be invalidated by removing all the entries referring to that path from all related routing tables.

Adaptation Depending on network conditions, decisions can be made to invalidate paths or reestablish previously revoked paths.

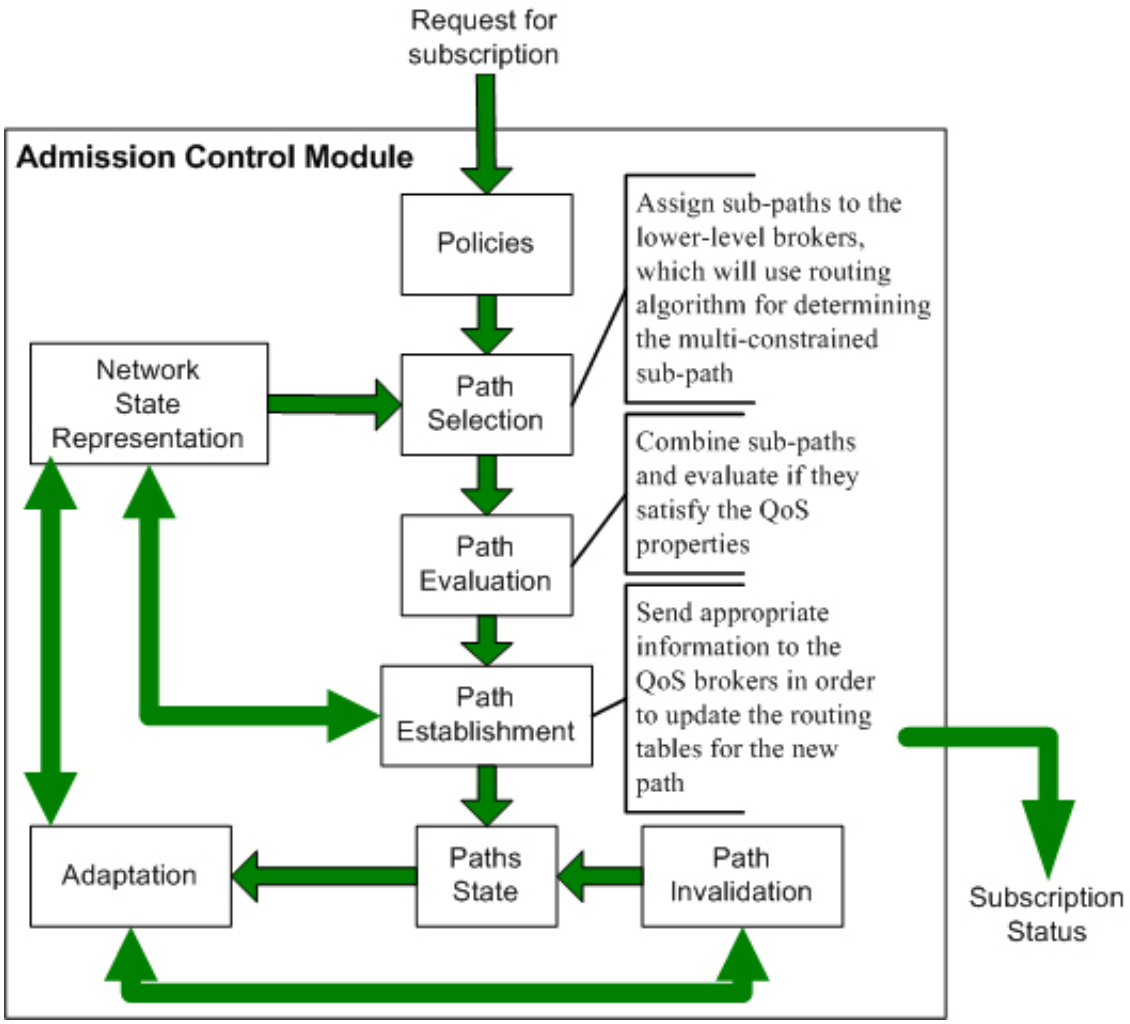

Figure 6: Admission Control Flowchart for Hierarchical QoS Path Allocation

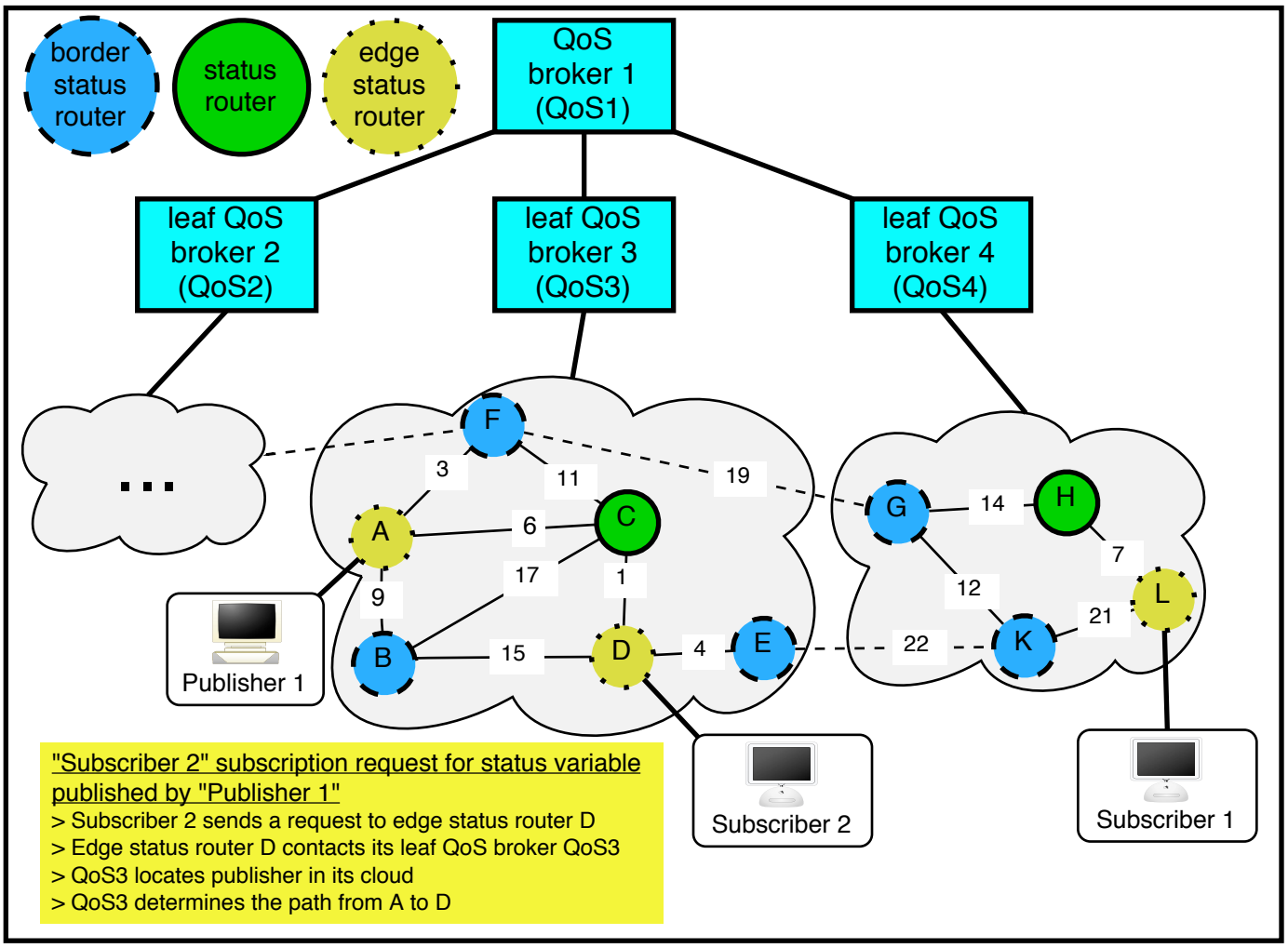

Figure 7: Intra-routing

There are two cases of admission control; one performed at the lowest-level QoS broker (leaf QoS broker) and the other one taking place at an upper-level QoS broker. Examples for each case are provided below:

Example of Intra-cloud Admission Control at leaf QoS broker

Consider Figure 7. For illustration reasons, the graph in this example is not directed although the actual network topology would be directed. Assume that "Subscriber2" is interested in receiving a status event from "Publisher1". Using the subscription interface[3] <publisher name, status name, timeliness>, it sends the request <QoS1/QoS3/A/Publisher1, event1, 20> to its edge status router D. Edge status router D forwards the subscription request to the leaf QoS broker QoS3, which locates the publisher in its domain. Therefore, this broker can locally determine whether a feasible path exists or not. In this case, the path <{A,C,D},7> is returned and the appropriate path establishment actions are taken.

Example of Inter-cloud Admission Control at QoS broker

[3]The actual subscription interface allows for multiple QoS requirements.

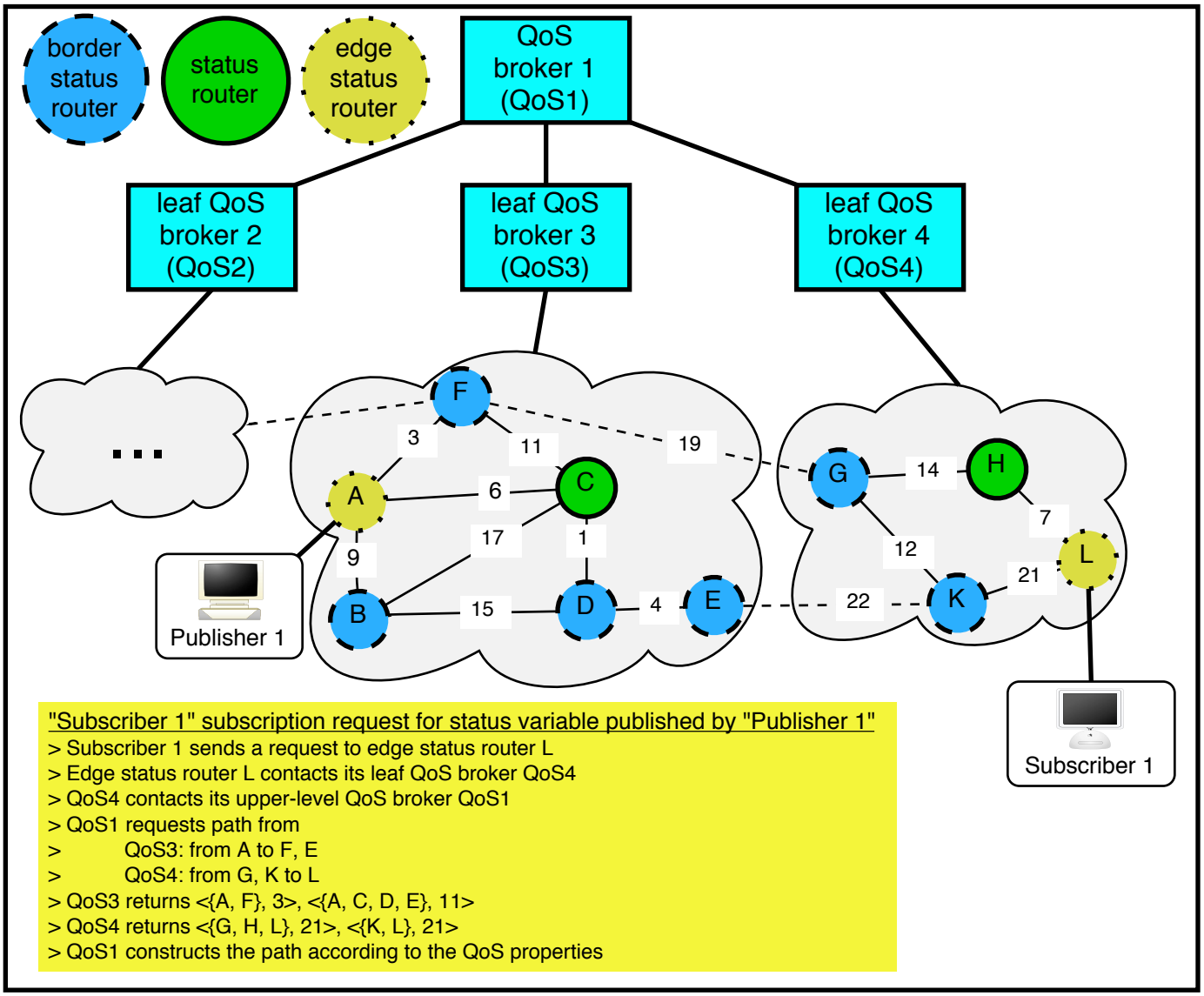

Figure 8: Inter-routing

Consider Figure 8. Assume that "Subscriber1" wants to receive an alert published by "Publisher1". The first step taken by "Subscriber1" is to submit the subscription request, using the subscription interface <publisher name, status name, timeliness> to its edge status router L. Suppose that the request received is <QoS1/QoS3/A/Publisher1, alert1, 40>. Edge status router L will forward it to its leaf QoS broker QoS4. QoS4 forwards the request to its QoS broker one level up, QoS1. This QoS broker locates the publisher under its controlling domain and determines which clouds are involved in delivering the information from "Publisher1" to "Subscriber1". The clouds involved are the ones managed by QoS3 and QoS4. Therefore, QoS1 sends commands to QoS3 to select paths from edge status router A to border status router F and E, and to QoS4 to select a path from border status routers G and K to edge status router L.

The leaf QoS brokers deliver the paths <{A,F},3>, <{A,C,D,E},11>, <{K,L},21> and <{G,H,L},21> to QoS1. There are 2 possible paths that can be constructed and these are <{A,F,G,H,L},3+19+21> and <{A,C,D,E,K,L},11+22+21>. None of the paths satisfy the requested timeliness. The subscriber is notified through its edge status router that the request could not be satisfied.

3.7 Network Security Requirements for Status Dissemination Middleware

Typical publish-subscribe architectures lack the capacity to support a uniform network security framework accommodating diverse security requirements such as information flow integrity and confidentiality, or to set security requirements for the communications paths that information will take [27]. In the GridStat framework, this is an issue since the publishers, the subscribers, and the owners of the communication paths may all be competitors. Information about the kinds of status that is being passed by a competitor or the ability to delay a competitor's status information may provide significant and unfair competitive advantage. These concerns apply to both the network security requirements of the application and of the infrastructure. Additions to the traditional publish-subscribe model in the context of the GridStat environment that will support different requirements for information protection, integrity, and access control are part of the GridStat network security framework. Developing there enhancements involves identifying both the security needs of all active GridStat entities at an individual level and the security requirements during their interactions, namely publisher–subscriber, publisher(or subscriber)–status routers, publisher(or subscriber)–QoS hierarchy and status routers–QoS hierarchy interactions.

In addition to the network security issues, trust management is required for establishing trust relationships between the various participants [19]. There are several open issues regarding trust in an open environment, such as how to define trust, build trust relationships, evaluate them and re-evaluate them in the face of new information [18], [25] . In the critical infrastructure domain, issues of inter-organizational trust compound the difficulty of creating effective security mechanisms. By definition, trust is not a transitive relation, yet effective models for composing trust paths are required. Similarly, in an environment such as GridStat it is not likely to be feasible to establish trust relationships between all parties. For instance, some companies may not wish to have their identities known as users of a GridStat communication network. Others may be joining the network for such a short period of time that it is inefficient to fully assess their trust relationships with all other parties yet it is desirable to allow them to have distinctive security policies with regard to information handling.

4 Mechanisms to Provide Services for Status Dissemination Middleware

This section presents the mechanisms used in GridStat to meet the requirements for status dissemination middleware. First a baseline set of mechanisms to deliver status variables is presented. Then a number of mechanisms that enhance the efficiency of such a system is described, followed by a section about mechanisms to enhance resilience of the overall status dissemination middleware framework. Finally, how such a status dissemination middleware framework helps with programmability for the application developer is explained.

4.1 Baseline Set of Status Delivery and their Control Interfaces

The baseline set of mechanisms GridStat provides carry out three principal activities.

Data plane configuration mechanisms These mechanisms are used to establish links between GridStat entities based on the knowledge of the network topology and network state in the configuration of the QoS brokers. GridStat masks heterogeneity between different network protocols and hardware to provide a common interface to the application layer. GridStat uses an underlying communication network infrastructure, either a point-to-point network between the entities or an overlay network over some physical communication network, to provide a network of status routers. The underlying communication network provides an interface that GridStat can use to establish real (point-to-point network) or "virtual" (overlay network) communication links. GridStat relies on the existing network infrastructure to provide mechanisms for controlling network resources such as bandwidth and usage of network routers. Currently, GridStat operates over the TCP/IP protocol. For a real deployment of GridStat, a dedicated network would be the most appropriate approach, so that GridStat could control all the network resource usage.

QoS hierarchy configuration mechanisms These mechanisms manage the hierarchy of QoS brokers. QoS brokers exchange command messages with their parents and children. A command message is sent by a child to its parent whenever its state or the state of any inter-cloud communication link that it manages changes. When the state of an inter-cloud communication link changes, the information propagates up the hierarchy until a QoS broker that is an ancestor to clouds at both ends of the communication link is reached.

Subscription mechanisms These mechanisms establish, restore and update subscription paths from publishers to subscribers. These activities were described in Section 3.6.2.

4.2 Mechanisms to Enhance Efficiency

The mechanisms that enhance efficiency are designed to save bandwidth and computation time for the entities in the GridStat framework. They optimize the datapath for subscriptions. QoS brokers use these mechanisms when allocating subscription paths, allowing more subscription paths to be allocated than would be otherwise possible.

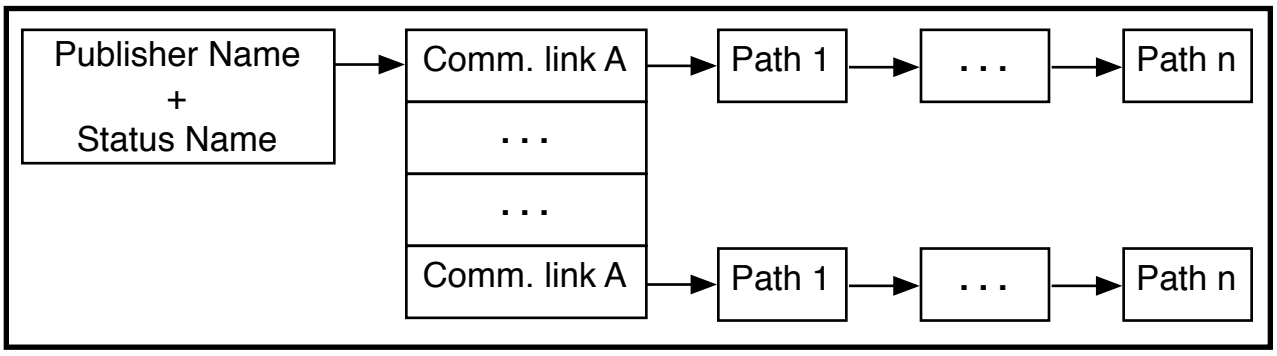

Figure 9: Routing Table Entry at Edge/status router

4.2.1 Route Aggregation

Route aggregation is accomplished by reusing already established paths, i.e. only one copy of a status event is sent on any output link. If multiple subscribers subscribe to the same status variable and the subscription paths go through the same status router, then bandwidth and computation are saved by only sending one copy of the status event.

Routing tables in the status routers are organized to support this optimization. In Figure 9, an entry in the routing table is shown. When the leaf QoS broker establishes a path it sends routing entries to the status routers. If a path already routes this status event to the same communication link, then the path is just appended to the list of paths that uses this routing entry. If there are no paths for this status variable using the communication link, a new entry is added. When the last path is removed for a routing entry, that entire routing entry is removed.

When a status event arrives, its publisher name and status name are extracted and used as an index into the routing table. Then the status event is sent out on each of the communication link entries, but only once on each communication link.

The route aggregation presented above differs from plain multicast in that the saved bandwidth is managed. The QoS brokers are aware of this optimization. The resulting system is a managed multicast system that can operate close to the maximum throughput without violating QoS requirements.

4.2.2 Packing of Status Events

Another optimization that saves bandwidth on the datapath is to send data packets as large as possible without exceeding the maximum protocol data unit (PDU) size of the physical communication layer. Status routers buffer status events for each link until a timeout occurs or the maximum PDU size is reached and then send a single data packet containing multiple status events. This increases the efficiency of bandwidth usage, but also increases end-to-end delay of status events. The trade-offs between improved efficiency, increased delay, decreased protocol processing overhead and increased buffering overhead associated with status events packing remain to be explored. It is noted that small packets map well onto some network technology with small payload, such as ATM. In this case packing of status events will not be necessary.

Since the leaf QoS broker is informed about the status of the communication links and the QoS

requirements for the subscription paths that go through each status router are known, the leaf QoS broker is in a position to issue commands to increase or slow down the time that status events are buffered at each status router. Leaf QoS brokers may issue commands to adjust the buffering and delay at status routers, allowing dynamic adaptation of the use of bandwidth compared with the end-to-end latency of status events.

4.2.3 Condensation Functions

A condensation function subscribes to a number of status variables in order to create a new status variable. GridStat provides for defining condensation functions and loading them into edge status routers. QoS policies control the placement of condensation functions on the edge status routers. Condensation functions whose output is broadly useful are best placed near the edge routers of the publishers of their inputs. Condensation functions with only a single subscriber are best placed at the edge router of the subscriber.

Interesting condensation functions include calculating the average, min or max of several status events, and calculating the min, max, or rate of change of a single status variable over some time.

4.3 Mechanisms to Enhance Resilience

A system that controls a critical infrastructure must be operational under all conditions. Occasionally when something goes wrong a domino effect is created, overloading other entities resulting in their failure and so on. GridStat provides mechanisms that make the system resilient even when it is operating close to its limits and in the presence of failures. These mechanism will now be explained.

4.3.1 Normal Events and Alert Events

GridStat handles normal status events and alert events differently.

Normal events are handled in a FIFO order at each status router as they are routed through the network. The leaf QoS brokers will make sure that they don't overload any status router so that the delay at the status routers is bounded, and status events will never be dropped due to overload.

Alert events are routed through the network with the smallest possible delay. Alert events are handled before normal events. If multiple alerts arrive at the same time they are routed in FIFO order.

4.3.2 Cache Extrapolation

Transient network omission failures can be masked for the application by extrapolating from previous received values. The subscriber specifies if it wants this feature turned on or off and the extrapolation function to be used. In order to use cache extrapolation the steady-state behavior of a status variable needs to be understood and captured in the extrapolation function. Since steady state behavior may not continue during a power anomaly, an alert event may be configured to turn off the use of extrapolation.

4.3.3 Link and Status Router Failure Recovery

The leaf QoS brokers and the status routers provide mechanisms for recovering from communication link failure and status router failure.

Link failure When a communication link fails, the status router that detects the failure marks it as a failed link and informs its leaf QoS broker with a command message. The status router tries to reconnect to the failed link at a regular interval. All status events that are destined for this link are dropped. As soon as the link is reconnected, the status router sends a command message informing the leaf QoS broker that the link is operational again. Status events that are destined for this link again flow through it.

A special case is when a link between a leaf QoS broker and one of its status routers fails. The status router tries to reconnect to its leaf QoS broker at regular intervals, while it continues its normal operation i.e. routing status events that it receives. When the link is re-established the leaf QoS broker sends updates, if any, to the status router that have occurred while the status router was disconnected.

Status router failure When a leaf QoS broker detects that one of the status routers in its cloud has failed, the status router is marked as unavailable. There are two possibilities why it failed: either the communication link to the status router failed or the status router itself failed. The failed status router informs the leaf QoS broker of the reason when it reconnects to the leaf QoS broker. The case of a link failure was explained above in "link failure". If the status router failed, it restarts in a clean state. The leaf QoS broker restores the state of the status router by issuing commands to establish communication links to other status routers and to re-establish its routing table. Each leaf QoS broker keeps enough information so that it can restore the state of all the status routers in its cloud.

4.4 Features to Enhance Programmability

Middleware frameworks provide services at a higher level to ease the task of the developers. GridStat provides features that make it easier to develop applications using status dissemination. Some of these features are described below.

4.4.1 Caches of Latest Value

The subscriber applications provide local object references to contain the values of status events. When the application needs the value of a status variable it can just do an atomic read of the local object and use the value. The application programmer does not need to deal with callback objects to handle the arrival of status messages.

4.4.2 Typed Status Values

The type of each status value is known to the infrastructure, so type checking is performed by the middleware framework and the programming language of the publisher and subscriber. When the subscriber subscribes to a status variable, it gives a typed reference to the object where the value is to be stored. Before the subscription is allowed, type checking is done on the reference and the

type of the published status variable. If there is a mismatch, an error message is returned to the subscriber. This feature helps the application developer in two ways; the application does not need to do type checking and the application never crashes with a type mismatch or illegal cast error.

4.4.3 Status Attributes and Status Patterns

In addition to being able to subscribe to status variables values, the middleware framework provides the ability to subscribe to other dynamic status variable attributes and status patterns. Dynamic attributes such as the value and min/max/average over some time, are published by the publishers. Status patterns, such as alerts when the change in a status variable exceeds a threshold, or when a status variable derived from the difference between two other variables is out of the allowed range, will be published by the condensation functions at edge status routers.

By providing attributes and patterns, the application developer's task is eased. With the wide range of variables available, the developer can concentrate on the logic of the application instead of the logic for retrieving the variables needed for the application.

5 Experimental Evaluation

In this section we present preliminary experimental results for the performance of a GridStat prototype. The experiments were performed on the graph topology presented in Section 5.3, where there is only one level of QoS management. The primary metric was the end-to-end latency from publisher to subscriber, while varying the number of status routers and subscriptions.

The main purpose of the experiments was to evaluate the cost of one hop through the status routers. This was done by setting up and running a system with one status router, and then adding one more status router and running the same experiments. Such performance data is an important input to the path evaluation process that the GridStat framework uses to guarantee end-to-end latency.

Prior to hardware implementation, the software prototype will provide valuable data and experience concerning which features and characteristics are most important in status dissemination systems.

5.1 Software

GridStat currently is implemented using SUN's Java SE version 1.4.1 on both Microsoft Windows XP and Linux 7.2 and in Apple's Java SE version 1.4.1 on MacOS X. CORBA interfaces are used between the major components. The CORBA ORB used is ORBacus for Java version 4.1.2.

5.2 TestBed Hardware

The hardware testbed used for evaluating the performance of GridStat consists of:

- 1 Dell Dual Xeon, 1.5 GHz, running Microsoft Windows XP
- 2 Dell P4, 1.5 GHz, running Linux 7.2

- 2 Dell P4, 1.5 GHz, running Microsoft Windows XP
- 1 CM P3, 1.13 GHz, running Microsoft Windows XP
- 1 Apple PowerBook, 1.0 GHz PowerPC G4, running MacOS X
- 1 Apple eMac, 800 MHz PowerPC G4, running MacOS X

5.3 Configurations

Figures 10 and 11 illustrate the graph topology used for the experimental evaluation. It consists of 20 edge status routers (10 used for the publishers and 10 used for the subscribers), and *r* (1 in Figure 10 and 2 in Figure 11) status routers. The publishers and subscribers are divided into two systems. The first one is called the 10-to-10 system (running on the Apple PowerBook). It consists of:

- 10 publishers, each connected to one of the edge status routers and publishing *p* status variables every 200ms.
- 10 subscribers, each connected to one of the edge status routers and subscribing to *s* status variables every 200ms.

The second system is called the 1-to-1 system (running on the Apple eMac). It consists of:

- 1 publisher connected to the 10th edge status router and publishing 1 status variable every 200ms.
- 1 subscriber connected to the 10th edge status router and subscribing to the published status variable every 200ms.

The following nodes were executing on the following hardware:

- The single[4] leaf QoS broker was running on the Dell Dual Xeon.
- Each of the *r* status routers was running on one of the Dell P4 running Linux.
- The 10 edge status routers that the publishers connected to were all running on the CM P3.
- 5 edge status routers (for the subscribers) were running on each of the Dell P4 running Microsoft Windows XP.
- The 10-to-10 system publishers and subscribers were running on the Apple PowerBook.
- The 1-to-1 system publisher and subscriber were running on the Apple eMac.

Running multiple components on each of the machines imposes additional cost due to CPU sharing that would not be present in an actual system. The measurements derived from this configuration bound how slowly an actual system would perform.

[4]The hierarchical management is limited to one level. Hierarchical management is implemented, but no performance testing has been done.

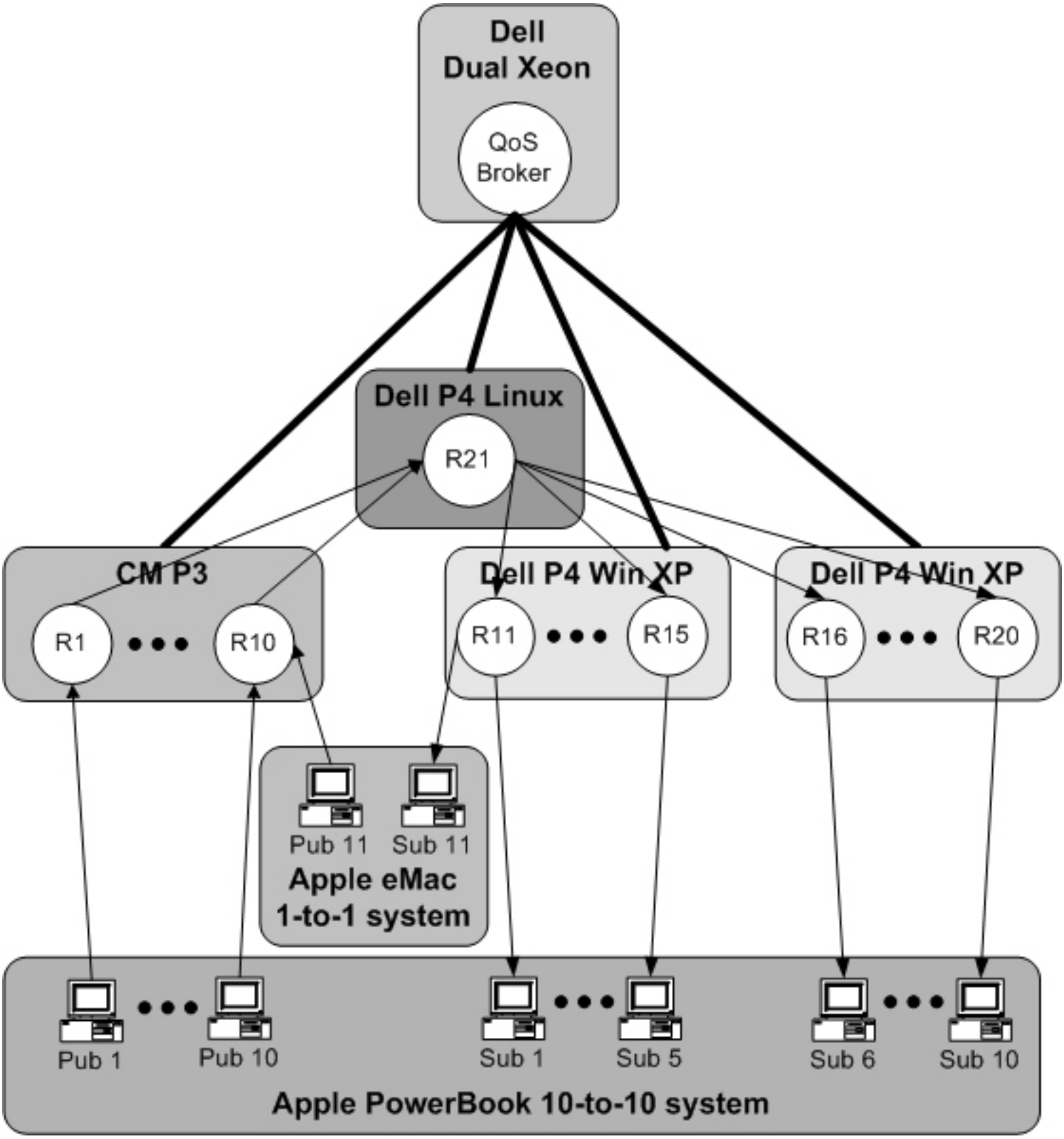

Figure 10: GridStat Basic Graph Configuration with 1 status router

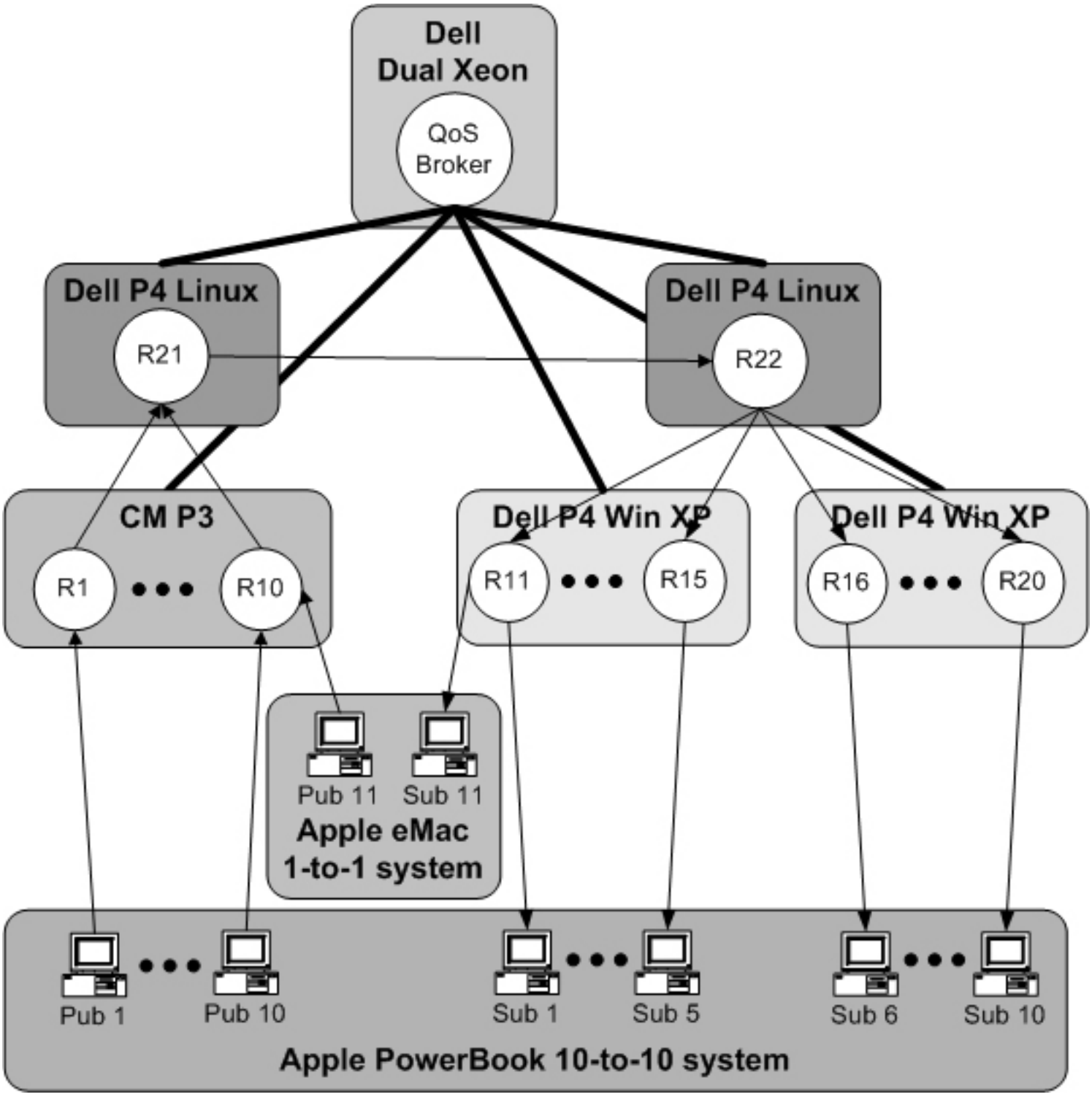

Figure 11: GridStat Basic Graph Configuration with 2 status routers

Table 3: 10-to-10 system with 1 status router

Subscriptions	0–5ms	0–10ms	0–15ms	0–20ms	0–25ms	0–30ms	0–40ms	0–50ms
500	58.8%	89.1%	94.4%	96.3%	97.3%	97.9%	98.7%	99.3%
750	32.7%	62.7%	74.3%	80.4%	84.4%	87.3%	91.3%	94.1%
1000	5.9%	17.0%	26.7%	35.0%	41.8%	47.2%	56.7%	63.8%

Table 4: 1-to-1 system with 1 status router

Subscriptions in the 10-to-10 syst.	0–5ms	0–10ms	0–15ms	0–20ms	0–25ms	0–30ms	0–40ms	0–50ms
500	83.4%	97.3%	98.0%	98.6%	98.9%	99.1%	99.4%	99.6%
750	76.8%	92.9%	96.4%	97.6%	98.3%	98.8%	99.1%	99.4%
1000	69.3%	80.2%	85.4%	88.6%	91.4%	93.0%	95.1%	96.3%

We tested the two configurations depicted in Figure 10 and 11 while varying the load on status routers (R21 and R22); i.e., varying the numbers of status variables subscribed to by each of the subscribers in the 10-to-10 system. Experiments were run with $s \in \{50, 75, 100\}$ and $p = 10$. Including the 1-to-1 system this corresponds to 501, 751, 1001 total subscriptions passing through R21 and R22.

Each experiment ran for approximately 30 minutes. The status events received during the first 5 minutes of each run were excluded due to initialization of the Java VM.

5.4 Performance

The following tables show the percentage of subscriptions whose end-to-end delay lies within the ranges 0–5ms, 0–10ms, 0–15ms, 0–20ms, 0–25ms, 0–30ms, 0–40ms, and 0–50ms. These delays were collected by taking the average latency for all the subscribers in the 10-to-10 system. Delays were measured with precision 1 ms. (Since publishers and subscribers run in the same system, accuracy of the measurement depends only on accuracy of the local system clock. Synchronization of the clocks across the distributed system is not necessary.)

5.4.1 One Status Router ($r = 1$) and Subscriptions $s \in \{50, 75, 100\}$

Table 3 shows the results for the 10-to-10 system, measured over all the subscriptions in the system. Each subscriber received, on average, 400201, 617689, and 845472 status events corresponding to $s \in \{50, 75, 100\}$ respectively.

The 1-to-1 system was running at the same time as the 10-to-10 system using the same status routers. Table 4 shows the results for the 1-to-1 system, for the single subscription in the system. The subscriber received, on average, 8698, 8700, and 8699 status events corresponding to $s \in \{50, 75, 100\}$ in the 10-to-10 system respectively.

Table 5: 10-to-10 system with 2 status routers

Subscriptions	0–5ms	0–10ms	0–15ms	0–20ms	0–25ms	0–30ms	0–40ms	0–50ms
500	46.7%	85.3%	93.2%	95.8%	97.0%	97.7%	98.6%	99.2%
750	27.8%	60.5%	73.9%	80.6%	84.8%	87.7%	91.7%	94.5%
1000	4.7%	15.0%	24.5%	32.6%	39.5%	45.1%	54.0%	61.0%

Table 6: 1-to-1 system with 2 status routers

Subscriptions in the 10-to-10 syst.	0–5ms	0–10ms	0–15ms	0–20ms	0–25ms	0–30ms	0–40ms	0–50ms
500	85.5%	96.8%	98.4%	98.7%	98.9%	99.0%	99.3%	99.5%
750	75.6%	92.6%	96.1%	97.6%	98.3%	98.7%	99.2%	99.4%
1000	70.8%	80.2%	85.6%	88.6%	91.2%	92.9%	94.9%	96.0%

5.4.2 Two Status Routers (r = 2) and Subscriptions $s \in \{50, 75, 100\}$

Table 5 shows the results for the 10-to-10 system, measured over all the subscriptions in the system. Each subscriber received, on average, 400174, 617672, and 845192 status events corresponding to $s \in \{50, 75, 100\}$ respectively.

The 1-to-1 system was running at the same time as the 10-to-10 system using the same status routers. Table 6 shows the results for the 1-to-1 system, for the single subscription in the system. The subscriber received 8699, 8700, and 8700 status events corresponding to $s \in \{50, 75, 100\}$ in the 10-to-10 system respectively.

5.4.3 Status Routers $r \in \{1, 2\}$ with 1 Subscription

Table 7 shows the results collected by having only one of the publishers publish 1 status variable and one of the subscribers subscribe to this 1 published status variable in the 10-to-10 system. The subscriber received 8702 status events corresponding to both r=1 and r=2.

5.5 Analysis

5.5.1 Analysis for the 10-to-10 and 1-to-1 Systems Experiments

As the load is increased by adding more subscriptions, some parts of the system become overloaded. With 1000 subscriptions, the CPU of the Apple PowerBook is executing at 100% which explains the poor performance of the 10-to-10 system. The CPU utilization for the Dell P4 Windows machines running the edge status routers for the subscribers, is within the ranges 20%–40%,

Table 7: 10-to-10 system with 1 subscription

Status routers	0–5ms	0–10ms	0–15ms	0–20ms
1	99.74%	99.87%	99.91%	99.92%
2	99.54%	99.89%	99.90%	99.92%

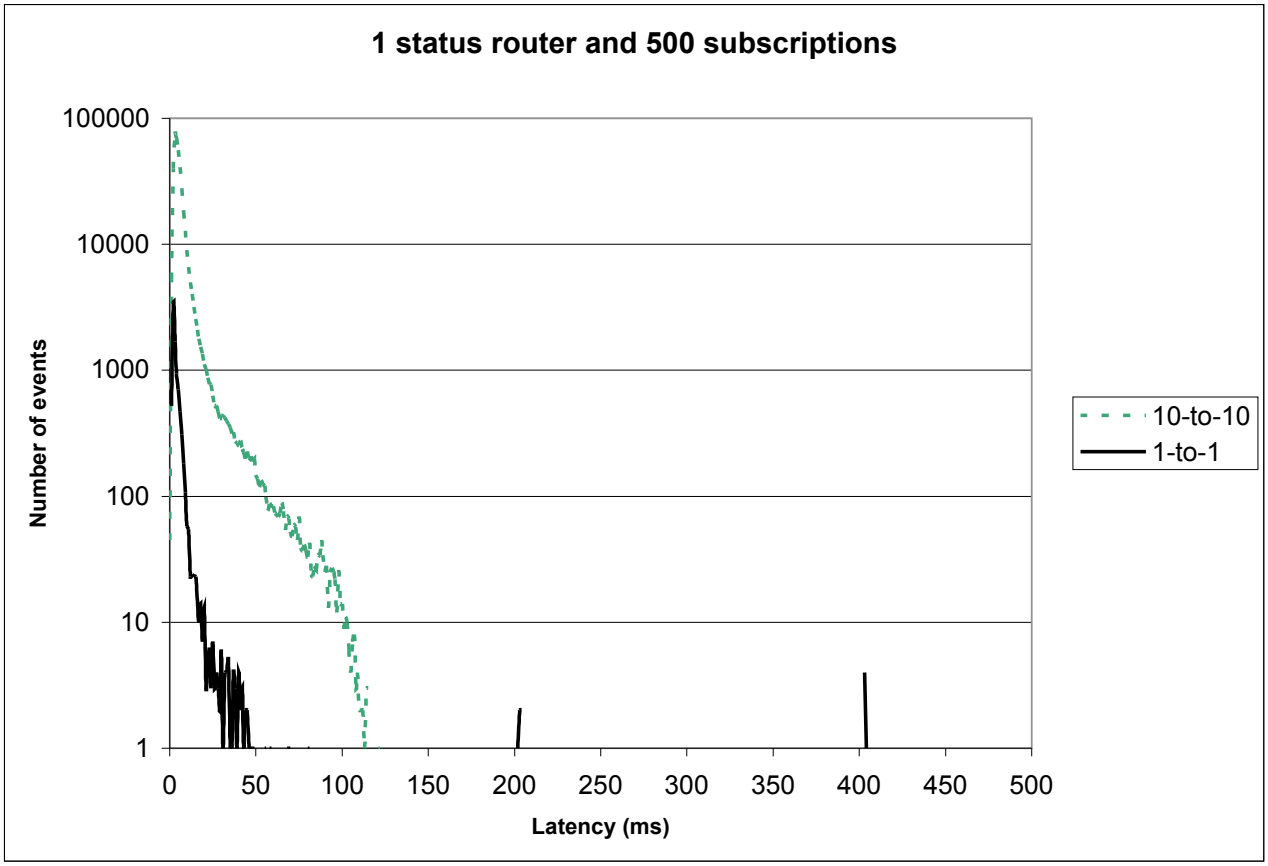

Figure 12: Latency distribution for 500 subscriptions routed by 1 status router every 200ms

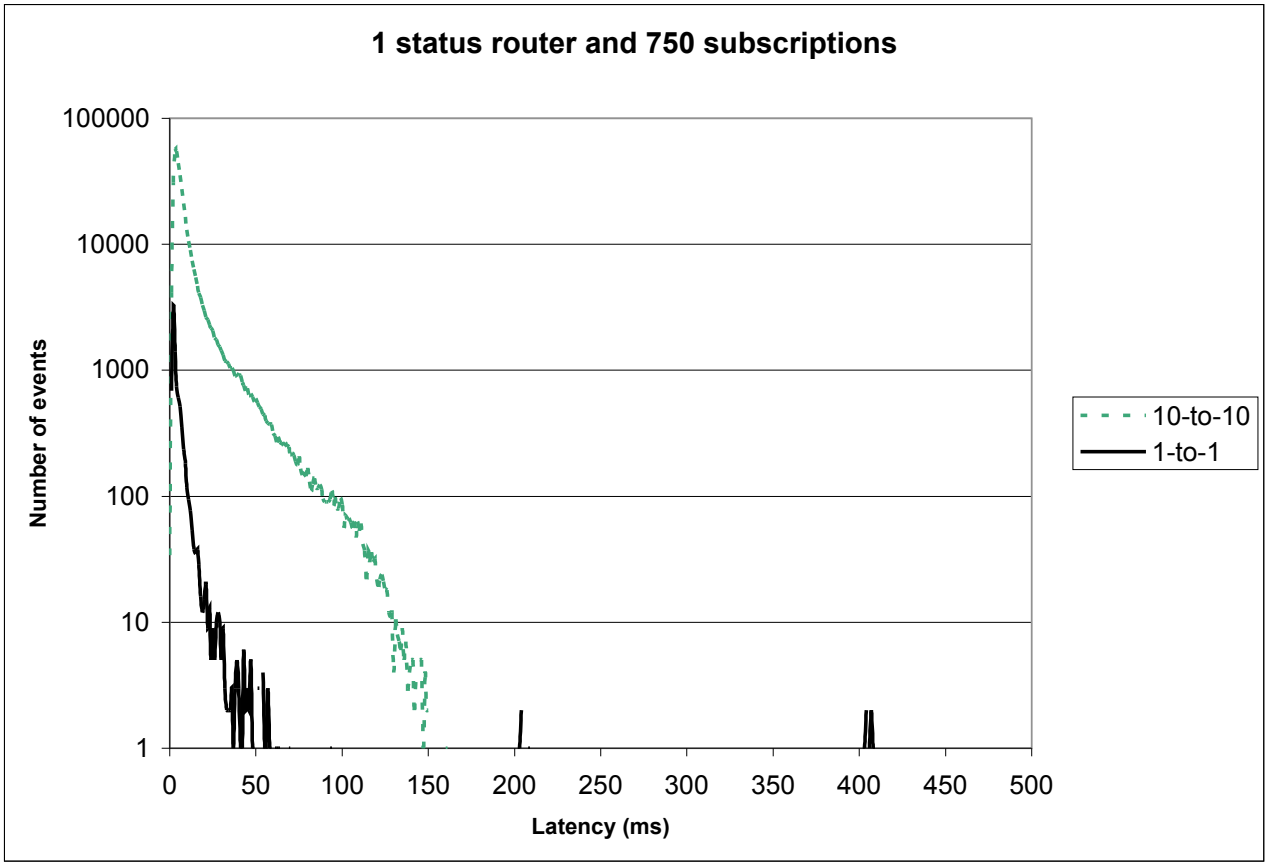

Figure 13: Latency distribution for 750 subscriptions routed by 1 status router every 200ms

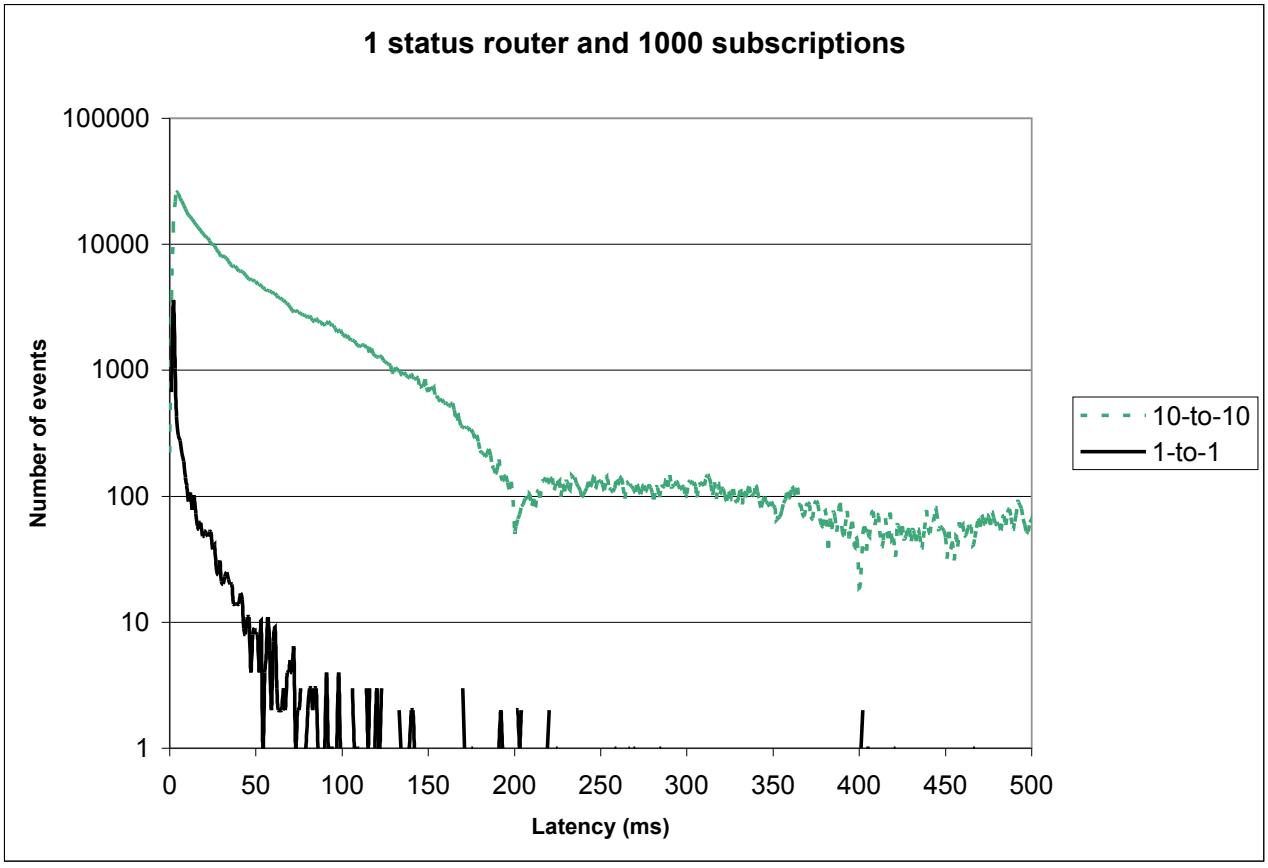

Figure 14: Latency distribution for 1000 subscriptions routed by 1 status router every 200ms

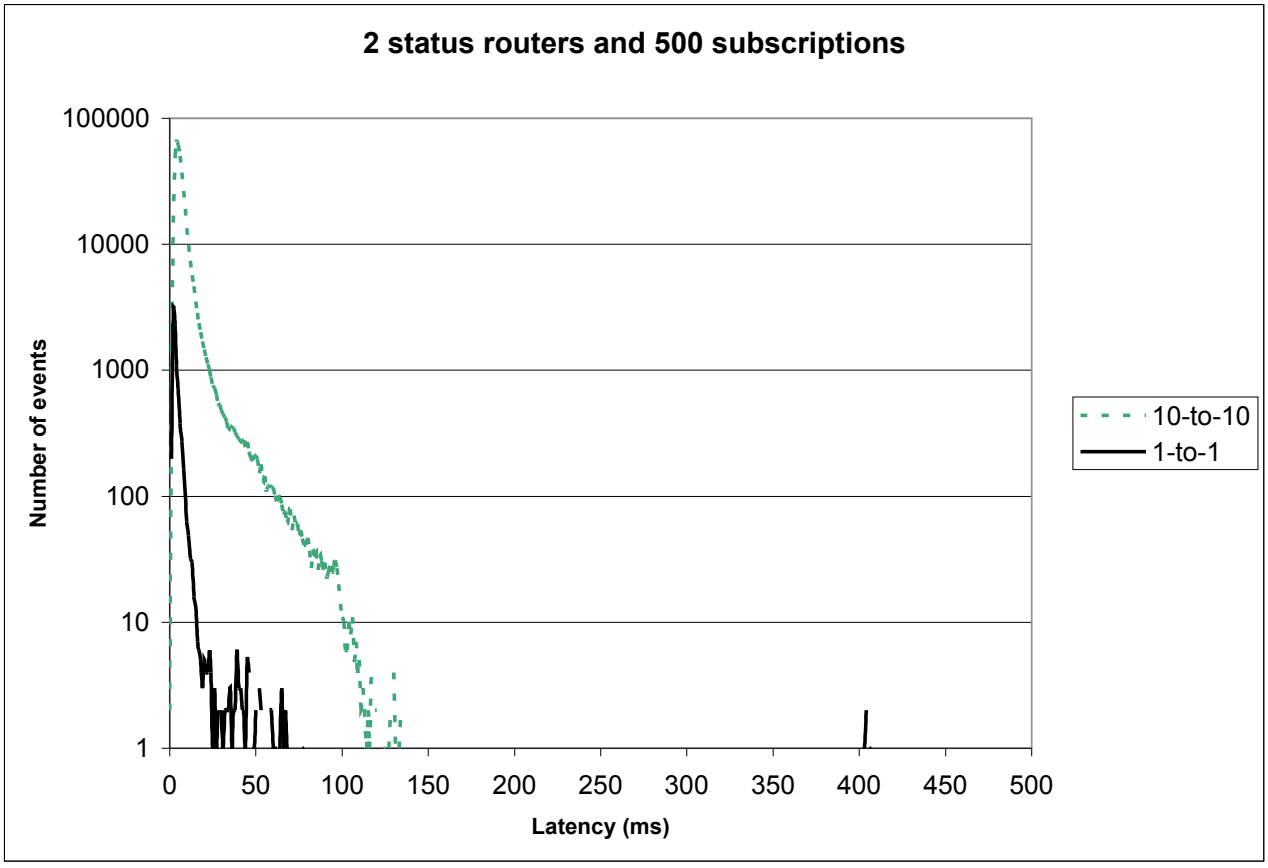

Figure 15: Latency distribution for 500 subscriptions routed by 2 status routers every 200ms

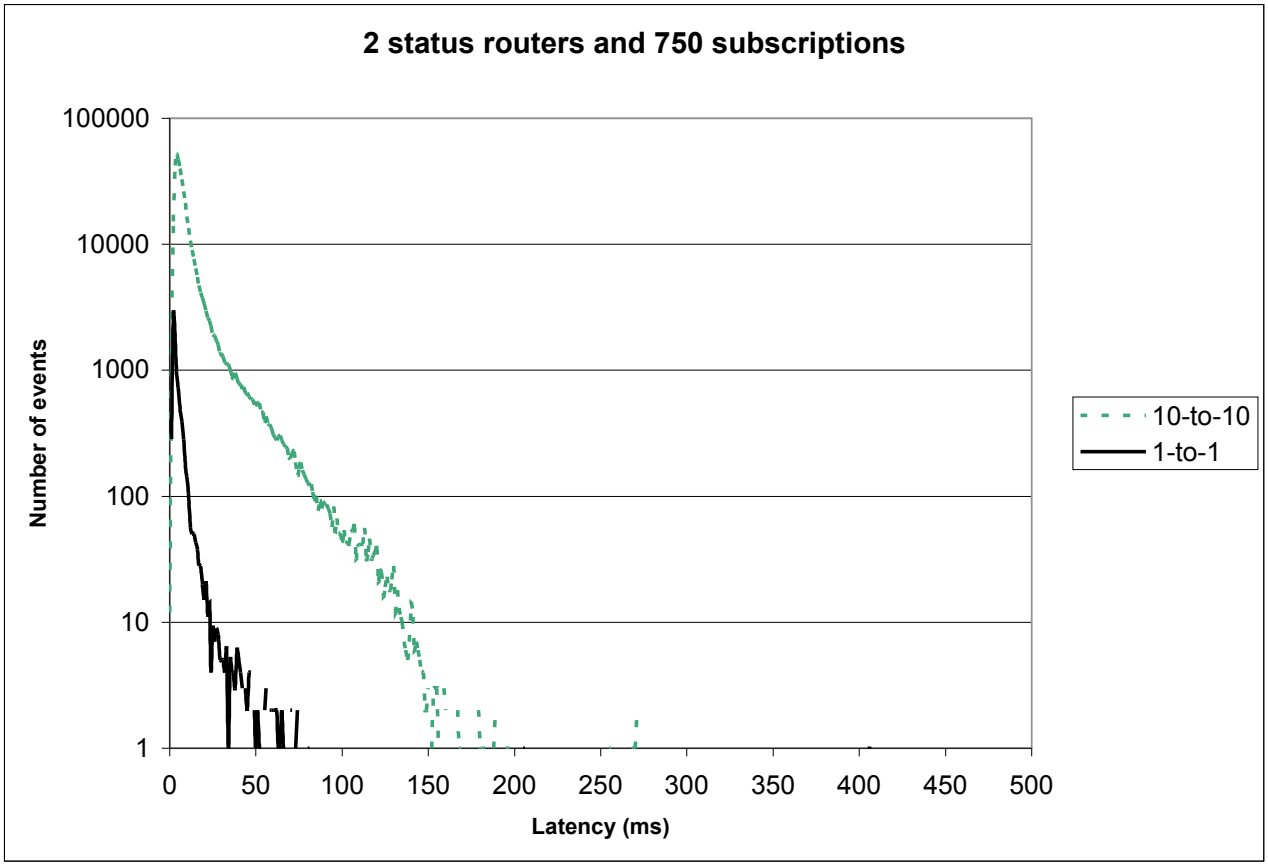

Figure 16: Latency distribution for 750 subscriptions routed by 2 status routers every 200ms

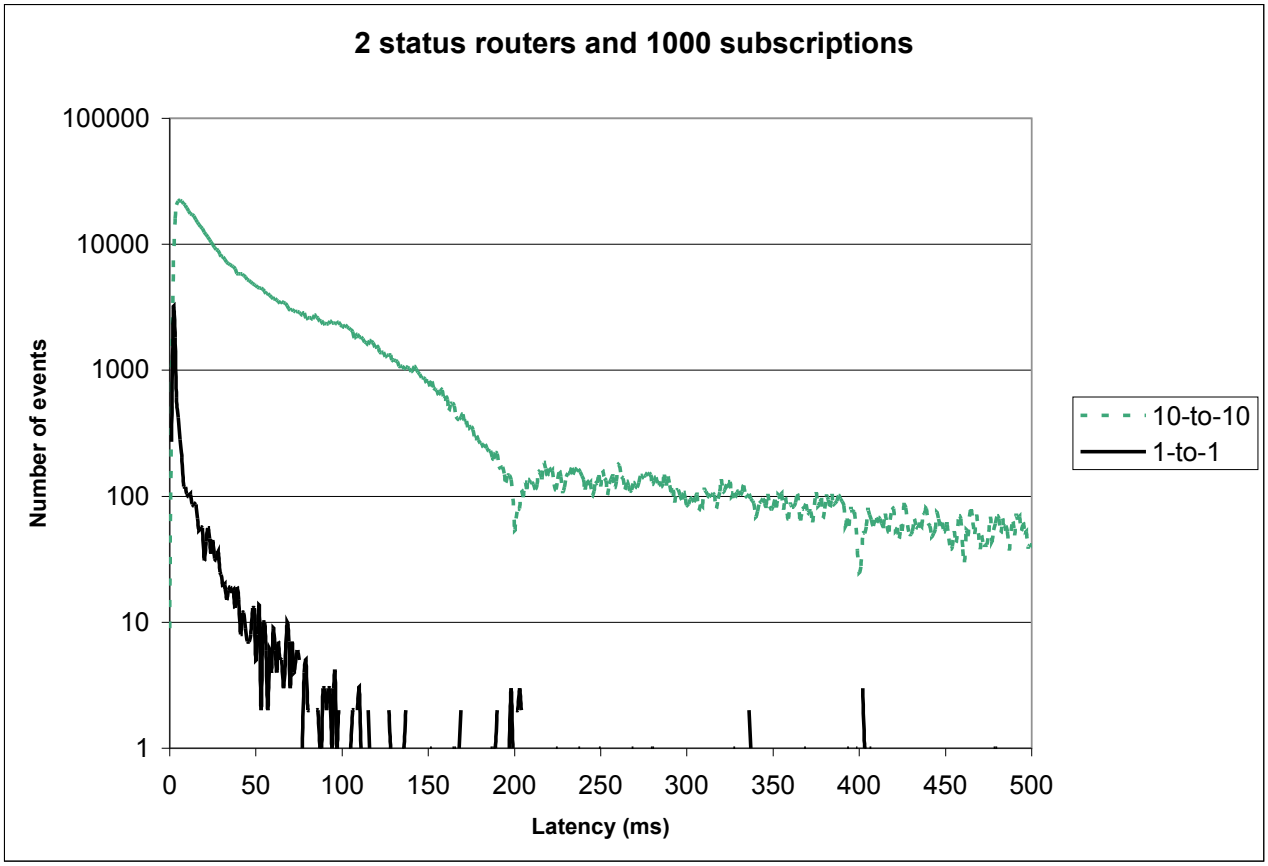

Figure 17: Latency distribution for 1000 subscriptions routed by 2 status routers every 200ms

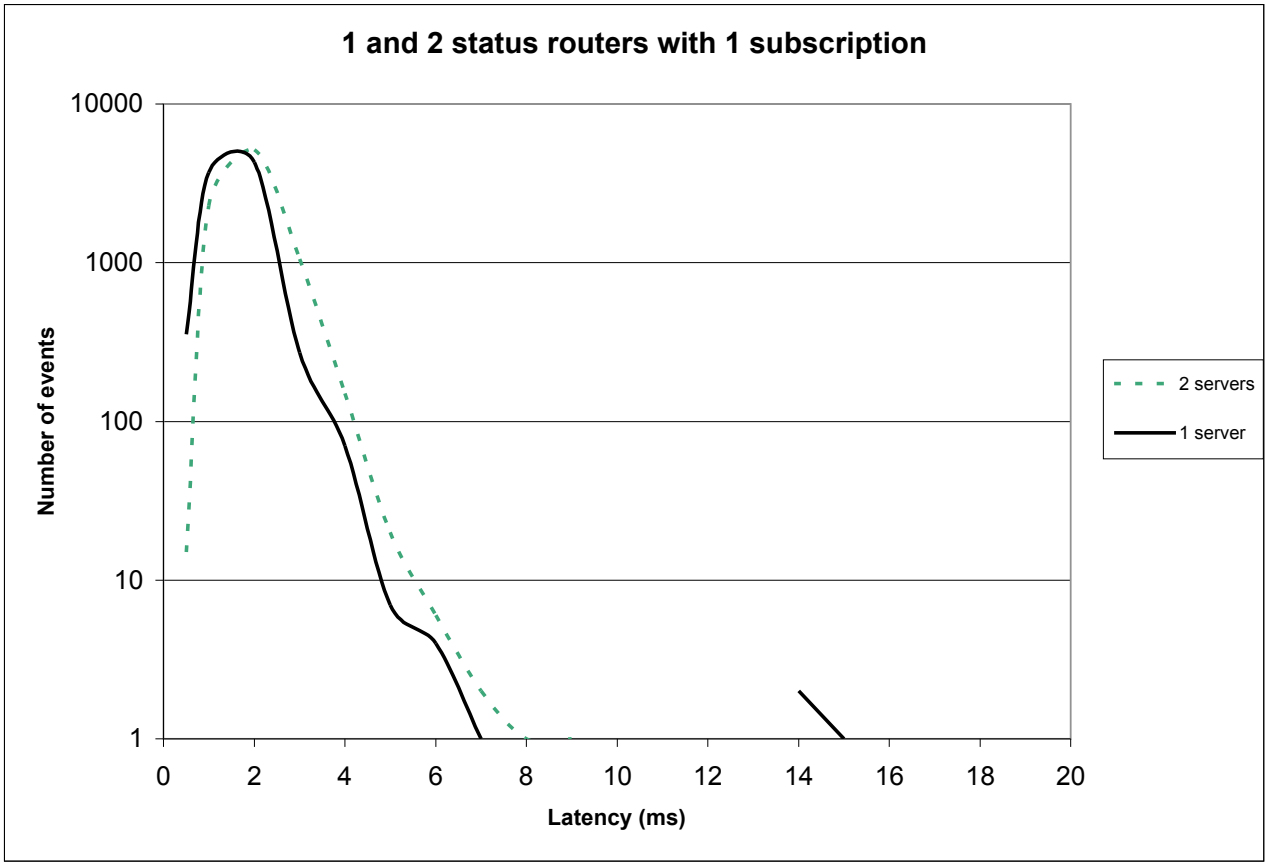

Figure 18: Latency distribution for 1 subscription routed by 1 and 2 status routers every 200ms

35%–55% and, 50%–70% for 500, 750 and, 1000 subscriptions respectively. Since the load of the status routers is increased, the performance of the 1-to-1 system should decline and this can be seen from the results. We are currently investigating if the decline in performance for the 1-to-1 system can be entirely explained by the load of the status routers, or if there is a flaw in the design of the edge status routers that allows slow subscribers to influence other subscribers.

A short in-depth analysis follows for both tested configurations. We concentrate on the 1-to-1 system because the primary purpose of the 10-to-10 system is to generate load on the status routers.

The experiments with 1 status router:

For the experiment with 1 status router, the results show the following:

500 subscriptions For the experiment with 500 subscriptions the average end-to-end latency was 4.8ms with the best 95% having an average delay of 3.3ms and the remaining 5% having an average delay of 44.5ms (Figure 12).

750 subscriptions For the experiment with 750 subscriptions the average end-to-end latency was 5.3ms with the best 95% having an average delay of 3.7ms and the remaining 5% having an average delay of 44.7ms (Figure 13).

1000 subscriptions For the experiment with 1000 subscriptions the average end-to-end latency was 14.1ms with the best 95% having an average delay of 5.9ms and the remaining 5% having an average delay of 181.9ms (Figure 14).

The experiments with 2 status routers:

For the experiment with 2 status routers, the results show the following:

500 subscriptions For the experiment with 500 subscriptions the average end-to-end latency was 4.6ms with the best 95% having an average delay of 3.3ms and the remaining 5% having an average delay of 29.7ms (Figure 15).

750 subscriptions For the experiment with 750 subscriptions the average end-to-end latency was 5.3ms with the best 95% having an average delay of 3.9ms and the remaining 5% having an average delay of 33.1ms (Figure 16).

1000 subscriptions For the experiment with 1000 subscriptions the average end-to-end latency was 12.4ms with the best 95% having an average delay of 5.9ms and the remaining 5% having an average delay of 133.7ms (Figure 17).

5.5.2 Analysis for the 1 publisher 1 subscriber experiment

The load on the system during these two experiments was minimal, so the CPUs of all the machines were executing at 0%–3%. These experiments show the best case scenario for the current implementation, shown in Figure 18.

For the experiment with 1 publisher and 1 subscriber, the results show the following:

The experiments with 1 status router:

The average end-to-end latency was 1.6ms for 1 status router with the best 95% having an average delay of 1.5ms and the remaining 5% having an average delay of 3.7ms.

The experiments with 2 status routers:

The average delay was 1.9ms with the best 95% having an average delay of 1.8ms and the remaining 5% having an average delay of 3.9ms.

5.5.3 Conclusions from the Experiments

The purpose of the experiments was to evaluate the latency across each hop under different loads (number of subscriptions).

From the first set of experiments (the 10-to-10 and 1-to-1 system), it is hard to conclude the cost of adding one more hop, since the results for the 1-to-1 system with 2 status routers are better than those with 1 status router. The results from the 10-to-10 system show that the experiment with 1 status router has a larger percentage of status events arriving with a shorter end-to-end latency than the experiment with 2 status routers.

For the second set of experiments, when the load of the system was kept at a minimum, the cost of one hop could be estimated. For the best 95% of the status events the added latency of the additional status router was:

$1.8ms - 1.5ms = 0.3ms$

6 Implementation Status

This section describes the current status of GridStat, and looks ahead to the pragmatic issues deployment.

6.1 Status

The current prototype of the framework includes support for hierarchies, redundant routing, and path allocation for meeting timeliness guarantees.

Status variable types supported are integers, floats and booleans. Recovery from both data link failure and management link failure is supported. Normal and alert event types are present. Rates of status streams on each link are adjusted to meet the needs of subscribers downstream of that link. Furthermore, rate-controlled status multicast allows links and nodes common to multiple subscription paths to carry only one copy of a given status event.

We also support graphical visualization at leaf QoS Brokers (not discussed in this paper). Visualizations include: status routers and the links between them within an administration domain; resource usage at status routers and communication links; on-demand display of published variables at each edge status router with variable and subscription details available on demand; graphical display of the cloud's status routers and the subscriptions that go through them, again with details available on demand. In addition, strip charts of the received status events at any subscriber, depicting values received and their latencies are supported.

In the current implementation, the default QoS policy is to allow all requests without restricting access to subscriptions. The design of a policy language and mechanisms implementing policies is one of the immediate goals.

6.2 Pragmatic Issues Deployment

A few notes on pragmatic issues involved in an actual deployment of GridStat on a wide scale are now in order. In the following, it is essential to distinguish between using internet protocols such as TCP/IP and UDP/IP and using the Internet itself. The former is feasible and provides economic

and practical benefits. However, deploying GridStat on the best-effort Internet would not allow GridStat to meet its QoS requirements, namely timeliness of delivery. In practice, some of the traffic would have to go over dedicated lines, or, as is being contemplated, a private network for power grid communication [17]. Such a dedicated network could be managed by GridStat, if status routers were extended to utilize real-time scheduling mechanisms or if they were implemented with network processors such as Intel's Internet Exchange Architecture (IXA). It may be possible to use the best-effort Internet for one of the paths for many if not most subscriptions, especially if bandwidth reservation mechanisms such as INTSERV [20] or DIFFSERV [21] were available.

We also note that, while both finding paths under multiple additive QoS constraints and the redundant path problems are NP-hard, the practical implications of this may not be serious. Further investigation of this issue is required prior to any deployment.

7 Future Work

GridStat is a work in progress. In the near future, condensation functions, subscriber cache extrapolation, and status patterns, presented in Section 4, will be implemented.

The existing algorithms assume that link delays and link bandwidth are known constants. Instrumentation tools for obtaining actual link delays directly from the network itself would extend the practicality of the system.

The area of probabilistic performance is also promising. Based on probabilistic descriptions of delays for each network link, a probabilistic model might be used to calculate status delivery delay probabilities. The link specification would include a steady state availability probability indicating the fraction of time that the link is available. It would also include a link latency distribution characterizing the delay experienced on the link. The subscriber would submit its subscription request with a desired latency and the status dissemination middleware framework would compose the probability density functions for all links involved in the subscription path in order to calculate various probabilities. For example, the probability of missing no packets (steady state), one packet in a row, n packets in a row are probabilities that can be derived from these link functions, provided that the various probabilities are independent.

The value of event packing at the status routers is to be investigated further. The amount of bookkeeping and processing at each status router may outweigh the saving of resources. The existing status routers are designed for simplicity: they only know the next hop for each of the paths that pass through them. Multiple-hop packing would increase the complexity of the routers. Simplicity is important when considering status routers hardware implementations – something that could be of considerable benefit for improving performance and lowering cost.

QoS network spatial redundancy relies on the existence and discovery of node-disjoint multi-constrained paths. To the best of our knowledge, there are no algorithms that determine such paths in polynomial time.

The existing implementation does not support a directory service. Directory service is an essential part of any distributed system [26] to locate entities based on their attributes.

A complete fault tolerance framework for status dissemination middleware is to be designed, including replication of QoS brokers. Similarly, a security framework for status dissemination middleware, including investigation of trust management is the subject of current research.

A future research goal is to develop a stochastic model of the network interactions that occur

during status dissemination and analyze the performance of the communication network, under normal and anomalous operating conditions. In addition, the effect of optimization features such as message priority and cache extrapolation will be evaluated using this network model.

We emphasize that the current implementation of GridStat is at the early prototype stage. Future optimizations of the software and implementing the status routers in hardware will improve the results shown below. Hardware options include custom silicon, field-programmable gate arrays (FPGAs), and programmable network processors such as Intel's Internet Exchange Architecture (IXA).

8 Related Work

Several research efforts are exploring ideas similar to GridStat. The most similar work in the domain of status dissemination publish-subscribe systems with end-to-end QoS is PASS (Piecewise Asynchronous Sample Service) developed at BBN and Georgia Institute of Technology [29]. PASS is a service that provides information about the status of communication network resources, indicating their availability or not. The information that flows is in the form of name-value pairs encapsulated in records. PASS provides low-level status dissemination mechanisms, but does not provide the management or adaptation of the resources in the system. Furthermore, all subscribers have the same QoS parameters. It has been fielded in military exercises but is not presently applicable for the power grid.

Sienna is an Internet-scale event notification middleware framework [9]. It provides the ability to organize the communication network of status routers in a hierarchical, acyclic peer-to-peer or general peer-to-peer manner. Publishers and subscribers connect to one of these status routers, which becomes their access points. Sienna does not provide for any QoS properties since it is a best-effort service. The main difference between Sienna and GridStat is that the former is a general purpose content-based publish-subscribe where the focus is to maximize the expressiveness in the selection of status events while still being able to scale to wide-area networks, compared to the latter which limits the expressiveness in the selection to typed status variables but provides management and adaptation of QoS requirement.

The Spinglass project [6] at Cornell University is a gossip-based probabilistic atomic broadcast protocol. Gossip-based protocols can overcome scalability problems faced in large-scale systems. One of the optimization tools in the Spinglass project called gravitational gossip can be used as a mechanism for publish-subscribe middleware. It is designed to support large numbers of communication groups providing them a high probability of message delivery for a specified percentage of the total number of messages. It provides both security and reliability. Gravitational gossip pushes all the complexity to the multicast layer, which provides for high performance but reduces the ability to take advantage of the semantics of the data to drop unnecessary messages. Since the purpose of the Spinglass project was to provide probabilistic guarantees for large-scale systems, it inherently lacks the ability to provide to QoS guarantees like timeliness and delivery rate.

Xie et al. in [28], point out the major deficiencies in current communication and information systems and propose a new information architecture tailored for the power grid. They address the need for redundancy in various grid entities. GridStat middleware provides the tools needed to build robust, flexible, manageable and secure distributed status dissemination software over the low-level communication infrastructure such as proposed there.

9 Conclusions

New data dissemination requirements within the electric power grid have been mandated in order to provide wide-area controllers for the power grids. Wide-area controllers will provide better utilization of the transmission infrastructure, resulting in more efficient and stable electric power grids. Furthermore, recent deregulation marks the beginning of a new competitive electric power market, which will support many participants requiring status information delivered to them in order to perform their tasks successfully. The new services and optimizations require more quantity, timeliness and diversity of the information flow than is currently being provided. GridStat status dissemination middleware provides solutions in a flexible, manageable and secure manner. The goal is that grid operation engineers can utilize GridStat in power system applications without any further assistance, mainly due to its simple interface. Applications and application programmers do not need to be aware of the internal workings of GridStat. GridStat active entities can communicate status information with one another while relying on the communication infrastructure managed by GridStat for delivering Quality of Service guarantees.

Preliminary experimental results for the performance of a GridStat prototype characterize the behavior of the system without any optimizations. Status dissemination and status routers are new concepts that need to be understood before hardware can be effectively deployed. We are pursuing research opportunities employing hardware options mentioned in Section 7. We believe that such hardware, in conjunction with dedicated or reserved network links and realtime scheduling mechanisms in status routers, will allow delivery latencies of 3–10 milliseconds across a geographic range of hundreds of miles. This will enable much broader non-local control than is presently possible.

Information sharing within the electric power industry is only a small part of the entire picture. Critical infrastructures will have to exchange data and alerts so that terrorist attacks will be prevented. Status dissemination middleware is flexible so that it can be utilized for disseminating information within a specific critical infrastructure but also for communicating status between critical infrastructures. Therefore, we anticipate that status dissemination middleware frameworks will be used in other critical infrastructures and applications besides the electrical power grid.

Acknowledgments

The work was funded in part by the US Department of Commerce, National Institute of Standards and Technology Grant #60NANB1D0016 (Critical Infrastructure Protection Program), in a subcontract to Schweitzer Engineering Labs Inc, and by the National Science Foundation under Grant CCR-0326006. We thank A. David McKinnon, Kevin Tomsovic, Sudipto Bhowmik and Liz Wilhite for their valuable comments on this research. We thank Deb Frincke for her suggestions on the security requirements for GridStat.

References

[1] M. Adamiak, D. Baigent, and S. Evans. Practical considerations in application of UCA GOOSE. In *Georgia Tech Relay Conference*, Atlanta, USA, May 2000.

[2] M. Adamiak and W. Premerlani. The role of utility communications in a deregulated environment. In *Proceedings of the 32nd International Conference on System Sciences*, Maui, Hawaii, January 1999.

[3] M. Amin. Security challenges for the electricity infrastructure. *Special Issue of the IEEE Computer Magazine on Security and Privacy*, pages 8–10, April 2002.

[4] D. Bakken. Middleware. In P. Dasgupta and J. Urban, editors, *Encyclopedia of Distributed Computing*. Kluwer Academic Publishers, to appear 2003. Also available from http://gridstat.eecs.wsu.edu.

[5] D. Bakken, A. Bose, C. Hauser, I. Dionysiou, H. Gjermundrød, L. Xu, and S. Bhowmik. Towards for extensible and resilient real-time information dissemination for the electric power grid. In *Proceedings of Power Systems and Communications Systems for the Future, International Institute for Critical Infrastructures*, Beijing, China, September 2002.

[6] K. P. Birman, R. Renesse, and W. Vogels. Spinglass: Secure and scalable communications tools for mission-critical computing. In *International Survivability Conference and Exposition*, Anaheim, California, June 2001.

[7] A. Bose. Power system stability: New opportunities for control. In Derong Liv and Panos Antsaklis, editors, *Stability and Control of Dynamical Systems and Applications*. Birkhauser (Boston), 2003. Also available from http://gridstat.eecs.wsu.edu.

[8] A. Campbell and K. Nahrstedt (editors). *Building QoS into Distributed Systems*. Chapman and Hall on behalf of International Federation of Information Processing (IFIP), 1997.

[9] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Design and evaluation of a wide-area event notification service. *ACM Transactions on Computer Systems*, 19(3):332–383, August 2001.

[10] C. DeMarco. Opportunitites and perils in ubiquitous data availability for the open access envronment. Panel: Grand challenges in elecric power engineering, IEEE Power Engineering Society, Winter Meeting, January 30 2002.

[11] Federal Energy Regulatory Commission. Open access same-time information system and standards of conduct. Order 889, 1997.

[12] EPRI. Utility Communications Architecture (UCA) version 2.0. Technical Report 1550–1999, IEEE-SA, November 1999.

[13] EPRI. Smart power delivery – a vision for the future. *EPRI Journal Online*, June 9th 2003.

[14] EPRI-ISI. Infrastructure security issues. Project status report, EPRI, March 2003.

[15] P. Eugster, P. Felber, R. Guerraoui, and A.M. Kermarrec. The many faces of publish-subscribe. Technical Report DSC ID:2000104, EPFL Lausanne, January 2001.

[16] A. Farrell, L. Lave, and G. Morgan. Bolstering the security of the electric power system. *Issues In Science and Technology*, XVIII(3):49–56, Spring 2002.

[17] Barton Gellman. Cyber-attacks by Al Qaeda feared. *Washington Post*, June 27 2002.

[18] T. Grandison and M. Sloman. A survey of trust in internet applications. *IEEE Communications Surveys and Tutorials*, 4(4):2–16, 2000.

[19] T. Grandison and M. Sloman. Specifying and analyzing trust for internet applications. In *2nd IFIP Conference on e-Commerce, e-Business, e-Government*, Lisbon, October 2002.

[20] IETF Network Working Group. Integrated services in the internet architecture: an overview. RFC 1633, 1994.

[21] IETF Network Working Group. An architecture for Differentiated Service. RFC 2475, 1998.

[22] F.A. Kuipers, T. Korkmaz, M. Krunz, and P. Van Mieghem. A review of constraint-based routing algorithms. Technical report, Technical University Delft, The Netherlands, 2002.

[23] P. Oman and J. Roberts. Barriers to a wide-area trusted network early warning system for electric power disturbances. In *Proceedings of the 35th International Conference on System Sciences*, 2002.

[24] R. E. Schantz. Quality of service. In P. Dasgupta and J. Urban, editors, *Encyclopedia of Distributed Computing*. Kluwer Academic Publishers, to appear 2003.

[25] E. Terzi, Y. Zhong, B. Bhargava, Pankaj, and S. Madria. An algorithm for building user-role profiles in a trust environment. In *Data Warehousing and Knowledge Management Conference*, Aix en Provence, France, September 2002.

[26] P. Verissimo and L. Rodrigues. *Distributed Systems for System Architects*. Kluwer Academic Publishers, 2001.

[27] C. Wang, A. Carzaniga, D. Evans, and A.L. Wolf. Security issues and requirements for internet-scale publish-subscribe systems. In *Proceedings of the 35th IEEE Hawaii International Conference on System Sciences*, Big Island, Hawaii, January 2002.

[28] Z. Xie, G. Manimaran, V. Vittal, A.G. Phadke, and V. Centero. An information architecture for future power systems and its reliability analysis. In *IEEE Transactions on Power Systems*, volume 17, pages 857–863. August 2002.

[29] J. Zinky, L. O'Brien, D. Bakken, V. Krishnaswamy, and M. Ahamad. PASS: A service for efficient large scale dissemination of time varying data using CORBA. In *International Conference on Distributed Computing Systems*, pages 496–506, Austin, Texas, USA, June 1999.

Author Biography Briefs

Kjell Harald Gjermundrød is a PhD candidate in Computer Science in the School of Electrical Engineering and Computer Science at Washington State University (WSU). His research interests include distributed systems, message-oriented middleware and concurrent programming.

Ioanna Dionysiou is a PhD candidate in Computer Science in the School of Electrical Engineering and Computer Science at Washington State University (WSU). Her research interests include distributed systems, security and trust in publish-subscribe models.

David E. Bakken is an Assistant Professor in the School of Electrical Engineering and Computer Science at Washington State University (WSU). His research interests include middleware, fault tolerance, and quality of service frameworks. Prior to joining WSU, he was a scientist at BBN, where he was an original co-architect and PI on the Quality Objects (QuO) framework.

Carl Hauser is an Associate Professor in the School of Electrical Engineering and Computer Science at Washington State University (WSU). His research interests include distributed systems, networking, concurrent programming models and mechanisms, and programming language implementations. Prior to joining WSU he was at Xerox Palo Alto Research Center for 17 years and at IBM San Jose Research Laboratory (now IBM Almaden Research Laboratory) for 4 years.

Anjan Bose is a Distinguished Professor in Power Engineering in the School of Electrical Engineering and Computer Science at Washington State University (WSU). He is a member of the National Academy of Engineering for his research on the electric power grid and is an IEEE Fellow. His research interests include a wide variety of issues involving the operation, control, and simulation of the electric power grid.