Adaptive Information Flow Mechanisms and Management for Power Grid Contingencies

Stian F. Abelsen[*], Erlend S. Viddal[†], K. Harald Gjermundrød[‡], David E. Bakken[§] and Carl H. Hauser
Washington State University
School of Electrical Engineering and Computer Science
Pullman, Washington, USA
{stian.abelsen, eviddal, haraldg}@gmail.com, {bakken, chauser}@wsu.edu

Abstract

GridStat is a QoS-managed publish-subscribe framework for data delivery for the electric power grid. GridStat's Data Plane delivers data updates through a network of middleware-level Status Routers. Subscriptions are managed by GridStat's hierarchical QoS Management Plane. The path allocation computations are typically done offline and beforehand, but are complex, not only due to the multiple QoS constraints but the number of status routers that would be involved an entire power grid. In a crisis, many entities may suddenly wish to add a large number of subscription requests, which would in practice overwhelm the subscription allocation mechanisms. In this paper we present a mechanism called modes, which lets GridStat change routing tables quickly. Modes can be either global, or only active at a given scope within the hierarchy. We present the design and experimental evaluation of GridStat modes and of two different mode change algorithms that provide different tradeoffs of performance and consistency.

1 Introduction

The electrical power grid is highly dependent on data monitoring and control capabilities in order to better understand and manage power transmissions over a highly complex network of transmission lines and substations. SCADA (Supervisory Control and Data Access) has in the last 40 years served as the electrical power grid's communication system and incorporates the requirements and network technologies from the time that it was developed. The requirements for communication in the electrical power grid are changing [7]. Growing concerns about terrorist attacks, changes in the power flow structure after the regulatory restructuring in the 1990's, new uses of technologies (IntelliGrid [3]) and an increased overall load to capacity ratio of the transmission line system demand a more flexible and adaptive communication network. The SCADA communication system features a centralized star-topology, point to point communication, lack of multicast, severe bandwidth constraints and proprietary protocols which are not sufficient to meet the requirements of today's grid. Further discussion of the limitations of SCADA can be found in [6] and [2].

GridStat is designed to address the need for a flexible and robust communication system for the electrical power grid using a specialized publish-subscribe paradigm [5]. GridStat middleware manages network resources, enables reliable delivery of data to any point and provides QoS (Quality of Service) for data streams. GridStat hides the details of lower-level network capabilities from application developers in order to enable the communication system to be deployed across different network technologies, operating systems, programming languages and device types. GridStat is divided into two planes; the management plane and the data plane. The management plane consists of a hierarchy of QoS brokers that collectively manage resources and subscriptions in the data plane. The data plane is a virtual message bus that lets publishers provide data to the network and enables subscribers to establish subscriptions to status data through a status router network. The use of QoS, on a per-subscription basis, allows subscribers to specify multiple redundant delivery paths (spatial redundancy), subscription interval and delay. Furthermore, GridStat provides status data delivery to multiple recipients at different rates through the multicast property and the ability to control and switch routing tables in the status router network at run-time through the use of *modes*.

[*]Current affiliation: Eltek Valere, Drammen Norway
[†]Current affiliation: Simula Innovation, Oslo Norway
[‡]Current affiliation: University of Cyprus, Nicosia Cyprus
[§]Contact author

A mode contains the necessary forwarding rules for a set of subscriptions. Using pre-configured modes allows the status router network to quickly switch between bundles of subscriptions, an action called a *mode change*. The process of establishing individual subscriptions is a resource-intensive operation requiring deallocation and allocation of subscriptions. Doing these computations at run-time is expensive and may result in unsatisfactory subscription delays [8]. Modes allow subscription bundles to be allocated and pre-loaded into the status routers' routing tables and then control which routing tables the status router network utilizes at any given time.

The mode change mechanism will help utility companies' control centers, regional control centers, ISOs and nation-wide monitoring centers to perform pre-contingency planning for communication needs and to switch subscription bundles when contingencies do occur in the electrical power grid. For example, modes are one way to exploit GridStat's capability for data load shedding in the communication infrastructure in analogy with the power load shedding in the electrical power grid. For example, subscribers could specify two QoS sets: the desired QoS and minimal acceptable QoS, and the management plane can automatically switch between them when the network is congested.

The research contributions of this paper are:

- Global and hierarchical modes: QoS brokers define and use modes to adapt communication in their respective administrative domains.
- Multiple simultaneously active routing tables in the data plane and the ability to switch between routing tables at run-time.
- Design and implementation of two mode change algorithms with different tradeoffs.
- Experimental evaluation comparing the mode change algorithms in terms of performance, resource usage and variance (time) in the presence of various temporary network conditions.

2 Status Dissemination and GridStat

GridStat is a publisher-subscriber framework that targets application domains where the majority of data are made available at periodic time intervals. It is mainly designed to serve as a flexible and robust communication system for the electrical power grid. Figure 1 shows a small scale GridStat deployment subdivided in a management plane and a data plane. The management plane consists of *QoS broker* modules that collectively control and manage resources in the data plane. The data plane is populated by *status routers*, *publishers* and *subscribers*, where publishers provide data and subscribers can subscribe to data. The management hierarchy handles subscription requests and establishes paths from the publisher to the subscriber through a sequence of status routers. More detailed information about GridStat and other baseline mechanisms can be found in [4] and [6].

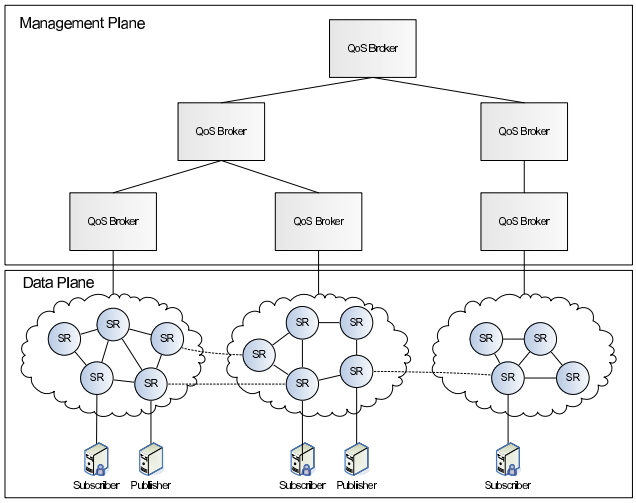

Figure 1. Status dissemination architecture

2.1 Management Plane

The lowest level of the management plane consists of *leaf QoS brokers*. A leaf QoS broker manages and provides services to a set of status routers, publishers and subscribers. The leaf QoS broker manages a flat collection of status routers, called a *cloud*. A leaf QoS broker has complete control over all available resources and the corresponding resource usage in the cloud. The resources include status routers, publishers, subscribers and the links connecting them. Leaf QoS brokers must control and make sure no allocated subscriptions exceed an event channel's bandwidth constraints. Additionally, the leaf QoS brokers must ensure that routing tables and computational resources are not overloaded in the status routers. The main responsibility of a leaf QoS broker is to control the allocation or deallocation of subscription paths between publisher-subscriber pairs in its cloud, and to ensure that the allocated path satisfies the QoS requirements specified by the subscriber.

Interior QoS brokers denote all non-leaf QoS brokers in the management hierarchy. Interior QoS brokers manage multiple clouds and offer services to lower-level QoS brokers, and whose main responsibility is to allocate and deallocate inter-cloud subscriptions.

2.2 Data Plane

The data plane is a virtual message bus where subscription data flows between publishers and subscribers. The virtual message bus consists of interconnected status routers, whose main purpose is to forward status events from publishers to the subscriber applications that requested the data. A status router is in effect a router with additional functionality to provide forwarding of status events when subscribed to and at the right rate (*rate filtering*). The management plane controls the content of the routing tables in the status routers, and leaf QoS brokers inform status routers to add, remove or modify the content corresponding to a subscription allocation or deallocation request.

Since resources are monitored and controlled, the latency from the publisher to the subscriber can be bounded. When registering a subscription, subscribers associate a set of QoS parameters with the subscription request, and among these are a subscription interval, a latency request and redundancy. The management hierarchy attempts to find one or more disjoint paths (QoS redundancy) between the publisher and subscriber that meet the latency request parameter. If no such path exists, the subscription request is rejected.

3 Mode Change Mechanisms and Management

In this section we present the foundation for global and hierarchical modes and introduce the notion of a *mode change*.

3.1 Overview and Mode Terminology

A *mode definition* consists of an ID, a name and a set of data plane subscriptions. Each mode definition is owned by a single QoS broker in the management hierarchy. Modes defined and owned by a QoS broker constitute a *mode set*. Exactly one of the modes in a mode set is active at any given time and is called the current *operating mode* for that mode set. Every QoS broker always operates in one mode. When a QoS broker operates in a given mode all subscriptions contained in that mode are active in the data plane.

Status routers use modes to route status events that belong to the currently active set of operating modes. More specifically, a status router forwards a status event if it belongs to a subscription that is in the operating mode of at least one QoS broker. Each status router operates in as many modes as there are levels in the management hierarchy above it. For instance, with x levels in the management hierarchy, a status router has x QoS broker ancestors (called its *ancestor scope*), and will therefore always operate in x modes simultaneously. This enables coarse resource provisioning between mode sets, e.g. a top level QoS broker could control 40% of the available resources in its hierarchical scope while the QoS brokers beneath it would control the remaining 60% of the resources.

Each status router maintains a separate routing table for every mode defined in its ancestor scope. For example, in a GridStat configuration with a management hierarchy consisting of two levels, every status router will always operate in two modes, and therefore use two separate routing tables for routing (see Figure 2). Although a status router might have tens or even hundreds of routing tables pre-loaded, either in memory or on disk, only the routing tables that correspond to the current operating modes of the QoS brokers are used for status event forwarding.

3.2 Propagation of Modes

QoS brokers read mode definitions from their respective configuration files and store the modes internally. Each leaf QoS broker also acquires information about the modes that are defined in its ancestor scope. The leaf QoS broker requests all the modes that are defined in its ancestor scope by contacting its parent QoS broker, which recursively repeats the process until the request reaches the root QoS broker. When the request returns, interior QoS brokers add their respective mode identifiers and current operating mode to the return packet. The leaf QoS broker stores the information returned from the request for each ancestor QoS broker. The ancestor mode set is used to inform status routers about all defined modes and operating modes they will operate in. The leaf QoS broker is responsible for updating the ancestor mode set during mode change operations to reflect the operating modes of its parent QoS broker.

A subscriber application is informed about all modes that are defined in its ancestor scope when it registers with its edge status router and later when changes occur. Knowing the set of modes allows a subscriber application to select modes for its subscriptions and include this as part of its subscription requests to the management hierarchy.

3.3 Mode Change Operations

Using modes, a QoS broker can quickly switch routing tables in the data plane. This enables the management plane to decide which status events are allowed to be forwarded through the data plane at any given time and delivered to the subscriber applications. Figure 2, shows how operating modes are used in both the management plane and in the data plane. Each QoS broker has its own mode set and operating mode. All status routers in cloud *B* operate in Green and Stable as the QoS brokers in their ancestor scope, QoS broker A and B, operate in Green and Stable, respectively.

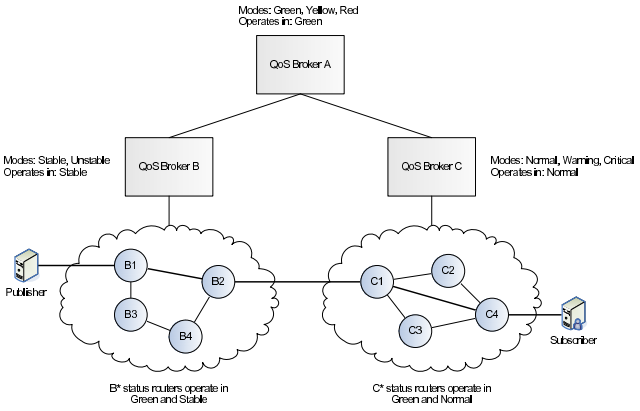

Figure 2. Status dissemination architecture.

Inter-cloud subscriptions can operate in one or more of QoS broker A's three modes Green, Yellow and Red since those modes control and manage routing tables in both clouds. Note that an inter-cloud subscription is unaffected by the modes in which leaf QoS broker B and C operate. If a subscription is configured to operate in mode Yellow only, status router B1 does not forward status events to B2 as it is using mode Green's routing table which contains no forwarding information for the subscription.

A QoS broker acts as a *coordinator* in a mode change operation. A mode change operation that is *initiated* by a coordinator affects all the status routers and QoS brokers in its hierarchical scope. For example, the hierarchical scope of QoS broker A is QoS broker B, cloud B, QoS broker C and cloud C. A mode change operation informs all status routers in the hierarchical scope of the coordinator to switch to the routing table associated with the operation. The number of status routers involved in a mode change operation varies with the population of status routers in the affected clouds and at what level in the management hierarchy the mode change operation was initiated. Local mode change operations (within a cloud) might involve tens or hundreds of status routers, while a mode change operation initiated at an interior QoS broker can potentially involve several thousand status routers. One of the major challenges is to ensure that all the status routers involved in a mode change operation receive and switch to the proper routing table.

A mode change operation that switches the routing tables in all of the involved status routers at the expected time is called a *consistent* mode change operation. Otherwise the operation is called an inconsistent mode change operation. Additional recovery mechanisms (see Section 3.5) must be used in order to restore the operating modes on the status routers that are inconsistent. An inconsistent mode change operation will most likely result in QoS violations for some subscriptions, which will not occur after a consistent mode change operation. However, since subscription traffic is rate-based, the loss of some status events during a mode change operation is tolerable because the next status update value for a particular subscription is due to arrive within a short time period.

Two mode change algorithms have been implemented in GridStat. The *hierarchical mode change* algorithm uses the management hierarchy to disseminate mode change operations and gather acknowledgements from status routers and QoS brokers. The hierarchical mode change algorithm enables all subscriptions that are registered to operate in both the coordinator's current and new modes to flow during the entire mode change. Thus, subscribers with subscriptions in both the current and new modes receive an uninterrupted stream of status events during hierarchical mode change operations. The hierarchical mode change algorithm is discussed in more detail in Section 4.

The flooding mode change algorithm disseminates mode change operations directly on the data plane using GridStat's flooding mechanism. When a status router receives a mode change operation it forwards it on all outgoing event channels, except the event channel on which it received the operation. Redundant copies are discarded. Mode change operations in this scheme carry a target time (in the future) for the mode change to occur. The flooding mode change algorithm is discussed in more detail in Section 5.

The two mode change algorithms provide different tradeoffs. The hierarchical mode change algorithm is a resource intensive algorithm using five message phases in order to enable subscriptions present in both the *before mode* and *after mode*, e.g., from Green to Yellow, to deliver status updates throughout the transition. In each message phase the coordinator initiates and propagates a mode change phase message down to all status routers in its hierarchical scope. The next phase cannot be initiated until the previous phase has completed which occurs when the coordinator has received aggregated acknowledgments from all status routers and QoS brokers in its scope. The flooding mode change algorithm, on the other hand, is a best-effort algorithm. It is efficient, in terms of resource use and performance, but it does not guarantee that all subscriptions will flow during the mode change operation. The flooding mode change algorithm directs the status routers to switch modes at a given time and therefore requires status routers to be time synchronized.

3.4 The RPC Mechanism

Communication related to mode change operations utilizes some of the unique features of GridStat's RPC mechanism [11]:

- Mode change communication over RPC within the management hierarchy or from the management hierarchy to the data plane can utilize spatial redundancy.

The degree of spatial redundancy is itself adjustable using the mode mechanism.

- RPC delivery confirmations provide the means for the client to resend a mode change message, or schedule one for a later time, when a delivery confirmation is not received within the expected time window.

An RPC connection can be configured to resend the call when a delivery confirmation is not received within the expected time window. More specifically, the RPC mechanism resends the call after a preconfigured timeout and employs the temporal redundancy scheme as many times as the connection setup states.

3.5 Recovery Mechanisms and Acknowledgment Aggregation

In order to tolerate some degree of network failures and to ensure that mode change operations are eventually consistent, a recovery mechanism was implemented to assist the hierarchical and flooding mode change algorithms. The recovery mechanism is triggered by a QoS broker that detects missing mode change acknowledgments caused by failed QoS brokers, failed status routers or link failures. It then attempts to resolve the inconsistency.

An important part of a mode change operation is to gather acknowledgments from the participants in the operation. A status router that receives a mode change operation responds with a mode change acknowledgment to its leaf QoS broker. When a leaf QoS broker receives the first acknowledgment for a particular mode change operation it immediately starts an aggregation round to gather acknowledgments from its status routers. The leaf QoS broker stores the name of the status router and the mode change identifier, and starts a timer. Additional acknowledgments are registered in a similar manner. The leaf QoS broker stops the aggregation round when all expected acknowledgments have been received or if the timer expires. If all expected acknowledgements have been received and the leaf QoS broker was itself the initiator of the mode change it updates the operating modes table in its state and marks the mode change operation as complete. If the coordinator is located at a higher level in the management hierarchy, the leaf QoS broker updates its ancestor modes table and sends an acknowledgment to its parent QoS broker for further processing.

Interior QoS brokers go through the exact same sequence of steps as the leaf QoS brokers, starting aggregation rounds when the first acknowledgment from one of their direct children QoS brokers is received. The process continues until the coordinator of the mode change operation has finalized the aggregation round.

When an aggregation round times out, a QoS broker initiates the recovery mechanism by storing the mode change operation and then, in collaboration with its children QoS brokers, continually attempts to restore the state of the deemed inconsistent status routers or QoS brokers. More details on the recovery mechanism can be found in [1].

4 Hierarchical Mode Change Algorithm

Recall that a main goal of the hierarchical mode change algorithm is to ensure that subscriptions that exist in both the *before mode* and *after mode* continue to meet their QoS delivery promises throughout the mode transition. This involves ensuring that routing tables in the status routers continue to forward status events for these subscriptions and also ensuring that no links are overloaded with traffic during the transition. The hierarchical mode change algorithm is divided into five distinct phases. The five phases for a mode change initiated by QoS broker A from Green to Yellow are:

1. **The inform phase** - All edge status routers, at the direction of QoS broker A, inform their subscribers about the upcoming mode change. This phase ensures that all subscribers know that QoS violations may occur for some of their subscriptions prior to switching routing tables.

2. **The prepare phase** - Edge status routers switch to the temporary routing table Green $\cap$ Yellow. The higher subscription interval (lower rate) of each subscription in the intersection is used in this phase in order to reduce the load on downstream status routers. This phase ensures that subscription traffic that belongs in both modes (Green and Yellow) is forwarded through the status router network. Subscription traffic that belongs to only Green or Yellow is dropped at the edge status routers. This step in the hierarchical mode change algorithm eliminates any possibility of link overload during the transition.

3. **The internal change phase** - Internal status routers switch to Yellow's routing table. Since all edge status routers operate in a temporary routing table and only forward a smaller set of subscriptions (in mode Green and Yellow), the internal status routers can safely switch to mode Yellow without overloading any status routers downstream.

4. **The edge change phase** - Edge status routers switch from the temporary routing table Green $\cap$ Yellow to Yellow's routing table. Since internal status routers operate in Yellow and expect to receive subscription traffic for mode Yellow, it is safe for edge status routers to finally switch.

5. **The commit phase** - Edge status routers inform their subscribers that the mode change is complete. This phase ensures that subscribers have correct expectations regarding their subscriptions' quality of service.

Common for all the phases in the hierarchical mode change algorithm is the propagation of the operation down the management hierarchy towards the data plane. The coordinator of the mode change operation sends the operation to all its children QoS brokers, and they repeat the process until the operation reaches the leaf QoS brokers. The leaf QoS brokers forward the mode change operation to each individual status router and await acknowledgements (see Section 3.5). When a leaf QoS broker has received acknowledgment from all status routers in its cloud, it forwards a single acknowledgment to its parent QoS broker, which then repeats the process until all acknowledgements have been gathered by the coordinator. The coordinator proceeds to initiate the next phase of the hierarchical mode change algorithm. More details on the hierarchical mode change algorithm can be found in [1].

5 Flooding Mode Change Algorithm

The flooding mode change algorithm is an alternative to the hierarchical mode change algorithm that switches modes more quickly but with greater chance that delivery will be interrupted during a transition. As previously described, the flooding mode change algorithm delivers mode change operations directly to the status routers using GridStat's flooding mechanism. In order to utilize the flooding mechanism, each QoS contains a publisher instance that connects to some edge status router in the QoS broker's hierarchical scope. The QoS broker publishes mode change operations through the publisher instance and status routers forward the operation to all their status router neighbors. The flooding eventually stops when all status routers have been informed. The flooding mechanism benefits from the redundancy in the data plane and is thus more resilient to network failures than is the hierarchical mode change algorithm. Whereas the hierarchical mode change algorithm attempts to preserve the subscriptions registered in the two involved modes, the flooding mode change algorithm switches directly to the new mode. That is, upon receiving a mode change operation through the flooding mechanism, a status router immediately responds with an acknowledgment to its leaf QoS broker and it activates the new mode at the destined future time. Figure 3 shows a flooding mode change initiated by QoS broker A. The operation is delivered to all participants within five message rounds. The diagram assumes an equal link delay, and the link labels refer to the message round in which the operation is transmitted across the link.

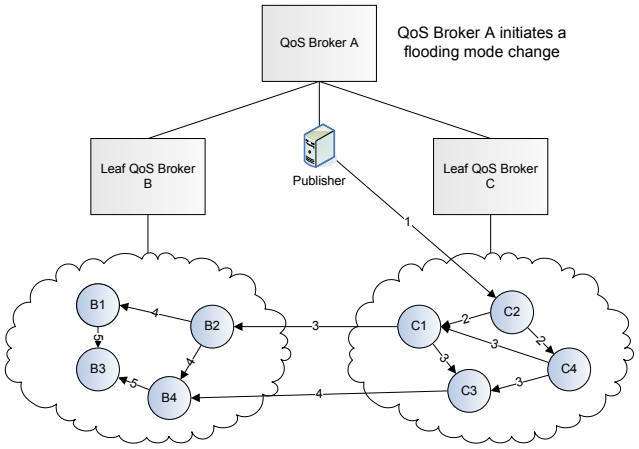

Figure 3. The flooding mechanism.

More details on the flooding mode change algorithm can be found in [1].

6 Experimental Evaluation

The following sections describe a subset of the experimental evaluation that we conducted. Refer to [1] for a more in-depth analysis. The experiments were conducted on a 14-node cluster. The hardware and software consisted of:

- 14 Intel Dual Xeon 3.06 GHz, 1 GB of RAM and 1 Gb network interface running Redhat 9 (2.4.20-8smp kernel).
- 1 Intel Pentium III (Coppermine) 1 GHz, 512 MB of RAM and 100Mb network interface running Ubuntu 6.10 (Edgy) Linux Distribution (2.6.17.10 kernel).
- Java Standard Edition 5.0 (build 1.5.0 11-b03).

The cluster nodes were used to run all the GridStat entities necessary to conduct the various experiments. The Ubuntu system ran a link emulator that provided controlled link latency and link loss on a per-link basis for the data plane links.

6.1 GridStat Settings

Figure 4 and Figure 5 show the GridStat experimental setup with 7 QoS brokers and 20 status routers for the hierarchical and flooding mode change algorithms, respectively. Each cloud consists of five status routers: three edge status routers and two internal status routers. The data plane executes on a total of 8 nodes; in each cloud, two and three status routers share a node. The management hierarchy executes on a total of 6 nodes; *A* and *B1* share a node, and every other QoS broker executes alone on the remaining nodes.

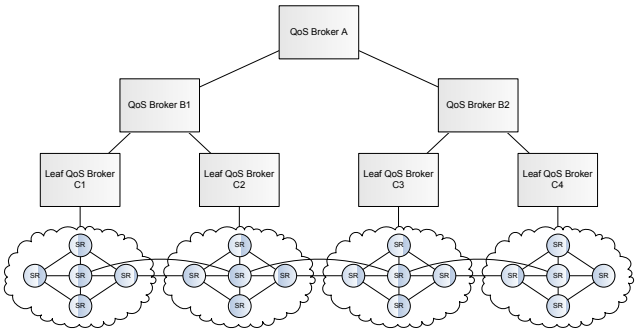

Figure 4. GridStat experimental setup - hierarchical algorithm.

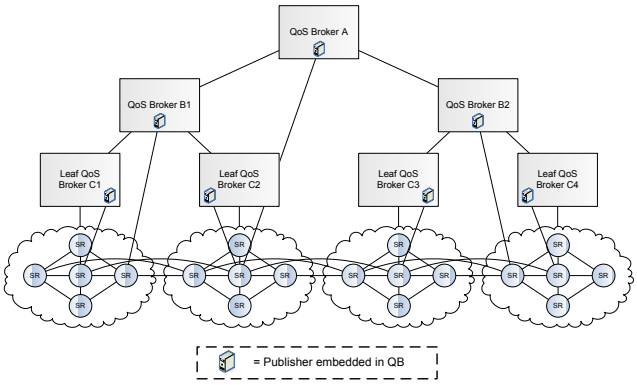

Figure 5. GridStat experimental setup - flooding algorithm.

QoS brokers are configured to communicate with other QoS brokers, and leaf QoS brokers with status routers, through dedicated RPC connections. RPC connections between leaf QoS brokers and status routers utilize two redundant paths, while inter-QoS broker RPC connections utilize one path only due to a current limitation in GridStat. Additionally, whenever a GridStat entity does not receive an RPC acknowledgment after *some* timeout, it resends the RPC call. The RPC retry timeout value depends on the experimental setup as the round trip time for the call and acknowledgment depend on the link latency and the number of event channel hops (Table 1).

The link emulator associates each link with a latency, a probability of packet loss and a burstiness setting. When a packet loss is triggered, the link consecutively loses as many packets as the burstiness setting suggests. If the burstiness setting is variable, e.g. 3-5, the link will at a minimum lose 3 consecutive packets, but no more than 5 (the actual number is subject to a uniform distribution). Furthermore, the probability of triggering a packet loss is adjusted to the desired packet loss probability setting divided by the mean

Link Latency	Top Level	Second Level	Leaf Level
0 ms	10	10	10
1 ms	15	15	15
2 ms	30	25	20
4 ms	60	40	30
8 ms	120	70	55

Table 1. Experiment RPC retry timeouts.

burstiness setting. For example, with a packet loss probability setting of 8% and a burstiness setting of 3-5, the trigger probability is 8% / 4 = 2%. It is important to note that a link will not trigger a new packet loss when in the middle of an ongoing loss sequence, and that edge links, those from publishers or subscribers to edge status routers, do not lose packets.

6.2 Algorithm Comparison

Figure 6 depicts the average completion time (averaged over 100 operations) for hierarchical mode change operations activated from the root QoS broker in the management hierarchy. The y-axis denotes the average time in ms and

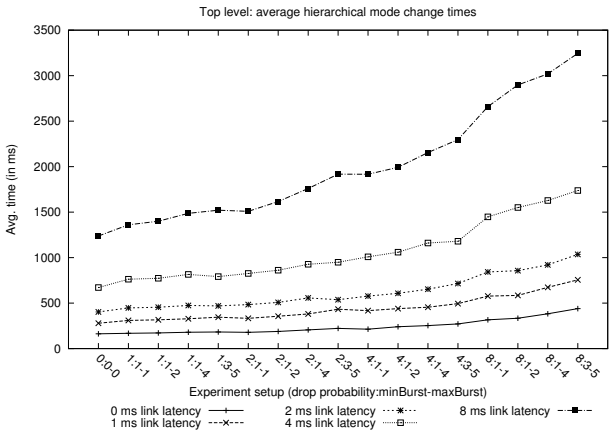

Figure 6. Hierarchical mode change operations at the top level in the management hierarchy.

the x-axis denotes the experiment setup in the form *per link packet loss rate: minimum consecutive packet loss - maximum consecutive packet loss*.

As expected, the mode change times increase when the per link probability of packet loss increases. When increasing the link latency, the increase in mode change completion times are more notable, which correlates to higher RPC latencies and RPC retry timeout values.

Figure 7 depicts the average completion time (averaged over 100 operations) for mode change operations activated from the leaf level in the management hierarchy. The experiments conducted at the various link latency settings show a relatively flat trend, which suggests that the two redundant paths utilized by the RPC connections between the leaf QoS broker and its status routers are able to withstand link loss up to 4% without any significant impact on the mode change times. An 8% link loss setting increases the average mode change times which clearly illustrates the impact of having only two redundant paths between the leaf QoS broker and its status routers. Figure 7 also shows the benefit of having redundant paths down to each individual status router, whereas the experiments in Figure 6 are more prone to link loss as inter-QoS broker communication is conducted over one communication path.

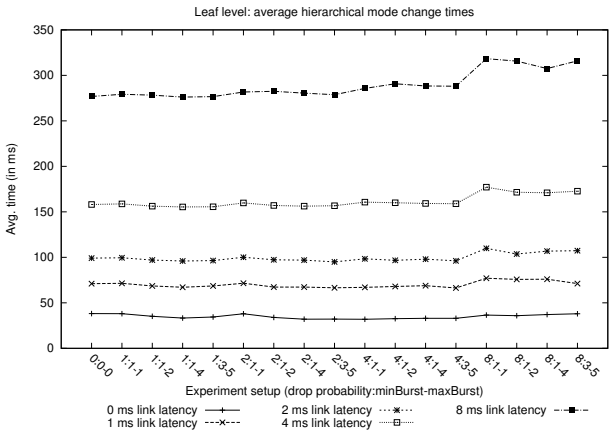

Figure 7. Hierarchical mode change operations at the leaf level in the management hierarchy.

Figure 8 shows the average completion time (averaged over 300 operations) for flooded mode change operations activated from the root in the management hierarchy. As mode change operations are disseminated directly onto the data plane, the five graphs, one per latency setting, illustrate how resilient the flooded mode change algorithm is against lossy links with various burstiness settings. With a link latency setting set to 8 ms, the flooded mode change reaches all the target status routers after approximately 45 ms, whereas the hierarchical algorithm requires 1200-3200 ms (Figure 6) depending on the link loss and burstiness setting. The experiments conducted with 8% link loss passes a threshold where the redundancy available in the data plane is not able to propagate the mode change message to the farthest away status router according to the best path, or close to the best path, so the average mode change time increases.

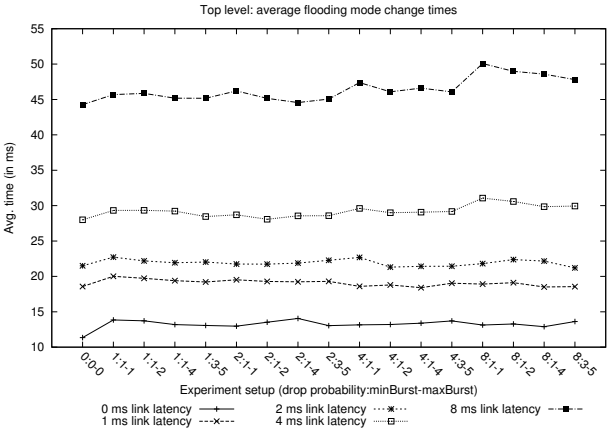

Figure 8. Flooding mode change operations at the top level in the management hierarchy.

6.3 Scalability Results

The following diagrams compare the same experiments conducted at the three levels in the management hierarchy in order to see how the algorithms scale when increasing the hierarchical scope of the hierarchical algorithm or the flooding domain of the flooding algorithm.

Figure 9 depicts experiments conducted at the three levels in the management hierarchy with 8 ms link latency by using the hierarchical algorithm. With no link loss, the diagram shows an increase in time between the experiments conducted at the leaf and second level which is approximately double the mode change times achieved at the leaf level, and the same results are seen between the experi-

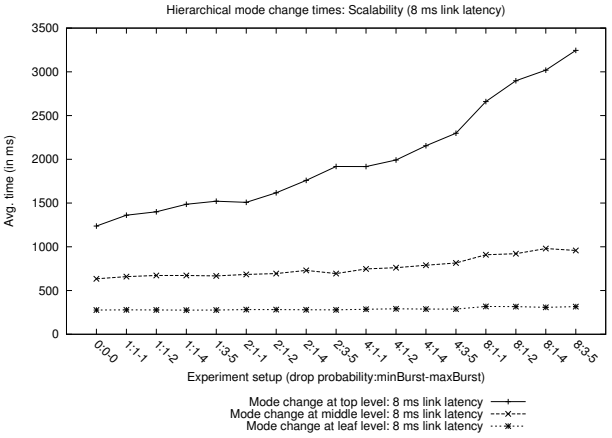

Figure 9. Scalability results using the hierarchical algorithm (8 ms link latency).

ments conducted at the second and top level. With increasing link loss settings, the mode change completion times for higher-level QoS broker activators increase, which illustrates the current limitation of supporting single inter-QoS broker communication paths. The results are expected to improve dramatically when inter-QoS broker communication paths support spatial redundancy.

The hierarchical mode change algorithm depends on the number of levels in the management hierarchy, the *length* of the RPC connections between the QoS brokers and the corresponding delays. The experimental layout for the hierarchical mode change algorithm in Figure 4 shows a binary management tree, where the lengths of the RPC connections are: 5 links between A and B1, 4 links between B1 and C1 and 3 links between C1 and each status router, and suggests that the paths become longer higher up in the management hierarchy. However, the lengths of RPC connections depend on the topology of the data plane and at which status router the QoS broker's publisher and subscriber connect to the data plane.

Figure 10 depicts experiments conducted using the flooding algorithm initiated at three different levels in the management hierarchy assuming 8 ms link latency. It

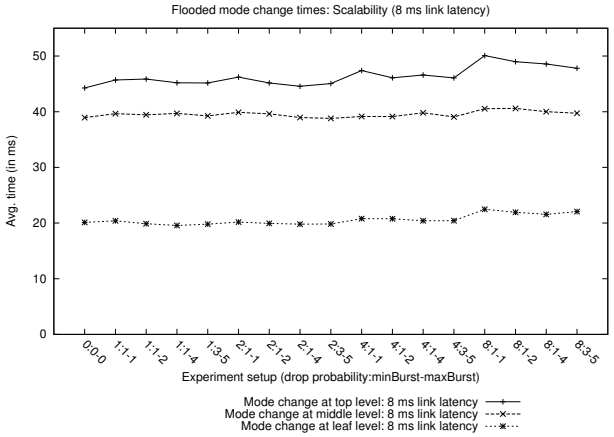

Figure 10. Scalability results using the flooding algorithm (8 ms link latency).

presents the average time at which all status routers have received the mode change operation and gives a notion of how much time should be allocated for flooding a mode change operation to all status router participants. The GridStat topology in Figure 5 and the results from Figure 10 show that flooding mode change operations initiated by the top level QoS broker require approximately 50 ms to deliver the operation to all status router participants, even under difficult network conditions.

These experiments show an increase in mode change completion times, caused by larger flooding domains, when the coordinator resides higher up in the management hierarchy. A larger flooding domain increases the number of message rounds for the mode change operation to reach all status router participants. Another factor is the starting point of the flooding mechanism, e.g., flooding from the *middle* of the flooding domain is more efficient than initiating a flood from an edge in the flooding domain. An example of this is shown between the experiments conducted at the leaf and second level. Flooding mode change operations activated from the second level initiate the flooding mechanism from an edge in the flooding domain, while the leaf QoS broker initiates the flooding mechanism from the center status router in its single administrative cloud. This, in effect, means that the leaf level requires two message rounds to disseminate the operation to all status router participants, while the second level requires four message rounds.

6.4 Link Traversals

Table 2 shows the number of links traversed by both mode change algorithms. The number of traversals is averaged over all the experiments conducted at a specific level in the management hierarchy. For example, the hierarchical algorithm at the top level traverses a total of 2124 event channels, averaged over all experiments conducted at that level. The hierarchical mode change algorithm requires more link traversals since the coordinator disseminates mode change operations through the management hierarchy and towards the data plane. The flooding mode change algorithm, on the other hand, saves a trip through the management hierarchy and uses it only for aggregating acknowledgements up towards the coordinator.

Algorithm	Top Level	Second Level	Leaf Level
Hierarchical	2124	833	340
Flooding	367	165	84

Table 2. Link traversals

7 Related Work

Given that both GridStat and the global and hierarchical mode change mechanism address novel problems there is little closely related work. One loosely related work is [10], which develops the principle of *Selective Notification* (SN). The underlying assumptions of this work are very different from that of GridStat (GS) and this mode change work. In terms of the manangement relationship, GS assumes that this is largely static while SN assumes it is highly dynamic. This allows GridStat to optimize data delivery and also switch many subscriptions at once, while SN is considering

more general issues involving dynamically (re)establishing the management relationship of a given node.

Previous work has been done in network routing using multiple simultaneously-active routing tables for differentiated QoS routing [9, 12]. More specifically, two routing tables are used; one for QoS traffic and another for best-effort traffic. This allows for differentiated routing strategies for the two traffic classes. For example, QoS traffic requiring lower drop rates could be forwarded along less-loaded paths reducing the probability of drops due to congestion at the expense of longer paths and thus higher delay. The mechanisms proposed in this paper share the property of previous work in the area of supporting multiple routing tables, but differ in that a status router is collectively managed by a set of QoS brokers where each QoS broker controls a distinct set of routing tables, and has the ability to switch between the routing tables at run-time.

8 Conclusions

This paper has presented an implementation of global and hierarchical mode change mechanisms and management in GridStat, which provides a novel way to switch between bundles of subscriptions at run-time. The power grid industry can benefit from this mechanism in GridStat by identifying and creating distinct subscription sets to provide specifically needed information during various critical situations in the electrical power grid. Furthermore, GridStat's modes implementation enables load shedding for data streams and corresponds to how load shedding enables transmission adjustments in the electrical power grid.

Two mode change algorithms that offer different tradeoffs with respect to consistency, resource usage and speed have been presented. The hierarchical mode change algorithm enables subscriptions present in both the old and new modes to flow during a mode change operation through five execution phases. In addition, the hierarchical mode change algorithm prevents bandwidth, status router computational resource and queue overloads. The flooding mode change algorithm is an efficient, best-effort algorithm that informs status routers to change modes through the flooding mechanism. However, the flooding mode change algorithm is not able to prevent overloads and does not guarantee continuous delivery during mode change operations.

The experimental evaluation addressed the mechanisms' performance under various network conditions. The results show that the hierarchical mode change algorithm scales linearly when increasing the hierarchical scope of a mode change operation. However, the algorithm adds a significant delay to the overall mode change completion time in the presence of link loss. The reason for this behavior is that QoS brokers do not utilize redundant subscription paths in their established RPC connections. The hierarchical mode change algorithm is expected to perform much better during poor network conditions when GridStat supports redundant communication paths in the management hierarchy. The results show that the flooding mode change algorithm completes a mode change operation more than an order of magnitude faster than the hierarchical mode change algorithm. Furthermore, the flooding mode change algorithm is less affected by link loss showing only minimal change in its mode change completion times.

9 Acknowledgements

This work was funded in part by the National Science Foundation through the CNS 05-24695 (CT-CS: Trustworthy Cyber Infrastructure for the Power Grid(TCIP)) grant.

References

[1] S. F. Abelsen. Adaptive gridstat information flow mechanisms and management for power grid contingencies. Master's thesis, Washington State University, June 2007.

[2] D. E. Bakken, C. H. Hauser, H. Gjermundrød, and A. Bose. Towards more flexible and robust data delivery for monitoring and control of the electric power grid. Technical report, TR-GS-009, School of Electrical Engineering and Computer Science, Washington State University, May 2007.

[3] EPRI/CEIDS. The integrated energy and communication systems architecture, Volumes I-IV, July 2004.

[4] K. H. Gjermundrød. *Flexible QoS-managed status dissemination framework for the electrical power grid*. PhD thesis, Washington State University, 2006.

[5] K. H. Gjermundrød, D. E. Bakken, C. H. Hauser, and A. Bose. GridStat: A flexible QoS-managed data dissemination framework for the power grid. *IEEE Transactions on Power Delivery*, 2008. To appear.

[6] C. Hauser, D. E. Bakken, and A. Bose. A failure to communicate: next generation communication requirements, technologies, and architecture for the electric power grid. *Power and Energy Magazine, IEEE*, 3:47–55, March-April 2005.

[7] C. H. Hauser, D. E. Bakken, I. Dionysiou, K. H. Gjermundrød, V. ta S. Irava, J. Helkey, and A. Bose. Security, trust, and QoS in next-generation control and communication for large power syst ems. *International Journal of Critical Infrastructures (Inderscience)*, 4(1/2), 2008. To appear.

[8] V. Irava. *Low-cost delay-constrained multicast routing heuristcs and their evaluation*. PhD thesis, Washington State University, 2006. Available via http://www.dissertations.wsu.edu/Dissertations/Summer2006/v%5Firava%5F072106.pdf.

[9] H. Kochkar, T. Ikenaga, Y. Hori, and Y. Oie. Multi-class QoS routing with multiple routing tables. In *Communications, Computers and Signal Processing, 2003. PACRIM. 2003 IEEE Pacific Rim Conference on, Vol.1, Iss., 28-30 Aug*, pages 388–391, 2003.

[10] J. C. Rowanhill, P. E. Varner, and J. C. Knight. Efficient hierarchic maangement for reconfiguration of networked information systems. In *Proceedings of the 2004 International Conference on Dependable Systems and Networks, IEEE*, pages 517–526, 2004.

[11] E. S. Viddal. Ratatoskr: Wide-area actuator RPC over GridStat with timeliness, redundancy, and safety. Master's thesis, Washington State University, Pullman, Washington, USA, December 2007.

[12] Y. Wang, R. Kantola, and S. Liu. Adding multi-class routing into the DiffServ architecture. In *Systems Communications, 2005. Proceedings, Vol., Iss., 14-17 Aug*, 2005.