# 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 GridStat modes and of two different mode change algorithms which 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 grids communication system and incorporates the requirements and network technologies back to when it was developed. The requirements for communication in the electrical power grid are changing. Growing concerns about terrorist attacks, changes in the power flow structure after the deregulation in 1996, new uses of technologies (IntelliGrid [?]) and an increased overall load to capacity ratio of the transportation 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 todays grid. [?] and [?] discuss the limitations of SCADA in more detail.

GridStat is designed to address the need for a flexible and robust communication system in the electrical power grid, and provides a specialization of the publisher-subscriber paradigm. 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 which collectively manage resources and subscriptions in the data plane. The data plane is a virtual message bus and 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 in run-time through the use of modes.

A mode contains the necessary forwarding rules for a

---

[*] Current affiliation: Eltek Valere, Drammen Norway

[†] Current affiliation: Simula Innovation, Oslo Norway

[‡] Current affiliation: University of Cyprus, Nicosia Cyprus

[§] Contact author

set of subscriptions and 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 in which the deallocation and allocation of subscription bundles at run-time is expensive and may result in unsatisfactory subscription delays. GridStat enables subscription bundles to be allocated and pre-loaded into the status routers routing tables where operating modes control which routing tables the status router network will utilize. Depending on the GridStat deployment, status routers can utilize several routing tables corresponding to the operating modes, while inactive routing tables lie dormant.

We believe the mode change mechanism will help utility companies (control centers), regional control centers, ISOs and nation-wide monitoring centers in pre-contingency planning for communication needs and to switch subscription bundles when contingencies do occur in the electrical power grid. Furthermore, modes enable data load shedding in the communications infrastructure in a similar manner as the electrical power grid utilizes power load shedding. For example, subscribers could specify two QoS sets; desired QoS and least desirable QoS, and 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.
- The design and implementation of two mode change algorithms with different tradeoffs.
- An experimental evaluation which compares the mode change algorithms in terms of performance, resource usage and variance (time) in the presence of various temporal network conditions.

## 2 Status Dissemination and GridStat

GridStat is a publisher-subscriber framework the targets application domains where the majority of data is made available at periodic time intervals, and is mainly designed to serve as a flexible and robust communication system in 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 [**?**] and [**?**].

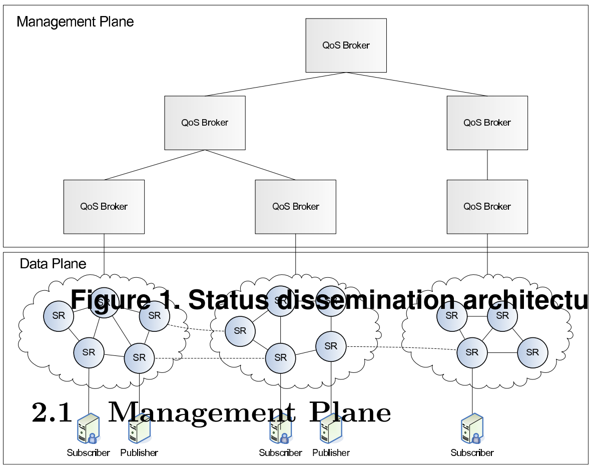


**Figure 1. Status dissemination architecture.**

### 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*, where the leaf QoS broker has complete control over all available resources and the corresponding resource usage. The resources include event channels, status routers, publishers and subscribers. Event channels serve as communication links between status routers, in which 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 and subscribers 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 term used to describe a virtual message bus where subscription data flows between publishers and subscribers. The virtual message bus consists of status routers and event channels, 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 and the subscriber can be bound. 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 are bound by the latency request parameter. If no such path exists, the subscription request is rejected. Subscription traffic is therefore bound by a specified delay and is able to flow on several disjoint paths towards the subscriber, providing timeliness and reliability QoS to subscriber applications.

## 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*. Further, an overview of the RPC mechanism and the support for multiple active routing tables in the status router network (data plane) is presented.

### 3.1 Overview and Mode Terminology

A *mode definition* consists of an ID, a name and a set of data plane subscriptions, and is owned by a single QoS broker in the management hierarchy. Modes defined and owned by a QoS broker constitute a *mode set*, and exactly one of the modes in a mode set is active at any time; an operating mode. This means that every QoS broker always operates in one mode, or in a *default* mode if no modes are defined. A QoS broker that operates in a mode implies that all subscriptions contained in the subscription set of 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 to be utilized in at least one of the modes the status routers are currently operating in. Since every QoS broker in the management hierarchy always operate in a mode, all status routers operate in as many modes as there are levels in the management hierarchy above them. For instance, with $x$ levels in the management hierarchy, a status router has $x$ QoS broker ancestors (*ancestor scope*), and will therefore always operate in $x$ modes simultaneously. This implication on the status router network enables coarse resource provisioning between mode sets, e.g. a top level QoS broker controls 40% of the available resources in its hierarchical scope while the QoS brokers beneath it must further provision the remaining resources between them.

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). However, status routers might have tens or even hundreds of routing tables pre-loaded, either in memory or on disk, but only the routing tables that correspond to the active set of modes are used.

### 3.2 Propagation of Modes

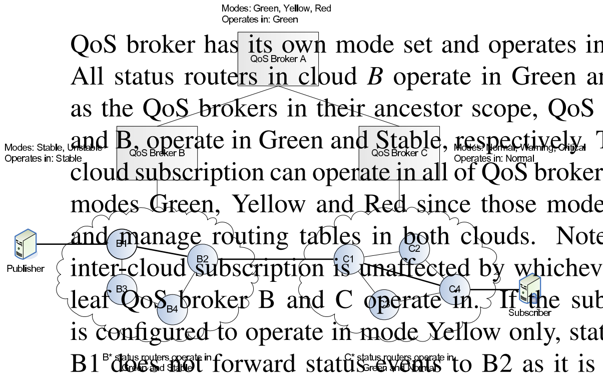
All QoS brokers read the mode definitions defined in their respective configuration files and store the modes internally. The leaf QoS broker requires additional state information on the modes that are defined in its ancestor scope. The leaf QoS broker requests all the modes that are defined in its ancestor 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 operating mode. 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 when they connect, or fail and reconnect, to GridStat. 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.

The subscriber receives all modes that are defined in its ancestor scope from and after it has registered with its edge status router. The set of modes allows the subscriber application to select in what modes a subscription will operate in and are sent with subscription requests to the management hierarchy.

## 3.3 Mode Change Operations

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

Figure 2. Status dissemination architecture.



A QoS broker can only change between modes that are defined in its mode set and 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 and cloud B. The main purpose of a mode change operation is to inform 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 from. 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 corresponding 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, and additional recovery mechanisms (see Section 3.6) must be utilized in order to restore the operating modes on the status routers that are considered inconsistent. An inconsistent mode change operation will most likely result in some subscribers not being able to see subscribed data, as they otherwise would have 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 as 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 v4. 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 registered to operate in the coordinators current and new mode to flow during the entire mode change. Thus, subscribers with subscriptions registered in both the current and new mode continue to receive status events during hierarchical mode change operations that switches between those modes. The hierarchical mode change algorithm is discussed in more detail in Chapter 4.

The flooding mode change algorithm disseminates mode change operations directly out on the data plane by using the limited flooding mechanism in GridStat. When a status router receives the operation it forwards the operation on all outgoing event channels, except the event channel from which it received the operation. As status routers may receive multiple copies of the same operation, redundant copies are discarded. Status routers are informed to change from the current mode to the new mode at some future timestamp. The flooding mode change algorithm is discussed in more detail in Chapter 5.

The two mode change algorithms provide different tradeoffs. The hierarchical mode change algorithm is a resource intensive algorithm which is split into five message phases in order to enable transferred subscriptions present in both modes in a mode change, e.g., from Green to Yellow, to flow. A message phase means the coordinator has to initiate and and propagate a mode change phase (message) down to all status routers in its hierarchical scope, and the next phase cannot be initiated until the previous phase has completed; the coordinator has received aggregated acknowledgements from all status routers and QoS brokers. The flooding mode change algorithm, on the other hand, is a best-effort algorithm, and disseminates mode change operations directly out on the data plane where status routers are informed to switch at a predetermined future time. The flooding mode change algorithm is efficient, in terms of resource usage and performance, but does not guarantee any subscriptions to flow during the mode change operation. The flooding mode change algorithm relies on the status routers ability to switch modes at the exact same time and therefore requires all status routers to be time synchronized.

## 3.4 The RPC Mechanism

The RPC mechanism is a newly added feature to GridStat v4 that allows connection-oriented communication to be conducted on top of GridStats publisher-subscriber architecture [**?**]. The RPC mechanism in GridStat facilitates two-way communication between GridStat entities whereas the standard publisher-subscriber paradigm does only support one-way communication between publishers and subscribers.

The RPC mechanism offers several advantages compared to CORBA as it is built on top of the publisher-subscriber paradigm in GridStat:

- RPC connections can utilize the spatial redundancy property in GridStat.
- RPC connections can easier utilize temporal redundancy through a flexible timeout-management scheme.
- Pre- and post-conditions on the client and server provide sufficient control mechanisms to ensure that certain properties are in place prior to or after the RPC call has been executed at the receiving end. For instance, a pre-condition on a substation actuator may employ a security check to verify that a line has been de-energized prior to switching a circuit breaker.

In order to utilize the RPC mechanism, both the *client* and *server*, must embed a publisher and subscriber instance and establish bi-directional subscriptions through the services that the management hierarchy provides. The management hierarchy establishes the two subscriptions with the requested QoS; temporal and spatial redundancy.

### 3.4.1 Spatial and Temporal Redundancy

Conducting communication over the RPC mechanism provides more flexible control of message delivery. Since RPC connections utilize data plane resources one can associate QoS requirements with any RPC connection as is possible with standard subscriptions. Temporal and spatial redundancy allow RPCs to span several paths across the data plane and increases the probability of message delivery. Subscription latency demands enable the client to take additional measures when a delivery confirmation is not received when expected. In addition, the RPC mechanism will benefit from further development of technologies in the data plane as it completely utilizes the publisher-subscriber paradigm and mechanisms in GridStat. An ongoing project, for instance, investigates how to secure data plane communications through encryption, from which the RPC mechanism will directly benefit through its reuse of GridStat technologies [**?**].

Communication related to mode change operations utilize some of the overall properties provided by the RPC mechanism:

- Mode change communication over RPC within the management hierarchy or from the management hierarchy to the data plane can utilize spatial redundancy if and when needed through the use of modes.

- 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. All QoS brokers add an additional layer on top of the temporal redundancy scheme provided by the RPC mechanism, and thereby have the ability to queue outgoing mode change messages and schedule them for sending by using the RPC mechanism at a later time.

## 3.5 Recovery Mechanisms and Acknowledgement Aggregation

In order to tolerate some degree of network failures and to eventually ensure consistent mode change operations, 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 acknowledgements caused by failed QoS brokers, failed status routers or link failures, and attempts to resolve these situations when possible.

An important part of a mode change operation is to gather acknowledgements from the participants in the operation. Status routers which receive a mode change operation respond with a mode change acknowledgement up to their leaf QoS broker. When a leaf QoS broker receives the first acknowledgement for a particular mode change operation it immediately starts an aggregation round to gather acknowledgements from its status routers. The leaf QoS broker stores the name of the status router and the mode change identifier, and starts a timer which times out to stop the aggregation round. Additional acknowledgements are registered in a similar manner. The leaf QoS broker stops the aggregation round when all expected acknowledgements have been received; otherwise, it times out. The leaf QoS broker finalizes the aggregation round and prepares a response to the coordinator that initiated the mode change operation after the aggregation round has completed successfully. If the leaf QoS broker is the coordinator, it updates the operating modes table in its state and marks the mode change operation as complete. Otherwise, if the coordinator is located at a higher level in the management hierarchy, the leaf QoS broker updates its ancestor modes table and prepares an acknowledgement and sends it up to its parent QoS broker for further processing.

Interior QoS brokers go through the exact same sequence of steps as the leaf QoS brokers and start aggregation rounds when the first acknowledgement from one of its direct children QoS brokers are received. The process continues until the coordinator of the mode change operation has finalized the aggregation round.

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

## 4 Hierarchical Mode Change Algorithm

The hierarchical mode change algorithm is divided into five distinct phases which enable transferred subscriptions present in both modes of a mode change operation, e.g., from Green to Yellow, to flow. Furthermore, the five phases eliminate any status router overload scenarios during a hierarchical mode change operation. The following list shows how the hierarchical algorithm affects the status routers in a mode change from Green to Yellow:

1. **The inform phase** - Edge status routers inform their subscribers about the upcoming mode change. This phase is a standalone phase in order to ensure that all subscribers have been informed about potential QoS violations prior to switching routing tables.
2. **The prepare phase** - Edge status routers switch to the temporary routing table Green $\cap$ Yellow. The highest subscription interval (lowest rate) of transferred subscriptions are 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 either Green or Yellow is dropped at the edge status routers. This step in the hierarchical mode change algorithm eliminates any status router overload scenarios (incoming queues and outgoing queues) as subscriptions are only removed.
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 about the completed mode change. This phase is a standalone phase in order to ensure that all modification to routing tables in the status router network is complete and that subscribers will receive status events conforming to the desired QoS.

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 awaits acknowledgements (see Section 3.5). When the leaf QoS broker has received acknowledgement from all status routers in its cloud, it forwards a single acknowledgement 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 [**?**].

## 5 Flooding Mode Change Algorithm

The flooding mode change algorithm is an alternative to the hierarchical mode change algorithm and offers better statistical delivery guarantees to the data plane. The flooding mode change algorithm delivers mode change operations directly to the status routers through the limited flooding mechanism in GridStat. In order to utilize the limited flooding mechanism, the QoS brokers embed a publisher instance that connects to some edge status router in the

QoS brokers hierarchical scope. The QoS broker can publish mode change operations through the publisher instance, where status routers forward the operation to all their status router neighbors, except the one they received the operation from. The flooding mechanism will eventually stop when all status routers have been informed. The limited flooding mechanism benefits from the amount of redundant paths in the data plane and is thus more resilient to network failures than 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 acknowledgement to its leaf QoS broker and will activate the new mode at the destined future timestamp. Figure 3 shows a flooding mode change initiated by QoS broker A which floods the mode change operation directly out on the data plane through its embedded publisher instance, and is able to deliver the operation to all participants within five message rounds. The diagram assumes an equal link delay, and the event channel labels refer to the message round in which the operation is flooded.

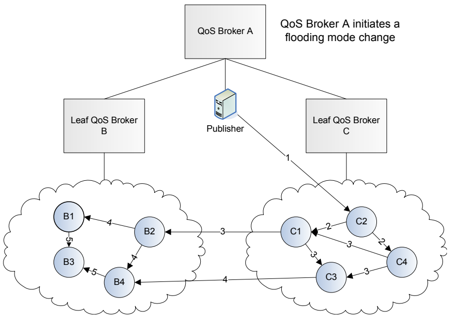


**Figure 3. The flooding mechanism.**

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

# 6 Experimental Evaluation

The following sections contain only a subset of the experimental evaluation, and we refer to [?] for a more in-depth analysis. The experiments were conducted on a 16-node cluster at the Electrical Engineering and Computer Science department at Washington State University. The hardware and software specifications are described in detail below:

- 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 which was used to emulate link latency and link loss on a per-link basis (data plane links) in GridStat.

## 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. A cloud consists of five status routers: three edge status routers and two internal status routers.

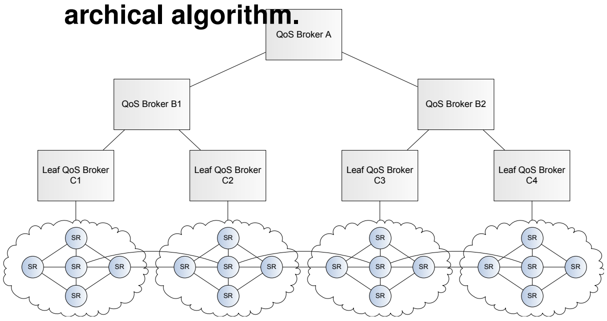


**Figure 4. GridStat experimental setup - hierarchical algorithm.**

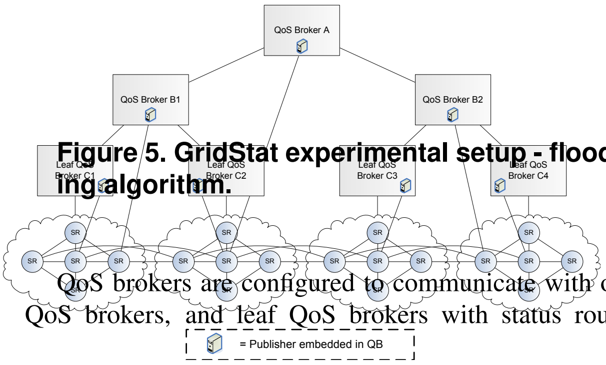


**Figure 5. GridStat experimental setup - flooding algorithm.**

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 acknowledgement after *some* timeout, it resends the RPC call. The RPC retry timeout value is subject to an experiment setup as the traversal time for the call and acknowledgement depend on the link latency and the number of event channel hops (Table 1).

| 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.**

The link emulator associates an event channel with a latency, a probability of packet loss and a burstiness setting. When a packet loss triggers, the link will consecutively lose as many packets as the burstiness setting suggests. If the burstiness setting is variable-sized, 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 burstiness setting. More specifically, 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 mention that a link will not trigger a new packet loss when in the middle of an ongoing loss sequence, and that edge links, publisher to edge status router or subscriber to status router, will not lose packets.

## 6.2 Algorithm Comparison

Figure 6 depicts the average completion time per experiment (100 operations) for hierarchical mode change operations activated from the top level QoS broker in the management hierarchy. The y-axis denotes the average time in ms and the x-axis denotes the experiment setup in the form *overall packet loss : minimum consecutive packet loss - maximum consecutive packet loss*.

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

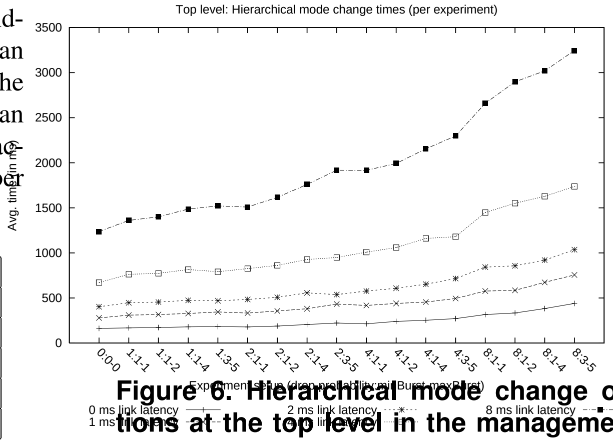


**Figure 6. Hierarchical mode change operations at the top level in the management hierarchy.**

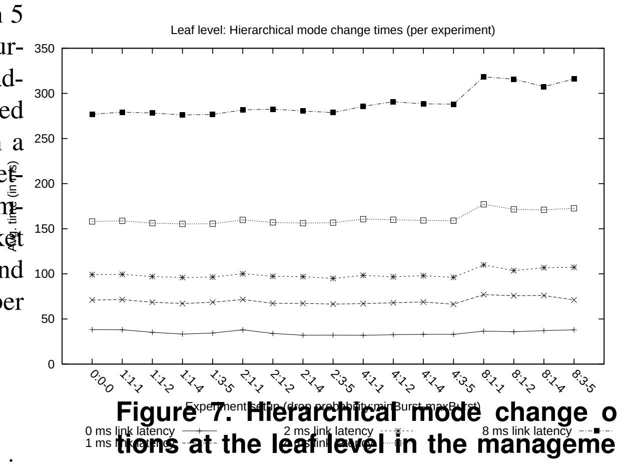


**Figure 7. Hierarchical mode change operations at the leaf level in the management hierarchy.**

Figure 7 depicts the average time per experiment (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 implication 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 8 shows the average time per experiment (300 operations) for flooded mode change operations activated from the top level in the management hierarchy. As mode change operations are disseminated directly on to 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, in which the average mode change time increases.

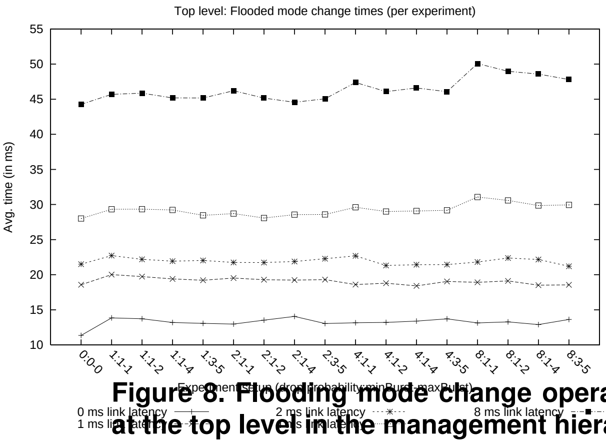


**Figure 8. Flooding mode change operations at the top level in the management hierarchy.**

## 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 the double of the mode change times achieved at the leaf level, and the same results are seen between the experiments 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 redundancy.

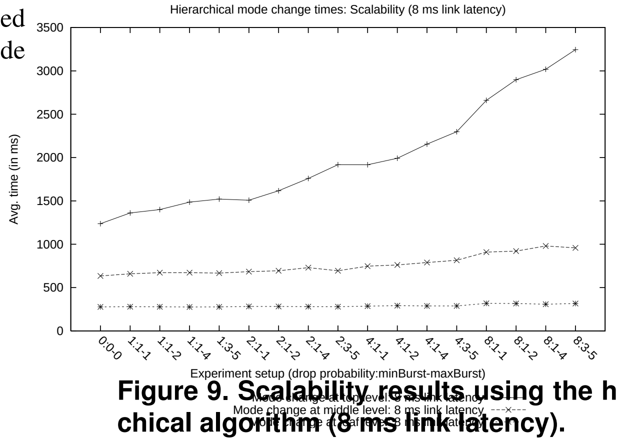


**Figure 9. Scalability results using the hierarchical algorithm (8 ms link latency).**

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 length 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 length of RPC connections depend on the topology in the data plane and at which status router the QoS brokers publisher and subscriber are connected to. Compared to a binary management hierarchy, a trinary management hierarchy would most likely result in a wider management hierarchy and slightly less populated GridStat clouds. It is important, however, to note that the status router network remains the same; the number of status routers and the underlying network topology. This suggests that a trinary tree has fewer levels than a binary tree, and that the length of RPC connections between QoS brokers are longer. In that case, a binary tree and a trinary tree should yield approximately the same results.

Figure 10 depicts experiments conducted at the three levels in the management hierarchy with 8 ms link latency by using the flooding algorithm. The diagram represents the average time at which all status routers have received the mode change operation (with 8 ms link latency), and give a notion of how much time should be allocated for flooding a

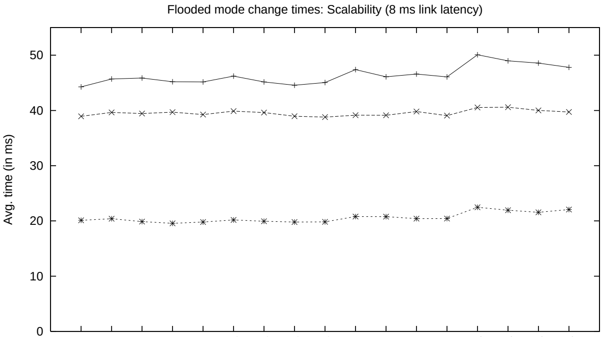


**Figure 10. Scalability results using the flooding algorithm (8 ms link latency).**

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 stringent network conditions. This means that, in the average case, status routers can safely switch modes at any time after that.

The experiments conducted at the three levels with 8 ms link latency show an increase in mode change completion times that is caused by larger flooding domains when the coordinator resides higher up in the management hierarchy, and with 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 (event channels) 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. In addition, the leaf QoS brokers utilize the spatial redundancy property in GridStat through its established RPC connections with the status routers it controls, which increases the number of link traversals for both algorithms.

| **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 are novel, there is not much related work that closely resembles the contributions of this paper. Previous work has been done in network routing using multiple routing tables, but this work has addressed multiple simultaneously active routing tables for differentiated QoS routing[**?**][**?**]. 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 emphasizing 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 is novel 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 them in run-time.

## 8 Conclusions

This paper has presented an implementation of global and hierarchical mode change mechanisms and management in GridStat, which is a novel way to switch between bundles of subscriptions in run-time. The power grid industry can benefit from this mechanism in GridStat by identifying and creating distinct mode holders for the information that is needed during various critical situations in the electrical power grid. Furthermore, GridStats 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 efficiency 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 resources and queues to become exhausted. The flooding mode change algorithm is an efficient best-effort algorithm that informs status routers to change modes through the flooding mechanism, and therefore provides a high statistical delivery guarantee. However, the flooding mode change algorithm is not able to prevent computational resources and queues in the status router network to become overloaded.

The experimental evaluation shows the mode change completion times for both mode change algorithms under various network conditions. The experimental 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 with 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 corresponding experiment conducted with the hierarchical mode change operation. Furthermore, the flooding mode change algorithm is less prone to link loss and shows a minimal impact on the overall mode change completion times. In term of scalability, the flooding mode change algorithm scales linearly and clearly shows that the width of the flooding domain and the point at which QoS brokers initiate the flooding mechanism from increases the 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, volms 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] 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.

[6] 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.

[7] E. Solum. Modular over-the-wire security in managed publish-subscribe systems. Master's thesis, Washington State University, December 2007.

[8] E. S. Viddal. Ratatoskr: Wide-area actuator rpc over gridstat with timeliness, redundancy, and safety. Master's thesis, Washington State University, August 2007.

[9] 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.*