Overcoming Communications Challenges in Software for Monitoring and Controlling Power Systems

Kenneth P. Birman[†], Jie Chen[‡], Ken Hopkinson[†], Bob Thomas[‡], Jim Thorp[‡], Robbert van Renesse[†], Werner Vogels[†]

[†]Department of Computer Science. Cornell University

[‡]School of Electrical and Computer Engineering, Cornell University

Abstract

The restructuring of the electric power grid has created new control and monitoring requirements for which classical technologies may be inadequate. The most obvious way of building such systems, using TCP connections to link monitoring systems with data sources, gives poor scalability and exhibits instability precisely when information is most urgently required. Astrolabe, Bimodal Multicast and Gravitational Gossip, technologies of our own design, seek to overcome these problems using what are called "epidemic" communication protocols. This paper evaluates a hypothetical power monitoring scenario involving the New York State grid, and concludes that the technology is well-matched to the need.

Introduction

The restructured electric power grid challenges operators, who find existing options for monitoring grid status increasingly inadequate. This was highlighted in the August 2003 power outage. Transcripts documenting operator interactions during the event made it clear that as the crisis unfolded, large-scale perturbations were creating increasingly confusing local conditions [Lipton03]. A critical monitoring system failed outright, but even outside of the region directly impacted by this event, operators were unable to make sense of fluctuating voltages and line frequencies observed over a period of several hours. Lacking data about the state of the grid on a large scale, they were not able to formulate appropriate control interventions on the basis of the limited information available from purely local instrumentation.

Unless new technologies are brought to bear on the problem, it seems all too plausible that there could be repeat events of a similar nature. The restructuring of the grid has changed the monitoring problem in fundamental ways. Fundamental improvements in monitoring technology are required in response.

[1] Authors' address: Cornell University, Ithaca New York 14853. This research was funded in part by DARPA, administered by AFRL under grant number IFGA/F30602-99-1-0532.

Monitoring and control problems arise at many time scales. The shortest deadlines are seen in relay control algorithms for equipment protection systems, which must react to events within fractions of a second. Assisting a human operator who is trying to make sense of an evolving power shortage or some other slower contingency involves tracking data that evolves over periods measured in minutes. Still other forms of data change over hours or days. Our focus here is on the second case (minutes). Our group has also looked at problems on shorter time scales, but the associated techniques are outside the scope of this paper [Coury00, Coury02]. For tracking grid state over long time scales, technologies such as databases, web sites and even email are adequate.

Our effort approaches the problem at several levels. First, using simulation tools, we explored proposed communication standards for the restructured grid. Most researchers anticipate that an Internet-like *utilities network* (dedicated to the power grid and isolated from the public Internet) will emerge in the coming decade. On this, TCP would be the primary communication protocol. To accurately model the behavior of TCP in such a setting we coupled the NS/2 protocol simulator to the PSCAD power grid simulator, and then looked at scenarios that might arise in a power grid during periods of stress [Hopkinson03]. Our study uses a realistic model of the New York State power grid as the target environment. NS/2 includes especially detailed simulations of TCP. The combination is therefore as close to reality as is currently possible with simulators.

This investigation reached a pessimistic conclusion. Our studies suggest that TCP is likely to exhibit problematic behavior under stressful conditions, and that this behavior could emerge as a serious obstacle for monitoring software systems running over it. Some of these issues arise primarily at very short time scales, but not all of them. Moreover, this comes as no surprise: TCP is the protocol used to download web pages and to support most Internet media players, and most readers will be familiar with the occasionally balky behavior of such systems. Indeed, one of the pioneers of the Internet, Butler Lampson, has observed that the "In a strict sense, the Web doesn't really work. But of course, it doesn't really need to work perfectly to be a great success [Lampson99]." Others have pointed out that the Internet was built to deliver email and move files, not to perform critical control tasks, and that many of the applications now migrating to the network are mismatched with its properties. Moreover, beyond these issues, we also identify practical difficulties in using TCP for grid monitoring problems.

The second aspect of our research involved the development of a new monitoring technology that avoids these TCP-specific problems by building on a very different style of communication, inspired by the architecture of peer-to-peer file sharing tools of the sort that have gained such notoriety. Unlike a peer-to-peer file sharing tool, our peer-to-peer protocols share data captured by the monitoring subsystem. They run on the Internet, but using UDP instead of TCP, and thus avoid the problematic behavior just cited. Moreover, they employ a pattern of "gossip" communication in which data spreads much like an epidemic in a population, routing around disruptions.

We used these new protocols to implement a system, Astrolabe, which offers a high degree of flexibility in terms of what data can be monitored. Astrolabe can be reconfigured at runtime, and will adapt to new demands within seconds even in a system with millions of nodes. The system is flexible with respect to the data collected and the manner in which that data is summarized. It is exceptionally scalable, and remains stable under stresses that would cripple many alternatives. Finally, Astrolabe is serverless: just as peer-to-peer file sharing systems create the illusion of a shared file system, Astrolabe creates the illusion of a database, updated in real-time, and yet has no central server. This eliminates single-point failures.

The last aspect of our work explores potential uses for Astrolabe in the kinds of monitoring and control scenarios that arise in the restructured grid. We used our simulation to study the quality of information that would be available to a hypothetical New York grid operator as a severe perturbation is introduced by disrupting power generation in Long Island, a situation analogous to the one that arose in the hours prior to the August 2003 blackout. As noted earlier, we focus on time scales of tens of seconds or minutes. This preliminary study supports the belief that Astrolabe could be valuable if it were actually deployed in this manner[2]. Astrolabe also includes a security architecture, and we show that with this mechanism, it might be possible to strike a balance between the legitimate need of power suppliers to safeguard data of high commercial value and the needs of grid operators to access that data in the face of a crisis.

The remainder of this paper addresses each of these topics in turn. To avoid repeating material that has appeared in print previously, we limit ourselves to a high-level description of Astrolabe. Readers interested in understanding exactly how the technology is implemented are referred to [VanRenesse03, Birman01]. A detailed description of our multi-mode simulator can be found in [Hopkinson03].

The Monitoring and Control Impact of Restructuring

The restructured power grid poses new kinds of monitoring and control problems. In this section, we briefly survey the issues, and also point to progress within the power industry towards communications standards intended as a response to these needs.

For purposes of this paper, restructuring can be understood as a process with two major elements. One element is regulatory: whereas classical power grids were centrally controlled and operated, changes in the regulatory structure now encourage independent ownership of generators and favor the emergence of competitive mechanisms by which organizations can enter into bilateral or multilateral power generation contracts. The second element is a consequence of the first involving large-scale operation of the grid. In the past, this was a centralized task. In the restructured climate, a number of competing power producers must coordinate their actions through a set of independent service operators.

Restructuring has been an incremental process. In its earliest stages, we saw the breakup of large monopoly-styled utilities into smaller companies with more specialized roles. At the same time, we've seen slow but steady growth in the numbers of long-distance contracts. Finally, the grid is being operated closer and closer to its capacity and limits. All of these trends are only just reaching a point where grid monitoring and control will need to be revisited.

[2] It should be noted that Astrolabe is a experimental prototype, not a commercial product aimed at the power systems community. Inquiries about research use of the system may be directed to the authors.

An example of how these trends complicate grid control involves *load following*, which is the problem of matching power generated to power consumed. Classically, load following was implemented regionally, by monitoring line frequency. Since line frequency is uniform throughout any well-connected region of the power grid, if power generation becomes mismatched to power consumption, operators can detect this by noticing that frequency was sagging or rising, and in the classical grid, could respond in a coordinated way (by increasing or decreasing generated power). Since a single utility was responsible for each region, all of its generators share a commercial incentive in responding to changing loads.

Now consider the same task in the presence of bilateral contracts. If Acme Smelters contracts to purchase power from Ithaca Wind Power, Acme and IWP need to coordinate power production and consumption on a pairwise basis. Other companies may be willing to participate in load following for the grid as a whole, but will not want to adjust their power production to deal with load and supply variations in the Acme-IWP contract – after all, they don't make money on that contract. To some degree, this challenge can be ameliorated by supporting only fixed-capacity contracts. Yet one can never exclude the possibility that some unanticipated event might shut Acme down suddenly, or cause IWP to take a generator offline. Unless both parties react immediately and in a coordinated manner, such an event will impact operators throughout the region.

Thus bilateral contracts are of global interest: the status of such a contract may impact the way an individual operator determines power output, may have implications for grid protection, and may impact pricing. On the other hand, knowledge of other operators' commitments also creates opportunities for a producer to discover and exploit market power. Regulators and operators are reluctant to release such information.

Worse still, the introduction of bilateral contracts invalidates the traditional use of line frequency as an implicit communications channel. Yes, all entities monitoring the frequency see the same value. But in a world of bilateral or multilateral contracts line frequency changes alone are not enough to formulate appropriate local actions (or even to decide if local action is needed). Much more information is required.

In the restructured grid, a tremendous number of problems that had been solved for monopoly-structured power systems will need to be revisited. We find ourselves in a new world of not merely bilateral contracts, but also mutually suspicious operators, loath to share information about the states of their systems, their available capacities, loads on private lines, pricing, and equipment states. Aging long-distance lines are being subjected to greatly increased loads. Aging relays are being operated closer and closer to their trip points. In a post 9-11 era, grid operators must be cognizant of terrorist threats, both to their equipment and to their control systems and software. Finally, with the grid operating closer and closer to its limits, control requires quick reactions and an increasingly fine hand.

A New Monitoring Problem

Considerations such as the ones just cited suggest that the restructured grid will require a new form of monitoring system. What should be the properties of that system?

To first approximation, setting security considerations to the side, a power grid monitoring system can be imagined as a set of large, dynamic, databases collecting data over an Internet-like communications network. The rows in these databases contain information collected at the many points where the grid can be instrumented: data such as voltage, line frequency, phase angle of the phasor, status of equipment, and so forth. There would be different databases for each major category of components, and perhaps also for different categories of observers: the ISO, operators, consumers, etc.

To make sense of this sort of data, one also needs a more static kind of information: a "map" of the power grid, showing buses and generators and loads, and annotated with the points at which measurements are being taken. In practice, of course, such a map is also a database. Notice that the map is not completely static: on any given day, there may be hundreds of road crews at work on lines throughout a region, and their activities will change the layout of the grid as a whole. Unless a widespread environmental disruption occurs, this type of information changes very slowly. However, when a disruption *does* occur, we may need to propagate the information urgently.

Today, we lack this kind of information. As noted, operators have many reasons to conceal the states of their networks, and this extends to all of these forms of databases. An operator might be willing to disclose certain data to the ISO, but not to competitors. Yet more information is certainly needed in order to avoid repeats of the wide-scale disruptions experienced in August of 2003.

This tension lies at the core of what follows. It is not enough to postulate a monitoring methodology. The need is for a methodology responsive to many goals: the purely technical goal of getting data through on the time scale in which it will be useful, the more pragmatic goals of protecting sensitive information unless a crisis arises, the security considerations associated with managing a vital component of the nationally critical infrastructure, and the pragmatic cost-saving goal of adhering, as much as possible, to widely supported commercial standards, so that off-the-shelf (so-called "COTS") technologies can be employed in building the new control systems.

System Model and Statement of Objectives

Rigorous development methodologies reside upon explicit models of the target environment, rigorous definitions of the protocols used, and the use of formal tools to show that the protocols achieve well-defined objectives when run in the target environment. While brevity precludes us from undertaking all of these tasks in the present paper, our research effort follows this outline. In this subsection we give a more precise characterization of the target environment and the goals for our communications infrastructure. Elsewhere, we've taken steps to close the loop, for example using an automated theorem proving tool called NuPrl to show that some of our protocols achieve the desired behavior[3]. We see these as first steps in a longer-term effort to achieve similar rigor in all parts of our work.

[3] More precisely, we've used NuPrl to prove properties of a previous class of protocols that we developed to solve fault-tolerance problems of the sort seen in stock exchanges and air traffic control

Our work models the utilities network as an *asynchronous distributed system*, consisting of a set of computing nodes, an infrastructure permitting them to exchange messages, and a message-passing layer. Messages are transmitted *unreliably*: a given message may be lost in the communication system, damaged, delivered out of order, or even delivered more than once. There are no *time limits* on message latency: a message can be arbitrarily delayed, even if some other message, sent almost at the same time, arrives with no delay at all. Although our work currently makes no use of this feature, nodes can be assumed to have high-quality GPS clocks; unless such a clock malfunctions, it provides time information accurate to hundreds of microseconds.

We assume that computers within the network fail by crashing, and that there is no way to distinguish such a crash from a timeout caused by transient packet loss. Traditionally, one would assume that failures are uncorrelated, but this is not realistic in systems structured like the Internet. Accordingly, we do assume that computer failures are uncorrelated, but we only assume that message failures are uncorrelated *on a long time scale, or for messages on path-disjoint routes*. Thus, for short periods of time, it may be impossible for computer A to communicate with computer B, even if both computers are operational. However, even if this occurs, A will probably be able to communicate with some node C, that can communicate to B. Moreover, if A and B remain operational, they will eventually be able to communicate. The time constants are, however, unknown within the system.

We note that this non-transitive view of communication is a common element of models of the Internet. For example, the same approach underlies MIT's work on Resilient Overlay Networks [Anderson01]. Indeed, MIT's RON system would be of possible value in a power systems network, although we have not explored this option.

Obviously, the situation in any real network is not quite as bleak as this model might permit. A trivial instance of the above model would be a system in which all the nodes have crashed, or one in which the network never delivers any messages at all. A utilities network, on the other hand, would presumably work fairly well.

Accordingly, we introduce a probabilistic element. We assume that there are constants p_c governing the probability that a computer c is operational, p_{clock} giving the probability that its clock operates within a bounded error and skew relative to true GPS time, and p_m giving the probability that a given message m will be delivered in a timely fashion. There is a mean communications delay δ (for messages not impacted by a failure). Moreover, within a given network, we may have good estimates for each of these constants.

Our abstract problem can now be refined as follows. Not only do we wish to support a database of sensor information, but we would like to do so in a manner that can be shown to guarantee timely delivery of information to monitoring agents throughout the network, with high probability, provided that the network behaves consistently with the model. Additionally, we will want solutions to "scale well", meaning that the overheads

---

systems. We have yet to apply NuPrl to Astrolabe or to a power scenario, and doing so would involve surmounting a number of technical challenges. However, we are optimistic that they can be overcome.

imposed on the network are either constant, or grow slowly, as the size of the system increases, and that delays in reporting data remain low.

In the remainder of this paper, constraints of brevity preclude formalization of the mechanisms we discuss. However, interested readers are referred to [VanRenesse03] for examples of the form of analysis made possible by this model. In particular, the Astrolabe technology we'll describe below satisfies the goals just enumerated in the model we've outlined. A second protocol, called Bimodal Multicast, could be used for scalable notifications when important events occur and large numbers of nodes must be notified rapidly [Birman99, Jenkins01]; it too can be proved to have the desired behavior in the target system model. In contrast, there has been little analysis of commercial Internet technologies using formal models or methods, and (as will now be discussed further), they apparently lack the desired properties.

The Power Internet

The power industry became aware of the trends we've summarized long ago, and has invested almost a decade in study to the matter. This work has yielded a number of proposals for new ways to protect the grid, and also a number of proposed standards for a future utility Internet, isolated from the public Internet but based on the same off-the-shelf technologies and running the standard protocols. TCP has been designated as the standard communications protocol for the utility internet, and proposals for standardizing the format of data that would be transmitted over these TCP connections are now being debated.

As mentioned earlier, TCP is the same protocol used when downloading web pages, transferring files and emails, supporting chat, and even transmitting video or audio data. Perhaps less widely appreciated is the degree to which TCP is tightly integrated with the routers used within the network itself. Routers drop packets as they become congested and this is viewed as a signal to the TCP endpoints that they should throttle back. Indeed, one router policy, called Random Early Drop, or RED, attempts to anticipate overload and starts to discard packets even before the router becomes completely overloaded, by randomly picking packets and discarding them[4]. TCP, for its part, varies the sending rate continuously, ratcheting the rate up steadily (linearly) from some low initial value, then scaling back (exponentially) if loss is detected. A TCP channel thus has a sawtooth behavior: throughput rises slowly, then falls quickly, then rises again. If a TCP connection is idle for a period of time, it resets to a low data rate – this is called the TCP "slow start" mechanism.

The effect of this sawtooth behavior is to ramp the TCP data rate to the maximum level that the network will bear, and then to test the threshold periodically, trying to push its rate up, then backing off just a bit below the limit. One could not casually disable

[4] It may be counter-intuitive that the network should warn of future congestion by discarding perfectly good data before the problem has actually occurred. The intuition is that because round-trip latencies are often a substantial fraction of a second, TCP may need a second or two to detect and react to packet loss.

these mechanisms. They are central to the scalability of the modern network. Without them, it is generally believed, the network would melt down when load surges occur.

The behavior just described is mismatched to the needs of power systems and other real-time applications. In [Coury02, Coury00] we used a simulator to evaluate a protection mechanism over an idealized "instant" communications channel, and then over a TCP connection that accurately models the behaviors outlined above. We found that in an idle network, the protection mechanism works in both cases. But in a network shared with other sources of traffic – even minor traffic, like web page downloads that happen to pass through the same routers, TCP ratchets loads up until packet loss occurs, and then begins to exhibit significant fluctuations in throughput and latency. These problems degrade the functioning of the protection protocol. Of course that study focused on a smaller time-scale than interests us here, but in fact the same issue is also seen (albeit less often) at a scale of seconds or even minutes. Any user of a web browser has experienced this issue. Clearly, such behavior is problematic for software monitoring the grid.

The use of TCP as the core transport protocol for data within the power grid also raises a great number of pragmatic problems. Suppose that normally, operator A is unwilling to disclose the structure of her network to operator B, but agrees to do so during an emergency. One day, a confusing situation arises and operator B tries to access the information in question. Should we now presume that some application running on site B opens (for the first time) a lockbox listing the contact points at which A's network can be instrumented, establishes secure connections to those points (A will want to know who obtained this sensitive data), and monitoring then ensues? We seem to be imagining a rather time-consuming process, particularly since A may have dozens or even hundreds of monitoring points. How would A authenticate the request? If B lies, is the auditing capability of the protocol adequate to "prove" what happened to a skeptical judge? How would B interpret the data without a detailed map of A's system?

Scale intrudes as a further concern. In a crisis, there may be dozens of B's that suddenly all need access to A's data. Thousands of connections will need to be made into A's network. Indeed, it is likely that everyone will want to connect to everyone else. Thus, just as the network starts to exhibit troubling behavior, we may be faced with creating and starting to use literally hundreds of thousands or millions of TCP connections. And all of them will start in the "slow start" mode.

The reader has no doubt experienced an analogous situation. It occurs when fast-breaking news events turn our attention to some little-frequented web site; not the CNN or MSNBC site, which invests heavily to be able to handle "flash loads", but a minor site. The web site goes down, not by crashing, but by responding so slowly that it might as well have crashed. The same problem will arise in power grids if TCP is used this way.

Without belaboring the point, we believe that TCP is poorly matched to the monitoring problem. To summarize:

1. TCP is a balky and unpredictable protocol ill-suited for the proposed uses. Unless a non-standard TCP implementation is selected (unlikely), the power industry will need to live with TCP slow-start and congestion control.

2. It is known that Internet-based systems may exhibit periods of non-transitive connectivity. TCP is unable to "route around" disruptions and would disconnect or run very slowly in such situations.
3. The TCP load on the center of the network may grow as $s*m$, where s is the number of sensors and m is the number of systems monitoring them. If the network grows large enough, this could become a significant cost.
4. Establishing the necessary connections poses deep practical problems, particularly with respect to authentication. It will be costly to keep lists of who should connect to whom up-to-date as the system evolves over time. It may not be practical to reconfigure monitoring systems rapidly when a problem arises.
5. The utilities Internet will support many applications, some having lower priority, and many shipping very large files. TCP lacks any notion of priority, hence low-priority file transfers compete for the same resources as do high-priority urgent notifications. The greedy bandwidth consumption approach underlying TCP ensures that when this happens, routers will become overloaded – bandwidth escalation that overloads routers is a design feature of TCP and the modern Internet.
6. The TCP security model, SSL, permits a client of a server to authenticate a server, and then encrypt sensitive data such as a credit card number. But that pattern is remote from the security issues that arise between mutually suspicious operators.
7. The lightening-fast spread of viruses and worms, and the constant threat of disruptive denial of service attacks, are a fact of life in the Internet. TCP is easily disrupted. A utility network will face similar threats. Thus, intentional disruption may be an issue.

Taken as a whole, it is our conclusion that the power systems community will inevitably come to see standardization around TCP as premature. TCP is a good tool, and the marriage of TCP to the Internet made ubiquitous email and web browsing possible. Nonetheless, it is not apparent that TCP is a suitable protocol for monitoring and control on a large scale, under probable stress, in a setting where the consequences of failure entail massive economic disruption and the significant risk of lose of life.

Alternatives to TCP

When we speak of TCP in this paper, or of the power systems community using TCP as a proposed standard, the TCP protocol in question is just one of a family of TCP protocols. One can identify at least twenty TCP-like protocols, all of which share the basic TCP interface, but using implementations that differ in significant ways. As used up to this point, "TCP" refers to the most common TCP implementations, found in desktop and PC and server systems worldwide. However, there are many variant forms of TCP, and one could ask whether our findings would hold for these other TCP implementations. In particular, there are two or three real-time protocols, designed to use a TCP-like interface, in which reliability is relaxed to support deadlines, or where retransmission is done more aggressively. A real-time protocol might overcome several of our concerns. For want of resources and time, our group has not explored these options.

Astrolabe

Our group at Cornell University developed the Astrolabe system in response to scalability and stability challenges encountered in large-scale monitoring, data mining and control applications. Believing that Astrolabe might be well-matched to the power systems requirement, we set out to systematically evaluate the potential. In this section, we describe Astrolabe and the way it might be used in power settings. The subsequent section presents our evaluation of Astrolabe in a simulation of such a setting.

The Astrolabe system builds a distributed database by monitoring an underlying network and populating the tuples (rows) of the database with data extracted from the instrumented system components. In the utility Internet, these components would be the sensors associated with relays, generators, major load points, and so forth.

Not all components are identical. Astrolabe offers two ways to deal with heterogeneous component sets. The first is to run Astrolabe more than once, with each instance focused on a relatively homogeneous subset of the sensors. Not only might separate instances of Astrolabe be used for different categories of data, but this is also a means of dealing with systems that have different categories of users, who are permitted to access different classes of data. For example, less sensitive data might be placed in an Astrolabe accessible to all operators, while more sensitive data is collected in a different Astrolabe database accessible only to ISO employees[5]. We should note that the ability to report data into Astrolabe does *not* imply the ability to read that data back out. Both to provide data to Astrolabe, and to read data from the system, an application must present credentials "strong enough" to authorize the desired form of access.

We see this as a partial response to the security concerns outlined earlier. The Astrolabe security architecture is based on a public-key cryptographic system, similar to the one used in SSL, and thus adheres to a popular standard. In the remainder of this paper, we'll focus on a single Astrolabe system.

Returning to the issue of heterogeneity, notice that even within a relatively uniform set, there may be sensors that measure values that other sensors don't report. For such situations, we include a vector of Boolean flags in the sensor data. If a sensor reports value *x*, it also raises the associated $flag_x$. An application can check the flag to distinguish a non-reported value from a reported value of zero.

Astrolabe can represent a variety of data types. The system "knows" about the usual basic data types: integers, character strings, floating point numbers, URLs, etc. But it can also handle XML encodings of other data types. The main constraint imposed is that the total size of a tuple should be limited to a few k-bytes. If the instrumentation output for some node involves larger amounts of data, we normally place just a summary into Astrolabe; applications needing access to more detailed data can use Astrolabe to

[5] In such a configuration, it would be important to run Astrolabe in a separate address space, not linked directly to the user's program, and also to encrypt communication between these Astrolabe agents. The system already supports such a separate agent, and by running it over a *virtual private network* (VPN) the latter property is readily achieved. A VPN does, however, require some means to authenticate a machine when it boots – for example, an operator may need to type in a password, or swipe a smart-card.

identify sensor nodes of interest, then drill down using, for example, tools built with the widely standard Web Services architecture.

Astrolabe is flexible about just what data it will monitor. Suppose that a utility monitoring architecture is defined. One might imagine that a tremendous number of parameters could be of interest in one situation or another. Astrolabe encourages the user community to define that full set of "monitorable" data. The system then uses *configuration certificates* to identify the *subset* of the monitorable information that will actually be tracked in real-time. As the needs change, these configuration certificates can be changed, and the new value will take effect within a few seconds, even in a system with hundreds of thousands of monitored nodes. Indeed, Astrolabe's data collection mechanisms are powerful enough to extract data from databases on the monitored nodes, or to extract data from files or spreadsheets, if permissions are set up to authorize such actions. Thus, subject to security policies controlled by the owners of the computing nodes, Astrolabe can be reconfigured on the fly as requirements evolve over time.

Astrolabe supports online "data mining". One can ask participating systems to check for information of interest, and they will do this on the fly, in a massively parallel search. Each node does a small amount of local work, and the Astrolabe database soon reports the collective responses. The computational power of such a data mining capability can be seen to grow in proportion to the size of the network. In contrast, consider the many issues posed by shipping data to a central node and analyzing it there: the latter approach would require that *all* data be shipped to that central node, imposing enormous loads on it and on the network, and the central node would then become a single point of failure and a system-wide security problem.

Data mining out at the edges leaves administrators with the power to configure a security policy appropriate to their installation; deciding what data to share with others, and what forms of authorization will be required before access is permitted. For example, a policy might dictate that normally, node A limits itself to reporting voltage data and the phase-angle of the power phasor, measured locally, but when the ISO announces a "contingency", A may be willing to report far more detailed data. Node A would require a configuration certificate authorizing contingency-mode reporting, and could log this information for subsequent audit.

As mentioned in the introduction, Astrolabe uses a peer-to-peer protocol to track the evolution of this data. This protocol operates as follows. Each node is given a list of peers within the utility internet – for our purposes here, one can assume that this list tracks the full set of nodes in the system (but with a relaxed consistency requirement – the list may lag reality). These lists of peers are updated continuously as nodes come and go [VanRenesse98].

In an unsynchronized manner, each node periodically selects a peer at random and sends it a *gossip* message. The period is determined by a parameter; for the present paper, we used a 1-second period (a fast rate for Astrolabe; in other settings, we've used periods of 10 to 30 seconds).

Gossip can be transmitted over TCP, but we don't require the form of reliability TCP provides, and Astrolabe can equally well run over a less reliable protocol such as UDP. Astrolabe doesn't require that these gossip messages get through – the system operates probabilistically, and even if a fairly large percentage of messages are lost (we've experimented with loss rates of as much as 50%), Astrolabe will continue to give correct behavior. Accordingly, gossip messages are fired off as timers expire, but no effort is made to ensure that they get through, or to resend a message that is damaged or severely delayed in the network.

The contents of a gossip message consist of a list of information available at the sender node. This includes timestamps for rows of the monitoring database, versions of configuration certificates, and so forth.

We limit the size of gossip messages and other messages, hence there may sometimes be more information available than will fit in a message. For this paper, we assumed that the maximum size of an Astrolabe packet is 64KB. Astrolabe biases itself to report "fresh" data and, if this would still overflow our limit, drops some random subset of the items to stay below the threshold.

Now, suppose that node B receives a gossip message sent by node A. B reacts in two ways. First, it identifies information that A possesses but B lacks. Second, it identifies information that B holds but that A lacks. Now B prepares a response message, soliciting data it needs and including data that it believes A is lacking. Again, the size of the message is limited and, if the message prepared according to this logic would be excessively large, Astrolabe biases itself to focus on data having the highest likely value to the destination node. If the resulting message is non-empty, B sends it (without worrying about reliability) to A. Assuming that A receives that gossip reply, it can now update its tuple set with the more current versions from B (if any, and if they are still more current), and send B a message containing the requested objects. Yet again, the message is size-limited and sent without special concern for reliability.

Networks are usually reliable, even when we don't use TCP as a transport. Thus, most messages get through. Although correlated failures can be a problem in most networks, they typically reflect overload on certain routers or certain routes. Thus the type of randomized peer-to-peer communication just described is likely to experience only a low rate of uncorrelated packet loss. It then follows that the theory of epidemic spread will describe the mathematics of how Astrolabe behaves.

Suppose that node A reports some event. After one time unit, the odds are good that node B will know about A's event, and hence the information will now spread from 2 nodes. After an additional time unit passes, 4 nodes will be involved. Although "reinfection" and packet loss will slightly slow the protocol, within expected time logarithmic in the size of the system, all nodes should know about A's event. Log being a small number even in a fairly large network, one sees that within a few seconds, A's status is known throughout the network. For example, log(128) is 7, and log (16,384) is 14. Thus, squaring the size of the network only doubles the expected delay, from a little more than 7 seconds to a little more than 15 with our current gossip rate.

Now, we've already outlined the basic behavior of Astrolabe. Each participant is able to track the state of the overall system, keeping an accurate copy of its own tuple and replicas of the tuples for other nodes in the system. This fully replicated database won't be perfectly consistent, of course, but it will track changes in real-time, and it does support all the usual database query and computational tools (other than "transactions", a consistency model that isn't available within Astrolabe). Thus one can access this database in control software in a natural manner, "drag and drop" the monitoring data into other databases or spreadsheets, which will be updated as conditions change, etc. One can even build web pages that will be automatically updated as data evolves.

A problem with the flat database we've described is that as the network grows large, the amount of data in the typical gossip message will rise linearly in system size. Clearly this leads to a non-scalable architecture. In the case of the utility network, and indeed in most systems where Astrolabe might be used, we believe that there is also limited value to having every node see the current state of every other node. For both of these reasons, as a system grows in size, Astrolabe moves from a flat database to one in which the system is broken into regions of roughly equal size (for practical reasons, we aim for a size of about 100 tuples per region). The nodes that comprise a region should be related in a logical sense – they benefit from tracking one-another's states. They might also be located physically close to one-another, but this is not required by our system. Astrolabe can form regions automatically, but more often, a human administrator does this by assigning nodes to regions when they are first connected with the system.

Thus, in a large system, Astrolabe will look like a stack of databases or a sheaf of spreadsheets. Each node "sees" the data associated with the peers in its own region. Nonetheless, Astrolabe provides an indirect way for a node in one region, say Long Island, to track the status of the remainder of the network. This is done using what we call *aggregation queries.*

The basic idea is simple. In addition to the objects previously mentioned, the administrator of an Astrolabe system can define some set of queries that summarize the contents of a region in the form of a single row. For example, if the data in a region includes line frequency measurements, the aggregate could calculate the mean, variance, and highest and lowest values within the region, thus summarizing as many as 100 values in a small tuple of 4 numbers. The tuples resulting from aggregation are now glued together to create a new kind of virtual database, in which each tuple – each row – summarizes the state of an entire region. A node in Long Island has accurate values for its own region, as well as summary data for the remainder of New York State, and both are updated as underlying data changes. This process can continue: We could define a higher level of the hierarchy in which each state is summarized as a row in some sort of national status database. Such a row would compress the state of perhaps tens of thousands of sensors into a single set of numbers, hence one loses a great deal of detail. Yet it will be updated in real-time, just like the local data.

Without delving into excessive detail, wide-area gossip, used to maintain these aggregated levels of the hierarchy, works much as local area gossip does, and is performed by representatives elected from each local region. The overall consistency and convergence properties of the system are very similar – any event that occurs will be globally visible within time logarithmic in the size of the network as a whole.

Just as one can change the configuration of Astrolabe while it is running, the aggregation queries can be modified as conditions change. A system administrator does this by introducing new aggregation queries that override existing ones or extend the basic set. Each administrator is associated with some level of the Astrolabe database hierarchy, and has authority over the nodes "below" that point in the hierarchy. Thus, an operator in Long Island can perform a kind of online data mining and, in real-time, change the behavior of Astrolabe in Long Island. A New York State ISO operator can change the behavior of the system for the state as a whole. A national operator (if any such entity is ever defined) could change the behavior of the entire national monitoring grid.

Earlier, we mentioned that a node posting information into Astrolabe must first present appropriate authorization keys, and that a node wishing to read data out of the system must similarly present a key authoring such access. In fact, Astrolabe makes extensive use of security keys to control all aspects of the protocol just described, and can even protect itself against many kinds of malfunctions in its own agents. Only the holder of a valid key can perform a given action, and each tuple is always signed by the node that generated it. Thus, damaged or falsified data can be identified and rejected. Keys also allow local operators to determine who is making a request, authorize (or refuse) that request according to data control policies, and audit events after the fact.

The combination of epidemic gossip with a security architecture makes Astrolabe unusually robust against disruption. Because information travels along so many possible paths, a network overload or failure is unlikely so slow things down by more than a round or two of gossip – a worst case outage, then, may delay data by a second or so, but not more. Even the loss of as many as 25% to 50% of packets, network-wide, will only slow Astrolabe down by a round or two of gossip– a few seconds. The system is also unusual in having a strictly bounded communication load that doesn't increase even when things go wrong. Obviously, when a network is very passive, gossip packets will be empty, and load is minimized in this case. But at worst, every maximum-sized gossip packet triggers exactly two other maximum-sized packets. Thus, if the limits are picked intelligently, Astrolabe simply cannot produce load surges that might melt down the network.

In the configuration we tested, the worst-case load on a participating node is three maximum-sized packets sent, and three received, per second. With parameters set as described above (1 second gossip rate, 64KB maximum packet size) the worst-case I/O load is about 1/50th of the maximum for typical desktop computers on a high-speed local area network at the time of this writing.

We see Astrolabe as a very good match to the needs of the utility Internet. First, we are convinced that the "regular" structure of the system will bring benefits because uniformity facilitates the design of general purpose monitoring and system control tools. The automated membership mechanism eliminate the need to keep track of who should connect to whom: a new monitoring node is simply assigned appropriate credentials and told where it will reside in the Astrolabe hierarchy; it then connects to the system and the database expands automatically to include it, without any changes to the configurations of programs that read the data (of course, one might also need to update one or more map database, but this is outside the scope of our paper).

The consistency properties of Astrolabe are also of potential value. In the past, control of large sectors of the grid was easy because of the uniformity of line frequency. All operators saw the same frequency at the same time, hence without explicit communication, could react to changes in load on the system. Systems like Astrolabe bring a similar kind of consistency to the monitoring problem, this encourages consistency in the control decisions made by participating machines. Finally, the ability to configure Astrolabe selectively, and to run multiple instances side by side, gives architects of the utilities network many options for limiting the disclosure of sensitive data. The robustness of the technology would even offer a considerable degree of protection against attacks on the control system launched by intruders within the network.

Bimodal Multicast and Gravitational Gossip

At the start of this paper, we noted that monitoring arises at several time scales. For purposes such as relay control, or notifying operators when a line trips or a generator goes offline or comes online, rapid event delivery is often required. However, Astrolabe targets problems requiring information accurate within tens of seconds or minutes, not shorter time scales. Thus, one could use Astrolabe to track slowly-varying properties of the grid, equipment status, line status, and other similar attributes. In contrast, the technology is *not* suitable for rapid response when a contingency suddenly occurs, such as the sudden loss of a major long-distance power line.

As noted in the introduction, Astrolabe is just one of several technologies our group is evaluating. Bimodal Multicast [Birman99] and Gravitational Gossip [Jenkins01] are examples of other systems, based on the same kinds of peer-to-peer epidemic protocols as are used in Astrolabe, but targeted to settings in which urgent notifications must be sent to large numbers of nodes as rapidly and reliably as possible. These can also support streams a stream of data sent from a few sources to a large number of receivers at high data rates (so-called "multicast" data dissemination patterns). We believe that Bimodal Multicast seems to be a good match to urgent notification problems, and Gravitational Gossip may be especially well-matched to load-following scenarios where streams of events report the status of the load-following contract. For reasons of brevity and focus, however, we have chosen to omit detailed discussion and analysis of these mechanisms from the present paper.

Experimental Study

We now report on an experimental evaluation of Astrolabe in the target setting. As mentioned earlier, we worked with a novel simulation system known as EPOCHS that combines several best-of-breed simulators into one platform. NS/2 is a protocol simulator, best known for its high-quality simulations of the most widely used versions of TCP, and for its ability to accurately model the behavior of commercial routers. PSLF is used for electromechanical transient simulation. PSLF can simulate power systems with tens of thousands of nodes and is widely used by electric utilities to model electromechanical stability scenarios. EPOCHS synchronizes these simulators and insulates its users from many simulation details through a simple agent-based framework.

The use of EPOCHS allowed us to examine the effect that a large-scale deployment of an Astrolabe system could have in preventing large-scale blackouts such as the one experienced earlier this year in the Northeastern United States. The Northeast blackout is a good example of an event that might have been prevented had a trustworthy large-scale information architecture been available.

The electric power grid is divided into a number of relatively large regions including New York, New England, and PJM (Pennsylvania-New Jersey-Maryland). A central control center resides in each center and monitors the state of the loads, generators, and utilities within its domain. Supervisory Control and Data Acquisition (SCADA) systems allow these control centers to get an accurate picture of the system state and allows the center to take action if problems arise. SCADA information is polled as often as once every four seconds from each location and provides a relatively rich picture of a region's current operation. Unfortunately, this information is not shared between control centers. Areas can get some sense of their neighbor's state by monitoring the tie lines that connect them, but this knowledge is extremely limited. It is rare that information is shared between areas with any great frequency. This serves as a major constraint to preventing outages that cascade between areas.

SCADA traffic is light on a per-bus basis, but its large volume in aggregate making it impractical to forward it between regions. An attractive alternative is to aggregate the most useful data for use by other regions that can use it to monitor the general health of their neighbors. During the Northeast blackout, American Electric Power (AEP) was able to detect the dangerous state of its neighbor in northern Ohio and disconnected itself before serious consequences occurred. However, many of the region's other neighbors were not able to react in time. Analysis of the August 14 event may continue for many years, but it already seems clear that a major cause of the system disturbance was a gradual decline in voltage in Northern Ohio over a period of hours. If aggregate information had been available to neighboring control regions then there would have had ample time for each of them to take action to minimize the impact of this disturbance on the electric power grid.

One of the defining characteristics of the August 14 blackout was the voltage collapse that occurred on portions of the transmission system both surrounding and within the northern Ohio and eastern Michigan load center during the blackout. The transmission system's voltage is somewhat similar in function to water pressure. Sufficient voltage is needed to transfer electric power through a power network. Reactive power plays a major role in maintaining proper system voltages. Reactive power sources usually need to be close to reactive loads such as major load centers. The reason for this is that reactive power cannot travel long distance due to considerable loss in transmission lines. For an area with only limited local reactive supplies, increased power loads will cause the region's voltage to drop.

A so-called voltage collapse can occur if a reactive power shortage is severe. These system disturbances typically take minutes or even hours before they become serious enough to cause widespread disruption. As such, Astrolabe would be well suited to aid in monitoring and preventing this class of problems. We designed a series of experiments that mimicked the conditions that led up to the August 14 blackout in order to demonstrate Astrolabe's utility.

Our experiments centered on the (NYPP) New York Power Pool, a 3,000-bus system that is shown in Figure 1. The NYPP system consists of 2,935 buses, 1,304 generators, 1,883 loads, and 7,028 transmission lines with a base load of 11,553 MW. We used the New York system to model a much larger region of the electric power grid. That is, we divide the New York system into eight regions and assumed that each area was independently operated by its own control center. Each of these areas was assigned a letter from A to H. A sequence of outage events was created to illustrate the effectiveness of preventing major blackouts from spreading in the manner of the latest August 14, 2003 blackout via timely communication between control regions.

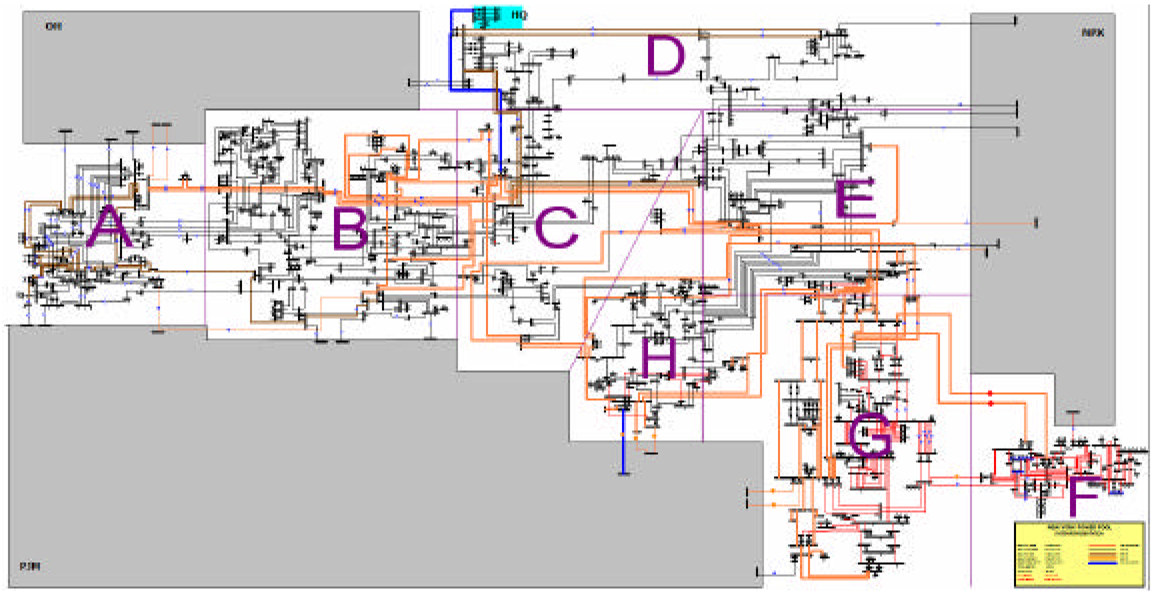

Figure 1: The New York Power System

The New York Power Pool's cascading sequence mimics the Northeast voltage collapse. The disturbance starts with a gradual load increase in Region F over relatively long period of time, for example, from noon to afternoon in a hot summer day. Region F is a "load pocket" in system. This means that large power imports are needed from the neighboring area, Region G.

Air conditioners utilize motors that require that they be supplied with reactive power. The aggregate increase in demand can become relatively large on the hot afternoon that we have envisioned for our experimental study. The demand for reactive power could normally be met by generators on the southeast corner of area G through tie lines if that became necessary, but the tie lines that connect F to G are unusually heavily loaded due to the region's large power imports. This means that the reactive power demand in Region F is mainly supplied by local resources.

The increasing load will create a slowly declining voltage profile in Region F. Region F will face voltage problems when significant numbers of bus voltages drop below a certain threshold. The North American Electric Reliability Council (NERC) suggests 0.95 per unit as a typical threshold value. In our scenario, a relay misoperation occurs in the afternoon after an hour of exposure to the system's low voltage profile. This operation opens up the *ShoreHam – WildWood* 138kV line, making the already inadequate voltage condition worse. *Five minutes* later, the *BrookHvn* Unit (rating 121MW) trips by the under-voltage protective relay. *Six minutes after that*, *HoltSvle-UnionAve* 138kV line trips, again due to low voltage, which also disconnects the *Union Avenue* Unit (rating 138 MW) from the power grid. This action creates a voltage collapse situation in Region F.

The voltage collapse in Region F causes two generators go offline. This in turn leads to a power mismatch in Region F that requires larger imports from Region G. Unfortunately, F-G tie lines were already run at close to capacity before the increased demand occurred. The added power demand overloads the *DunWoodie – Shore_Rd* 345kV tie-line connecting Region F and G. The corresponding overload protection relay reacts by opening up this tie-line. The other tie-lines between Regions F and G overload soon afterwards. The other three tie-lines, the *EgrdnCty – SprainBrook* 345kV line, the *LakSucss – Jamaica* 138kV line, and the *Jamaica – VallyStr* 138kV line, trip in quick succession due to the overloads. A blackout results leaving the load center in Region F unserved. And because Region G withdraws about 1160MW in this case from the tie-lines to Region F, there is a sudden large power mismatch in Region G causing the local frequency to speed up. The power mismatch in Region G is more than 25% of local capacity meaning that the frequency goes beyond the normal level of 60Hz to roughly 61 Hz or more. Most of generators will trip in this overfrequency situation. The outage will cascade into Region G and will potential spread farther into other regions if no appropriate actions taken. That is, if tie lines are not opened and the outputs of local generators are not decreased after a region's control center has sensed that the blackout is moving towards its area.

Although complex, this scenario shares many characteristics with the August 14 Northeast blackout. Mechanisms that could assist in preventing a New York State blackout in our example scenario could potentially be used on a larger scale to prevent a recurrence of the August 14 event.

We performed our experiments under three different situations. In the first, there was no timely communication between regions. Events unfolded according to the description that was just given and resulted in a large power blackout. In the second case, aggregate system information was shared between regions using standard Internet protocol TCP/IP. TCP/IP was chosen due to the momentum that the power industry has toward it widespread adoption in standards such as the Utility Communication Architecture (UCA). Finally, we ran a scenario where system information was between regions using Astrolabe.

TCP/IP connections were made using NS2's FullTCP implementation. Packets has a 0.1% chance of being dropped per link traversed. Nodes within the same region were typically within a few hops of each other making this an appropriate drop rate.

Astrolabe does not have a native NS2 implementation. However, Astrolabe has been studied extensively enough to let us model its behavior without implementing a new event-level simulation of the Astrolabe protocols per-se. With adopted this approach to model the propagation time and latency distribution for events reported through Astrolabe and communicated using its peer-to-peer gossip protocols. The approach is believed to yield realistic predictions about the behavior to be expected from the Astrolabe system if it were deployed in our experimental scenario.

With both TCP/IP and Astrolabe, SCADA data was collected and filtered at individual buses so that the current bus voltage could be determined. Bus voltages are aggregated at the control centers. We felt that the key information that neighboring control centers would need in determining the likelihood of an impending voltage collapse was the average voltage at the highest 10% and lowest 10% of all nodes in the region. A series of 11 steps were created to emulate a slow decrease in voltage in region F. The final step reduces the average voltage in region F to close to 0.82 and would result in a blackout in a real system. A graph of the voltage drop can be found in Figure 2. The goal of either the TCP/IP or Astrolabe information systems is to alert neighboring control centers to the impending voltage collapse so that appropriate action can be taken.

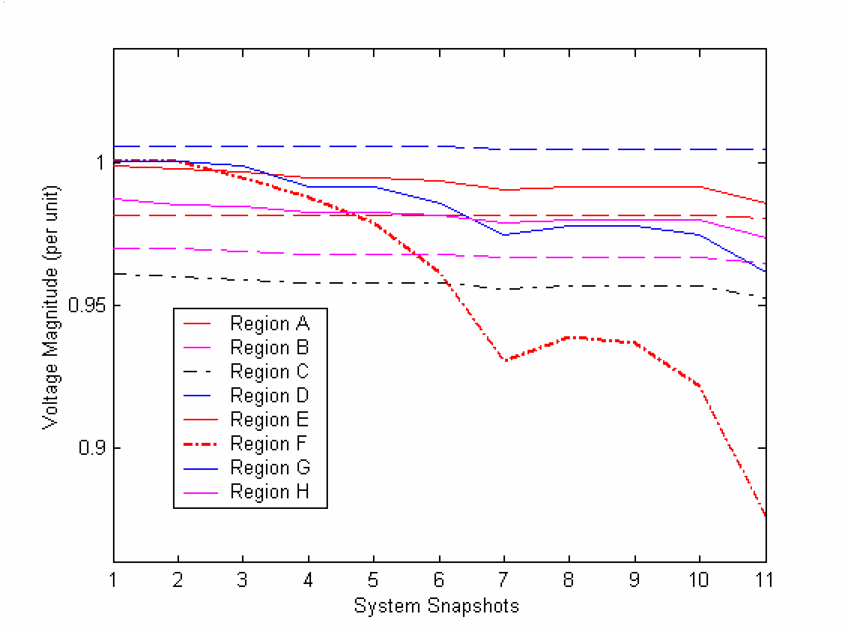

Figure 2. System snapshots of the average voltage for each region

After each snapshot, we ran an experiment in the EPOCHS environment to see how long it took for information about the previous event to travel to all control centers. The experiment began with the change in the system state and ended when all buses had reported their condensed state in aggregate to every control center in the system. Astrolabe's results were in a tight range between 14.1 seconds and 15.6 seconds. We repeated this process replacing Astrolabe's gossip process with TCP/IP connections. Our initial TCP trial took 7.1 seconds before all nodes had received notification of dangerous voltage levels once their presence had begun. However, notification times varied widely. For example, Region G was notified of the voltage situation in less than 0.1 seconds.

This trial was repeated placing a loss rate of 25% at each of the four tie lines connecting areas F and G. All other lines remained at a loss rate of 0.1%. The trial result showed that notification did not reach region F for 6.1 seconds compared with Astrolabe's 3.7 second time delay. Data had still not reached some control centers after *one and a half minutes* into the simulation in the TCP/IP case.

A 25% drop rate across tie lines is not unusual within a network build using standard Internet components and shared among multiple users, precisely for the reasons discussed in the first sections of this paper. TCP is designed to overload the network and will increase its data rates until packet loss is detected. Large data transfers of 2.4 MB or more have been proposed between a bus and its owner's engineering center after a fault has taken place. It is not hard to imagine a situation where a bus near the boundary between two regions is owned by a company that has an engineering center in its neighboring area. For example, a fault in Long Island might trigger a data transfer to a corporate center in New Jersey over the proposed Utility Intranet. Thus, during periods of potential concern, one would expect the Utility Internet to come under a variety of loads. By design, the data loss rate will rise rapidly and could often reach the 25% level. And when this occurs, a TCP-based monitoring infrastructure – even the simple one we analyzed in our experiments – will already begin to behave erratically. Astrolabe, on the other hand, remains stable, and indeed can be shown to remain stable even under extreme conditions, such as might be created in deliberate "denial of service attacks" (such as could arise if a virus or worm were to infect machines on the Utility Internet, or an intruder were to gain access).

The conclusion to be drawn from these experiments is that a robust protocol like Astrolabe is well-suited to monitoring the electric power grid for disturbances that take place over a time scale of minutes or more. TCP/IP is an excellent transport protocol for use when a network will not be heavily loaded. However, real networks often experience heavy traffic loads at unexpected times. This makes TCP/IP a questionable choice in real-time situations. This is particularly true when communication is used to support critical infrastructures like the electric power grid. Fortunately, the proposed infrastructure that would support TCP/IP traffic can easily support stronger protocols that are compatible with the IP standard like the Astrolabe system described in this document.

Conclusions

Our paper explored options for supporting a new generation of electric power system monitoring and control algorithms. We identified concerns about the tentative utilities internet standards, notably the assumption that standard versions of TCP running on standard platforms should be used to transport monitoring data. These issues include several stemming from the TCP protocol, and others reflecting the practical difficulty of configuring a utilities monitoring system using pair-wise connections and, as that system evolves through time, maintaining it. Used on a large scale, it also seems likely that a TCP monitoring infrastructure would suffer from performance problems, of the sort seen in the Internet when a web site becomes overloaded.

We also described the Astrolabe system, a flexible and extensible mechanism for large-scale monitoring, control and data mining. Astrolabe's peer-to-peer gossip protocols result in a system robust against many kinds of disruption, including the patterns of localized network disruption typical of network overloads and distributed denial of service attacks. Astrolabe's protocols "route around" such problems. A simulation study suggests that Astrolabe could be a valuable tool for solving some aspects of the utility network monitoring problem.

At the same time, we found that Astrolabe has some drawbacks. Information propagates reliably, but too slowly for notifications when an urgent event occurs. There may be security issues associated with the Astrolabe model, in which all users can see the database and aggregation results. And Astrolabe offers weak consistency, perhaps too weak for certain applications. Although we have proposed solutions for all of these problems, our initial evaluation is not detailed enough to validate this overall vision. Far more work will be needed.

We also described a model, which can be formalized, within which it is possible to derive an analytical prediction of Astrolabe's performance and robustness to disruption. Such analysis is confirmed by experiments on the current implementation, even under network conditions known to disrupt protocols using techniques other than the kind of epidemic peer-to-peer mechanism employed by Astrolabe For example, Astrolabe is known to maintain its reliability, with only slightly inc reased data reporting latencies, with message loss rates of as much as 50%.

Finally, we pointed to other work in which we apply similar techniques to study problems at smaller time-scales.

Our work supports several broad conclusions. First, we have come to believe that the power industry is allowing itself to drift towards standardization using technologies that just may not be the right choices for this setting. A scientific effort to evaluate options carefully before declaring them to be standards is ur gently needed. The industry will need to use the best-of-breed solutions if it is to solve the new generation of problems, resulting from a variety of trends: restructuring, the emergence of competitive markets, and the desire to operate the grid closer to its limits. Standardizing on the Internet protocols simply because they are widely used may not be appropriate: the Internet was not designed to support mission-critical monitoring and control software.

The good news in our study is that the same Internet infrastructure can be used in ways that work around at least some of the problems. Astrolabe is an example of the kinds of technologies that could respond to these needs, but is certainly not the only technology needed. A broad, industry-backed initiative to rigorously quantify the needs, systematically evaluate the options, and standardize on the best technologies will be required if we are to advance towards a power grid that is not merely restructured, but also as secure and reliable as the one from which it evolved.

References

[Lipton03] Overseers Missed Big Picture as Failures Led to Blackout. Eric Lipton, Richard Perez-Pena and Matthew Wald. *New York Times.* September 13, 2003.

[Coury02] An Agent-based Current Differential Relay for use with a Utility Intranet. D.V. Coury, J.S. Thorp, K.M. Hopkinson, K.P. Birman. IEEE Transactions on Power Delivery, January 22, 2002, Vol. 17, No 1, pp 47-53.

[Hopkinson03] EPOCHS: Integrated COTS Software for Agent-based Electric Power and Communication Simulation.2003 Winter Simulation Conference.7-10 of December 2003, New Orleans, USA.

[Birman03] Navigating in the Storm: Using Astrolabe for Distributed Self-Configuration, Monitoring and Adaptation Ken Birman, Robbert van Renesse, Werner Vogels. 5th Annual International Active Middleware Workshop (AMS 2003), Seattle , WA June 2003.

[Anderson01] Resilient Overlay Networks. David G. Andersen, Hari Balakrishnan, M. Frans Kaashoek, Robert Morris. Proceedings of the 18th ACM Symposium on Operating Systems Principles, Banff, Canada, October 2001.

[Lampson99] Computer systems research: past and future. Butler Lampson, invited lecture, ACM Symposium on Operating Systems Principles. 17th ACM Symposium on Operating Systems Principles. Dec. 12-15, 1999. Kiawah Island Resort, near Charleston, SC, USA

[VanRenesse03] Astrolabe: A Robust and Scalable Technology for Distributed System Monitoring, Management, and Data Mining. Robbert van Renesse, Kenneth Birman and Werner Vogels. ACM Transactions on Computer Systems, May 2003, Vol.21, No. 2, pp 164-206

[Birman01] Spinglass: Secure and Scalable Communications Tools for Mission-Critical Computing. Kenneth P. Birman, Robbert van Renesse and Werner Vogels. International Survivability Conference and Exposition. DARPA DISCEX-2001, Anaheim, California, June 2001.

[Birman99] Bimodal Multicast. Kenneth P. Birman, Mark Hayden, Oznur Ozkasap, Zhen Xiao, Mihai Budiu and Yaron Minsky. ACM Transactions on Computer Systems, Vol. 17, No. 2, pp 41-88, May, 1999.

[Coury00] Agent Technology Applied to Adaptive Relay Settings for Multi-Terminal Lines. D.V. Coury, J.S. Thorp, K.M. Hopkinson, K.P. Birman, Cornell University, Computer Science TR2000-1792, March 2000. Submitted to the IEEE Summer Power Conference.

[Jenkins01] A Gossip Protocol for Subgroup Multicast. Kate Jenkins, Ken Hopkinson and Ken Birman. International Workshop on Applied Reliable Group Communication (WARGC 2001), Phoenix, Arizona, April 2001.

[VanRenesse98] A Gossip-Based Failure Detection Service. Robbert van Renesse, Yaron Minsky, and Mark Hayden. In Proc. of Middleware '98, the IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing, pp. 55--70. England. September 1998.