Delivery Requirements and Implementation Guidelines for the NASPInet Data Bus

David E. Bakken, Carl H. Hauser
School of Electrical Engineering and Computer Science
Washington State University
Pullman, Washington, USA
{bakken,hauser}@eecs.wsu.edu

Harald Gjermundrød
Department of Computer Science
University of Nicosia
Nicosia, CYPRUS
harald@unic.ac.cy

Abstract— **The bulk power system faces many challenges as load growth continues to far outstrip new transmission capacity, utility operators are retiring in large numbers, and renewable sources of energy and microgrids with very different power characteristics must be integrated into electricity grids and be actively managed. NASPInet holds great promise to help improve the reliability, efficiency, and cyber-security of the bulk power system. In order to achieve this potential, NASPInet must support a wide range of guarantees for latency, rate, and availability. It also must support delivery with extremely stringent values of low latency and high availability, and do so across a wide area even in the face of potential IT failures and cyber-attacks. In this paper, we describe the baseline delivery requirements for performance and fault-tolerance that NASPInet must provide if it is to meet those goals. We then offer twenty implementation guidelines that we argue must be met if the delivery requirements are to be implementable and at reasonable cost. These guidelines are based on a number of sources, including our 11 years of designing and developing GridStat (an implementation of the NASPInet Data Bus), the experience of us and others on projects by DARPA and others involving wide-area, real-time, fault-tolerant, and secure middleware for wide-area networks, and the state of the art and practice in networking and distributed computing. Finally, we summarize the coverage of those delivery requirements and implementation guidelines that middleware, networking protocols IP Multicast and MPLS, and power protocols C37.118 and IEC 61850, provide.**

I. INTRODUCTION

Data deliver in the power grid today is, for the most part, hard-coded, tedious to implement and change, and does not provide any real end-to-end guarantees. The one exception to this is where isolated networks are used, for example for protection applications. In this case, there are no real QoS mechanisms provided by the network, but rather massive over-provisioning provides low latencies and high availability (at least in the steady state, but not necessarily in the face of IT failures, bugs in software or hardware that cause spurious traffic, or cyber-attacks).

This practice of using isolated networks will, in our opinion, soon become unsustainable as more applications, such as those outlined above, that can exploit coherent, real-time data delivery emerge. Further, so will the common practice today in the electricity industry of designing a new communication system for each new application (or family thereof). Fortunately, the state of the art in distributed computing, real-time systems, and fault-tolerant computing does support providing strong guarantees with data delivered to many applications. If done right, a data delivery system leveraging this state of the art (and state of practice in other industries) can be a disruptive technology: greatly lowering the barrier to entry (in both time and money) to deploy new power applications. If done wrong, it will be something that should be discarded in ~5 years because it cannot keep up with increasing demands. Further, these data delivery systems will have a long life, and no single network-level mechanism (for multicast or security or QoS) can be assumed to be everywhere. It is thus crucial that such data delivery systems for the power grid's IT backbone need to have interoperability between different kinds of network mechanisms providing the same property such as delay guarantees [1].

In this paper, we examine the conceived NASPInet Data Bus (NnDB), which is they only effort for grid modernization that is deeply considering data delivery issues. We first explain and justify the requirements which NnDB must meet in the areas of performance and reliability. We then present implementation guidelines, based on best practices in other industries and in the field of distributed computing systems, that we believe must be followed in order to achieve these requirements in NnDB, which by nature must be designed to have a long lifetime. Finally, we analyze how middleware, network protocols, and power protocols map onto those delivery requirements and implementation guidelines.

We note that there is a very strong inter-dependence between the power dynamics of the power grid and its data delivery infrastructure; for more information, see [2].

II. BASELINE DELIVERY REQUIREMENTS FOR NASPINET

We now overview delivery requirements (DR) that NnDB must meet [2,3] not including the details of cyber-security related ones. Here, we denote *QoS+* to mean not just the typical latency but also rate and criticality/availability and cyber-security.

A summary of these requirements and their rationale follows (they are denoted **DR*x*** below); more is in [2]:

1. **Hard, end-to-end (E2E) guarantees must be provided over an entire grid.**

   If the guarantees are soft or non-existent, then of course it is foolish to build protection and control applications that depend on the data delivery. The guarantees must thus be deterministic: met unless the system's *design criteria* have been violated (e.g., traffic amount, number of failures, and severity of cyber-attack).

2. **NASPInet must have a long-lifetime and thus be designed with future-proofing in mind.**

   This is crucial in order to make data delivery much easier to use and its costs amortized over many projects, utilities, grids, etc. The goal of NnDB is to last at least 30 years.

3. **Multicast (one→many) is the normal mode of communications, not point-to-point.**

   Increasingly, a given sensor value is needed by multiple power applications.

4. **End-to-end guarantees must be provided for a wide range of QoS+.**

   Data delivery for the grid is not "one size fits all" [3]. For example, to provide very low latencies, very high rates, and very high criticality/availability to all applications would likely be prohibitively expensive. Fortunately, many applications do not require these stringent guarantees, but their less stringent requirements must of course be met.

5. Some merging and future SIPS and control applications require **ultra-low latencies**, delivered (one-way) on the order **of** a half or full power cycle (8-16 msec in the US) over hundreds of miles and possibly across most of a grid [4]. Thus, any forwarding protocols must not add more than about one millisecond of latency (through all forwarding hops) on top of the speed of light in the underlying communication medium, which is roughly 100 miles/msec. Otherwise, many opportunities will be lost.

   These latencies must be provided in a way that is:

   a. **Predictable, and guaranteed for each update message**, not a (much weaker) aggregate guarantee over longer periods of time, applications, and locations like MPLS provides.

      Each sensor update needs to arrive within its required guaranteed deadline. As shown in [2], virtually all technologies widely deployed in today's best effort internet do not provide such per-packet guarantees. (ATM is a notable exception, but it does not provide multicast, and it is not a realistic end-to-end solution for the entire grid for other reasons.)

   b. **Tolerating (benign) failures** in NnDB.

      No system can possibly tolerate unlimited kinds and numbers of failures. However, much like the power grid must continue in the face of one or more knowable contingencies, the IT infrastructure on which it increasingly depends upon must provide these hard, E2E guarantees as long as its design criteria are met.

   c. **Tolerating (malicious) cyber-attacks**.

      Power grids, especially in the USA, are known to be subjects of extensive study and probing by multiple organizations that have significant information warfare capabilities, including nation states, terrorist organizations, and organized crime. Similar to benign IT failures, NnDB must adapt and continue to deliver data despite cyber-attacks of a designed severity (a bar which should be designed to be easily increasable over the life of the system).

6. Extremely **high throughput** is required.

   Today's synchrophasor applications are generally limited to 30 or 60 Hz in the USA, largely because the communications systems they use are not designed to support higher rates. To not provide much higher sustainable throughput would greatly limit the number of new applications over the next few decades which can help the grid's stability. Indeed, not just synchrophasors but digital fault recorders and IEDs in substations provide a wealth of data which is not tapped today. It is quite conceivable, and arguably likely, that "If you build it, they will come" and there will be many thousands of synchrophasors, DFRs, and other sources of sensor updates across a grid. Indeed, today's DFRs output at 720 Hz and are typically sample at 8 Khz.

These six requirements must all be met if power engineers are to justify depending on data delivery. We note that **we are not aware of any commercial or military market that has such stringent requirements, nor exploits the potentially controllable aspect, over a wide area**. In our opinion, these requirements are quite achievable, based on state-of-the-art in distributed real-time embedded (DRE) computing. That is, as long as a careful end-to-end (E2E) analysis is done [5], and in particular data delivery mechanisms are not saddled with unnecessary features. Such features can be provided by other mechanisms as part of NnDB, but they must not be allowed to impede the core mechanisms that must meet the stringent requirements outlined in this section.

III. Implementation Guidelines

The requirements outlined in the previous subsection were kept to a bare minimum. In order to achieve them, however, we believe it will be necessary to utilize a number of *implementation* guidelines (IG), many which are quite different from what is provided in today's best-effort internet and what has been the conventional wisdom in networking research.

Table 1 overviews these IGs and the DRs which mandate them. In this section we enumerate and explain these IGs, which are denoted **IG*x*** below:

1. *Avoid post-error recovery mechanisms*. Traditional protocols for the internet in general and reliable multicast protocols from the fault-tolerant computing research community use post-error recovery. Here the receiver either sends a positive acknowledgement (ACK) when it

TABLE I. IMPLEMENTATION GUIDELINES AND DELIVERY REQUIREMENTS THAT MANDATE THEM

DR1: Hard E2E WAN guarantees	DR2: Future-Proofing	DR3: Multicast	DR4: Wide Range of QoS+…	4a: Latency & Rate	4b: Criticality/Afailability	4c: Cyber-Security	DR5: Ultru-Low Latencies…	5a: Per-update & predictable	5b: Tolerating failures	5c: Tolerating Cyber-Attacks	DR6: High Throughput	IGx Prerequisited	*Summary of Implementation Guideline IGx*
X								X	X				IG1: Avoid post-error recovery mechanisms
X				X				X	X	X	X		IG2: Optimize for Rate-Based Sensors
		X									X		IG3: Provide Per-Subscriber QoS+
		X									X		IG4: Provide efficient multicast
												2,3	IG5: Provide Synchronized Rate Down-Sampling
X					X			X			X		IG6: Don't depend on priority-based "guarantees"
X	X			X	X	X							IG7: Provide E2E interop. across diff./new QoS+ mech.
X								X			X		IG8: Exploit a priori knowledge of traffic
X								X	X	X		8	IG9: Have systematic, quick internal instrumentation
X								X					IG10: Exploit smaller scale of NnDB
X								X				8-10	IG11: Use static, not dynamic, routing
X								X	X	X			IG12: Enforce complete perimeter control
X								X	X	X	X	12	IG13: Reject unauthenticated. messages quickly & locally
											X	2,8	IG14: Provide only simple subscription criteria
								X			X	2	IG15: Support transient, not persistent, delivery
								X			X		IG16: Don't over-design consistency & (re)ordering
											X	2,8,14-16	IG17: Minimize forwarding-time logic
X	X			X	X	X							IG18: Support >1 QoS+ mech. for diff. operating conditions
								X			X	17	IG19: Inspect only message header, not payload
X								X			X		IG20: Manage aperiodic traffic

receives a message or byte, or it sends a negative acknowledgment (NACK) when it concludes that the message or byte will not arrive. However, both add considerable latency when a message[1] gets dropped: three one-way latencies (OWL) are required plus a relatively large timeout. The better alternative is to send out sensor updates (messages) proactively over multiple disjoint paths, each of which meets the latency and rate requirements, which is what is done in GridStat [6–10]. Indeed, if multiple independent messages, each going over a QoS-managed path, cannot meet the delivery deadline, then sending ACKs or NACKs is very unlikely to help, and indeed will only make things worse.

2. *Optimize for rate-based sensors.* NnDB can be made with much higher throughput and robustness if it is not over-engineered. General-purpose publish-subscribe systems offer a wide range of traffic types, because they are designed to support a wide range of kinds of applications. However, in NnDB, the vast majority of the traffic will be rate-based. Design accordingly.

3. *Provide per-subscriber QoS+.* It is crucial that different subscribers to the same sensor variable be able to have different guarantees in terms of latency, rate, and criticality/availability. If not, then a lot of bandwidth will be wasted: all subscribers will have to be delivered that sensor's updates at the most stringent QoS+ that any of its subscribers requires. Such per-subscriber QoS+ is possible while still optimizing for rate-based sensors (DR2) [2].

4. *Provide efficient multicast.* In order to achieve the highest throughput possible, it is imperative to avoid unnecessary network traffic. Thus, never send an update over a link more than once. Also, as a sensor update is being forwarded through the network, if it is not needed downstream in the multicast tree (e.g., those subscribers require it at a lower rate than other subscribers), the update message should be dropped. We call this efficient multicast, which can best be implemented as *rate down-sampling* mechanism as is done in GridStat [6–10].

This guideline and the previous three add up to a need for multi-cast routing heuristics that provide multiple, disjoint paths to each subscriber with each path meeting the subscriber's latency requirement. A family of heuristics developed for this multicast routing problem [11,12] confirms the feasibility of the approach at the anticipated scale (see IG10) if routing decisions are made statically (IG11).

[1] We use the term "message" rather than "packet", because in many cases we are describing middleware-layer mechanisms above the network and transport layers. See Section IV.A for more information on middleware in this context.

5. *Provide synchronized rate down-sampling*. In providing rate down-sampling, it is import to not downsample in a way that destroys the usefulness of some data times. For example, synchrophasors are used to take a direct state measurement at a given microsecond. If some subscribers require only a small fraction of the updates for a set of synchrophasor sensors, it is important that the updates that are all passed through are the same for different synchrophasors that the application subscribes to. For example, if a subscriber only requires a tenth of the updates from two different variables, then it would not be useable to get Updates {#1, #11, #21, …} from one synchrophasor and then Updates {#2, #12, #22} from another synchrophasor, because the given measurements (e.g., #1 vs #2) do not measure the same exact time (they are not the same snapshot), which is the main point of synchrophasors.

6. *Don't depend on priority-based "guarantees"*. Publish-subscribe delivery systems typically offer a way to specify a priority, so if the traffic gets to heavy less important traffic can be dropped. However, this is not providing a hard end-to-end guarantee (DR1) to subscribing applications, and even applications that are not of the highest criticality still need their DRs to be met. Thus, given that they are not designed to provide absolute guarantees, don't build NnDB assuming they do! Instead of priorities, mechanisms must be used that exploit the characteristics NnDB (as outlined in these guidelines) to provide each subscriber firm assurances that its guarantees will be met so long as there are not more than the agreed upon number of failures or severity of cyber-attack. If such strong guarantees are not provided, then power engineers have no reasonable basis to depend upon the data delivery system, and opportunities for improving the grid's efficiency and reliability will be lost.

7. *Provide end-to-end interoperability across different/new IT technologies (providing QoS+: multicast, latency, rate, etc)*. A grid-wide NnDB will *ipso facto* have to span many utility and network organizations. It is unlikely that the exact same mechanisms will be present across all these organizations. And, even if they are today, if NnDB gets locked into the lower-level APIs and semantics of a given multicast or QoS mechanisms, it will be difficult to "ride the technology curve" and utilize newer and better mechanisms that inevitably be available over the long lifetime of NnDB; this is a stated goal of the GridWise community, for example [13]. Fortunately, it is possible to use middleware to span these different underlying technologies and provide guarantees that span this underlying diversity; in fact, middleware is required to meet the stated goals of the "smart grid" community [1].

8. *Exploit a priori knowledge of predictable traffic.* Internet routers cannot make assumptions or optimizations based on the characteristics of the traffic that they will be subjected to, because they are intended to be general-purpose and support a wide range of traffic types. NnDB, however, will have traffic that is not just rate-based, but it is almost all known months ahead of time (e.g., when an engineering survey is made of a new power application). This common case can be optimized, as described in later IGs below.

9. *Have systematic, quick internal instrumentation.* In order to provide E2E guarantees across a wide area despite failures and cyber-attacks, IG8 must be exploited to provide systematic and fast instrumentation of the NnDB. This allows for much quicker adaptations to anomalous traffic, whether accidental or malicious in origin.

10. *Exploit smaller scale of NnDB.* **This is a crucial if the challenging delivery requirements are to be met over a wide area with reasonable cost**. However, this requires rethinking the conventional wisdom in networking research and commercial middleware products. NnDB will be orders of magnitude less in scale than the internet at large[2], so it is feasible for the entire configuration to be stored in one location. Additionally, academic computer science researchers historically consider something that is $O(N^2)$ for path calculation with N routers or forwarding engines to be infeasible; see for example [14]. However, this assumption ignores two key factors for NnDB. First, N is not in the neighborhood of 10^8 as in the Internet, but rather is more likely $\sim 10^3$ at least for the next 5-10 years; this is very computable even if it is $O(N^2)$. (We note that this is the number of routers or pub-sub forwarding engines, which is less than the number of end locations/devices.) Second, as a rule, power engineers do not decide that they need a given sensor's values seconds before they really need it, due in part to the fact that today's data delivery infrastructure requires them to recode hard-coded socket programs and then recompile. Rather, power engineers plan their power contingencies (and what data they will need in them) months ahead of time with detailed engineering studies, and similarly for their monitoring, protection, control, and visualization needs. Thus, the routing/forwarding decisions involved in path selection can be done offline well ahead of time, while still allowing for handling a modest number of subscription requests at runtime.

    Finally, networking and security researchers generally assume that the membership of multicast groups (or a set of subscribers) may change rapidly; see for example [14]. However, as noted above, that is not the case with NnDB. Thus, security and other features need not be overly concerned about providing their capabilities for a group that is changing relatively often.

11. *Use static, not dynamic routing and naming.* Much stronger latency guarantees can be provided when using complete knowledge of topology coupled with static routing. The latter is a reasonable assumption in a managed NnDB, given that it will be a carefully managed

[2] For example, in the Eastern US grid there are on the order of 10K generators and busses, and in the entire USA there are approx 3500 companies that participate in the grid. We thus believe that the number of router-like forwarding engines that would be required for a NnDB backbone (at least in the case of broker-based publish-subscribed; define later) is at most 10^4 and likely only around 10^3.

critical infrastructure with complete admission control. Also, almost all of the sensors and power applications will be known well ahead of time, so optimizations for static (or slowly-changing) naming can potentially be useful and can be done while still providing more flexible and dynamic discovery services at a much lower volume.

12. *Enforce complete perimeter control*. All traffic put onto NnDB must pass admission control criteria (permissions based on both security and resource management) via a management system: the publisher registering the sensor variable (at a given rate) and the subscribers asking for a subscription with a given rate and E2E latency. This allows for **much stronger guarantees, and on a per-message granularity**, than is otherwise possible. It also enables quicker adaptations.

13. *Reject unauthorized messages quickly and locally*. Messages that have gone around the admission control perimeter should be rejected as soon as possible, ideally the next NnDB forwarding engine, rather than going most or all the way across an entire grid through NnDBs, and hence consuming many more resources. This detection of such unauthorized packets is of course anomalous traffic and evidence of a failure or cyber-attack that needs to be reported to the management infrastructure. When sufficient evidence over sufficient time is collected, an appropriate adaptation can occur.

14. *Provide only simple subscription criteria.* This is exactly the opposite of what is usually done with academic research and commercial products: both strongly favor a complicated subscription criteria which is inevitably more expensive to calculate when each update is forwarded through the system (think of complex "topics"). For example, in GridStat, the subscription criteria are latency, rate, and #paths, and, as noted below, the forwarding decision is done completely based on rate, with static routing. Note also that the lower-level ID of a sensor variable could still be looked up through a complicated discovery service; this guideline is concerned with avoiding unnecessarily runtime forwarding logic.

15. *Support transient delivery only, not persistent delivery*. Most publish-subscribe systems offer persistent delivery, whereby if an event cannot be immediately forwarded it is stored for some time and then the delivery retried. This harms throughput, however, as well as potentially the per-packet predictability (because it requires storing the data). In our experience, it is completely unnecessary for real-time visualization, control and protection, due to the temporal redundancy inherent in rate-based update streams: the next update will be arriving very soon anyway, so the usefulness of a given update fades very quickly. Further, multiple disjoint paths provide additional availability. Thus, it is inadvisable to complicate delivery mechanisms to support persistent delivery (it can be provided "on the side" by other mechanisms). Further, in the power grid, historian databases are already required for archiving data; there is no reason to complicate the design or otherwise bog down the fastest and highest availability mechanisms of NnDB to delivery historical data.

16. *Don't over-design unnecessary consistency and (re)ordering*. Research in fault-tolerant multicast tends to provide different levels of ordering between updates from the same publisher, or between different clients of the same server, as well as consistency levels between different replicas or caches of a server. There is no need for anything like this in NnDB, for three reasons. First, in our experience, IT personnel in today's grids rarely have learned of such advanced delivery properties. Second, even for those that do, their present data delivery software provides no kind of consistency at all, so power applications assume nothing in terms of consistency and ordering (other than. The only such consistency that we have found is required is IG5 for synchrophasors, and the only ordering of any kind is where a PDC combines updates from different PMUs into one message to pass on (with errors corrected, computed angles added, etc). Third, with devices such as synchrophasors that have accurate GPS clock the order of events can be directly known and no delivery ordering mechanism is required other than that which is done by a PDC.

17. *Minimize forwarding-time logic*. In order to provide the highest throughput, the forwarding logic that decides how a packet or update is to be forwarded on should be kept as simple as possible. On the GridStat project, however, forwarding decisions are done based solely on the subscription rate of subscribers downstream in the multicast tree [6–10]. Given that the traffic is rate-based (IG2), known ahead of time (IG8), subscription criteria are kept simple (IG14), only transient delivery is supported (IG15), and there are no consistency mechanisms (except IG5: IG16); it follows that much logic can be pushed off to subscription setup time or even offline. This can drastically reduce the logic necessary when an update arrives at a forwarding engine (or peer-to-peer middleware mechanisms at an edge) and hence greatly improve throughput and latency as well as their predictability.

18. *Support multiple QoS+ mechanisms for different runtime conditions*. A given mechanism that provides guarantees of latency and security, for example, will not be optimal (or perhaps even appropriate) for all the runtime operating conditions for which a long-lived NnDB may have to operate with. This is because different implementations of a given QoS+ mechanism can require very different amounts of lower-level resources such as CPU, memory, and bandwidth [15].

19. *Inspect only packet header, not payload*. In order to provide the highest throughput, ensure that subscription criteria and consistency semantics allow a forwarding decision to be based solely on a packet header. Unfortunately, this is not possible for publish-subscribe middleware that has complicated subscription topics as is typical with commercial and research systems. For them, data fields in the payload also have to be inspected.

20. *Manage aperiodic traffic*. Any traffic that is aperiodic (i.e., not based on rate but on a condition) must be isolated from rate-based periodic traffic and managed accordingly. This can be done deterministically, for example with (OSI

Layer 1) optical wave division multiplexing (OWDM) hardware. Further, aperiodic traffic should be aggregated intelligently rather than sending all alarms/alerts to a central location within a utility or a grid.

IV. Meeting the Delivery Requirements and Implementation Guidelines

We now summarize how middleware, networking protocols, and power protocols meet (or don't) the delivery requirements and implementation guidelines presented above. We note that the services that will support NASPInet are generally not required to be nearly as high performance as the NnDB. They will also likely be hierarchical [6,14].

A. Middleware

Simply put, middleware is required to meet the stated goals of smart grids, by the state of the art today [1]. Similarly, it is important to note that meeting IG3 (and others) requires the data delivery system to be provided at the middleware layer. This is because network-level mechanisms, for example providing some delay guarantee, only know about packets and IP addresses. They do now know of, let alone provide guarantees on the basis of, middleware-layer sensor variables and the power applications that require guarantees on the granularity of a particular sensor variable.

B. Networking Protocols

Traditional networking protocols are not designed to provide most of the DRs and IGS [16]. In particular, **the *combination of MPLS and IP Multicast (which some recent projects are using for a stand-in for NnDB) does not provide many of the DRs and IGs that NnDB needs.***

C. Power Protocols

Similar to networking protocols, power protocols such as DNP3, IEC 61850, and C37.118 simply are not designed to provide any of the DRs over a wide area, and certainly employ none of the IGs. However, we note that C37.118 is being extended to be able to utilize different underlying transports. Thus, using the NnDB for this transport could meet those DRs and utilize all IGs.

Acknowledgment

This research was funded in part by Department of Energy Award Number DE-OE0000097 (TCIPG). Disclaimer: "This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof."

References

[1] David E. Bakken, Richard E. Schantz, and Richard D. Tucker. "Smart Grid Communications: QoS Stovepipes or QoS Interoperability", in *Proceedings of Grid-Interop 2009*, GridWise Architecture Council, Denver, Colorado, November 17-19, 2009. **(Best "connectivity" Paper award.)** http://gridstat.net/publications/TR-GS-013.pdf.

[2] David E. Bakken, Anjan Bose, Carl H. Hauser, Edmond O. Schweitzer, David E. Whitehead, Greg C. Zweigle. "Smart Generation and Transmission with Coherent, Real-Time Data"[3]. Technical Report TR-GS-015, Washington State University, August 2010.

[3] Electric Power Research Institute (EPRI), The Integrated Energy and Communication Systems Architecture, Vol. IV: Technical Analysis, 2004.

[4] Horowitz, S. Novosel, D. Madani, V. Adamiak, M. "System-Wide Protection", *IEEE Power & Energy Magazine*, 6(5), September 2008, 34-42.

[5] J. Santzer, D. Reed, and D. Clark. "End-to-End Arguments in System Design". Transactions on Computer Systems, Association of Computing Machinery (ACM), 2(4), November 1984, 277–288.

[6] K. Harald Gjermundrød, Ioanna Dionysiou, Carl Hauser, Dave Bakken, and Anjan Bose "Flexible and Robust Status Dissemination Middleware for the Electronic Power Grid". *Technical Report EECS-GS-003*, School of Electrical Engineering and Computer Science, Washington State University, September 2003.

[7] K. Tomsovic, D. Bakken, M. Venkatasubramanian, and A. Bose, *Designing the Next Generation of Real-Time Control, Communication and Computations for Large Power Systems*. In *Proceedings of the IEEE* (Special Issue on Energy Infrastructure Systems), page 93(5), May 2005.

[8] K. Harald Gjermundrd. "Flexible QoS-Managed Status Dissemination Framework for the Electric Power Grid". Ph.D. Dissertation, Washington State University, 2006.

[9] D. Bakken, C. Hauser, H. Gjermundrød, and A. Bose. "Towards More Flexible and Robust **Data Delivery for Monitoring and Control of the Electric Power Grid",** *Technical Report EECS-GS-009*, Washington State University, May 2007.

[10] K. Harald Gjermundrød, David E. Bakken, Carl H. Hauser, and Anjan Bose. "GridStat: A Flexible QoS-Managed Data Dissemination Framework for the Power Grid", *IEEE Transactions on Power Delivery*, 4(1), 2009, 136–143.

[11] V.S. Irava, and C. Hauser, "Survivable low-cost low-delay multicast trees," Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM), Nov. 2005.

[12] V.S. Irava, "Low-cost delay-constrained multicast routing heuristics and their evaluation," PhD Dissertation, Washington State University, August 2006.

[13] GridWise Architecture Council, *Interoperability Constitution Whitepaper (v1.1)*, December 2006

[14] Rakesh Bobba, Eric Heine, Himanshu Khurana, and Tim Yardley. "Exploring a Tiered Architecture for NASPInet", in *Proceedings of the IEEE Conference on Innovative Smart Grid Technologies*, Gaithersberg, MD, January 2010.

[15] Zinky J., Bakken D., Schantz R. Architectural Support for Quality of Service for CORBA Objects. Theory and Practice of Object Systems, April 1997.

[16] K. Hopkinson, G. Roberts, X. Wang, and J. Thorp, *Quality of Service Considerations in Utility Communication Networks*, IEEE Transactions on Power Delivery, 24(3), July 2009.

[3] This is a preliminary (and expanded) version of an invited paper for a 2011special issue on smart grids of Proceedings of the IEEE.