# Exploring Convergence for SCADA Networks

Erich Heine, Himanshu Khurana *Senior Member, IEEE,* Tim Yardley *Member, IEEE*

***Abstract*—This paper describes the design and implementation of the CONES (COnverged NEtworks for SCADA) real-time middleware toolkit aimed at supporting convered SCADA networks. CONES provides 1) host based process management, 2) network resource management and 3) host based network resource management. These capabilities collectively provide a basis for convergence of SCADA networks while ensuring that isolation and timeliness properties of all SCADA applications (e.g., protection, monitoring and maintenance) are supported as per the requirements of those applications. We provide an architecture, an advanced prototype implementation and extensive experimentation results to demonstrate the feasibility of CONES.**

## I. Introduction

In this work, we consider SCADA systems (especially in power systems) and the digital communication systems between control centers and end-use devices (e.g., PLCs and relays) that provide protection, monitoring and maintenance. SCADA systems provide centralized monitoring and control of field devices spread over large geographic areas.

SCADA systems use a wide variety of networking technologies to enable transmission of field data from field sites to the control centers, and of control information from control centers to field sites. Once field data reaches the control system, software systems provide capabilities to visualize and analyze the data. Based on automated or human-driven analysis, action is taken as needed such as recording the data, processing alarms, or sending control information back to field sites. This data acquisition and supervisory control is undertaken by a combination of hardware and software systems connected by a multitude of networking technologies. Hardware includes PLCs, Remote Terminal Units (RTUs), IEDs, relays, SCADA servers (a.k.a. Master Terminal Units or MTUs), communication routers, workstations, and displays. These hardware systems run a variety of software such as data input and output processing, data transfer protocols, state estimators, visualization tools, data historians, equipment controllers, alarm processing and reporting, and remote access. All of these hardware and software systems are connected via local and wide-area networks depending on their proximity. Standard and proprietary communication protocols running over serial communications are used to transport information between the control center and field sites via communication mediums such as telephone line, cable, fiber, and radio frequency, such as broadcast, microwave, and satellite.

Exchange of data and commands over SCADA networks provides support for several classes for applications such as protection, monitoring and maintenance. Typically, operational data supports protection, non-operational data supports monitoring and maintenance data supports maintenance. These applications require isolation from each other and timely delivery of data and commands to support the overall real-time needs for grid applications.

Common architectures deployed in the field today use multiple physical networks to carry different types of data and ensure isolation and timeliness properties. This approach is based on the accepted use of dedicated hosts and networks for such critical applications. Furthermore, many of the hosts are special-purpose hardware systems. In contrast, use of Quality of Service (QoS) managed solutions [1] offer the potential for integration of all SCADA services (protection, monitoring and maintenance) and applications over a single integrated network. Not only would such an approach greatly reduce management overhead as control centers would have to manage only a single network and its redundant backup, it would also allow for easy deployment of new applications that can execute safely over existing shared resources. There is extensive relevant research in the areas of QoS, real-time systems and resource management such as [2], [3] and [4], [5].

A primary objective in this work is to design, implement and experiment with a real-time middleware system that allows convergence of SCADA networks and, in turn, allow existing and emerging SCADA applications and services to benefit from high-speed networking abilities. A key observation supporting this work is the increasing use of commercial platforms and operating systems (e.g., Windows and Linux) in SCADA and other power grid environments. These systems first motivate the need for developing such a real-time middleware solution because they typically lack effective resource management but then also allow for development of the middleware because of their inherent flexibility.

In this work we first conduct a detailed study of SCADA application requirements and demonstrate the need for strong host-level CPU and network resource management capabilities that support the ability to meet millisecond-level deadlines. Based on these requirements we design, implement and experiment with our real-time middleware solution, CONES: COnverged NEtworks for SCADA. The middleware provides 1) host based process management, 2) network resource management, and 3) host based network resource management. The primary contributions of this work are 1) a study of SCADA application requirements to specify resource management needs for convergence of SCADA networks, 2) a

All the authors are with University of Illinois at Urbana-Champaign. e-mail: {eheine,hkhurana,yardley}@illinois.edu

This material is based upon work supported in part by the Department of Energy Contract No. DE-AC05-76RL01830 managed by the Pacific Northwest National Laboratory and in part by the Department of Energy under Award Number DE-OE0000097 We would like to thank Jeff Dagle, Bill Sanders, David Nicol, Klara Nahrstedt, Rakesh Bobba and Pete Sauer for discussions on this project. We would like to thank our industry partners at Entergy, Tennessee Valley Authority and SISCO for feedback on SCADA application characteristics. We would also like to thank Long Vu for related survey on real-time middleware tools and technologies.

practical implementation on the Linux platform that provide fine-grained host-level CPU and network resource management, 3) extensive experimentation of an integrated system in a testbed environment to demonstrate feasibility. The CONES implementation leverages the iDSRT software implementation in [3] and complements ongoing work in the GridStat project that looks at QoS over wide area power grid networks such as those for supporting sharing of synchrophasor data [6].

The rest of this report is organized as follows. Section 2 discusses requirements. Section 3 provides our architecture and approach. Section 4 describes our implementation. Section 5 presents the experimental setup and results. Section 6 discusses related works. Section 7 concludes and discusses future work.

## II. Requirements

Our vision for this work is to develop a communication system that supports convergence of SCADA applications while ensuring isolation and timeliness properties of individual applications. Furthermore, in keeping with product development and deployment trends in the electric sector we aim to do so by using commercial platforms and operating systems. This section explores and enumerates technical requirements that need to be met by the desired communication system.

As a first step in the process we studied some common SCADA applications and their Quality of Services (QoS) requirements including bandwidth, latency, priority, etc. Figure 1 below provides one summary of our study that was refined by feedback from several electric sector industry experts. This summary focuses on identifying a few key applications across the various classes and studying their properties with an emphasis on bandwidth requirements.

In addition, IEEE Standard 1646 [7] provides a detailed study of communication delivery performance requirements for substations. This standard details data delivery time ranging from 4 ms to minutes for supporting a variety of applications. The standard includes detailed timing on tens of applications. We identified two key applications that drive the technical requirements below:

(1) *Peer-to-peer substation protection*: this application requires occasional communication between IEDs in a substation with high priority and a data delivery duration of 4 ms.

(2) *System protection with synchrophasor data*: this application requires high rate communication (30 to 120 samples per second) to control centers from substations with medium priority and data delivery duration of 20 to 30 ms.

After studying these applications and general QoS requirements we have identified the following three technical requirements for achieving our goals for a converged network for SCADA. It is our vision that these requirements be met using commercial platforms and operating systems. This will allow for widespread deployment of cost effective solutions. A particularly noteworthy implication of this is it includes the use of IP networking technologies for communications.

(1) *Host-based Process Management*. It is well understood that in order to support real-time applications, CPU resources at hosts must be managed. On commercial platforms and operating systems, this management includes the ability to associate priorities and deadlines with processes and then scheduling the processes based on priorities/deadlines [8], [9]. In practice, one can achieve soft real-time guarantees that matches requirements for a range of applications [2]. Our requirements are to meet the end-to-end real-time requirements for SCADA systems outlined above and exemplified by the two key applications of substation and system protection.

(2) *Network Resource Management*. Again, it is well understood that in order to support real-time applications that involve data delivery over router networks, there is a need to provide QoS assurances over the network. This includes managing network bandwidth, delay, jitter and packet loss. For IP networks, common technologies used to provide this management include IntServ [10] DiffServ [11], [12], and MPLS [13]. Our requirement in this area is to leverage some of these existing technologies but integrate them in overall architecture to provide end-to-end real-time guarantees for SCADA applications.

(3) *Host-based Network Resource Management.* As part of the system-wide QoS objectives, it is important to manage the network resource at the host level; in particular, managing the scheduling of *outgoing* packets at each host machine. This has been discussed recently in the context of wireless network where such scheduling intuitively affects real-time delivery [14], [15]. However, this has not been considered much of an issue in wired networks where gigabit capacity networks offer seemingly sufficient resources. In this work, we demonstrate that such network resources need to be managed to meet strict deadlines in typically SCADA environments. Our requirements are then to develop solutions that provide host-based network resource management and integrate them in an overall architecture to provide end-to-end real-time guarantees for SCADA applications.

## III. Architecture and Approach

In this section we discuss our approach of building a real-time middleware system to address the requirements identified earlier. We call this middleware, "CONES: COnverged NEtworks for SCADA", to identify its primary objective of enabling the use of a single physical network to carry data and information for all current and future SCADA applications.

Using middleware to address the needs of managing distributed real-time and embedded systems with a distributed computing infrastructure in a flexible and cost-effective manner is considered to be an effective approach [2]. We adopt this approach in our work with an emphasis on low level properties and coordination to achieve the millisecond level deadlines identified earlier. Recently, *iDSRT* has been proposed as an example middleware solution that explores the use of wireless network for power grid environments [3]. We build on *iDSRT* and use many of existing features while significantly enhancing others to form one of the core components of the CONES system.

Figures 2 and 3 below describe the CONES vision and architecture respectively in the context of SCADA networks. We envision SCADA host machines (e.g., IEDs, aggregators)

| Power Systems Application | Traffic Type | Traffic Path | Qualitative Quality of Service (QoS) Parameters | Packet Characteristics (size, timing) per device | Scalability considerations | Stream Bandwidth Characteristics (per device, total) |
|---|---|---|---|---|---|---|
| **Protection/ Control** | SCADA | IED(substation) -> Control Center | Low latency, high priority, no loss | Size: 256B – 1KB Frequency: 1 packet every 2-4s | ~5 devices per bus | .5KB/s per device 2.5-5KB/s per bus |
| | SMV/ GOOSE | IED -> IED | High speed/low latency[1], high priority. | Size: typically less than 1 Ethernet frame Frequency: | 1 event per second per bus | 1-15KB per protection event |
| **Monitoring** | PMU | IED/PMU -> Phasor Data Concentrator (Control Center) | Low latency, medium priority. | Size: 128 Bytes Frequency: 30 – 120 samples/sec | 2 PMUs per bus | 30Kbps per device, 60Kbps per bus |
| | Other Monitoring Data | IED/master -> Control Center | Low latency, medium priority. | Size: 32-64 Bytes Frequency: 1 sample/ms | 20-25 Devices/substation | 256-512Kbps per device 1-5 Mbps per substation (not all data leaves the substation) |
| **Engineering** | Interactive[2] | Control Center <-> Substation | Medium latency, medium priority | N/A (these are not critical timings and can vary greatly) | | 1M per occasional request |
| | Data Transfer[3] | Control Center <-> Substation | Low priority | N/A (Big packets, but not a standard size) | A flow 1-2 times per day | 1-5M per occasional request |
| **Surveillance** | Video | Substation -> Control Center | Medium – High latency, medium priority. | Varied video frame sizes and rates | 2-10 cameras per substation. | 100 Kb/s -1Mb/s per camera[4] ~5Mbps per substation |

[1] Assuming a 1-2 cycle requirement for the communications to complete, and decisions to be reached.
[2] Traffic of this type is assumed to be from a human at a terminal, e.g. ssh traffic.
[3] This is data transfer, either for backup, or new configurations, or non-critical data.
[4] Cameras are assumed to be low quality, low frame rate, unless something "interesting" is happening, at which point frame rate and quality increase. Interesting is defined as an occurence worth capturing in higher detail.

Fig. 1. Key SCADA applications and their bandwidth requirements

to be connected via IP LAN technologies in a substation environment and that a subset of these host machines are connected to Control Center host machines (e.g., SCADA master, engineering workstation) via IP WAN technologies. All host machines use an enhanced Linux kernel where the CPU scheduler and networking stacks are modified and the term "Smart" denotes such enhanced host machines. Furthermore, applications use our developed CONES API to utilize these modified kernel capabilities to manage CPU and networking resources at these hosts. On the WAN side we envision use of available tools for network provisioning and QoS assurance such as MPLS and DiffServ.

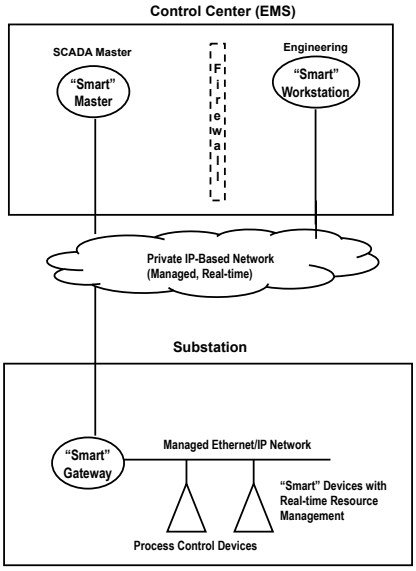


Fig. 2. Envisioned architecture for SCADA networks

## IV. Design and Implementation of CONES Real-Time Middleware

To meet the goals laid out in the Requirements and Architecture sections a middleware approach has been deemed most appropriate. This allows for simple application integration, while allowing for CPU and network resource management, across distributed SCADA hosts and providing soft real-time guarantees. To build this middleware several components had to be integrated, across various layers of systems in question. These layers include the operating system, at kernel and system library levels, as well as application layer programming interfaces. Further, all of the components needed to be readily available, off the shelf systems, or easily integrated with such systems.

For real-time scheduling we chose *iDSRT* for two reasons: it was a Linux extension, meaning it worked well in our environment, and it was a Linux-based real-time solution that also included explicit network resource scheduling. The architecture of *iDSRT* is fully described in [3]. In summary, the software uses the Earliest Deadline First (EDF) algorithm to schedule network traffic and CPU time. It uses both network and CPU deadlines in factoring its scheduling, allowing for

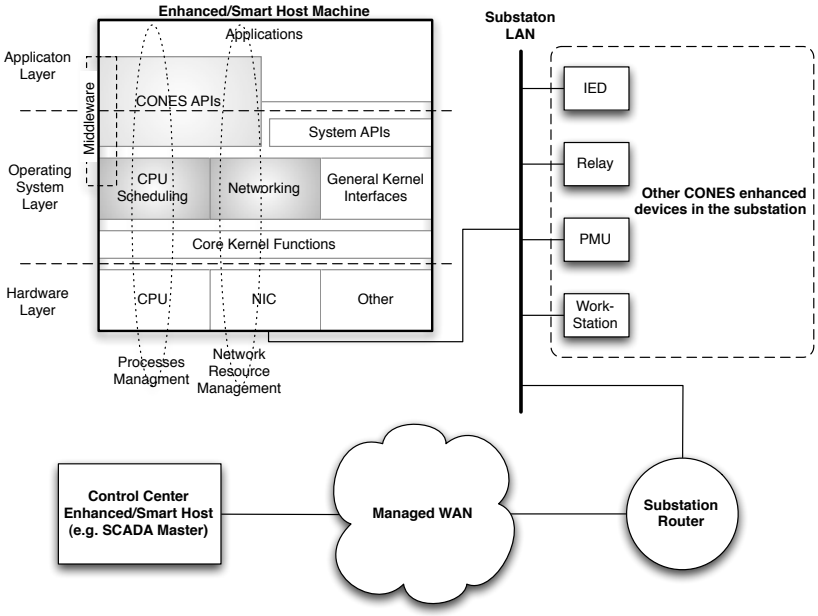


Fig. 3. CONES Architecture and Approach

a finer grained control of resources. Further, *iDSRT* takes advantage of a coordination service, which allows network coordination amongst various hosts. This service was designed for use on wireless networks where the radio spectrum is limited and frequently in contention via transmission collisions. As CONES is designed for use on a full duplex media, this service is not needed, and therefore not used.

For CPU scheduling, *iDSRT* uses EDF to select which process to run based on a known deadline, known worst-case run-time, and period, allowing for periodic time-bounded tasks. As a consequence of this selection process, all tasks will complete fully provided the CPU is not over-scheduled. To implement the EDF algorithm on Linux, the *iDSRT* team created a Linux kernel scheduler module which ties into the basic Linux process scheduling routines. If there are no EDF tasks which need to run, the module falls back to the regular best-effort scheduler. Additionally processes which overrun their allotted time run in "best effort" scheduling. For the purposes of CONES, the CPU scheduler of *iDSRT* was sufficient, and no enhancements were made.

The networking scheduler in *iDSRT*, called iEDF, also implements the EDF algorithm. To schedule network resources iEDF hooks into the packet queuing subsystem of the Linux networking stack. This subsystem is the lowest layer of the stack before device drivers. This effectively gives the scheduler control of the networking resource for the purpose of packet selection. Each time a packet is transmitted, iEDF first selects packets based on timing parameters and second based on the default Linux queuing mechanism. This allows for non-scheduled network traffic as well as scheduled traffic. The algorithm used in iEDF is actually an extension of EDF called Implicit EDF. The algorithm allows for the scheduling of network resources for non-existent tasks. This functionality is used by the coordination service in *iDSRT* for scheduling transmission over shared resources, preventing collisions by reserving the network for times a remote host is also transmitting. The implicit capability is not currently used in CONES, however future work may do so.

While the *iDSRT* project provided a good base, there were three major modifications to *iDSRT's* iEDF module to work in the CONES environment. The first deals with how iEDF tracked real-time processes. Prior to CONES, it simply tracked processes by the order in which they registered themselves as real-time. Further these were limited in number to 8 processes, due to iEDF's use of TOS bits to track and identify real-time packets (see below). The modifications now track scheduling info based on process id (PID). Specifically, the iEDF module maintains an array of scheduling information objects. The objects are assigned on a first come first served basis to processes registering, and a hash table is used to track the relationship of a process to its scheduling information.

The second modification is in the identification and processing of "real-time packets". In the original implementation this was done manually in the application code, modifying the IP packet's TOS bits to contain the id used for the scheduling information look-up. The size of the TOS field in an IP packet is 3 bits, making a limit of 8 real-time processes. The above mentioned id assignment, and PID mapping is also used for packet queuing. This frees the TOS bits to be used for its original purpose.

The third enhancement was to add a socket option to the Linux kernel. This is the widest reaching change needed for CONES on Linux. The option, SOCK_REALTIME, adds a socket option to the kernel to allow sockets to identify themselves as needing real-time scheduling. This combined with the look-ups based on PID allows a real-time program to have multiple traffic flows that are real-time scheduled, it further allows the program to have real-time and non-real time

(best effort) traffic simultaneously. This provides much more flexibility to the system.

A fairly substantial stylistic set of enhancements were also added to iEDF. This second set of enhancements focuses on standards compliance. The code has been modified to use standard Linux kernel constructs where possible. Examples include the use of appropriate macros in place of direct pointer dereferencing, use of kernel optimized linked lists and bitmaps instead of custom implementations, and the use of kernel defined data structures where possible.

## V. Experiments

### A. Testbed

In all cases the physical network is set up according to the plan in Figure 4. The control switch and network exist separate from the testing network, so all participating machines can be controlled as needed without interfering with the network. Further the control network is connected to the extended network allowing access for tests remotely. Host A is the computer which runs the virtual machines, in their CONES enhanced and generic states. The bridge on this host is a virtual layer 2 bridge connecting virtual machine to the physical network.

Host C does not participate in the timing experiments directly, instead it provides the time source for the Network Time Protocol(*NTP*) services, and when necessary for contention, provides extra network traffic to the appropriate recipient.

Host B is the host which receives all "forward" timing packets. These are the packets which have been sent by VMs for the purposes of timing. For more details please refer to the section on testbed setup.

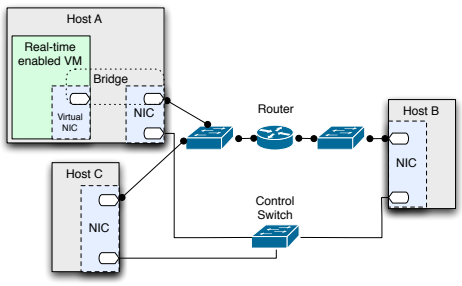


Fig. 4. Physical setup of the test bed

These physical setups are to simulate two different networks. The first is a simulated LAN environment. The LAN is routed, as is the case in many larger LAN environments, particularly when various security zones and administrative zones are required. The router is capable of full 1000Mbs bandwidth, as are all NIC cards in the hosts. In this setup an emulated IED and emulated substation computer communicate across a router. The IED in that architecture maps to the Virtual machine in Figure 4. The substation computer is Host B in Figure 4.

### B. Data Generation and Timing

We created a data generation tool called *trafficgen*, and a real-time extension called *rtgen* for emulating power and control networks. These tools are based on a popular networking library called *libevent*, and event-based framework which has minimal overhead, and provides determinism useful in timing. *trafficgen* emulates various scenarios by transmitting or exchanging messages following data flow and timing patterns found in various IEDs. This flexibility allows the same tool to emulate synchrophasors, various control device such as relays and breakers, as well as other sensors and IEDs. Further, each emulated device can be tracked separately or as part of a class, allowing for a wide variety of statistical analysis.

*rtgen* is a small subset of functionality from *trafficgen*. This software is designed to run tied into the cones middleware layer. This software is minimal in that it only allows one-way traffic emulation, for devices such as synchrophasors, or scenarios such as report-by-exception. This software is used to gather data on packet transmission in the real-time scenarios.

Timing is accomplished by using the host clock, which is synchronized over the network connections shown in Figure 4 which connect to the "Control Switch". These connections are out of band for the experiments, preventing interference between experimental traffic and time synchronization. The synchronization is accomplished via a modified NP, which sets the absolute time of each host rather than adjusts clock speed. This method allows for a maximum observed clock skew of 200 microseconds, and a average skew to be 75 microseconds.

### C. Experimental Cases

*1) Case 1: Host based process management:* The first test case examines the requirement for host base process management. This management is achieved in CONES with by using the real-time extensions, based on *iDSRT*, as described above. The need for process management becomes apparent when considering that many computing devices have a variety of concurrent tasks running to achieve functionality. A software bug, or even an unexpected confluence of events can result in a CPU contention scenario.

IEDs frequently need real-time capabilities for sensing capabilities and event response, computation time however is notoriously variable. Some data or events may require more processing than others. Frequently the confluence of several such cases can result in CPU contention at an inappropriate time. Other causes of CPU contention can be software bugs, or operating system level issues, such as heavy networking loads. In any case, CPU contention can create situations which cause computational deadlines to be missed. They can also create situations in which network latency is affected, if the software which sends data via the network cannot get the appropriate CPU time, the packets will be delayed.

To measure such effects, CPU contention must be created separate from network contention. We achieve this by using a program called "burn" (a component of the wrekavoc project [16]). The program takes a single parameter, which is the ceiling of CPU utilization by the program. To highlight the effects of CPU contention, several instances of the burn program are run, increasing the effects of contention and reducing out the noise.

Measuring the difference between a base operating system, and a CONES operating system is done by running the same

tests on an instance of each. The contention is created on the target host. Base traffic is then applied. The receiving host of the base traffic is under-utilized, to have minimal impact on the timings. The network used for these tests is the simulated LAN described above in the testbed section.

The results of these tests can be seen in Figure 5. Each chart is a histogram of per-packet network traversal times. The base traffic was run across the network under test conditions in six runs, each lasting 20 minutes. The two curves on the chart represent the runs which had the highest and lowest standard deviation.

*2) Test Case 2: Host Based Network Resource Management:* The second test case examines the requirement for host based network resource management. To achieve this management CONES utilizes and extends the network socket EDF scheduling described earlier. Like with CPU management, host network management becomes apparent in light of software bugs and unexpected confluences of events. With networking in particular, these could occur in the host with contention, or trigger contention in a separate host.

At the host level, contention is for the NIC itself. Such contention can be the result local processes attempting to send more packets possible over the interface. Causes of such contention can be a large number of polling requests (including such things as connection setup, and ping floods) from remote hosts, generating a large number local responses, or it can result from local processes trying to send a lot of information at one time. This case can be the result of an attack, or an event generating a lot of data, or some sort of engineering request or maintenance. Software bugs can also create such contention. Worn cables or faulty switches can result in lower speeds negotiated for line-rates.

Given modern networking speeds of 1000Mbps and faster, there was some concern of realistic network saturation at the host level. To determine if this was possible, we ran tests where multiple processes on a single host maximized network usage by sending packets as fast as possible. The packets were sent to multiple targets eliminating router and receiver contention issues. Aside from this contention creating traffic measurement traffic was being run between the host and an otherwise unused server. CPU usage for each core was consistently below 100%, eliminating CPU contention as a source of error.

As can be seen below in Figure 6 there is very little introduced latency, even with high network utilization on the host. When the network interface card has a reduced speed, specifically 100Mbps, the latencies introduced by contention become apparent. Further, when using a VM, as shown in Figure 7, the potential for contention is even higher.

To deal with these latencies, the iDSRT network scheduler was used. The test software *rtgen* was used to send packets in a real-time, scheduled manner, from a host under network contention. The results of this test are seen below in Figure 7, in histogram form. There were 6 runs of 20 minute duration for both CONES operating system, and vanilla Linux. Each graph shows the lowest and highest standard deviation run.

The results of this test, in the worst-case VM instance, show dramatic improvement of latency for network packets when the resource is scheduled.

*3) Test Case 3: Wide Area Network Resource Management:* A third test case was examined, that of Wide Area Network Resource Management. Due to space considerations this case cannot be fully presented. In summary however we confirmed widely accepted need for router based QoS, setting up channels for various traffic flows. These channels were set up with MPLS and DiffServ, and created classes for "Real-time" traffic and "Regular" traffic. Prior to management, incurred latencies of up to 25 ms, but averaging 15 ms were seen across a contended router. Post management, the latencies were reduced to 2 ms on average and 5 ms at maximum on the same contended router.

### *D. Analysis*

In this section we analyze the results of the three test cases against the requirements identified earlier in Section 2. Among all of the requirements mentioned in that section we discuss the two most stringent ones identified below. For each of these requirements we show that a CONES-based solution has the necessary capabilities for building a solution using commercial hardware/software systems.

(1) *Peer-to-peer substation protection*: this application requires occasional communication between IEDs in a substation with high priority and a data delivery duration of 4 ms.

For this requirement we look at host-based CPU and network management studied in Figures 5 and 7. Looking at how a host behaves in the presence of CPU contention from Figure 5 it is clear that a host machine requires real-time capabilities such as those provided by CONES to meet the 4ms requirement. The situation under network contention is worse for off-the-shelf systems as seen in Figure 7. Without real-time support delays of several seconds are possible. On the other hand with CONES even under extreme CPU or networking contention delays are limited to 1ms allowing sufficient time to meet the 4ms deadline.

(2) *System protection with synchrophasor data*: this application requires high rate communication (30 to 120 samples per second) to control centers from substations with medium priority and data delivery duration of 20 to 30 ms.

For this requirement we look at all three above also hold in this case. In addition, we consider that the latencies were significantly lowered in test case 3, regarding network resource management. We can see that without network resource management delays of 15 to 20 ms can be observed one-way in a WAN environment. However, with appropriate network management these delays come down to 2 ms making it easy to achieve the targeted deadlines for this requirement. Collectively, we can argue that a CONES-based solution (or a similar) one is necessary to satisfy these requirements.

Taking a wider view, the results show that under lab conditions, perhaps the strong resource management introduced by CONES is unnecessary. The lab equipment however is pristine and very powerful. Once adversity is introduced via various forms of contention and a lower network speed, it becomes clear that the protection offered by the resource management is needed for greater reliability. We show how contention quickly becomes an issue for the various resources, in the test

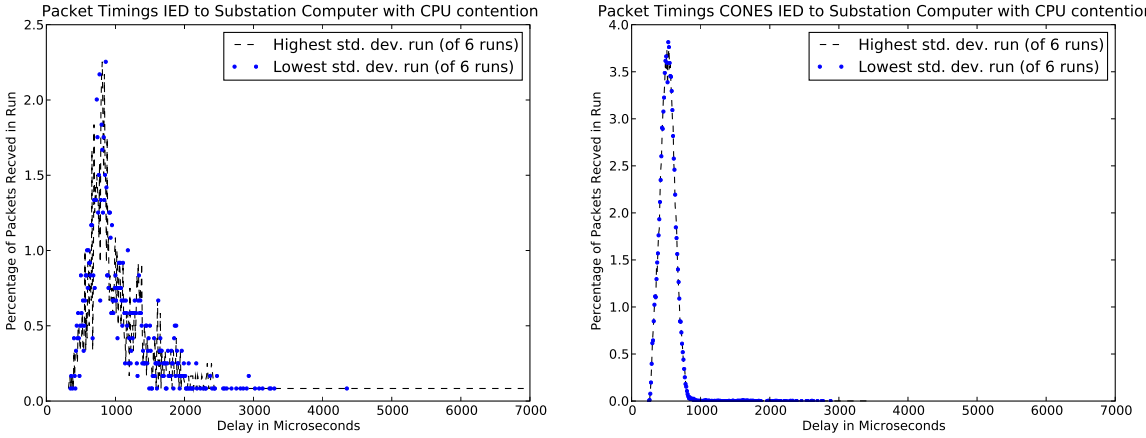


Fig. 5. Packet timings under CPU contention, with and without CONES enhancements

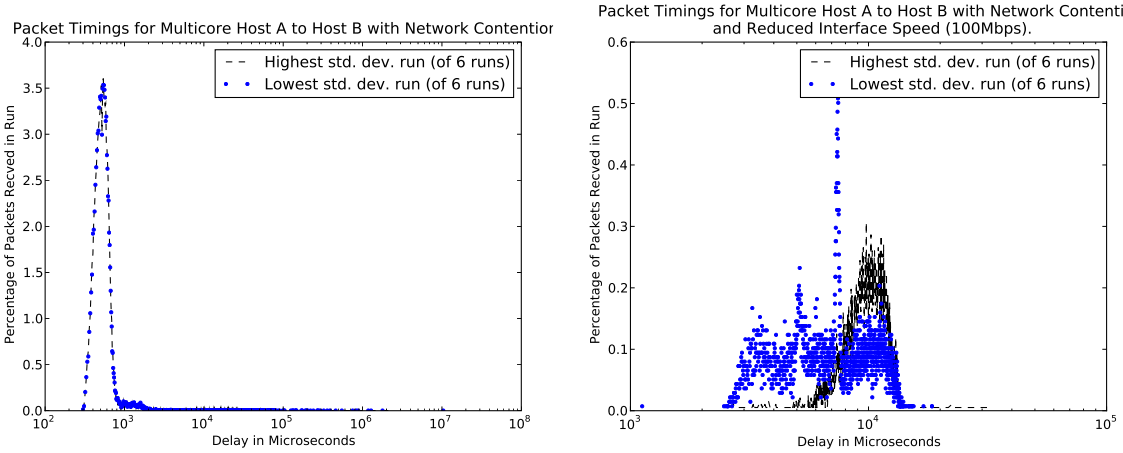


Fig. 6. Packet timings under Network contention for various speeds of Network Interface Card.

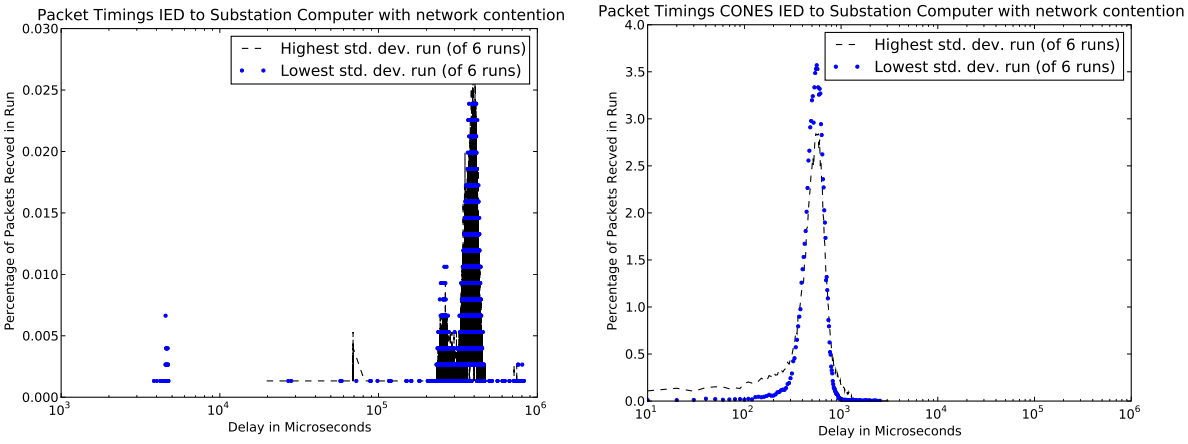


Fig. 7. Packet timings under network contention, with and without CONES enhancements

cases, and show the improvement that CONES brings when the potentially bad cases occur.

## VI. Related Work

This work falls under the broader category of Quality of Service (QoS) solutions for networked, distributed systems [1]. From a cyber infrastructure point of view, SCADA systems can be viewed as a form of embedded network systems for which QoS solutions with tight resource management are needed; e.g., as discussed in [2]. In this work we focus on an in-depth study of CPU and host network resource management requirements for SCADA applications. We then design, implement and experiment with a Linux-based solution for such resource management. In this area there are several

solutions that have been developed in the past. These solutions are based on a large body of work in the area of QoS, real-time systems and resource management.

The widely used set of patches to Linux known as PREEMPT_RT convert Linux into a fully preemptable kernel capable of supporting hard real-time. In recent versions, the trend has been to modularize other system resource management, such as the IO subsystem, as seen with the various IO scheduling options. These various components are leveraged in many commercial distributions of Linux, such as Timesys and Monte Vista Linuxes. Of particular note to this work, during the time our research was being conducted, the European software company Evidence released an EDF scheduling module called SCHED_DEADLINE, unfortunately it was too late to incorporate it this work. Also within Linux's networking stack is a complete complement of queuing management algorithms for packet networks (known collectively as $tc$, after the command used to configure them). To the authors' knowledge however, none of the above-mentioned systems include a deadline-based network scheduler. More recently, there is work from Illinois called iDSRT [3], which does provide network scheduling. It is built on the Linux components mentioned above, adding an EDF schedulers for both CPU and network resources. iDSRT was chosen for this reason as the basis of our work, see Section 5 for a full description of the enhancements and integration. The novelty of this work is using CONES for extensive experimentation of host CPU and network resource management in the context of representative SCADA applications with an eye towards network convergence. We demonstrate both the need for soft guarantees with such resource management and the ability of CONES to provide the needed guarantees.

There are several *middleware* systems that can provide messaging over wide area communication networks based on similar resource management capabilities addressed in this work. Examples of such middleware are, GridStat,Data Distribution Service for Real Time Systems (DDS), System of Systems Common Operating Environment (SOSCOE), and work by Schantz *et al.* [2]. Among these, Gridstat has been developed in the context of the power grid [5]. Research involving integration of CONES with such middleware solutions to achieve more fine grained end-to-end properties may help address power grid requirements beyond SCADA systems.

## VII. Conclusion

This paper describes the CONES (COnverged NEtworks for SCADA) real-time middleware toolkit aimed at supporting converged SCADA networks. CONES provides 1) host based process management, 2) network resource management, and 3) host based network resource management. These capabilities collective allow convergence of SCADA networks while ensuring that isolation and timeliness properties of all SCADA applications (e.g., protection, monitoring and maintenance) are supported as per the requirements of those applications. We provide an architecture, an advanced prototype implementation and extensive experimentation results to demonstrate the feasibility of CONES.

Future work in this area includes 1) the integration of security solutions that provide fine-grained access control and integrity and 2) further enhancements in specification, enforcement and dynamic adaptation for system-wide quality of service resources.

## References


[1] S. Chen and K. Nahrstedt, "An overview of quality of service routing for next-generation high-speed networks: problems and solutions," *Network, IEEE*, vol. 12, no. 6, pp. 64 –79, nov. 1998.

[2] R. E. Schantz, J. P. Loyall, C. Rodrigues, D. C. Schmidt, Y. Krishnamurthy, and I. Pyarali, "Flexible and adaptive qos control for distributed real-time and embedded middleware," in *Middleware '03: Proceedings of the ACM/IFIP/USENIX 2003 International Conference on Middleware*. New York, NY, USA: Springer-Verlag New York, Inc., 2003, pp. 374–393.

[3] H. Nguyen, R. Rivas, and K. Nahrstedt, ""idsrt: Integrated dynamic soft real-time architecture for critical infrastructure data delivery over wlan."," in *"Quality of Service in Heterogeneous Networks (QSHINE'09)"*, 2009, pp. 185–202.

[4] R. A. Johnston, C. H. Hauser, K. H. Gjermundrod, and D. E. Bakken, "Distributing time-synchronous phasor measurement data using the gridstat communication infrastructure," in *HICSS '06: Proceedings of the 39th Annual Hawaii International Conference on System Sciences*. Washington, DC, USA: IEEE Computer Society, 2006, p. 245.2.

[5] D. E. Bakken, C. H. Hauser, H. Gjermundrod, and A. Bose, "Towards More Flexible and Robust Data Delivery for Monitoring and Control of the Electric Power Grid," Technical Report EECS-GS-009, School of Electrical Engineering and Computer Science, Washington State University, May 2007.

[6] D. Bakken, D. Whitehead, and G. Zweigle, "Smart Generation and Transmission with Coherent, Real-Time Data," *Technical Report*, 2010. [Online]. Available: http://gridstat.net/publications/TR-GS-015.pdf

[7] IEEE, "Ieee standard communication delivery time performance requirements for electric power substation automation," *IEEE Std 1646-2004*, pp. 1–24, 2005.

[8] D. C. Schmidt, M. Deshpande, and C. O'Ryan, "Operating system performance in support of real-time middleware," in *Proceedings of the The Seventh IEEE International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS 2002)*, ser. WORDS '02. Washington, DC, USA: IEEE Computer Society, 2002, pp. 199–.

[9] S. Wang, S. Kodase, K. Shin, and D. Kiskis, "Measurement of os services and its application to performance modeling and analysis of integrated embedded software," in *Proceedings of the 8th IEEE Real-Time and Embedded Technology and Applications Symposium*, 2002, pp. 113 – 122.

[10] R. Braden, D. Clark, and S. Shenker, "Integrated services in the internet architecture: an overview (RFC 1633)," *Internet Request for Comments*, vol. 1633.

[11] K. Nichols, S. Blake, F. Baker, and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers: RFC 2474," *Internet Request for Comments*, 1998.

[12] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services RFC 2475," *Internet Request for Comments*, vol. 2475.

[13] E. Rosen, A. Viswanathan, and R. Callon, "RFC3031: Multiprotocol Label Switching Architecture," *Internet Request for Comments*, 2001.

[14] M. Caccamo, L. Y. Zhang, L. Sha, and G. Buttazzo, "An implicit prioritized access protocol for wireless sensor networks," in *Proceedings of the 23rd IEEE Real-Time Systems Symposium*, ser. RTSS '02. Washington, DC, USA: IEEE Computer Society, 2002, pp. 39–.

[15] T. Crenshaw, S. Hoke, A. Tirumala, and M. Caccamo, "Robust implicit edf: A wireless mac protocol for collaborative real-time systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 6, no. 4, p. 28, 2007.

[16] L.-C. Canon, O. Dubuisson, J. Gustedt, and E. Jeannot, "Defining and Controlling the Heterogeneity of a Cluster: the Wrekavoc Tool," *Journal of Systems and Software*, vol. 83, pp. 786–802, 2010. [Online]. Available: http://hal.inria.fr/inria-00438616/en/