Application of Congestion Notifications in a Cyber-Physical System

Stephen Jackson and Dr. Bruce McMillin

Missouri University of Science & Technology, Rolla, MO 65409, USA,
{scj7t4,ff}@mst.edu

Abstract. This work presents a new technique for protecting a cyber-physical, real-time, distributed system during network congestion. New applications of existing congestion avoidance and detection methods are used to ensure that the system can still perform useful work, without causing physical instability, during network congestion. By notifying processes of congestion, a process adjusts its behavior to continue to effectively manage physical devices during network congestion. We demonstrate with these techniques that the availability of the distributed system is improved under various scenarios.

Keywords: smart-grid, cyber-physical systems, real-time systems, distributed systems, network congestion

1 Introduction

The Future Renewable Electric Energy Delivery and Management (FREEDM)[12] smart-grid is a project focused on the future of the electrical grid. This smart grid project is an advanced cyber-physical system; it couples distributed cyber controllers with controllable physical devices. Major proposed features of the FREEDM cyber-physical system (CPS) rely on distributed local energy storage, and distributed local energy generation [1]. This vein of research emphasizes decentralizing the power grid, making it more reliable by distributing energy production devices. The Distributed Grid Intelligence (DGI) is a critical component of this system. The DGI performs automatic distributed configuration and management of power devices to manage an electrical grid. Energy management algorithms, a core feature of the DGI[2], must have access to a set of available processes to work with. Automatic reconfiguration using a group management algorithm allows algorithms like [2][17][8] to autonomously control distributed power devices.

Because the FREEDM smart-grid manages critical infrastructure in a distributed manner, its behavior during cyber or network fault conditions is of particular interest. In this work, we consider the effects of network congestion on a cyber communication network used by the FREEDM smart-grid. We create a model version of a large number of DGI processes in a simple partitioned setup. In the FREEDM smart-grid, the consequences of this congestion could result in several problematic scenarios. First, if the congestion prevents the DGI from autonomously configuring using its group management system, processes cannot work together to manage power devices. Secondly, if

messages arrive too late, or are lost, the DGI could apply settings to the attached power devices that drive the physical network to instability. These unstable settings could lead to problems in the power-grid like frequency instability, blackouts, and voltage collapse.

Additionally, this work has applications for other distributed systems. For example, in Vehicular Ad Hoc Networks (VANET)s[11][14], Drone Control and Air Traffic Control[15][3], an attacker could overwhelm the communication network and prevent critical information about vehicle position and speed from being delivered in a timely manner. As a consequence, the affected vehicles could collide, endangering the passengers.

In this work, we propose a technique to inform distributed processes of congestion. These processes act on this information to change their behavior in anticipation of message delays or loss. This behavior allows them to harden themselves against the congestion, and allows them to continue operating as normally as possible during the congestion. This technique involves changing the behavior of both the leader election[7] and load balancing algorithm during congestion.

To accomplish this, we extend existing networking concepts of Random Early Detection (RED), Explicit Congestion Notification (ECN)[13], and ICMP source quench[4]. When a network device detects congestion, it notifies processes that the network is experiencing congestion and they should react appropriately. We propose a practical application for these techniques in scenarios where the message delay is a primary concern, not the rate at which data is sent. With this information, they can adjust their behavior to compensate for the detected congestion.

In this work we demonstrate an implementation of the FREEDM DGI in a Network Simulator 3 (NS-3) simulation environment[6] with our congestion detection feature. The DGI operates normally until the simulation introduces a traffic flow that congests the network devices in the simulation. After congestion has been identified by the RED queueing algorithm, the DGIs are informed. We show that when the congestion notifications are introduced, the DGI maintains configurations which they would normally be unable to maintain during congestion. Additionally, we show a greater amount of work can be done without the work causing unstable power settings to be applied.

2 Background

2.1 DGI

The DGI is composed of controllers each with a scheduler and a series of modules which implement various features. Features provided by the DGI modules include automatic configuration, power management, and state collection. The DGI executes these modules using a round-robin real-time schedule. Processes synchronize their clocks and execute modules semi-synchronously. The load balancing module shares a quantum of power between a supply process and a demand process by performing a “migration.” In a migration, power devices are manipulated to share power on a shared bus. Each time the load balancing module is scheduled to execute it performs a fixed number of migrations during its execution phase. The schedule for the DGI is decided before the process is started, is shared by all processes and does not change when the DGI is running.

Leader Election. The DGI uses a variation on the leader election algorithm, "Invitation Election Algorithm," written by Garcia-Molina[7]. This modified algorithm allows the interactions of the processes in our semi-synchronous system to be modeled with a Markov chain. A version of this algorithm with a restriction on what processes could become coordinator is presented in [10]. We elected to use an improved version of the algorithm in this work because of its easy-to-follow group state.

The elected leader is responsible for making work assignments, identifying and merging with other coordinators when they are found, and maintaining an up-to-date list of peers. Group members monitor the group leader by periodically checking if the group leader is still alive by sending a message. If the leader fails to respond, the querying peers will enter a recovery state and operate alone until they can identify another coordinator. Therefore, a leader and each of the members maintain a set of currently reachable processes, a subset of all known processes in the system.

Using a leader election algorithm allows the FREEDM system to autonomously reconfigure rapidly in the event of a failure. Cyber components are tightly coupled with the physical components, and reaction to faults is not limited to faults originating in the cyber domain. Processes automatically react to crash-stop failures, network issues, and power system faults. The automatic reconfiguration allows processes to react immediately to issues, faster than a human operator, without relying on a central configuration point. However, it is important the configuration a leader election supplies is one where the system can do viable work without causing physical faults like voltage collapse or blackouts[5].

Power Management. In this work we utilize the load balancing algorithm from [2]. The load balancing algorithm performs work by managing power devices with a sequence of migrations[16]. In each migration, a sequence of message exchanges identify processes whose power devices are not sufficient to meet their local demand and other processes supply them with power by utilizing a shared bus.

The DGI algorithms can tolerate packet loss and is implemented using UDP to pass messages between DGI processes. Effects of packet loss on the DGI's group management module have been explored in [9] and [10]. The load balancing algorithm can tolerate some message loss, but lost messages can cause migrations to only partially complete, which can cause instability in the physical network. These actions can eventually lead to power instability through issues such as voltage collapse. In this work, we consider effects due to delays by congestion.

2.2 Random Early Detection

The RED queueing algorithm is a popular queueing algorithm for switches and routers. It uses a probabilistic model and an Exponentially Weighted Moving Average (EWMA) to determine if the average queue size exceeds predefined values. These values are used to identify potential congestion and manage it. This is accomplished by determining the average size of the queue, and then probabilistically dropping packets to maintain the size of the queue. In RED, when the average queue size avg exceeds a minimum threshold (min_{th}), but is less than a maximum threshold (max_{th}), new packets arriving at the queue may be "marked".

With RED, the probability a packet is marked varies linearly with the average queue size, and as a function and the time since the last packet was marked. If avg is greater than max_{th}, the probability of marking trends toward 1 as the average queue size approaches $2 * max_{th}$ In the event the queue fills completely, the RED queue operates as a drop-tail queue.

In a simple implementation of the RED algorithm, marked packets are dropped. For a TCP application, the result of the dropped packets causes the slow-start congestion control strategy to reduce the rate packets are sent. A more advanced implementation, using ECN, sets specific bits in the TCP header to indicate congestion. By using ECN, TCP connections can reduce their transmission rate without re-transmitting packets.

UDP applications have not typically utilized ECN. Although the ECN standard has flags in the IPv4 header, access to the IPv4 header is not possible on most systems. Furthermore, there is not a "one size fits all" solution to congestion in UDP algorithms. However, for the DGI and a class of similar real-time processes, congestion notification has great potential. If processes can adjust the amount of traffic they send based on the anticipated congestion (by disabling features, for example), they can decrease the effects of congestion.

3 Application

3.1 Usage Theory

When the RED algorithm identifies congestion it must notify senders of congestion. Since this approach is non-standard and most UDP applications would not understand the notification, we have opted to create an application that runs on switches and routers. Congestion is detected, the application sends a multicast beacon to a group of interfaces informing the attached devices of the level of congestion. For similarity with the RED algorithm and the NS-3 implementation, this notification is classified as either "soft" or "hard." A soft notification is an indication the congestion in the network is approaching a level where real-time processes can expect message delays that may affect their normal operation. A hard notification indicates the congestion has reached a level where messages are subject to both delay and loss.

3.2 Group Management

The group management module's execution schedule is broken into several periods of message generation and response windows. Because the schedule of the DGI triggers the execution of group management modules approximately simultaneously, the traffic generated by modules is bursty. The number of messages sent is $O(n^2)$ (where n is the number of processes in the system), in a brief window, which is dependent on how well the clocks are synchronized in the system. The duration of the response window is dependent on the amount of time it takes for messages to propagate to the hardest-to-reach process the DGI hopes to group with. Additionally, to contend with congestion, an additional slack must be added to allow the RED algorithm to detect congestion before it reaches a critical level. Figure 1 depicts typical queueing behavior for a network device serving DGI processes under different circumstances.

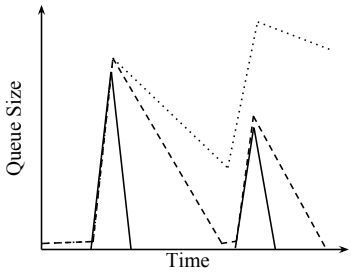

Fig. 1: Example of network queueing during DGI operation. DGI modules are semi-synchronous, and create bursty traffic on the network. When there is no other traffic on the network (solid line), the bursty traffic causes a large number of packets to queue quickly, but the queue empties at a similar rate. With background traffic (dashed line), the bursty traffic causes a large number of packets to be queued suddenly. More packets arrive continuously, causing the queue to drain off more slowly. When the background traffic reaches a certain threshold (dotted line), the queue does not empty before the next burst occurs. When this happens, messages will not be delivered in time, and the queue will completely fill.

For this work, the algorithm from [10] was used. This algorithm has a higher message complexity when in a group than the Garcia-Molina algorithm it is based on. However, it does possess a desirable memoryless property that makes it easy to analyze. This work uses an improved version of the algorithm which removes the restrictions in [10] where only one process could become the leader.

Soft ECN. A soft ECN message indicates the network has reached a level of congestion where the router suspects processes will not be able to meet their real time requirements. The soft ECN message encourages the DGI processes to reduce the number of messages they send to reduce the amount of congestion they contribute to the network, and to allow for reliable distribution techniques to have additional time to deliver messages (since fewer messages are being sent). In the case of potential congestion, the group management module can reduce its traffic bursts by disabling elections during the congestion. When the elections are disabled, messages for group management are only sent to members of the group. Processes do not seek out better or other leaders to merge with. As a consequence, the message complexity for processes responding to the congestion notification reduces from $O(n^2)$ to $O(n)$.

Hard ECN. In a hard ECN scenario, the router will have determined congestion has reached a threshold where the real-time processes will soon not be able to meet their deadlines. In this scenario, the real-time process will likely split its group. In an uncontrolled situation, the split will be random. It is therefore desirable when this level of traffic is reached to split the group. Splitting the group reduces the number of messages sent across the router for modules with $O(n^2)$ (where n is the number of processes in the original group) message complexity. For larger groups, splitting them provides a

significant savings in the number of messages that must be queued by the router, especially since the traffic is very bursty.

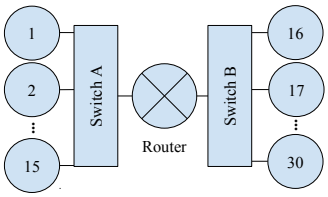

Fig. 2: Example of process organization used in this paper. Two groups of processes are connected by a router.

Suppose a network like one depicted in Figure 2, where processes are divided by a router. In Figure 2, there are n processes on one side of the network and m on the other. In normal operation the omission-modelable algorithm has an $O(n^2)$ message complexity. In Soft ECN maintenance mode, the reduced number of messages reduces the complexity to $O(n)$ by disabling elections.

During elections (and with each group update) the leader distributes a fallback configuration that will coordinate the division of the groups during intense congestion. When the ECN notification is received the processes will halt all current group management operations and enter a splitting mode where they switch to the fallback configuration. The leader of the group distributes a fallback notification to ensure all processes in the group apply their new configuration. The complexity of distributing the notification is linear $O(n)$ and processes that already received the notification will have halted their communication. This approach will ideally avoid the burst/drain phenomena from figure 1.

The design of the fallback configuration can be created to optimize various factors. These factors include cyber considerations, such as the likely network path the processes in the group will use to communicate. By selecting the group around the network resources, the group can be selected to minimize the amount of traffic that crosses the congested links in the future. Additionally, considerations from the physical network can be considered. Fallback groups can be created to ensure they can continue to facilitate the needs of the members. This can take into the consideration the distribution of supply and demand processes in the current group. By having a good mix of process types in the fallback group the potential for work can remain high.

3.3 Cyber-Physical System

For a real-time CPS, message delays could affect coordinated actions. As result, these actions may not happen at the correct moments or at all. Since the two-army problem prevents any process from being entirely certain a coordinated action will happen in concert, problems arising from delay or omission of messages is of particular interest. In particular, we are interested in the scenario from [5], where only half of a power

migration is performed. Other power management algorithms could have similar effects on the power system based on this idea of a process performing an action that is not compensated for by other processes.

Soft ECN. In a soft congestion notification mode, the process being informed of the congestion can reduce its affect on the congestion by changing how often it generates bursty traffic. Processes running the load balancing algorithm make several traffic bursts when they exchange state information and prepare migrations. As shown before, if the interval between these bursts is not sufficient for the queue to drain before the next burst occurs, then critical, overwhelming congestion occurs. Since the schedule of the DGI is fixed at run-time processes cannot simply extend the duration of the load balancing execution phase. However, on notification from the leader, the process can, instead, reduce the number of migrations to increase the message delivery interval. This notification to reduce the schedule originates from the coordinator as part of the message exchange necessary for the process to remain in the group. Every process in the group must receive this message to participate in load balancing, ensuring all processes remain on the same real-time schedule. Using this approach, the amount of traffic generated is unchanged but the time period a process waits for the messages to be distributed is increased.

Hard ECN. When the DGI process receives a hard congestion notification, the processes switch to a predetermined fallback configuration. This configuration creates a cyber partition. By partitioning the network, the number of messages sent by applications with $O(n^2)$ message complexity can be reduced significantly. Each migration of load balancing algorithm begins with an $O(n^2)$ message burst and so benefits from the reduced group size created by the partition.

Suppose there is a network like the in Figure 2 with n processes on one half and m on the other. The number of messages sent across the router for the undivided group is of the order $2mn$ as the n processes on side A send a message to the m on side B and vice-versa. Let i_1 and j_1 be the number of processes from side A and side B (respectively) in the first group created by the partition. Let i_2 and j_2 be the number of processes in the second group created by the partition under the same circumstances of i_1 and j_1. The number of messages sent that pass through the router, is then

$$2i_1j_1 + 2i_2j_2 \tag{1}$$

For an arbitrary group division, the following can be observed. Suppose i_1 and j_2 are the cardinality of two arbitrarily chosen sets of processes from side A and side B respectively. Following the same cut requirements as before:

$$i_2 = n - i_1 \text{ and } j_2 = m - j_1 \tag{2}$$

The the number of messages that must pass through the router for this cut is:

$$2i_1j_1 + 2(n - i_1)(m - j_1) \tag{3}$$

The benefits of the cut are minimized when i_1 and j_1 are $\frac{n}{2}$ and $\frac{m}{2}$:

$$2(2\frac{mn}{4} + mn - \frac{mn}{2} - \frac{mn}{2}) = mn \tag{4}$$

Which is a reduction of half as many messages. For systems with a large number of participating processes this represents a significant reduction in the number of messages sent across the router. As a consequence, this further extends the delivery window for processes sending messages.

4 Experimental Setup

Experiments were run in a Network Simulator 3.23[6] test environment. The simulation time replaced the wall clock time in the DGI for the purpose of triggering real-time events. As a result, the computation time on the DGIs for processing and preparing messages was neglected. However, to compensate for the lack of processing time, the synchronization of the DGIs was instead randomly distributed as a normal distribution. This was done to introduce realism to ensure events did not occur simultaneously. Additionally, the real-time schedules used by the DGI were adjusted to remove the processing time that was neglected in the simulation.

The DGIs were placed into a partitioned environment. The test included 30 nodes. Each of the nodes ran one DGI process. Two sets of 15 DGI were each connect to a switch and each switch was in turn connected to the router. This network is pictured in Figure 2. Node identifiers were randomly assigned to nodes in the simulation and used as the process identifier for the DGI.

The links between the router and the switches had a RED enabled queue placed on both network interfaces. The RED parameters for all queues were set identically. A summary of RED parameters are listed in Table 1. All links in the simulation were 100Mbps links with a 0.5ms delay. RED was used in packet count mode to determine congestion. ARP tables were populated before the simulation began. RED parameters were selected using results from [10].

Parameter	Value	Parameter	Value
RED Queueing Mode	Packet	RED Gentle Mode	True
RED Q_w	0.002	RED Wait Mode	True
RED Min Threshold	90	RED Max Threshold	130
RED Link Speed	100 Mbps	RED Link Delay	0.5 ms

Table 1: Summary of RED parameters. Unspecified values default to the NS-3 implementation default value

To introduce traffic, processes attached to each of the switches attempted to send a high volume of messages to each other across the router. The number of packets sent per second was a function of the data rate and the size of the packets sent. In

each simulation, half of the traffic originated from each switch. Due to the bottleneck due to the properties of the network links, the greatest queueing effect occurred at the switches.

5 Results

Figures 3 and 4 show the normal operation of the system. In this configuration, there is no congestion on the network. The DGIs start, group together and then begin migrating power between processes. Figure 3 plots the queue size over time for a queue used to send packets from a switch to the router. Figure 4 is a detailed view of a portion of Figure 3. Figure 4 shows the queue size during the normal operation of group management as well as the first migration of the load balancing module. The dotted line plots the EWMA of the size of the queue.

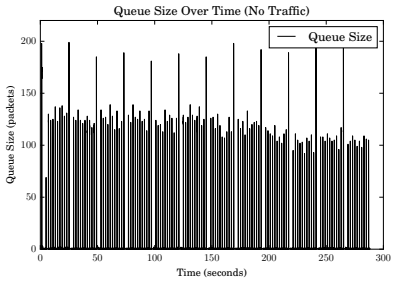

Fig. 3: Plot of the queue size for a queue from switch A to the router when only the DGI generates traffic.

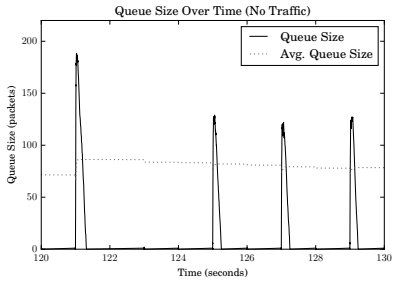

Fig. 4: Detailed view of Figure 3. The left most peak is from Group Management, and the 3 smaller peaks are from power migrations.

Figure 5 shows the queue size as the network traffic begins to increase. The DGIs in these experiments use a schedule that allows for some congestion to occur before processes are disrupted. This slack gives the network devices the opportunity to identify when the network congestion will go beyond the acceptable threshold.

Figure 6 shows an example of congestion affecting the physical network without ECN. As a result of the congestion in Figure 6, processes leave the main group. Additionally power migrations are affected: migrations are lost, or the supply process is left uncertain of migrations completions. Figure 10 plots the count of failed migrations over time.

Figure 7 shows an example of the ECN algorithm notifying processes of the congestion. Compared to the scenario in Figure 6, the ECN algorithm successfully prevents the group from dividing, and increases the number of migrations by reducing the number of attempted migrations each round.

Figures 9 and 8 show an example of a more extreme congestion scenario. In Figure 9, the RED algorithm shares a Hard ECN notification. This notification causes the DGI

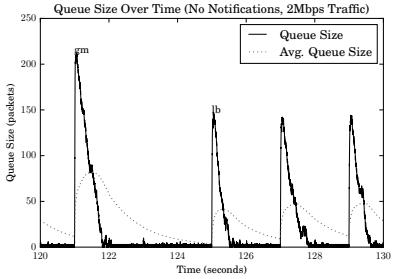

Fig. 5: Detailed view of the effect on queue size as other network traffic is introduced. Compared to Figure 4, the peaks are taller and wider. Background traffic causes the average queue size to be updated more frequently.

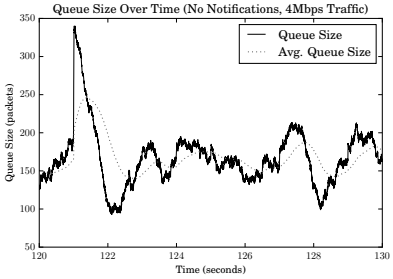

Fig. 6: Detailed view of the effect on queue size as other network traffic is introduced. With no ECN notifications, the peak from Group Management is much larger. The congestion is sufficient that the Group Management and Load Balancing modules are affected.

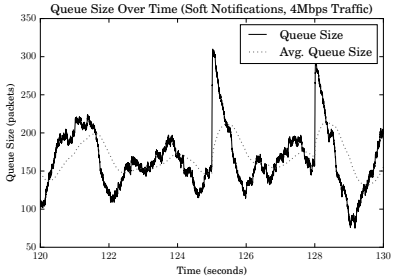

Fig. 7: Detailed view of the effect on queue size as other network traffic is introduced. In this scenario, the ECN notifications put Group Management into a maintenance mode that reduces its message complexity and switches Load Balancing to slower migration schedule, preventing undesirable behavior.

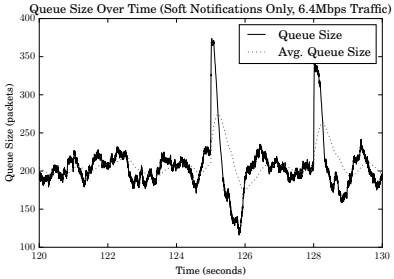

Fig. 8: Detailed view of the effect on queue size as a large amount network traffic is introduced. Groups are unstable and processes occasionally leave the main group. Some migrations are lost due to queueing delays.

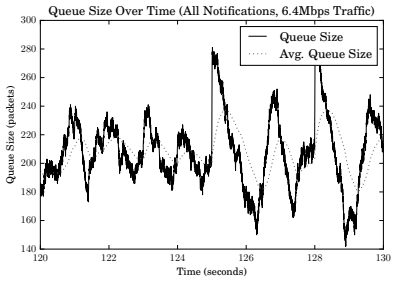

Fig. 9: Effect on queue size as a large amount of network traffic is introduced. Hard notifications cause the groups to divide. As a result of the smaller groups, the group management and load balancing peaks are smaller than those in 8. No migrations are lost.

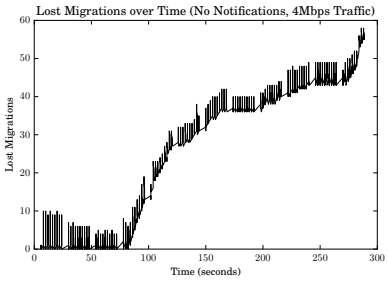

Fig. 10: Count of lost migrations from all processes over time. Migrations are counted as lost until the second process confirms it has been completed. Without congestion management, a large number of migrations are lost.

to switch to a smaller fallback configuration. This fallback configuration decreases the queue usage from Figure 8 to Figure 9. Without this fallback configuration behavior, the system is greatly affected by the traffic. However, with the fallback configuration the system remains stable and no migrations are lost.

6 Conclusion

In this work, we presented a technique for hardening a real-time distributed cyber-physical system against network congestion. The RED queueing algorithm and an out-of-band version of explicit congestion notification (ECN) were used to signal an application of congestion. Using this technique the application changed several of its characteristics to ready itself for the increased message delays caused by the congestion.

These techniques were demonstrated on the DGI, a distributed control system for the FREEDM smart-grid project. In particular, this paper demonstrated the hardening techniques were effective in keeping the DGI processes grouped together. Additionally, it helped ensure the changes applied to the DGI through cyber-coordinated actions did not destabilize the physical power network.

This technique will be important to create a robust, reliable CPS for managing future smart-grids. However, this technique could potentially be applied to any CPS that could experience congestion on its network, as long as it has the flexibility to change its operating mode. Potential applications can apply to both the cyber control network and the physically controlled process. For example, in a VANET system, the vehicles could react to congestion by increasing their following distance.

Acknowledgments. The authors acknowledge the support of the Future Renewable Electric Energy Delivery and Management Center, a National Science Foundation sup-

ported Engineering Research Center under grant NSF EEC-081212, and the United States Department of Education GAANN program.

References

1. Akella, R., Meng, F., Ditch, D., McMillin, B., Crow, M.: Distributed power balancing for the FREEDM system. In: Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on. pp. 7–12 (Oct 2010)
2. Akella, R., Meng, F., Ditch, D., McMillin, B., Crow, M.: Distributed power balancing for the FREEDM system. In: Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on. pp. 7 –12 (October 2010)
3. Bai, C., Zhang, X.: Aircraft landing scheduling in the small aircraft transportation system. In: Computational and Information Sciences (ICCIS), 2011 International Conference on. pp. 1019–1022 (Oct 2011)
4. Baker, F.: Requirements for IP version 4 routers (6 1995), rFC 1812
5. Choudhari, A., Ramaprasad, H., Paul, T., Kimball, J., Zawodniok, M., McMillin, B., Chellappan, S.: Stability of a cyber-physical smart grid system using cooperating invariants. In: Computer Software and Applications Conference (COMPSAC), 2013 IEEE 37th Annual. pp. 760–769 (July 2013)
6. Consortium, N..: Network simulator 3.23. `http://www.nsnam.org/`
7. Garcia-Molina, H.: Elections in a distributed computing system. Computers, IEEE Transactions on C-31(1), 48 –59 (January 1982)
8. Huq, K., Baran, M., Lukic, S., Nare, O.: An energy management system for a community energy storage system. In: Energy Conversion Congress and Exposition (ECCE), 2012 IEEE. pp. 2759–2763 (Sept 2012)
9. Jackson, S., McMillin, B.M.: The effects of network link unreliability for leader election algorithm in a smart grid system. In: Critical Information Infrastructures Security. pp. 59–70. Springer, Berlin, Heidelberg (2013)
10. Jackson, S., McMillin, B.: Markov models of leader elections in a smart grid system (under review). Journal of Parallel and Distributed Computing (2016)
11. Koscher, K., Czeskis, A., Roesner, F., Patel, S., Kohno, T., Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., Savage, S.: Experimental security analysis of a modern automobile. In: Security and Privacy (SP), 2010 IEEE Symposium on. pp. 447–462 (May 2010)
12. NSF FREEDM Systems Center: FREEDM, The Future Renewable Electric Energy Delivery and Management Systems Center, `http://www.freedm.ncsu.edu/`
13. Ramakrishnan, K., Floyd, S., Black, D.: The addition of explicit congestion notification (ECN) to IP (9 2001), rFC 3168
14. Rawat, D., Bajracharya, C., Yan, G.: Towards intelligent transportation cyber-physical systems: Real-time computing and communications perspectives. In: SoutheastCon 2015. pp. 1–6 (April 2015)
15. Sampigethaya, K., Poovendran, R.: Cyber-physical system framework for future aircraft and air traffic control. In: Aerospace Conference, 2012 IEEE. pp. 1–9 (March 2012)
16. Stanovich, M., Leonard, I., Sanjeev, K., Steurer, M., Roth, T., Jackson, S., Bruce, M.: Development of a smart-grid cyber-physical systems testbed. In: Innovative Smart Grid Technologies (ISGT), 2013 IEEE PES. pp. 1–6 (Feb 2013)
17. Zhang, Z., Chow, M.Y.: The leader election criterion for decentralized economic dispatch using incremental cost consensus algorithm. In: IECON 2011 - 37th Annual Conference on IEEE Industrial Electronics Society. pp. 2730–2735 (Nov 2011)