CapNet: Exploiting Wireless Sensor Networks for Data Center Power Capping

ABUSAYEED SAIFULLAH, Wayne State University, USA
SRIRAM SANKAR, Microsoft Corporation, USA
JIE LIU, Microsoft Research, USA
CHENYANG LU, Washington University in St. Louis, USA
RANVEER CHANDRA and BODHI PRIYANTHA, Microsoft Research, USA

As the scale and density of data centers continue to grow, cost-effective data center management (DCM) is becoming a significant challenge for enterprises hosting large-scale online and cloud services. Machines need to be monitored, and the scale of operations mandates an automated management with high reliability and real-time performance. The limitations of today's typical DCM network are many-fold. Primarily, it is a fixed wired network, and hence scaling it for a large number of servers increases its cost. In addition, with server densities increasing over recent years, this network also has to be cabled correctly and the management of this network parallels the complexity of managing a data network, since it needs to be networked with multiple switches and routers. In this article, we propose a wireless sensor network as a cost-effective networking solution for DCM while satisfying the reliability and latency performance requirements of DCM. We have developed CapNet, a real-time wireless sensor network for power capping, a time-critical DCM function for power management in a cluster of servers. CapNet employs an efficient event-driven protocol that triggers data collection only on the detection of a potential power capping event. We deploy and evaluate CapNet in a data center. Using server power traces, our experimental results on a cluster of 480 servers inside the data center show that CapNet can meet the real-time requirements of power capping. CapNet demonstrates the feasibility and efficacy of wireless sensor networks for time-critical DCM applications.

CCS Concepts: • **Networks** → **Network protocol design**; *Data center networks*; • **Computer systems organization** → *Real-time system architecture*;

Additional Key Words and Phrases: Wireless sensor network, real-time system, data center management, power capping, MAC protocol

ACM Reference format:
Abusayeed Saifullah, Sriram Sankar, Jie Liu, Chenyang Lu, Ranveer Chandra, and Bodhi Priyantha. 2018. CapNet: Exploiting Wireless Sensor Networks for Data Center Power Capping. *ACM Trans. Sen. Netw.* 15, 1, Article 6 (December 2018), 31 pages.
https://doi.org/10.1145/3278624

Authors' addresses: A. Saifullah, Wayne State University, 5057 Woodward Ave, Detroit, MI, 48202; email: saifullah@wayne.edu; S. Sankar, Microsoft Corporation, One Microsoft Way, Redmond, WA, 98052; email: sriram.sankar@microsoft.com; J. Liu, R. Chandra, and B. Priyantha, Microsoft Research, One Microsoft Way, Redmond, WA, 98052; emails: {jie.liu, ranveer, bodhip}@microsoft.com; C. Lu, Washington University in St. Louis, One Brookings Dr, St. Louis, MO, 63130; email: lu@cse.wustl.edu.

1550-4859/2018/12-ART6 $15.00
https://doi.org/10.1145/3278624

1 INTRODUCTION

Modern enterprise data centers continue to scale to meet growing cloud computing and storage demands by increasing the number of servers. The continuous, low-cost, and efficient operation of these large-scale data centers heavily depends on its management network and system. A typical data center management (DCM) system handles physical layer functionalities such as powering on/off a server, motherboard sensor telemetry, cooling management, and power management. Higher-level management capabilities such as system re-imaging, network configuration, (virtual) machine assignments, and server health monitoring [17, 34] depend on DCM to work correctly. DCM is expected to function even when the servers do not have a working OS or the data network is not configured correctly [1].

In today's data centers, DCM is typically designed in parallel to the production data network [18] (in other words, *out of band*), with a combination of Ethernet and *serial connections* for increased redundancy. There is a cluster controller for a rack or a group of racks, which are connected through Ethernet to a central management server. Within the clusters, each server has a motherboard microcontroller (Baseboard Management Controller, BMC) that is connected to the cluster controller via point-to-point serial connections. For redundancy reasons, every server is typically connected to two independent controllers on two different fault domains, so there is at least one way to reach the server under any single point of failure. Unfortunately, this architecture does not scale. The overall cost of management network increases super-linearly with the number of servers in a data center. At the same time, massive cabling across racks increases the chance for human errors and prolongs the server deployment latency.

This article presents a different approach to data center management network at the rack granularity by replacing serial cable connections with low cost wireless links. Low-power wireless sensor network technology such as IEEE 802.15.4 has intrinsic advantages in this application. Recently, the US Federal Energy Management Program has also recommended the adoption of wireless sensor technology as a cost-effective approach for data center management, real-time monitoring, and to optimize energy use [16].

- *Cost:* Low-power radios (i.e., IEEE 802.15.4) are cheaper individually than wired alternatives, and the cost scales linearly with the number of servers.
- *Embedded*: These radios are physically small and hence can be integrated onto server motherboard to save precious rack space.
- *Reconfigurability:* Wireless sensor networks can be self-configuring and self-repairing with the broadcast media to prevent human cabling error.
- *Low power:* With a small on-board battery, the DCM based on wireless can continue to function on batteries providing monitoring capabilities even when the rack experiences a power supply failure.

However, whether a wireless DCM can meet the high reliability requirement for data center operation is not obvious for several reasons. The amount of sheet metals, electronics, and cables may completely shield RF signal propagation within racks. Furthermore, although typical traffic on a DCM is low, emergency situations might need to be handled in real time, which could require the design of new protocols.

Power capping is an example of emergency event that imposes real-time requirements. Today, data center operators commonly oversubscribe the power infrastructure by installing more servers to an electric circuit than it is rated. The rationale is that servers seldom reach their peak at the same time. By over-subscription, the same data center infrastructure can host more servers than otherwise. In the rare event when the aggregate power consumption of all servers exceeds the

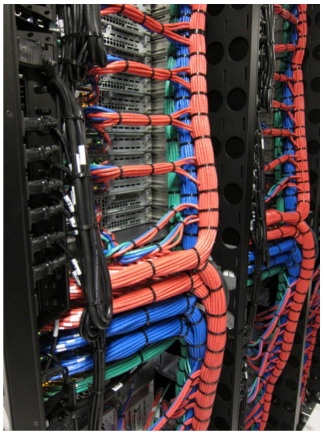


Fig. 1. Present cable (green) wiring for rack management [2].

circuit's power capacity, some servers must be slowed down (i.e., power capped), through dynamic frequency and voltage scaling (DVFS) or CPU throttling, to prevent the circuit breaker from tripping. Every magnitude of oversubscription is associated with a trip time that is a *deadline* by which power capping must be performed to avoid circuit breaker tripping.

This article makes the following key contributions.

- We study the feasibility and advantages of using low-power wireless for DCM. In two data centers, we empirically evaluate IEEE 802.15.4 link qualities in server racks to show that the overall packet reception rate is high.
- We further dive into the power capping scenario and design **CapNet**, a wireless **Net**work for power **Cap**ping, that employs an event-driven real-time control protocol for power capping over wireless DCM (that was initially published as a conference article [57]). The protocol uses distributed event detection to reduce the overhead of regularly polling all nodes in the network. Hence, the network throughput can be used by other management tasks when there is no emergency. When a potential power surge is detected, the controller uses a sliding window and collision avoidance approach to gather power measurements from all servers and then issues power capping commands to a subset of them.
- We deployed and evaluated CapNet in a data center. Using server power traces, our experimental results on a cluster of 480 servers in the data center show that CapNet can meet the real-time requirements of power capping. It demonstrates the feasibility and efficacy in power capping like wired DCM with a fraction of the cost.

In the rest of the article, Section 2 describes the motivation for wireless DCM. Section 3 gives an overview of the CapNet design. Section 4 presents the CapNet protocol. Section 5 describes fault tolerance of CapNet. Section 6 presents the experimental results. Section 7 describes future work. Section 8 reviews related work. Section 9 is the conclusion.

2 THE CASE FOR WIRELESS DCM (CAPNET)

The limitations of wired DCM solution (Figure 1) in data centers are many-fold. It is a fixed wired network and hence scales poorly with increase in number of servers. The serial-line based point-to-point topology incurs additional costs as we connect more of them together. In addition, with server densities increasing over recent years, this network also has to be cabled correctly, and the management of this network parallels the complexity of managing a data network, since it

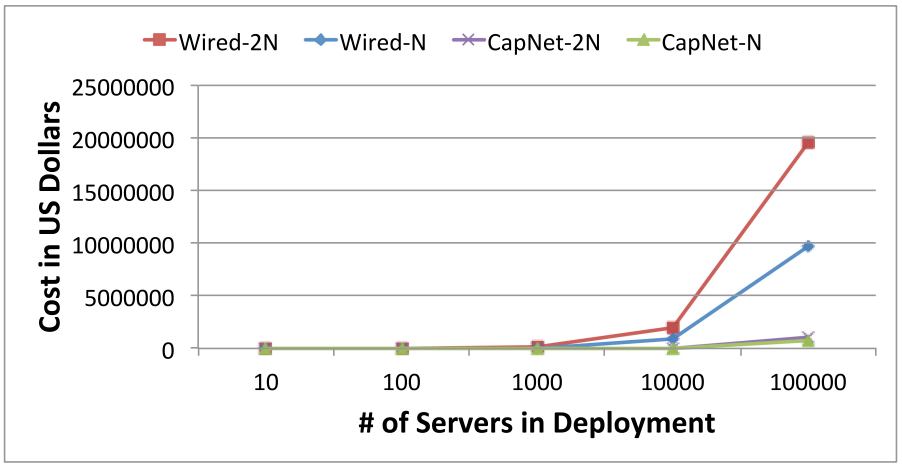


Fig. 2. System cost comparison and scalability.

needs to be networked with multiple switches and routers. Also, an important point to note here is that the management network is the last frontier; if this fails and the primary network is down, then we will not be able to manage the servers. While we can increase redundancy in a wired solution by constructing multiple paths and using redundant switches, the complexity and cost incurred increases as well. Here, we compare the costs of the wired DCM to our proposed wireless based solution (CapNet) by considering the cost of the management network and by measuring the quality of in-rack wireless links.

2.1 Cost Comparison with Wired DCM

To compare the hardware cost, we consider the cost of the DiGi switches ($3917/48port [3]), controller cost (approx. $500/rack [4]), cable cost ($2/cable [5]), and additional management network switches ($3000/48port on average [6]). We do not include the labor or management costs for cabling for simplicity of costing model, but note that these costs are also significant with wired DCMs. We assume that there are 48 servers per rack, and there can be up to 100,000 servers that need to be managed, which are typical for large data centers. For the wireless DCM-based CapNet solution, we assume IEEE 802.15.4 (ZigBee) technologies for its low-cost benefits. The cost of network switches at the top level layer stays, but the cost of DiGi can be significantly reduced. We assume $10 per wireless controller, which is essentially an Ethernet to ZigBee relay. For wireless receivers on motherboard, we assume $5 per server for RF chip and antenna as the motherboard controller is already in place [7].

We develop a simple cost model based on these individual costs and compute the total devices needed for implementing management over number of servers ranging from 10 to 100,000 (to capture how cost scales with the number of servers). We consider solutions across two dimensions: (1) Wired vs Wireless and (2) N-redundant vs. 2N-redundant (A 2N redundant system consists of two independent switches, DiGis, and paths through the management system). Figure 2 shows the cost comparison across these solutions. We see that a wired N-redundant DCM solution (Wired-N) for 100,000 servers is 12.5× the cost of a wireless N-redundant DCM solution (CapNet-N). If we increase the redundancy of the management network to 2N, then the cost of a wired solution (between Wired-2N and Wired-N) doubles. In contrast, the cost of a wireless solution increases only by 36% (due to 2N controllers and 2N switches at the top level). The resulting cost of

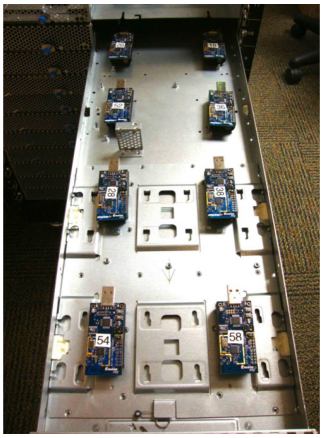


Fig. 3. Mote placed in bottom sled.

Wired-2N is 18.4× that of CapNet-2N. Given the significant cost difference between wired DCM and CapNet, we next explore whether wireless is feasible for communication within racks.

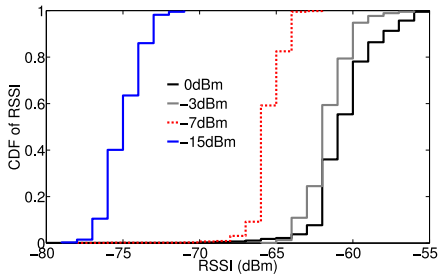
2.2 Choice of Wireless—IEEE 802.15.4

We are particularly interested in low-bandwidth wireless like IEEE 802.15.4 instead of IEEE 802.11 for a number of reasons. First, the payload size for data center management is small, and hence a ZigBee (IEEE 802.15.4) network bandwidth is sufficient for control plane traffic. Second, in WiFi (IEEE 802.11) there is a limit on how many nodes an access point can support in the infrastructure mode, since it has to maintain an IP stack for every connection, and this impacts scalability in a dense deployment. Third, to support management features, the data center management system should still work when the rack is unpowered. A small backup battery can power ZigBee longer at much higher energy efficiency. Finally, ZigBee communication stack is simpler than WiFi so the motherboard (BMC controller) microcontroller can remain simple. Although we do not rule out other wireless technologies, we chose to prototype with ZigBee in this article.

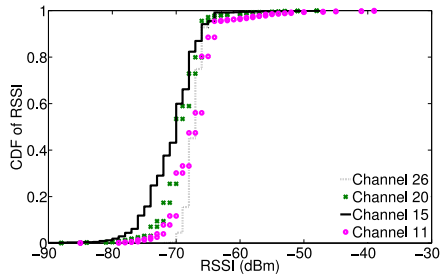
2.3 Radio Environment Inside Racks

We did not find any previous study that evaluated the signal strength within the racks through servers and sheet metal. The sheet metals inside the enclosure are known to weaken radio signal, giving a harsh environment for radio propagation inside racks. RACNet [43] studied wireless characteristics in data centers but only across racks when all radios are mounted at the top of the rack. Therefore, we first perform an in-depth 802.15.4 link layer measurement study based on in-rack radio propagation inside a data center of Microsoft Corporation.

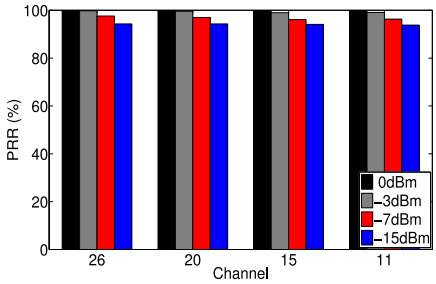
Setup: The data center used for measurement study has racks that consist of multiple chassis in which servers are housed. A chassis is organized into two columns of sleds. In all experiments, one TelosB mote is placed on top of the rack (ToR), inside the rack enclosure. The other motes are placed in different places in a chassis in different experiments. Figure 3 shows the placement of eight motes inside a bottom sled (shown as open in the figure but was closed in the experiment). While measuring downward link quality, the node on ToR is the sender, and the nodes in the chassis receive. Then we reverse the sender and the receiver to measure the upward link quality. In each setup, the sender transmits packets at 4Hz. The payload size of each packet is 29 bytes. Through a week-long test capturing the long-term variability of links, we collected signal strengths and



(a) **RSSI when Tx power varies (channel 26)**



(b) **On various channels (Tx power -3dBm)**



(c) **PRR at a receiver**

Fig. 4. Downward signal strength and PRR in bottom sled in data center (Redmond, WA).

packet reception rate (PRR). We perform the experiments in two different clusters in two different places of the data center.

Results: First we experiment the feasibility of low-power wireless communication in one cluster of the data center. This cluster is powered but the servers were not operating. Our next measurement as well as all subsequent experiments will be done in operating cluster. Figure 4(a) shows the cumulative distribution function (CDF) of Received Signal Strength Indicator (RSSI) values at a receiver inside the bottom sled for 1,000 transmissions from the node on ToR for different transmission (Tx) power using IEEE 802.15.4 channel 26. For −7dBm or higher Tx power, RSSI is greater than −70dBm in 100% cases. RSSI values in ZigBee receivers are in the range $[-100, 0]$. Previous study [61] on ZigBee shows that when the RSSI is above −87dBm (approx.), PRR is at least 85%. As a result, we see that signal strength at the receiver in bottom sled is quite strong. Figure 4(b) shows the CDF of RSSI values at the same receiver for 1,000 transmissions from the node on ToR on different channels at Tx power of −3dBm. Both figures indicate a strong signal strength, and in each experiment the PRR was at least 94% (Figure 4(c)).

For the same cluster, we now perform the experiments for upward communication. Specifically, we place the motes in different positions inside a rack that will transmit to the node on ToR. Placing the sender nodes in different places of a rack, we show the CDF of the RSSI values at the node on ToR (receiver) for 1,000 transmissions at Tx power of −3dBm (channel 26) for each sender in Figure 5. In the figure, "Position 6" indicates the position when the sender node is placed at the extreme end of the bottom sled. Other positions indicate the sender positions at different upper sleds. As the figure shows, RSSI is greater than −70dBm in 100% cases except for Position 6. For Position 6, RSSI is greater than −80dBm in more than 90% cases. Thus, the RSSI values indicate that signals are strong enough in upward communication also.

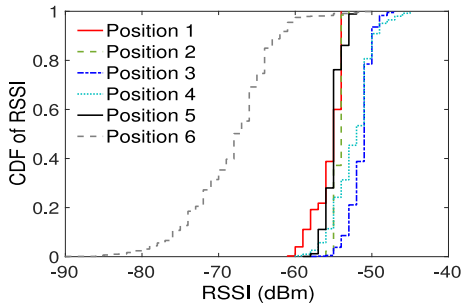
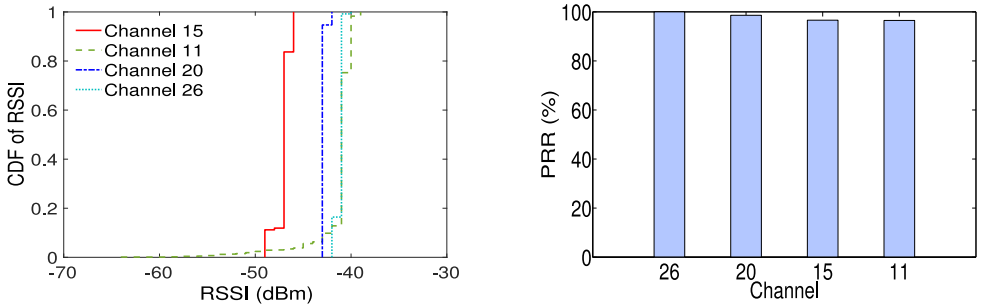


Fig. 5. Upward signal strength and PRR from senders positioned at different places in a rack in data center (Redmond, WA).



(a) **On various channels (Tx power -3dBm)** (b) **PRR at a receiver**

Fig. 6. Downward signal strength and PRR in bottom sled in an operating cluster in data center (Redmond, WA).

We next perform the similar experiments in another cluster of the data center. This is an operating cluster and is in a different location (on the same floor of the same data center) from the previous cluster. In this location, outside WiFi signal was quite low. Hence, there was no external interference from coexisting networks. Figure 6(a) shows the CDF of RSSI values at a receiver inside the bottom sled for 1,000 transmissions from the node on ToR on different channels (Channels 26, 20, 15, 11) at −3dBm Tx power. On every channel, RSSI is greater than −65dBm in 100% cases. Thus, here also the signal strength at the receiver in bottom sled is quite strong. Figure 6(b) shows that PRR on each channel was at least 95%. We observed similar results in all other setups of the measurement study and omit those results.

The measurement study reveals that low-power wireless, such as IEEE 802.15.4, is viable for communication within data center racks and can be reliable for telemetry purpose. We now focus on the power capping scenario and CapNet design for real-time power capping over wireless DCM.

3 CAPNET DESIGN OVERVIEW

Power infrastructure bears huge capital investment for a data center, up to ≈42% of the total cost (Figure 7) of a large data center that can cost hundreds of millions of U.S. dollars [31]. Hence, it is desirable to use the provisioned infrastructure to its maximum rated capacity. The capacity of a branch circuit is provisioned during design time, based on upstream transformer capacity during normal operation or UPS/Generator capacity when running on backup power.

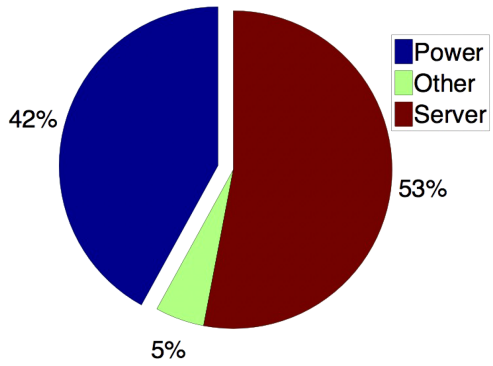


Fig. 7. Data center cost distribution.

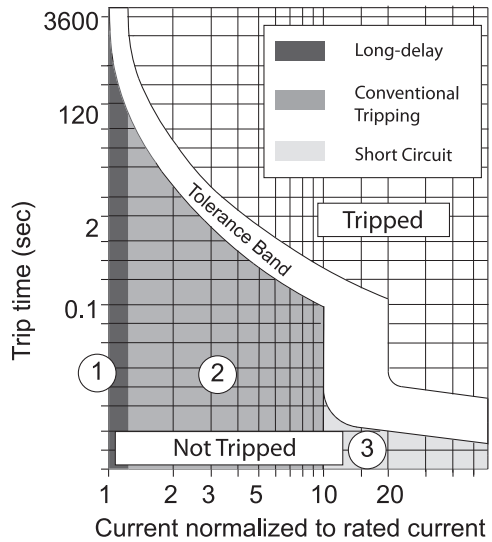


Fig. 8. The trip curve of Rockwell Allen-Bradley 1489-A circuit breaker at 40°C [8]. X-axis is oversubscription magnitude. Y-axis is trip time.

To improve data center utilization, a common practice in enterprise data centers is to do *oversubscription* [27, 30, 44, 50]. This method allocates servers in a circuit exceeding the rated capacity (i.e., *cap*), since not all servers reach their maximum power consumption at the same time. Hence, there is a circuit breaker (CB) that trips to protect expensive equipment. The peak power consumption above the cap has a specified time limit, called a *trip time*, depending on the magnitude of over-subscription (as shown in Figure 8 for Rockwell Allen-Bradley 1489-A circuit breaker). If the over-subscription continues for longer than the trip time, then the CB will trip and cause undesired server shutdowns and power outages disrupting data center operation. *Power capping* is the mechanism to bring the aggregate power consumption back to the cap. An overload condition under practical current draw trips the CB on a time scale from several hundred milliseconds to hours, depending on the magnitude of the overload [8]. These trip times are the *deadlines* for the corresponding oversubscription magnitudes within which power capping must be done to prevent CB tripping to avoid power loss or damage to expensive equipment.

3.1 The Power Capping Problem

Note that the need to manage peak power of an individual server is well understood and today most servers ship with mechanisms for power capping that allow limiting their individual peak power consumptions to a set threshold. However, such individual and static capping does not allow a server to take the advantage of another server's unused power. Such capacity waste can be avoided by coordinating the caps across multiple servers through centralized or global capping of these servers. In so-called coordinated capping, which is more desired in data centers, when some servers in a cluster (of multiple servers) are running at lower load, the power left unused could be used by other servers to operate at higher power levels than would be allowed by their individual static cap. In this article, we focus on coordinated power capping.

To enable power capping for a rack or cluster, a *power capping manager* (also called *controller*) collects all servers' power consumption and determines the cluster-level aggregate power consumption. If the aggregate consumption is over the cap, then the manager generates control messages asking a subset of the servers to reduce their power consumptions through CPU frequency modulation (and voltage if using DVFS) or utilization throttling. The application level quality of service may require different servers to be capped at different levels. So the central controller needs all individual server readings. In some graceful throttling policies, the control messages are delivered by the BMC Controller to the host OS or VMs, which introduce additional latency due to OS stack [22, 44]. To avoid abrupt changes to application performance, the controller may change the power consumption incrementally and require multiple iterations of the feedback control loop before the cluster settles down to below the power cap [44, 65]. These control policies have been studied extensively by previous work and are out of the scope of this article.

3.2 Power Capping over Wireless DCM

Servers in a data center are stacked and organized into racks. Figure 9 shows the wireless DCM architecture inside a data center. A *cluster* is a logical notion indicating a power management unit that consists of one or more racks. While it is possible that power management units are also hierarchical, we consider power management of a single cluster of n servers for protocol design. All servers in a cluster incorporate a wireless transceiver that connects to the BMC microcontroller. Each server is capable of measuring its own power consumption. Modern server hardware has built-in power metering capabilities (e.g., using motherboard or power supply–based power sensors) and several solutions exist to monitor power for older servers [36, 37]. At a server, power reading is taken using the corresponding API provided by the OS of the servers.

A cluster power capping manager can either directly measure the total power consumption using a power meter or, to achieve fine-grained power control, aggregates the power consumption from individual servers. We focus on the second case due to its flexibility. In this approach, power capping can be enabled for any subset of the servers. Besides, for capping purposes, we would need to know all individual server readings for power capping control algorithm that will determine a subset of the servers to be capped based on their individual power consumption (for example, by prioritizing the servers based on their current power consumption).

Power oversubscription and the corresponding trip times of the CB are mapped to power oversubscription and corresponding deadline at the power capping manager. Based on the aggregate power consumption and the magnitude of oversubscription, the power capping manager issues capping commands over wireless links to individual servers. The main difference compared to a wired DCM is the broadcast wireless media and challenge of scheduling communication to meet the real-time demands.

To reduce extra coordination and to enable spatial spectrum reuse, we assume a single IEEE 802.15.4 channel for communication inside a cluster. Using multiple channels, multiple clusters

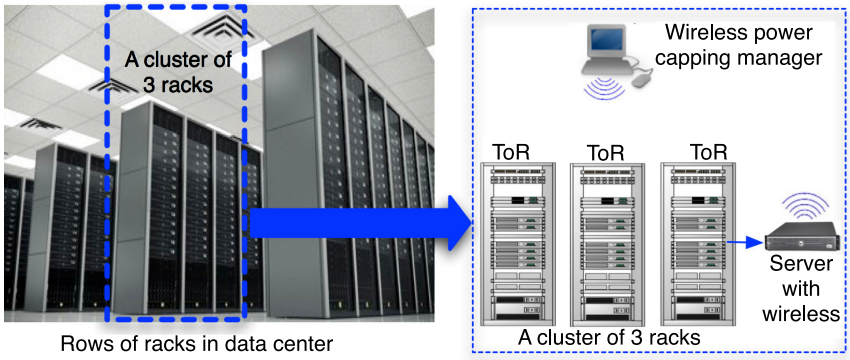


Fig. 9. Wireless DCM architecture.

can run in parallel. Channel allocation can be done using existing protocols that minimize inter-cluster interference (e.g., Reference [59]) and is not the focus of our article. Later, in Section 5, we shall consider dealing with failures in power capping managers through inter-cluster fail over.

CapNet forms a wireless network of n servers in a cluster and a power capping manager at ToR. It forms a star topology where the manager directly communicates with the wireless sensors on the servers. In our prototype of CapNet, wireless devices are plugged into the servers through their serial interface, which monitors server power consumption and wirelessly reports to the manager and receives control commands in response. Note that this is the first work to motivate low-power wireless for critical DCM operations such as power capping. Hence, in production servers, it is expected that wireless interface will be integrated into the motherboard in the future. Even if this is not done in the future, CapNet is still adoptable in data centers by using low-power wireless chips through serial interfaces of the servers. Besides, for smart racks [14] that can monitor individual server's power consumption, CapNet can be implemented by using ToR wireless devices.

Note that our experiments on wireless feasibility shown in Section 2 did not include the scenario when wireless interface is integrated into the motherboard. However, we performed experiments by placing a node in the extreme end of the bottom sled. That setup can be considered as an extreme scenario for wireless communication between a server and a manager. Our experiments observed reliable communication in that setup also as RSSI was greater than −80dBm in more than 90% cases. Therefore, we can expect reliable wireless communication even when wireless interface is integrated into the motherboard inside a server.

3.3 A Naive Periodic Protocol

A naive approach for a fine-grained power capping policy is to always monitor the servers by periodically collecting the power consumption readings from individual servers. The manager periodically computes the aggregate power. Whenever the aggregate power exceeds the cap, it generates a control message. On finishing aggregation and control in η iterations, it resumes the periodic aggregation again.

3.4 Event-Driven CapNet

Oversubscribing data centers may provision for the 95th (or more) percentile of the peak power and require capping for 5% (or less) of the time, which may be an acceptable hit on performance in relation to cost savings [22]. Thus power capping is a rare event, and the naive periodic protocol is an overkill as it saturates the wireless media by always preparing for the worst case. Other delay-tolerant telemetry messages cannot get enough network resources. An ideal wireless protocol

should generate significant traffic only when a significant power surge occurs. Therefore, CapNet employs an event-driven policy that is designed to trigger power capping control operation only when a potential power capping event is predicted. Due to the rareness and emergency nature of power surge, the network can suspend other activities to handle power capping. It provides real-time performance and a sustainable degree of reliability without consuming much network resource. The details of the protocol is explained in the next section.

4 POWER CAPPING PROTOCOL

We design a distributed event detection policy, where we assign local caps to each individual server from their global (cluster-level) cap. When a server observes a local power surge based on its own power reading, it can trigger the collection of the power consumption of all the servers to detect a potential surge in the aggregate power consumption of cluster. If a cluster-level power surge is detected, then the system initiates a power capping action. As many servers can simultaneously exceed their local caps; if we adopt a standard CSMA/CA protocol, then all of those servers will attempt to transmit at the same time. Such an approach will hence suffer from significant packet loss due to excessive contention and collisions (as we will also experimentally verify in Section 6), affecting the delay sensitivity of power capping. In fact, the CSMA/CA-based protocols may not provide predictable latency and hence are not preferred for real-time communication [58]. Hence, we also do not adopt the CSMA/CA approach. Finally, as power aggregate consumption can be quite dynamic, it may be infeasible to predict an upcoming power peak based on historical readings. Therefore, we also cannot adopt a predictive protocol that proactively schedule data collection based on historical power readings.

While a global detection is possible by just monitoring at the branch circuit level, say using a power meter, it cannot support fine-grained and flexible power capping policies such as those based on individual server-priority or reducing powers of individual servers based on their power consumptions. Also, a centralized measurement introduces a single point of failure. That is, if the power meter fails, power oversubscription will fail also. In contrast, our distributed approach is more resilient to failure. If individual measurement fails, then the system can always assume a maximum power consumption at that server and keep the whole cluster going.

The event-driven protocol runs in three phases as illustrated in Figure 10: **detection**, **aggregation**, and **control**. The event detection phase generates alarms based on local power surges. On detecting a potential event, CapNet runs the second phase that invokes a power aggregation protocol. False detection may happen when some servers generate alarms exceeding the local caps, but the aggregate value is still under the cap. This is corrected in the aggregation phase, where the controller determines the aggregate power consumption. The impact of a false positive case is that the system runs into the aggregation phase that incurs additional wireless traffic. The control phase is executed only if the alarms are true.

We normalize each server's power consumption value between 0 and 1 by dividing its instantaneous power consumption by the maximum power consumption of an individual server. This normalized power consumption value of server i is denoted by p_i, where $0 \leq p_i \leq 1$, and is used in this article as a server's power consumption. According to these normalized power consumption values, the cap of a cluster of n servers is denoted by c, and the total power consumption of n servers is considered as the aggregate power consumption and is denoted by p_{agg}. The important notations used in the protocol description are also summarized in Table 1.

Assigning local cap. If $p_{\text{agg}} > c$, then a necessary condition is that some servers' (at least one) individual power consumption values locally exceed the value $\frac{c}{n}$. Therefore, a possible way is to assign $\frac{c}{n}$ as each server's local cap. However, there can be situations where only one server exceeds

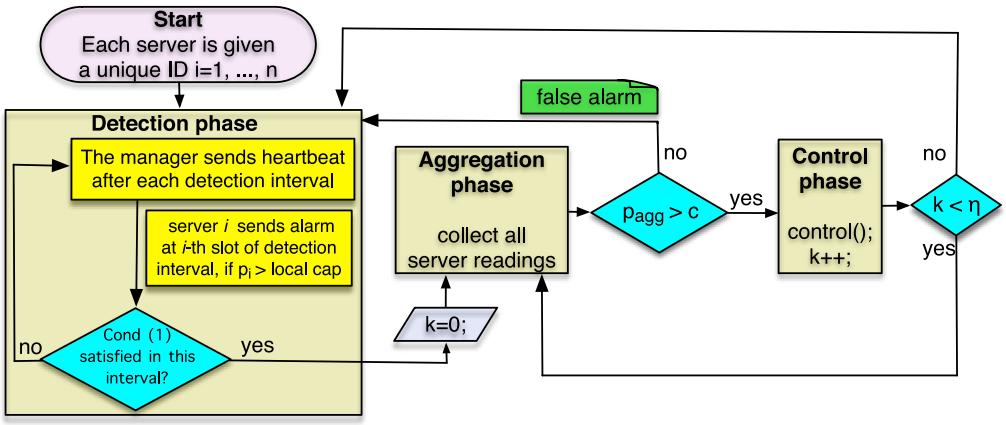


Fig. 10. CapNet's event-driven protocol flow diagram: Event detection phase generates alarms on detecting potential events (power surges) locally at server level. If an alarm is generated at the sth slot, then the manager checks Condition (1) (or Condition (2) if the network is considered unreliable). If this condition is satisfied, then it indicates that the estimated aggregate power may exceed the cap and the aggregation phase starts that determines actual aggregate power consumption. If the actual aggregate power consumption is indeed over the cap, then the control phase starts that selects several servers to reduce their power consumption through CPU throttling.

Table 1. Notations

Notation	Description
n	Total number of servers in the cluster
c	Cap of the cluster
p_i	Power consumption of server i
h	Detection interval
p_{agg}	Aggregate power consumption of the cluster
ω	Window size for sliding window protocol
τ_d	Maximum downward communication time
τ_u	Maximum upward communication time
L_{det}	Time spent in the detection phase
L_{agg}	Total aggregation latency
L_{os}	OS level latency
L_{hw}	Hardware level latency
L_{cap}	Total power capping latency in one iteration

$\frac{c}{n}$ while all other servers are under $\frac{c}{n}$, thereby triggering an aggregation phase on a single server's alarm. As a result, this policy will generate many false alarms that will trigger an aggregation phase causing unnecessary communication. Therefore, to reduce unnecessary aggregation phases, we assign a slightly smaller local cap and consider alarms from multiple servers before starting an aggregation phase. Thus, we use a value $0 < \alpha \leq 1$ close to 1 and assign $\frac{\alpha c}{n}$ as the local cap for each server. A server i reports alarm if $p_i > \frac{\alpha c}{n}$. In this strategy, the probability that multiple servers generate alarms can be less than the probability that one server generates alarm. Therefore, it reduces unnecessary aggregation phases.

Each server is assigned a unique ID i, where $i = 1, 2, \ldots, n$. The manager broadcasts a *heartbeat* packet at every h time units called *detection interval*. The detection interval of length h is slotted

into n slots, with each slot length being $\lfloor \frac{h}{n} \rfloor$. As the number of servers in a cluster changes, the slot length can be updated/adjusted. However, the value of h is selected in a way so that a slot is long enough to accommodate one transmission and its acknowledgement. After receiving the heartbeat message, the server clocks are synchronized.

4.1 Detection Phase

Each node i, $1 \leq i \leq n$, takes its sample (i.e., power consumption value p_i) at the ith slot in the detection phase. If its reading is over the cap, i.e., $p_i > \frac{\alpha c}{n}$, it generates an alarm and sends the reading (p_i) to the manager as an acknowledgement of the heartbeat message. Otherwise, it ignores the heartbeat message and does nothing. If an alarm is received at the sth slot, the manager determines, based on whether the network is reliable or not, whether an aggregation phase has to be started. Let the servers who have sent alarms in the current detection window so far be denoted by A.

Reliable Network. Let an alarm be generated in the sth slot of a detection interval. Considering a reliable network we can consider that no server message was lost. Therefore, each of the other $s - |A|$ servers among the first s servers has a power consumption reading of at most $\frac{\alpha c}{n}$ as it has not generated an alarm. Each of the remaining $n - s$ servers can have a power consumption value of at most 1. Thus based on the alarm at sth slot, the manager can estimate an aggregate power of $\sum_{j \in A} p_j + (s - |A|)\frac{\alpha c}{n} + (n - s)$. Hence, if an alarm is generated at the sth slot, then the manager will start aggregation phase if the following condition (Condition (1)) is true as shown in Figure 10,

$$\sum_{j \in A} p_j + (s - |A|)\frac{\alpha c}{n} + (n - s) > c. \tag{1}$$

Unreliable Network. Now we consider a scenario where some server alarms were lost. As a result, if an alarm is generated in the sth slot of a detection window, then each of the other $s - |A|$ servers among the first s servers may have a power consumption reading of at most 1 as its alarm is assumed to be lost. Therefore, each of the $n - |A|$ servers can have power consumption of at most 1, making an estimated aggregate power of $\sum_{j \in A} p_j + (n - |A|)$. Such a technique handles both link failure and node failure. Thus, if an alarm is generated in the sth slot, the manager will start aggregation phase if

$$\sum_{j \in A} p_j + (n - |A|) > c. \tag{2}$$

If there are no alarms in the detection phase or all alarm messages were lost due to transmission failure, the controller resumes the next detection phase (to detect the surges again using the same mechanism) when the current phase is over.

4.2 Aggregation Phase

To minimize aggregation latency, CapNet adopts a sliding window–based protocol to determine aggregate power consumption denoted by p_{agg}. The controller uses a window of size ω. At anytime, it selects ω servers (or, if there are fewer than ω servers whose readings are not yet collected, then selects all of them) in a round-robin fashion who will send their readings consecutively in the next window. These ω server IDs are ordered in a message. In the beginning of the window, the controller broadcasts this message and starts a timer of length $\tau_d + \omega\tau_u$ after the broadcast, where τ_d denotes the maximum downward communication time (i.e., the maximum time required for a controller's packet to be delivered to a server) and τ_u denotes the maximum upward communication time (server to controller). On receiving the broadcast message, any server whose ID is in order i, $1 \leq i \leq \omega$, in the message transmits its reading after $(i - 1)\tau_u$ time. Other servers ignore

the message. If the timer fires or packets from all ω nodes are received, then the controller creates the next window of ω servers that are yet to be scheduled or whose packets were missed (in the previous window). A server is scheduled in at most γ consecutive windows to handle transmission failures, where γ is the worst-case ETX (expected number of transmissions for a successful delivery) in the network. The procedure continues until all server readings are collected or there is no server that was retried γ times.

To reduce aggregation latency, we should set the window size ω large enough. However, it cannot be set arbitrarily large. A very large window size requires the payload size of the manager's message to be larger (since the packet contains ω node IDs indicating the schedule for next window). While it is possible to use bitmap technique to keep the manager's message size small, the window size is also restricted due to the clock drifts of the servers. Note that, on receiving the broadcast message of the manager, any server whose ID is in order i, $1 \leq i \leq \omega$, in the message transmits its reading after $(i-1)\tau_u$ time. Hence, a very large window size has the risk that two servers transmission times may overlap due to clock drift in the long time window. Hence, the window size should be assigned by considering both the message size of the manager and the server clock drifts.

4.3 Control Phase

On finishing the aggregation phase, if $p_{\text{agg}} > c$, where c is the cap, then it starts the control phase. The control phase generates a capping control command using a control algorithm, and then the controller broadcasts the message requesting a subset of the servers to be capped. To handle broadcast failures, it repeats the broadcast γ times (since the broadcast is not acknowledged). The servers react to the capping messages by DVFS or CPU throttling that incurs an operating system (OS) level latency as well as a hardware-induced delay [22]. If the control algorithm requires η-iteration, then after the capping control command is executed in the first round, the controller will again run the aggregation phase to reconfirm that capping was done correctly. The procedure iterates up to $(\eta - 1)$ more iterations. On finishing the control, or after the aggregation phase on a false alarm, it resumes the detection phase.

4.4 Latency Analysis

Given the time criticality for power capping, it is important for CapNet to achieve bounded latency. Here, we provide an analytical latency upper bound for CapNet's power capping latency that consists of detection phase latency, aggregation latency, OS level latency, and hardware latency. In practice, the actual latency is usually lower than the bound. The analysis can be used by system administrators to configure the cluster to ensure power capping meets the timing constraints.

Aggregation Latency: For n servers in the cluster, the total aggregation delay L_{agg} under no transmission failure can be upper bounded as follows. Note that each window of ω transmissions can take at most $\tau_u\omega + \tau_d$ time units. There can be at most $\lfloor \frac{n}{\omega} \rfloor$ windows where in each window ω servers transmit. Then, the last window will take only $\tau_u(n \bmod \omega) + \tau_d$ time to accommodate the remaining $(n \bmod \omega)$ servers. Hence,

$$L_{\text{agg}} \leq (\tau_u\omega + \tau_d)\left\lfloor \frac{n}{\omega} \right\rfloor + (\tau_u(n \bmod \omega) + \tau_d).$$

Considering γ as the worst-case ETX in the network,

$$L_{\text{agg}} \leq \left((\tau_u\omega + \tau_d)\left\lfloor \frac{n}{\omega} \right\rfloor + (\tau_u(n \bmod \omega) + \tau_d)\right)\gamma. \tag{3}$$

The above value is only an analytical upper bound, and in practice the latency can be a lot shorter.

Latency in Detection Phase: The time spent in the detection phase is denoted by L_{det}. In a detection window the protocol never will need the readings from the last $\lfloor c \rfloor - 1$ servers as an aggregation phase must start before this should a power capping needed (assuming that not all alarms were lost). Therefore, the alarms generated within the first $(n - \lfloor c \rfloor + 1)$ slots must trigger aggregation phase. Hence,

$$L_{\text{det}} \leq \left\lfloor \frac{h}{n} \right\rfloor (n - \lfloor c \rfloor + 1)\,. \tag{4}$$

Total Power Capping Latency: To handle a power capping event, a detection phase and an aggregation phase are followed by a control message that is broadcasted γ times and takes $\tau_d \gamma$ time. In addition, once the control message reaches a server, there is an operating system level latency, and after processor frequency changes, there is a hardware-induced delay. Let the OS level latency and the hardware level latency in the worst case be denoted by L_{os} and L_{hw}, respectively. Thus, the total power capping latency in one iteration, denoted by L_{cap}, is bounded as

$$L_{\text{cap}} \leq L_{\text{det}} + L_{\text{agg}} + \tau_d \gamma + L_{\text{os}} + L_{\text{hw}}.$$

A η-iteration control means that once power capping command is executed, the controller will again need to collect all readings from servers and reconfirm that capping was done correctly in $(\eta - 1)$ more iterations. Therefore, for η-iteration control, the above bound is given by

$$L_{\text{cap}} \leq L_{\text{det}} + (L_{\text{agg}} + \tau_c \gamma + L_{\text{os}} + L_{\text{hw}})\eta. \tag{5}$$

5 FAULT TOLERANCE

One important challenge is handling the failure of power capping manager in a cluster. In this article, we focus on the fail-stop model in power capping managers. We exploit a fail-over mechanism to allow the power capping manager in a nearby rack to take over the management functions of a rack whose power capping manager has failed. When some manager fails, a nearby one can take over its servers. We first show through cross-rack measurements that it is feasible for a nearby controller to communicate with nodes in the rack over a wireless sensor network.

5.1 Cross-Rack Measurement

A setup similar to that in Section 2.3 is used to evaluate radio propagation across racks. This is done in a data center at Washington University in St. Louis, Missouri. We place a transmitter node inside one rack and place a receiver node inside a different rack. We use channel 26 and a transmission power of 0dBm for transmitting, and record all signal strength values at the receiver. We repeat the same experiment placing the receiver node in different racks such that the number of rows between the two racks are different. Specifically, we put the sender node inside a rack of Row 1 and put one receiver in a rack in each of Rows 2, 3, 4, and 5. Figure 11 shows the CDF of RSSI values of 1,000 transmissions when the receiver node is placed at different racks (at varying distances from the sender node). The figure shows that radios can cross a few racks with receivable signal strength as transmissions from Row 1 can reach Row 4 with RSSI values above −70dBm. This result indicates that when a power manager fails, a nearby power manager will be able to communicate with its servers.

5.2 Failure Handling Mechanism

For each power capping manager (controller), we designate a backup power capping manager from a nearby rack that will take over the management of its servers when it fails. In our design, a power capping manager uses a separate 802.15.4 radio to communicate with its backup manager through a different channel that is not used for communication with servers in either cluster. The

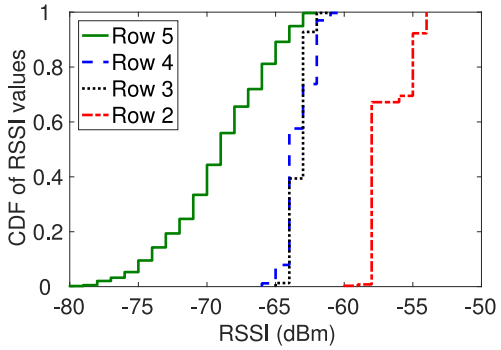


Fig. 11. Cross-rack signal strengths in data center (St Louis, MO).

backup manager can detect the failure of a manager based on heartbeat messages. Specifically, a manager B can periodically send its heartbeat to its backup manager A. If A does not receive a certain number of heartbeats (i.e., receives no heartbeat in a certain time window) from B, then it will decide that B has failed. For each power capping manager, a backup manager is assigned or set manually. Cascaded failures can be handled by assigning multiple backup managers.

We explain the failure handling mechanism assuming that when controller B fails, the backup controller A will take over B's servers. A changes its channel to B's channel and broadcasts a recovery message that is repeated γ times to ensure all nodes receive the broadcast, and after that all nodes of B join A. Let the servers of A be numbered as $1, 2, \ldots, n_a$ and those of B numbered as $1, 2, \ldots, n_b$. The detection interval of A is now slotted where each slot length is given by

$$\left\lfloor \frac{h}{n_a + n_b} \right\rfloor.$$

The detection interval h of A can be increased if a time slot is too short. Now each node with index i in B will be indexed as $(n_a + i)$ after joining A's cluster. The same event driven protocol then runs in A's cluster with $(n_a + n_b)$ servers. When B is back, A abandons B's servers and asks them to change back to the old channel.

6 EXPERIMENTS

In this section, we present the experimental results of CapNet. The objective is to evaluate the effectiveness and robustness of CapNet in meeting the real-time requirements of power capping under data center realistic settings.

Implementation. The wireless communication side of CapNet is implemented in NesC on TinyOS [15] platform. To comply with realistic data center practices, we have implemented the control management at the power capping manager side. In our implementation, wireless devices are plugged to the servers directly through their serial interface.

Workload Traces. We use workload demand traces from multiple geo-distributed data centers run by a global corporation over a period of 6 consecutive months. Each cluster consists of several hundreds of servers that span multiple chassis and racks. These clusters run a variety of workloads including Web-Search, Email, Map-Reduce jobs, and cloud applications, catering to millions of users around the world. Each cluster uses homogeneous hardware, though there could be differences across clusters. We use workload traces of two representative server clusters: C1 and C2. In both clusters each individual server has CPU utilization data of 6 consecutive months in every 2-minute interval. While we recognize that full system power is composed of storage, memory,

and other components, in addition to CPUs, several previous works show that a server's utilization is roughly linear to its power consumption [24, 25, 27, 53]. Hence, we use server's CPU utilization as a proxy for power consumption in all experiments.

6.1 Experimental Setup

6.1.1 Experimental Methodology. We experiment with CapNet using TelosB motes for wireless communication. First, we deployed 81 motes (1 for manager, 80 for servers) in Microsoft's data center in Redmond, WA. The servers in the cluster were powered and in operation. When we experiment with more than 80 servers to test scalability, one mote emulates multiple servers and communicates for them. For example, when we experiment for 480 servers, mote 1 works for first 6 servers, then mote 2 works for next 6 servers, and so on. This is feasible, since no two motes will transmit together in the detection phase or in aggregation phase. And the control phase sends a common broadcast message for all servers.

We place all 80 motes in racks. The manager node is placed on ToR and connected through its serial interface to a PC that works as the manager. No mote in the rack has direct line of sight with the manager. Using the workload demand traces, CapNet is run in a trace-driven fashion. For every server the reading at a time stamp sent from its corresponding wireless mote is taken from these traces at the same time stamp. Since we have the data of every 2-minute interval, for the sake of a trace-driven experiment, the power consumption value at a time stamp between two consecutive data points is taken through a linear interpolation between the two data points. While the data traces are 6 months long, our experiment does not actually run for 6 months. When we take a subset of those traces, say, for 4 weeks, the protocols skip the long-time intervals where there is no peak. For example, when we know (looking ahead into the traces) there is no peak between time t_1 and t_2, the protocols skip the times between t_1 and t_2. Thus our experiments finish in several days instead of 4 weeks.

6.1.2 Oversubscription and Trip Time. We use the trip times from Figure 8 as the basis to determine the different caps required in various experiments. in Figure 8, the X-axis shows the ratio of current draw to the rated current and is the *magnitude of oversubscription*. The Y-axis shows the corresponding trip time. In our experiment, for a particular magnitude of oversubscription, we consider the same value (magnitude of oversubscription) in the X-axis and take the corresponding trip time in the Y-axis as the deadline. The trip curve is shown as a tolerance band. The upper curve of the band indicates *upper bound (UB) trip times*, above which is the tripped area, meaning that the circuit breaker will trip if the duration of the current is longer than the UB trip time. The lower curve of the band indicates *lower bound (LB) trip times*, under which is the not-tripped area. This band between the two curves is the area where it is non-deterministic if the circuit breaker will trip. LB trip time is a very conservative bound. In our experiments, we use both LB and UB of conventional trip times to verify the robustness of CapNet.

6.1.3 CapNet Parameters. For all experiments, we use channel 26 and Tx power of −3dBm. The payload size of each packet sent from the server nodes is 8 bytes, which is enough for sending power consumption reading. The maximum payload size of each packet sent from the manager is 29 bytes, the maximum default size in IEEE 802.15.4 radio stack for TelosB motes. This payload size is set large to contain the schedules and control information. For aggregation protocol, window size ω is set to 8. A larger window size can reduce aggregation latency but requires the payload size of the manager's message to be larger (since the packet contains ω node IDs indicating the schedule for next window). In the aggregation protocol both τ_d and τ_u were set to 25ms. The manager sets its timeout using these values. These values are relatively larger compared to the maximum transmission time between two wireless devices. The time required for communication between

Latency component	Latency (ms)
OS	10 - 50
Wall power change	100 - 300

Fig. 12. OS and hardware level latencies in experimented in Intel Xeon L5520 (frequency 2.27GHz, 4 cores), Intel Xeon L5640 (frequency 2.27GHz, dual socket, 12 cores with hyper-threading), and an AMD Opteron 2373EE (2.10GHz, 8 cores with hyper-threading), each running Windows Server 2008 R2.

two wireless devices is in the range of several milliseconds. But in our design the manager node is connected through its serial interface to a PC. The TelosB's serial interface does not always incur a fixed latency for communication between PC and the mote. On experimenting and observing a wide variation of this time, we have set τ_d and τ_u to 25ms.

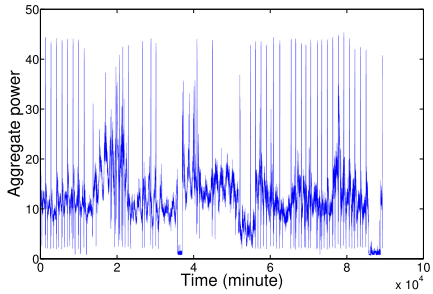
6.1.4 Control Emulation. In our experiments, we emulate the final control action, since we use workload traces. We assume that one packet is enough to contain the entire control message. To handle control broadcast failure, we repeat control broadcast $\gamma = 2$ times. Our extensive measurement study through data center racks indicated that this is also the maximum ETX for any link between two wireless motes. On receiving the control broadcast message, the nodes generate an OS level latency and hardware level latency. We use the maximum and minimum OS level and hardware level time required for power capping experimented on three servers with different processors: Intel Xeon L5520 (frequency 2.27GHz, 4 cores), Intel Xeon L5640 (frequency 2.27GHz, dual socket, 12 cores with hyper-threading), and an AMD Opteron 2373EE (2.10GHz, 8 cores with hyper-threading), each running Windows Server 2008 R2 [22]. The ranges of OS level and hardware level latencies found through experiments are shown in Figure 12. We generate OS and hardware level latencies using a uniform distribution in this range.

6.2 Power Peak Analysis of Data Centers

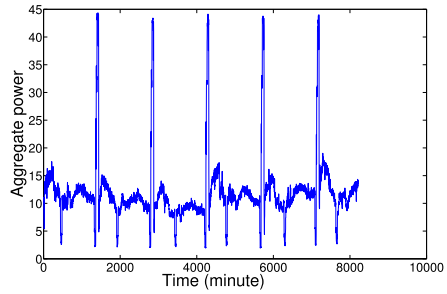
We first analyze whether CapNet protocol is consistent with the data center power behavior leveraging our data traces. For brevity, we present the trace analysis results of 3 racks: Racks R1 and R2 from Cluster C1, and Rack R3 from Cluster C2. To give an idea on how power consumption varies over time in a data center, Figure 13(a) shows the aggregate power of 60 servers on RACK R1 in cluster C1 for 2 consecutive months which is zoomed in for 6 consecutive days in Figure 13(b). For each rack, we use the 95th percentile of aggregate power over 2 consecutive months as the power cap (Figure 13(c)). Similar power traces are shown in Figure 14 for RACK R3 in cluster C2.

We first explore the power dynamics of the servers and the unpredictability of power capping events. Using 2-month data, Figure 15 shows that the time intervals between two consecutive peaks can range between few minutes to several hundred hours. We define *power jump* as the difference between the power that exceeds the cap and the preceding measurement that is below the cap. As Figure 15(b) shows that power jumps can vary between 0 and 51 for 60 servers in each rack (while their aggregate power is in the range $[0, 60]$). This result shows the motivation for an event-driven protocol.

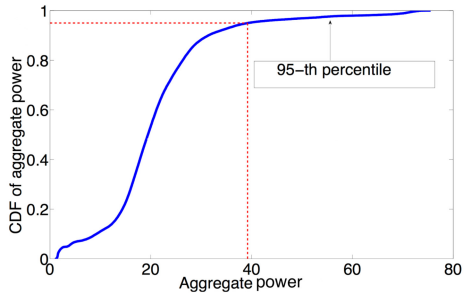
We first analyze the correlations between the power peaks of different servers within a cluster. In Figure 16(a), the scatter plot on 2-month raw data of three servers in Rack R3 indicates strong positive correlation between the power consumption of the servers, with their power peaks and valleys frequently coinciding with each other. Figure 16(b) shows the traces for cluster-level



(a) **Aggregate power (2 Months)**

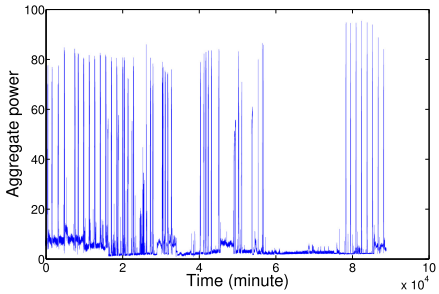


(b) **Aggregate power (1st 6 days zoomed-in)**

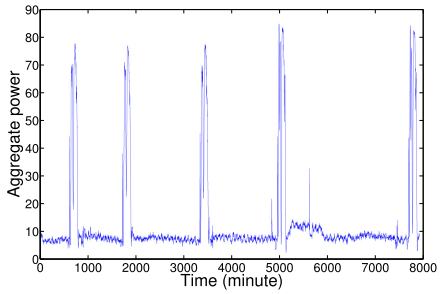


(c) **Cap: 95-th percentile of aggregate power**

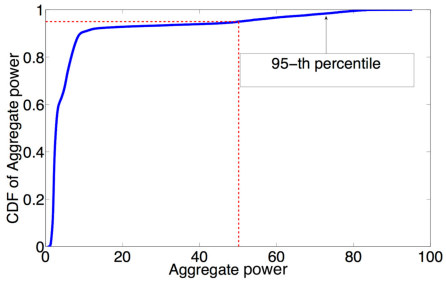
Fig. 13. 60 Servers on Rack R1 in Cluster C1.



(a) **Aggregate power (2 Months)**

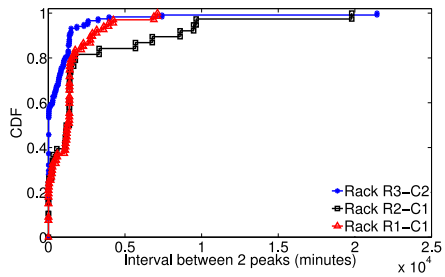


(b) **Aggregate power (6 days zoomed-in)**

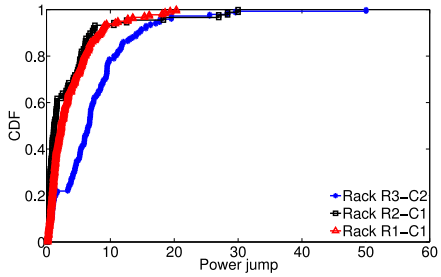


(c) **Cap: 95-th percentile of aggregate power**

Fig. 14. 60 Servers on Rack R3 in C2.

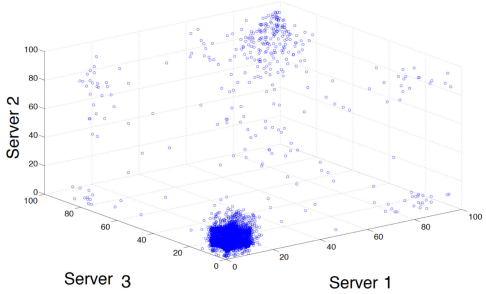


(a) **Time interval between 2 peaks**

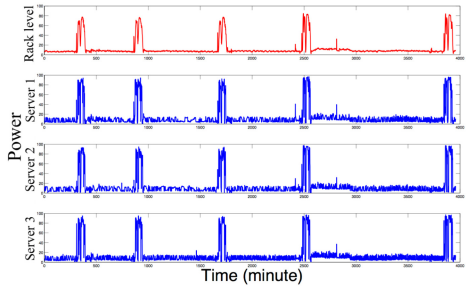


(b) **Power jump**

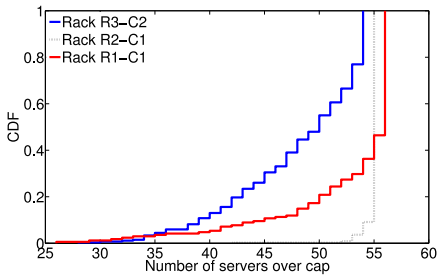
Fig. 15. Power characteristics (2-month data).



(a) **Power of 3 servers in Rack R3 C2 (2 month data)**



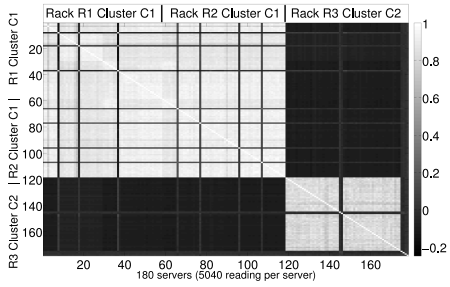
(b) **Similarity between Rack R3 aggregate power and its 3 servers (1 week)**



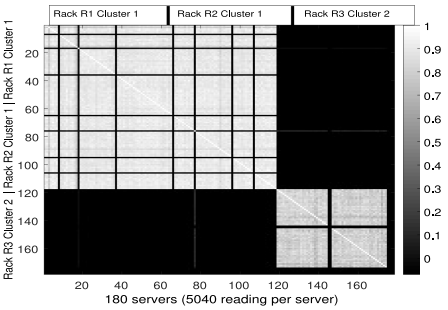
(c) **Total number (out of 60) of servers that exceed local cap $\frac{c}{60}$**

Fig. 16. Synchrony/asynchrony among servers and racks.

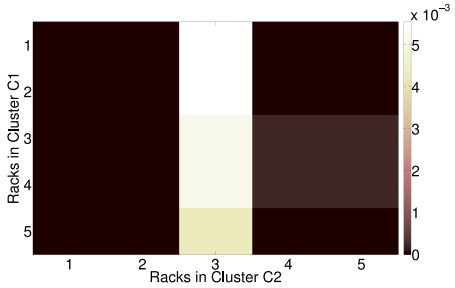
(or rack-level as the cluster has one rack) aggregate power consumption (top) and the individual consumption of three servers of that rack (for 1 week), which shows synchronous power peaks in the servers as well as the cluster in aggregation. This usually happens because the servers in the same cluster hosts similar workloads leading to synchronous power characteristics. We further assume a local cap of $\frac{c}{60}$ (considering $\alpha = 1$) for each individual server and show in Figure 16(c) the CDF of the number of servers that exceed local caps when the cluster-level aggregate power exceeds cap c. The figure shows that in 80% cases when the cluster-level (rack-level) aggregate power exceeds cap c, the numbers of servers (among 60 servers per rack) that are over the local cap are 43, 55, and 50 for Racks R3, R1, and R2, respectively. The strong *intra-cluster synchrony* in power



(a) **Correlations among 180*180 server pairs in 3 racks in 2 clusters**



(b) **Peak and non-peak correlation for Fig 17(a)**



(c) **Probability of simultaneous peak between two different clusters**

Fig. 17. Correlations among servers, racks, and clusters.

surge suggests the feasibility of detecting a cluster-level power surge based on local server-level measurements.

Figure 17(a) illustrates the correlations across 180 servers from different racks and clusters using their raw power consumption data over 1 week. The image is a visualization of a 180×180 matrix, indexed by the server number. That is, the entry indexed at $[i, j]$ in this matrix is the correlation coefficient of the values (5040 samples) between the ith and the jth servers. We can clearly see that the servers in the same rack are strongly positively correlated, and those in the same cluster are also positively correlated. But the servers between clusters are less or negatively correlated. Figure 17(b) shows similar results for correlations among the peaks and non-peaks (by considering a wave where peak is 1 and any non-peak is 0). This usually happens because the servers in the same cluster hosts similar workloads leading to synchronous power characteristics [24]. Figure 17(c) shows probabilities of different racks in two clusters to be at peak simultaneously. The entry indexed at $[i, j]$ in this 2D matrix is the probability that the ith rack in cluster 1 and the jth rack in cluster 2 are at peak simultaneously. The probabilities were found in the range $[0, 0.0056]$. This strong *inter-cluster asynchrony* implies that using an event-driven protocol (that performs wireless communications only on detecting an event) significantly minimizes inter cluster interference caused by transmissions generated by the event-driven CapNet in different clusters.

We observe strong synchrony in power behavior among the servers in the same cluster and strong asynchrony among between different clusters. The major implication of the trace analysis is that CapNet protocol is consistent with real data center power behavior. As the intra-cluster synchrony suggests the potential efficacy of a local event detection policy, our protocol is particularly effective in the presence of strong intra-cluster synchrony that exists in enterprise data

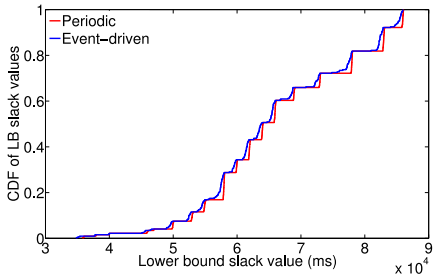
centers as observed in our trace analysis. However, in the absence of intra-cluster synchrony in power peaks, CapNet will not cause unnecessary power capping control or more wireless traffic than a periodic protocol. The synchrony only enhances CapNet's performance.

6.3 Power Capping Results

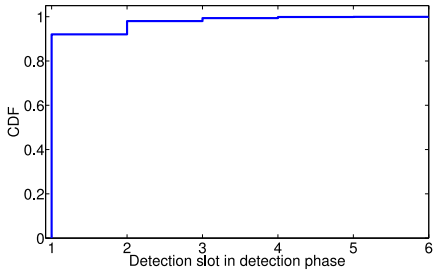
Now we present our experimental results with CapNet's event-driven protocol. First, we compare its performance with the periodic protocol and a representative CSMA/CA protocol. We then analyze its scalability in terms of number of servers. First, we experiment only for the simple case, where a single iteration of control loop can settle to a sustained power level, and then we also analyze scalability in terms of number of control iterations, where multiple iterations are needed to settle to a sustained power level. We have also experimented it under different caps and in presence of interfering clusters. Note that the caps across the clusters may be different as the cap of a cluster depends on the power consumption values of its servers as well as the number of servers. In all experiments, detection phase length, h, was set to $100 * n$ ms, where n is the number of servers. We set this value, because this makes each slot in the detection phase equal to 100ms, which is enough for receiving one alarm as well as for sending a message from the manager to the servers. Setting a larger value reduces the number of cycles of detection phase but reduces the granularity of monitoring. For assigning a local cap of $\frac{\alpha c}{n}$ to the servers, we first experiment with $\alpha = 1$. Later, we experiment under different values of α. Condition 1 is used for detection and starting an aggregation phase. In the results, ***slack*** is defined as the difference between the trip time (i.e., deadline) and the total latency required for power capping. That is, a negative value of slack implies a deadline miss. We use *LB slack* and *UB slack* to define the slack calculated considering LB trip time and UB trip time, respectively. In our results, in cases timing requirement can be loose, while there are cases where these are very tight, and the results are shown for all cases. We particularly care for tight deadlines and want to avoid any deadline misses.

6.3.1 Performance Comparison with Base Lines. Figure 18 presents the results using 60 servers on one rack for single-iteration control loop. We used 4-week-long data traces for this rack. We set the 95th percentile of all aggregate powers values of all data points in every 2-minute interval as its cap c. For assigning local cap we use $\alpha = 1$. In running the protocols using these traces, the protocols observe all peaks. The upper bound of aggregation latency (L_{agg}) given in Equation (3) was set as the period of the periodic protocol. Figure 18(a) shows the LB slacks for both the event-driven protocol and the periodic one. The figure only plots the CDF for the cases where the magnitude of oversubscription was above 1.5 for better resolution as the slack was too big for a smaller magnitudes (which are not of interest). Since UB trip times are easily met, we also omit those results. The non-negative LB slack values for each protocol indicate that it easily meets the trip times. Hence there is no benefit in using non-stop communications (i.e., the naive periodic protocol).

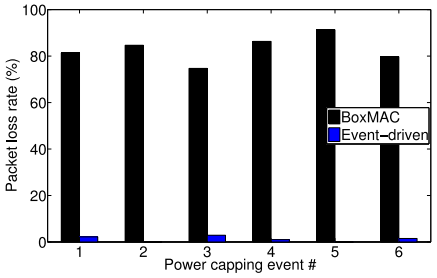
While the slacks in event-driven protocol are shorter than those in the periodic protocol because the former spends some time in the detection phase, in 80% cases event-driven protocol can provide a slack of more than 57.15s while the periodic protocol provides 57.88s. The difference is not significant, because as shown in Figure 18(b) in 90% cases among all power capping events the detection happened in the first slot of the detection cycle. Only in 10% cases, it was after the first slot of the detection phase, and all detection happened within the 6th slot, although the phase had a total of 60 slots (for 60 servers, one slot per server). These results indicate that CapNet's local detection policy can quickly determine the events. This is also an implication that experimental values of power capping latencies are quite different (or shorter) from the pessimistic analytical values derived in Equation (5). Also, in this experiment, 94.16% of the total detection phases did not have any transmission from the servers. Therefore, if we compare with the periodic protocol that



(a) **CDF of lower bound slack**



(b) **CDF of detection slots in detection phase**



(c) **Packet Loss Rate**

Fig. 18. Performance of Event-Driven protocol on 60 servers (4 weeks).

needs to continue communication always in the network, the event-driven protocol suppresses transmissions at least by 94.16% while the real-time performance of two protocols are similar.

We also evaluate the performance when BoxMAC (the default CSMA/CA-based protocol in TinyOS [15]) is used for power capping communication for up to first 6 capping events in the data traces. Figure 18(c) shows that it experiences packet loss rate over 74% while performing communication for a power capping event. This happens because all 60 nodes try to send at the same time, and the back-off period in 802.15.4 CSMA/CA under default setting is too short, which leads to frequent repeated collisions. Since we lose most of the packets, we do not consider latency under CSMA/CA. Increasing the back-off period reduces collisions but results in long communication delays. In subsequent experiments, we exclude CSMA/CA as it does not fit for power capping.

6.3.2 Scalability in Terms of Number of Servers. In our data traces each rack has at most 60 active servers. To test with more servers, we combine multiple racks in the same cluster, since they have similar pattern of power consumption (as we have already discussed in Subsection 6.2. For sake of experimentation time, in all subsequent experiments we set cap at 98th percentile (that would result in a smaller number of capping events). The lower bound slack distribution are shown in Figure 19 for 120, 240, and 480 servers by merging 2, 4, and 8 racks, respectively (for single iteration capping). Hence, for single iteration, the deadlines are easily met for even 480 servers (since in each setup, 100% of all slack values are positive).

6.3.3 Experiments under Varying α. Now we experiment with different values of α for assigning a local cap of $\frac{\alpha c}{n}$ to the servers using 480 servers. The results in Figure 20 show the tradeoff between false alarm rate and power capping latency under varying α. As we decrease the value of α from 1 to 0.80, the false alarm rate decreases from 45% to 2%. This happens because with decreased value of α, CapNet considers multiple alarms before detecting a potential event. Note that this alarm

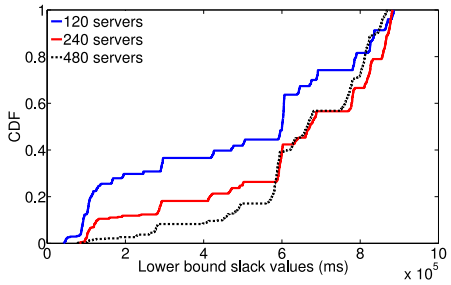


Fig. 19. CDF of LB slack under various numbers of servers (4 weeks).

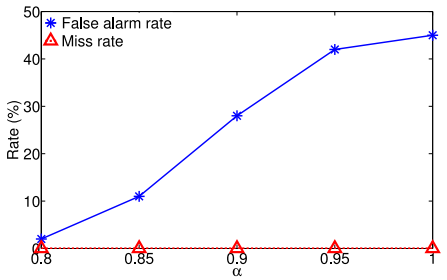
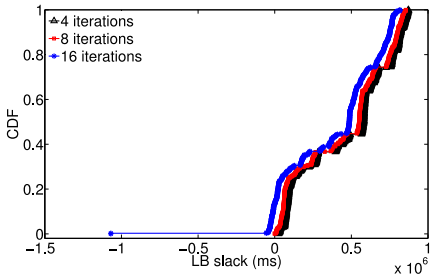


Fig. 20. Deadline miss rate and false alarm rate under varying α.

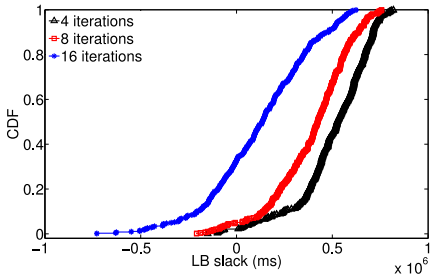
rate is very small compared to the whole time window, since power capping happens in at most 5% cases. Therefore alarms are also generated rarely. Since waiting for multiple alarms increases the latency in detection, the total power capping latency increases as the value of α decreases. However, as this latency increase happens only in the detection phase that is negligible compared to the total capping latency, there is hardly any impact on deadline miss rates. The figure shows a deadline miss rate of 0 under varying α.

6.3.4 Scalability in Number of Control Iterations. Now we consider a conservative case where multiple iterations of control loop are required to settle to a sustained power level [22, 44, 65]. The number of iterations required for the rack-level loop as experimented in Reference [65] can be up to 16 in the worst case (which happens very rarely). Hence, we now conduct experiments considering multiple numbers of control iterations (up to 16 assuming a pessimistic scenario). We plot the results in Figure 21 for various numbers of servers under various number of iterations. As shown in Figure 21(a), for 120 servers under the 16-iteration case, we have 13% cases with negative slack, meaning that the LB trip times were missed. However, the UB trip times were met in 100% cases. Note that we have considered a quite pessimistic setup here, because using 16 iterations as well as trying to meet the lower bound of trip times are both very conservative considerations. For 120 servers under 8 iterations, in 0.13% cases slacks were negative. However, in 80% cases the slacks were above 92.492s, 66.694s, and 22.238s for 4, 8, and 16 iterations, respectively, indicating that the trip times were easily met, and the system could oversubscribe safely. For 4 iterations, the minimum slack was 23.2s. To preserve figure resolution, we do not show the UB slacks, since they were all positive. For 480 servers (Figure 21(b) and (c)), 98.95%, 97.86%, 94.93%, and 67.2% LB trip times were met for 2, 4, 8, and 16 iterations, respectively. For 240 nodes, we miss deadlines in 5% cases under 8 iterations and 13.94% cases under 16 iterations.

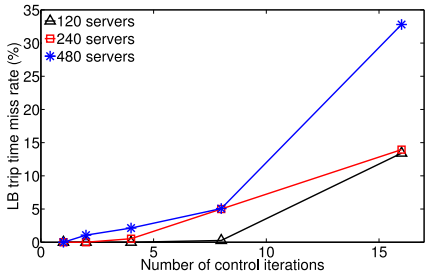
For all cases we met UB trip times in 100% cases. Note that assuming 16 iterations and considering the LB trip times are very conservative assumption as it can rarely happen. Hence, the above



(a) **LB slack for 120 servers**



(b) **LB slack for 480 servers**



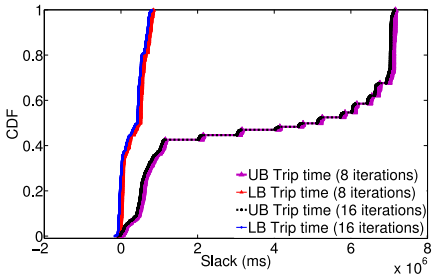
(c) **Miss rate (LB trip time)**

Fig. 21. Multi-iteration capping under event-driven protocol (4 weeks).

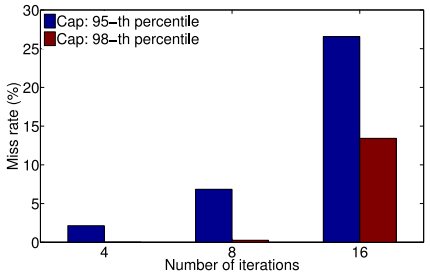
results show that, even for 480 servers, the latencies incurred in CapNet for power capping remain within the conservative latency requirements in most cases.

6.3.5 Experiments under Varying Caps. In all experiments we have performed so far, CapNet was able to meet UB trip times. Now we make some setup changes to encounter some scenario where UB trip times can be smaller by making oversubscription magnitude higher. For this purpose, we now decrease the cap to decrease the trip times so as to make scenarios to miss upper bound trip times to see the robustness of the protocol. Now again we set the 95th percentile of aggregate power as the cap. This would give the previous capping events shorter deadlines, since a smaller cap implies a larger magnitude of oversubscription. For the sake of experiment time, we only tested with 120 servers and their 4-week data traces. Figure 22 shows that we now miss more LB trip times and miss some UB trip times as well, since the deadlines now become shorter. However, UB trip times are missed only in 0.11% and 1.02% cases under 8 and 16 iterations, respectively, while LB deadlines were missed in 2.14%, 6.84%, and 26.56% cases under 4, 8, and 16 iterations, respectively. All deadlines were met for up to 3 iterations (not shown in the figures). We have shown the results only for higher number of iterations that rarely happen. The results show the robustness for larger magnitude of oversubscription in that even when we use 16 iterations only 1.02% UB trip times are missed.

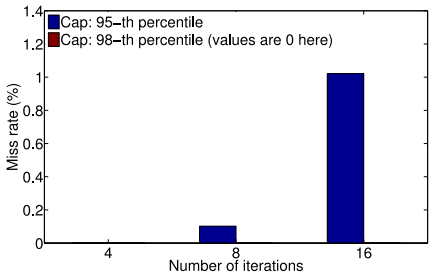
6.3.6 Experiments in Presence of Multiple Clusters. We have shown through data center trace analysis in Figure 17(c) that the probability that two clusters are over the cap simultaneously is no greater than 0.0056. Yet, in this section we perform some experiment from a pessimistic point of view. In particular, we perform an experiment and see the performance of CapNet under an interfering cluster.



(a) **CDF of slack values (Cap: 95-th percentile)**



(b) **LB trip time miss rate**

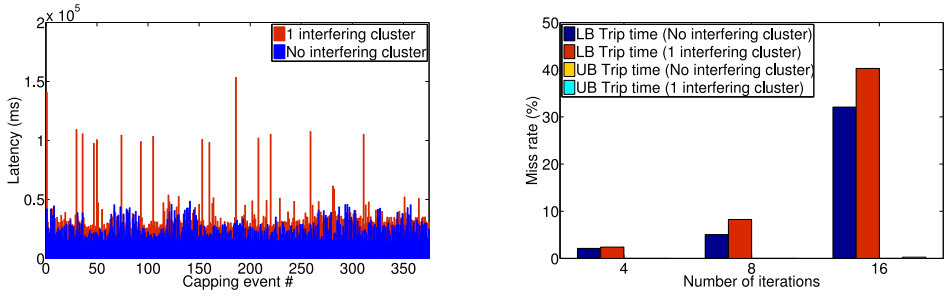


(c) **UB trip time miss rate**

Fig. 22. Capping under different caps on 120 servers (4 weeks).

We mimic an interfering cluster of 480 servers in the following way. We select a nearby cluster and place a pair of motes in the rack: one at the ToR and the other inside the rack. We set their Tx power at maximum (0dBm). The mote at the ToR represents its manager and carries on a pattern of communication like a real manager to control 480 servers. The mote inside the rack responds as if it were connected to each of 480 servers. Specifically, the manager executes a detection phase of $100 * 480$ms, and the node in the rack randomly selects a slot between 1 and 480. On that slot, it generates an alarm with probability 5%, since capping happens in no more than 5% cases. Whenever the manager receives the alarm, it generates a burst of communication in the pattern like what it would have done for 480 servers. After finishing this pattern of communication it resumes the detection phase.

We run the main cluster (system used for experiment) using 4-week data traces, and plot the results in Figure 23. Figure 23(a) shows the latencies for different capping events in 4-week data both under interference and without interference (when there was no other cluster). Under interfering cluster, the delays mostly increase. This happens because the event-driven protocol experiences packet loss and uses retransmission for those, thereby increasing network delays. While the maximum increase was 124.63s, in 80% of the cases the increase was less than 15.089s. We noticed that such big increase happened due to the loss of alarms in a detection phase that resulted in a detection in the next phase (i.e., while the phase length is 48s). Still power capping was successful in all cases but those when the control broadcast was lost. Among 375 events, 4 broadcasts were lost at some server even after 2 repetitions, resulting in control failure in 1.06% cases. This value became 0 in multi-iteration cases. For multi-iteration cases, at least one control broadcasts was successful that resulted in no capping failure for control message loss. However, as the delay due to transmission failure and recovery increased in detection phase, we experienced capping failure. For 16 iterations, we missed the upper bound of trip time in 40.27% cases and lower bound of trip



(a) **Capping latency** (b) **Miss rate**

Fig. 23. Capping for 480 servers under interfering cluster.

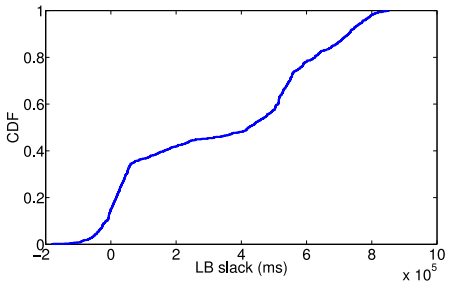


Fig. 24. CDF of LB slack under node failure (120 servers, 4 weeks).

times in 32.08% cases. However, we use a conservative assumption here. For 4 iterations, the miss rate was 5.06% and 8.26% only. And for 2 iterations they are only 2.13% and 2.4%, which are very marginal. The result indicates that even under interference, CapNet demonstrates robustness in meeting the real-time requirements of power capping.

6.3.7 Handling Node Failure. Note that, for each power capping manager, there is a preassigned backup power capping manager from a nearby rack that will take over the management of its servers when it fails. A power capping manager uses a separate 802.15.4 radio to communicate with its backup manager through a different channel that is not used for communication with servers in either cluster. Note that a manager B can periodically send its heartbeat to its backup manager, say, A. If A does not receive a certain number of heartbeats (i.e., receives no heartbeat in a certain time window) from B, then it will decide that B has failed. Since we do not have such manager with two radios physically designed, for experiments, we emulate a scenario where a manager B fails and manager A takes over the management of B's servers. We want to evaluate the performance on B's servers under this scenario. We take 120 servers under B and consider their 4-week data. A has another set of 120 servers. A uses channel 20, B uses channel 26. Both clusters use 8-iteration control.

To emulate a failure, we turn off B. A then initiates the fail over process as follows. First, it asks its own servers to switch to B's channel, and it itself switches to B's original channel, and asks B's nodes to join.All broadcasts are repeated 2 times. Figure 24 shows the CDF of lower bound slacks for 120 servers of B for entire 4 weeks. In approximately 50% cases, the latencies are very long, because for those cases A manages 120 servers of B in addition to its own 120 servers. In few cases, the LB deadlines were missed. This happened due to extra communication and time spent in the process of joining A's clusters. However, the UB deadlines were met in all cases. These results

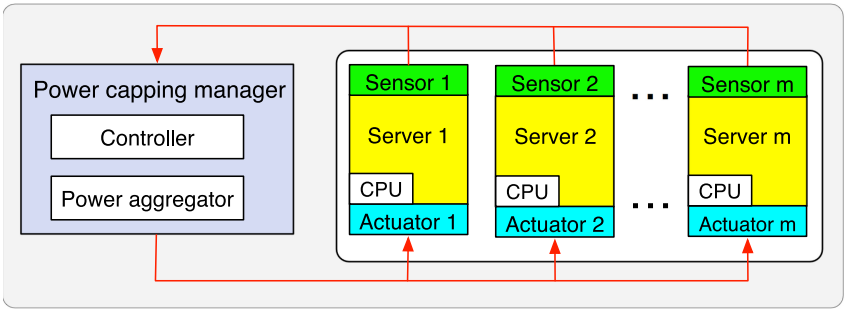


Fig. 25. Feedback control loops architecture for power capping.

show that power manager failure scenario can be handled in CapNet protocol without causing severe performance degradation.

7 DISCUSSIONS AND FUTURE WORK

While our article addresses feasibility, protocol design and implementation, several engineering challenges such as closing the control loops as a cyber-physical system, addressing security, electromagnetic interference (EMI), and compliance need to be studied in the future.

7.1 Closing the Feedback Loops Using Control Theory

We have developed network protocol for power capping over wireless. Such a system's performance can be optimized through a cyber-physical codesign approach, namely by developing data center power capping system as a cyber-physical system [46]. Such a cyber-physical codesign framework can be developed based on a control-theoretic approach [47]. In this way, more accurate power capping can be done compared to heuristic solutions as was studied in existing work for single server [41] and also for cluster-level power capping [64] over a wired network. Thus, implementing CapNet based on feedback control theory over wireless is a promising future work. Specifically, we can consider there are m servers in a cluster numbered as $1, 2, 3, \ldots, m$. To model the power consumption, let $p_i(k)$ be the power consumption of server i and $f_i(k)$ be the frequency level of the processor of server i at time k. The **cap** (set point) is c. The aggregate power consumption at time k of the cluster is $p_{\text{aggr}}(k) = \sum_{i=1}^{m} p_i(k)$. Figure 25 shows the feedback loops architecture of the cluster. The control goal is to guarantee that $p_{\text{aggr}}(k)$ converges to c within trip time (settling time). We can leverage on the relationship between the ***controlled variable*** $p_{\text{aggr}}(k)$ and the ***manipulated variable*** $f_i(k)$ studied in Reference [65] and represent the dynamic model of the system (in matrix form) as

$$\text{p(k + 1)} = \text{p(k)} + \text{A}\delta\text{f(k)},$$

where $\text{p(k)} = [p_1(k), \ldots, p_m(k)]^T$; $\delta\text{f(k)} = \text{f(k)} - \text{f(k-1)} = [\delta f_1(k), \ldots, \delta f_m(k)]^T$.

7.2 Extending CapNet beyond Power Capping

While we have considered only power capping management in our work, an enterprise data center needs a myriad of management functionalities. The continuous, low-cost, and efficient operation of large-scale data centers heavily depends on its management network and system. Such management functionalities include powering on/off a server, motherboard sensor telemetry, cooling management, various other power management, temperature, and humidity control for better and safer server operation. Higher-level management capabilities such as system re-imaging, network configuration, (virtual) machine assignments, and server health monitoring [17, 34] depend largely

on such managements. Hence, in the future, we would like to extend CapNet to include all such management functionalities as an important class of industrial Internet of Things [60].

There can be many ways for measuring the temperature and humidity distributions inside a data center. Many modern servers also have several onboard sensors that monitor the thermal conditions near key server components, such as the CPUs, disks, and I/O controllers. These sensors are used to detect and prevent hardware failures due to overheating. Some recent servers also have temperature sensors at the air intake, which our system can exploit to estimate room conditions from these sensors. Modern server hardware has built-in power metering capabilities (e.g., using motherboard or power supply based power sensors). As discussed before, to make CapNet more effective, future servers should be designed with embedded wireless chips on motherboard. Thus all of these sensors can be incorporated into the same CapNet framework to include all such monitoring functions. Integrating all management and monitoring functions of a data center into one framework can leverage our previous work on shared wireless sensor network that hosts multiple applications [23, 68].

7.3 Two-Tier Architecture

In the future, we plan to extend CapNet to cover an entire data center through a second-tier network where the power capping managers of all clusters are connected through TV spectrum White Space band [54–56]. TV *white spaces* refer to the allocated but unused TV channels and can be used by unlicensed devices as secondary users [9, 10]. They can easily penetrate obstacles and hence hold enormous potential for wireless sensor network applications that need long transmission range. Since White Spaces are available mostly everywhere, especially in indoor environments and can be used for communication over long distances [21], we shall be able to cover all power capping managers within a single-hop network for any existing data center. Therefore, White Spaces networking can be a promising approach for wireless DCM. Our recent design of sensor network over white spaces (SNOW) has shown the feasibility of enabling asynchronous, low power, bi-directional, and massively concurrent communications between numerous sensors and a base station over long distances [52, 54–56]. Thus, in the future, we shall explore to adopt the SNOW technology as low-power wide-area network for wireless DCM.

7.4 Security and Resilience

A key challenge and limitation of CapNet is that it raises security concerns of the data center management system. Since the system relies on wireless control, someone might be able to maliciously tap into the wireless network and take control of the data center. Using low-power wide-area network technology for data center management as discussed above can make this security issue more serious as its wireless communication can reach far outside the data center due to its long-range capability [35]. Such security issues will be studied in the future. In general, there are two typical approaches to handle this security issue. First, the signal itself should be attenuated by the time it reaches outside the building. We can identify secure locations inside the data center from which the controller can communicate and identify a signature for the controllers that would be known to the server machines. We can also use shielding within the data center to keep the RF signals contained within the enclosed region. Second, it is possible to encrypt wireless messages, for example, using MoteAODV (+AES) [20].

In the future, we shall also study to make the CapNet system more resilient against various faults. For example, currently to handle the failure of a sensor/server node we consider the maximum power consumption from it to keep CapNet operational. Such an approach leads to false positive events. Hence, more effective techniques can be developed to handle such faults in the future.

7.5 EMI and Compliance

While less emphasized in research studies, a practical concern of introducing wireless communications in data centers is that they do not adversely impact other devices. There are FCC certified IEEE 802.15.4 circuit design available (e.g., Reference [11]). Previous work has also used WiFi and ZigBee in live data centers for monitoring purposes [43]. In some data centers where there is excessive EMI, wireless management can be affected by EMI that we shall study in the future. We shall also study DCM fault detection, isolation, and node migration in future work.

8 RELATED WORK

To reduce the capital spending on data centers, enterprise data centers use an over-subscription approach as studied in References [27, 30, 44, 50], which is similar to over-booking in airline reservations. For better power utilization, most servers nowadays ship with mechanisms for power capping that allow limiting the peak consumption to a set threshold [41, 53]. Therefore, server vendors and data center solutions providers have started to offer power capping solutions [12, 13]. Power capping using feedback control algorithms [66] has been studied for individual servers. Capacity waste can be better avoided by coordinating the caps across multiple servers. That is, when some servers in a cluster or application are running at lower load, the power left unused could be used by other servers to operate at high power levels than would be allowed by their static cap. The study of this article concentrates on coordinated power capping, which is more desirable in data centers as it allows servers to exploit power left unused by other servers. While the methods that coordinate the power caps dynamically across multiple servers and applications have been studied before [28, 39, 44, 45, 51, 65, 69], all existing solutions rely on wired network for controller-server communication. In contrast, we focus on wireless networking for power capping.

In this article, we show the proof-of-concept results for using low-power wireless for data center management. We show that even when wireless network is used, we achieve the reliability and timeliness required and at the same time achieve the benefits of using a wireless network. Wireless networks can be built at very low cost, can self-configure, and are resilient to failures. As an added benefit, they are also optimized for low power consumption, even when the main power supply is down due to failures. We gain the flexibility to grow the network and eliminate the wiring complexity and management costs in dense server solutions. Previous work on using wireless network in data centers exists on applications to high bandwidth (e.g., with 60GHz radio) production data network [19, 29, 32, 33, 70]. In contrast, CapNet is targeted at data management functions that have much lower bandwidth requirement while demanding real-time communication through racks. This is the first exploration of a low-power wireless sensor network approach for data center management that is traditionally used for various outdoor monitoring applications such as civil infrastructure monitoring [38, 42, 49], habitat monitoring [26, 48, 62], environment monitoring [40], volcano monitoring [67], and target tracking [63].

RACNet [43] is a passive monitoring solution in the data center that monitors temperature or humidity across racks where all radios are mounted at the top of the rack. It also depends on a combination of wired and wireless communications to scale. Our solution enables active control and requires communication through racks and server enclosures and hence encounters fundamentally different challenges. Also, RACNet also does not have real-time features, while CapNet is designed to meet the real-time requirements in power capping and provides analytical latency bounds.

9 CONCLUSION

DCM is increasingly becoming a significant challenge for enterprises hosting large-scale online and cloud services. The continuous, low-cost, and efficient operation of a data center heavily

depends on its management network and system. Today, DCM is typically designed as out-of-band (through a controller that could be reached out-of-band on a separate path that is distinct from the primary network path and the operating system). The limitations of the typical management network are many-fold. Primarily it is a fixed wired network, and hence scaling it for a large number of servers increases its cost. In addition, with server densities increasing over recent years, this network also has to be cabled correctly and the management of this network parallels the complexity of managing a data network, since it needs to be networked with multiple switches and routers. Also, an important point to note here is that the management network is the last frontier; if this fails and the primary network is down, then we will not be able to manage the servers for which a significant amount of capital was invested. While we can increase redundancy in a wired solution by constructing multiple paths and using redundant switches, the complexity and cost incurred increases as well. To address the cost and complexity issues with current network, we have proposed a counter-intuitive solution—to use a low-cost wireless solution—and attempted to satisfy the reliability and timeliness requirements of complex management functionalities of a data center.

In our article, we have chosen power capping to be a representative of a challenging management functionality that data centers need to perform in a timely manner, when running on an over-subscribed power budget. We have first studied the feasibility in reliable communication through server enclosures in commercial data centers. Our results show that even when a wireless network is used, we achieve the reliability and timeliness required, and at the same time achieve the benefits of using a wireless network. Wireless networks can be built at very low cost, can self-configure, and are resilient to failures. As an added benefit, they are also optimized for low power consumption, even when the main power supply is down due to failures. We gain the flexibility to grow the network and eliminate the wiring complexity and management costs in dense server solutions. In our article, we have hence designed power capping policies using our proposal that perform within constraints required by the data center. We have discussed failure scenarios and identified solutions to address the scenarios.

As a proof-of-concept design, we have developed CapNet, a low-cost, real-time wireless management network for data centers and validated its feasibility for power capping. We deployed and evaluated CapNet in an enterprise data center. Using server power traces, our experimental results on a cluster of 480 servers inside the data center show that CapNet can meet the real-time requirements of power capping. CapNet represents a promising step towards applying low-power wireless networks to time-critical, close-loop control in DCM. In the future, we would like to close the control loops as a cyber-physical system for performance optimization, and address security, EMI, and compliance.

ACKNOWLEDGMENTS

This work was supported by Microsoft Research and NSF through grants CNS-1320921 (NeTS), 1144552 (NeTS) and 1646579 (CPS).

REFERENCES

[1] [n. d.]. Private communication with data center operators.
[2] [n. d.]. Retrieved from http://blog.softlayer.com/2011/before-they-were-softlayer-data-centers/.
[3] [n. d.]. Retrieved from www.cdwg.com/shop/products/Digi-Passport-48-console-server/1317701.aspx.
[4] [n. d.]. Retrieved from http://www.cdwg.com/shop/search/Servers-Server-Management/Servers/x86-Based-Servers/result.aspx?w=S62&pCurrent=1&p=200008&a1520=002200.
[5] [n. d.]. Retrieved from http://www.cdwg.com.
[6] [n. d.]. Retrieved from http://www.cdwg.com/shop/search/Networking-Products/Ethernet-Switches/Fixed-Managed-Switches/result.aspx?w=N11&MaxRecords=25&SortBy=TopSellers.

[7] [n. d.]. Retrieved from http://www.digikey.com/us/en/techzone/wireless/resources/articles/comparing-low-power-wireless.html.
[8] [n. d.]. Retrieved from http://literature.rockwellautomation. com/idc/groups/literature/documents/sg/ 1489-sg001_-en-p.pdf.
[9] [n. d.]. FCC, ET Docket No FCC 08-260, November 2008.
[10] [n. d.]. FCC, Second Memorandum Opinion and Order, ET Docket No FCC 10-174, September 2010.
[11] [n. d.]. Retrieved from http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en535967.
[12] [n. d.]. Retrieved from http://www.intel.com/content/dam/doc/case-study/data-center-efficiency-xeon-baidu-case-study.pdf.
[13] [n. d.]. Retrieved from http://h20000.www2.hp.com/bc/docs/support/SupportManual/c01549455/c01549455.pdf.
[14] [n. d.]. Smart Rack Solutions. Retrieved from https://www.raritan.com/landing/smart-rack-solutions.
[15] [n. d.]. TinyOS Community Forum. Retrieved from http://www.tinyos.net/.
[16] [n. d.]. Wireless Sensor Networks for Data Centers. Retrieved from https://energy.gov/eere/femp/wireless-sensor-networks-data-centers.
[17] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. 2008. A scalable, commodity data center network architecture. In *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication (SIGCOMM'08)*. ACM, 63–74. DOI: https://doi.org/10.1145/1402958.1402967
[18] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. 2008. A scalable, commodity data center network architecture. In *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication (SIGCOMM'08)*. ACM, New York, NY, USA, 63–74. DOI: https://doi.org/10.1145/1402958.1402967
[19] E. Baccour, S. Foufou, R. Hamila, and M. Hamdi. 2015. A survey of wireless data center networks. In *Proceedings of the 2015 49th Annual Conference on Information Sciences and Systems (CISS'15)*. 1–6. DOI: https://doi.org/10.1109/CISS.2015.7086853
[20] Werner Backes and Jared Cordasco. [n. d.]. MoteAODV – an AODV implementation for TinyOS 2.0. In *Proceedings of the Workshop in Information Security Theory and Practice (WISTP'10)*.
[21] Paramvir Bahl, Ranveer Chandra, Thomas Moscibroda, Rohan Murty, and Matt Welsh. 2009. White space networking with Wi-fi like connectivity. In *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication (SIGCOMM'09)*. ACM, New York, NY, 27–38. DOI: https://doi.org/10.1145/1592568.1592573
[22] A. A. Bhattacharya, D. Culler, A. Kansal, S. Govindan, and S. Sankar. 2012. The need for speed and stability in data center power capping. In *Proceedings of the 2012 International Green Computing Conference (IGCC'12)*. 1–10. DOI: https://doi.org/10.1109/IGCC.2012.6322253
[23] Sangeeta Bhattacharya, Abusayeed Saifullah, Chenyang Lu, and Gruia-Catalin Roman. 2010. Multi-application deployment in shared sensor networks based on quality of monitoring. In *Proceedings of the 2010 16th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'10)*. IEEE Computer Society, 259–268. DOI: https://doi.org/10.1109/RTAS.2010.20
[24] Gong Chen, Wenbo He, Jie Liu, Suman Nath, Leonidas Rigas, Lin Xiao, and Feng Zhao. 2008. Energy-aware server provisioning and load dispatching for connection-intensive internet services. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation (NSDI'08)*. USENIX Association, Berkeley, CA, 337–350. DOI: http://dl.acm.org/citation.cfm?id=1387589.1387613
[25] J. Choi, S. Govindan, B. Urgaonkar, and A. Sivasubramaniam. 2008. Profiling, prediction, and capping of power consumption in consolidated environments. In *Proceedings of the 2008 IEEE International Symposium on Modeling, Analysis and Simulation of Computers and Telecommunication Systems*. 1–10. DOI: https://doi.org/10.1109/MASCOT.2008.4770558
[26] J. P. Dominguez-Morales, A. Rios-Navarro, M. Dominguez-Morales, R. Tapiador-Morales, D. Gutierrez-Galan, D. Cascado-Caballero, A. Jimenez-Fernandez, and A. Linares-Barranco. 2016. Wireless sensor network for wildlife tracking and behavior classification of animals in donana. *IEEE Commun. Lett.* 20, 12 (2016), 2534–2537. DOI: https://doi.org/10.1109/LCOMM.2016.2612652
[27] Xiaobo Fan, Wolf-Dietrich Weber, and Luiz Andre Barroso. 2007. Power provisioning for a warehouse-sized computer. In *Proceedings of the 34th Annual International Symposium on Computer Architecture (ISCA'07)*. ACM, New York, NY, 13–23. DOI: https://doi.org/10.1145/1250662.1250665
[28] M. E. Femal and V. W. Freeh. 2005. Boosting data center performance through non-uniform power allocation. In *Proceedings of the 2nd International Conference on Autonomic Computing (ICAC'05)*. 250–261. DOI: https://doi.org/10.1109/ICAC.2005.17
[29] Avery John Francois. 2016. *Wireless 60 GHz Rack to Rack Communication in a Data Center Environment*. Master's thesis. Rochester Institute of Technology.
[30] Xing Fu, Xiaorui Wang, and Charles Lefurgy. 2011. How much power oversubscription is safe and allowed in data centers. In *Proceedings of the 8th ACM International Conference on Autonomic Computing (ICAC'11)*. ACM, New York, NY, 21–30. DOI: https://doi.org/10.1145/1998582.1998589

[31] Hamilton. 2008. Retrieved from http://perspectives.mvdirona.com.
[32] A. S. Hamza, J. S. Deogun, and D. R. Alexander. 2016. Wireless communication in data centers: A survey. *IEEE Commun. Surv. Tutor.* 18, 3 (2016), 1572–1595.
[33] Tao Huang, Jiao Zhang, and Yunjie Liu. 2016. A mechanism achieving low latency forwireless datacenter applications. *Comput. Sci. Inf. Syst.* 13, 2 (2016), 639–658.
[34] Michael Isard. 2007. Autopilot: Automatic data center management. *Operat. Syst. Rev.* 41, 2 (2007), 60–67.
[35] Dali Ismail, Mahbubur Rahman, and Abusayeed Saifullah. 2018. Low-power wide-area networks: Opportunities, challenges, and directions. In *Proceedings of the Workshop Program of the 19th International Conference on Distributed Computing and Networking (Workshops ICDCN'18)*. ACM, Article 8, 6 pages. DOI: https://doi.org/10.1145/3170521.3170529
[36] Aman Kansal, Feng Zhao, Jie Liu, Nupur Kothari, and Arka Bhattacharya. 2009. *Joulemeter: Virtual Machine Power Measurement and Management.* Technical Report. Retrieved from DOI: https://www.microsoft.com/en-us/research/publication/joulemeter-virtual-machine-power-measurement-and-management/
[37] Aman Kansal, Feng Zhao, Jie Liu, Nupur Kothari, and Arka A. Bhattacharya. 2010. Virtual machine power metering and provisioning. In *Proceedings of the 1st ACM Symposium on Cloud Computing (SoCC'10)*. ACM, 39–50. DOI: https://doi.org/10.1145/1807128.1807136
[38] Sukun Kim, Shamim Pakzad, David Culler, James Demmel, Gregory Fenves, Steven Glaser, and Martin Turon. 2007. Health monitoring of civil infrastructures using wireless sensor networks. In *Proceedings of the 6th International Conference on Information Processing in Sensor Networks (IPSN'07)*. ACM, New York, NY, 254–263. DOI: https://doi.org/10.1145/1236360.1236395
[39] V. Kontorinis, L. E. Zhang, B. Aksanli, J. Sampson, H. Homayoun, E. Pettis, D. M. Tullsen, and T. Simunic Rosing. 2012. Managing distributed UPS energy for effective power capping in data centers. In *Proceedings of the 2012 39th Annual International Symposium on Computer Architecture (ISCA'12)*. 488–499. DOI: https://doi.org/10.1109/ISCA.2012.6237042
[40] Koen Langendoen, Aline Baggio, and Otto Visser. 2006. Murphy loves potatoes: Experiences from a pilot sensor network deployment in precision agriculture. In *Proceedings of the 20th International Conference on Parallel and Distributed Processing (IPDPS'06)*. IEEE Computer Society, Washington, DC, 174–174. DOI: http://dl.acm.org/citation.cfm?id=1898953.1899108
[41] Charles Lefurgy, Xiaorui Wang, and Malcolm Ware. 2007. Server-level power control. In *Proceedings of the 4th International Conference on Autonomic Computing (ICAC'07)*. IEEE Computer Society, 4–. DOI: https://doi.org/10.1109/ICAC.2007.35
[42] B. Li, Z. Sun, K. Mechitov, G. Hackmann, C. Lu, S. J. Dyke, G. Agha, and B. F. Spencer. 2013. Realistic case studies of wireless structural control. In *Proceedings of the 2013 ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS'13)*. 179–188.
[43] Chieh-Jan Mike Liang, Jie Liu, Liqian Luo, Andreas Terzis, and Feng Zhao. 2009. RACNet: A high-fidelity data center sensing network. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys'09)*. ACM, New York, NY, 15–28. DOI: https://doi.org/10.1145/1644038.1644041
[44] Harold Lim, Aman Kansal, and Jie Liu. 2011. Power budgeting for virtualized data centers. In *Proceedings of the 2011 USENIX Conference on USENIX Annual Technical Conference (USENIXATC'11)*. USENIX Association, Berkeley, CA, 5–5. DOI: http://dl.acm.org/citation.cfm?id=2002181.2002186
[45] Yanpei Liu, Guilherme Cox, Qingyuan Deng, Stark C. Draper, and Ricardo Bianchini. 2017. Fast power and energy management for future many-core systems. *ACM Trans. Model. Perform. Eval. Comput. Syst.* 2, 3, Article 17 (Sept. 2017), 17:1–17:31 pages. DOI: https://doi.org/10.1145/3086504
[46] C. Lu, A. Saifullah, B. Li, M. Sha, H. Gonzalez, D. Gunatilaka, C. Wu, L. Nie, and Y. Chen. 2016. Real-time wireless sensor-actuator networks for industrial cyber-physical systems. *Proc. IEEE* 104, 5 (2016), 1013–1024.
[47] J. M. Maciejowski. 2002. *Predictive Control With Constraints.* Prentice Hall.
[48] Alan Mainwaring, David Culler, Joseph Polastre, Robert Szewczyk, and John Anderson. 2002. Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02)*. ACM, New York, NY, 88–97. DOI: https://doi.org/10.1145/570738.570751
[49] A. B. Noel, A. Abdaoui, T. Elfouly, M. H. Ahmed, A. Badawy, and M. S. Shehata. 2017. Structural health monitoring using wireless sensor networks: A comprehensive survey. *IEEE Commun. Surv. Tutor.* 19, 3 (2017), 1403–1423.
[50] Steven Pelley, David Meisner, Pooya Zandevakili, Thomas F. Wenisch, and Jack Underwood. 2010. Power routing: Dynamic power provisioning in the data center. In *Proceedings of the 15th Edition of ASPLOS on Architectural Support for Programming Languages and Operating Systems (ASPLOS XV)*. ACM, New York, NY, 231–242. DOI: https://doi.org/10.1145/1736020.1736047
[51] Ramya Raghavendra, Parthasarathy Ranganathan, Vanish Talwar, Zhikui Wang, and Xiaoyun Zhu. 2008. No "power" struggles: Coordinated multi-level power management for the data center. In *Proceedings of the 13th International*

Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS XIII). ACM, New York, NY, 48–59. DOI: https://doi.org/10.1145/1346281.1346289

[52] Mahbubur Rahman and Abusayeed Saifullah. 2018. Integrating low-power wide-area networks in white spaces. In *Proceedings of the ACM/IEEE Conference on Internet-of-Things Design and Implementation (IoTDI'18)*. 255–260.

[53] Parthasarathy Ranganathan, Phil Leech, David Irwin, and Jeffrey Chase. 2006. Ensemble-level power management for dense blade servers. In *Proceedings of the 33rd Annual International Symposium on Computer Architecture (ISCA'06)*. IEEE Computer Society, 66–77. DOI: https://doi.org/10.1109/ISCA.2006.20

[54] Abusayeed Saifullah, Mahbubur Rahman, Dali Ismail, Chenyang Lu, Ranveer Chandra, and Jie Liu. 2016. SNOW: Sensor network over white spaces. In *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems (SenSys'16)*. ACM, New York, NY, 272–285. DOI: https://doi.org/10.1145/2994551.2994552

[55] Abusayeed Saifullah, Mahbubur Rahman, Dali Ismail, Chenyang Lu, Jie Liu, and Ranveer Chandra. 2017. Enabling reliable, asynchronous, and bidirectional communication in sensor network over white spaces. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems (SenSys'17)*.

[56] Abusayeed Saifullah, Mahbubur Rahman, Dali Ismail, Chenyang Lu, Jie Liu, and Ranveer Chandra. 2018. Low-power wide-area networks over white spaces. *ACM/IEEE Trans. Netw.* 26, 4 (2018), 1893–1906. DOI: https://doi.org/10.1109/TNET.2018.2856197

[57] Abusayeed Saifullah, Sriram Sankar, Jie Liu, Chenyang Lu, Bodhi Priyantha, and Ranveer Chandra. 2014. CapNet: A real-time wireless management network for data center power capping. In *Proceedings of the 2014 IEEE Real-Time Systems Symposium*. 334–345. DOI: https://doi.org/10.1109/RTSS.2014.35

[58] A. Saifullah, Y. Xu, C. Lu, and Y. Chen. 2010. Real-time scheduling for wirelessHART networks. In *Proceedings of the 2010 31st IEEE Real-Time Systems Symposium*. 150–159. DOI: https://doi.org/10.1109/RTSS.2010.41

[59] A. Saifullah, Y. Xu, C. Lu, and Y. Chen. 2014. Distributed channel allocation protocols for wireless sensor networks. *IEEE Trans. Parallel Distrib. Syst.* 25, 9 (Sept. 2014), 2264–2274. DOI: https://doi.org/10.1109/TPDS.2013.185

[60] Emiliano Sisinni, Abusayeed Saifullah, Song Han, Ulf Jennehag, and Mikael Gidlund. 2018. Industrial internet of things: Challenges, opportunities, and directions. *IEEE Trans. Industr. Inf.* (2018).

[61] K. Srinivasan and P. Levis. 2006. RSSI is under appreciated. In *Proceedings of the Third Workshop on Embedded Networked Sensors (EmNets'06)*.

[62] Robert Szewczyk, Alan Mainwaring, Joseph Polastre, John Anderson, and David Culler. 2004. An analysis of a large scale habitat monitoring application. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys'04)*. ACM, New York, NY, 214–226. DOI: https://doi.org/10.1145/1031495.1031521

[63] B. Thiyagarajan, P. Ravisasthiri, P. Lalitha, P. Ambili, S. Thenmozhi, and K. Prem Kumar. 2015. Target tracking using wireless sensor networks: Survey. In *Proceedings of the 2015 International Conference on Advanced Research in Computer Science Engineering; Technology (ICARCSET'15)*. ACM, New York, NY, Article 57, 4 pages. DOI: https://doi.org/10.1145/2743065.2743122

[64] X. Wang and M. Chen. 2008. Cluster-level feedback power control for performance optimization. In *Proceedings of the 2008 IEEE 14th International Symposium on High Performance Computer Architecture*. 101–110. DOI: https://doi.org/10.1109/HPCA.2008.4658631

[65] Xiaorui Wang, Ming Chen, C. Lefurgy, and T. W. Keller. 2012. SHIP: A scalable hierarchical power control architecture for large-scale data centers. *IEEE Trans. Parallel Distrib. Syst.* 23, 1 (2012), 168–176.

[66] Zhikui Wang, Cliff Mccarthy, Xiaoyun Zhu, Partha Ranganathan, and Vanish Talwar. 2009. Feedback control algorithms for power management of servers. In *In Proceedings of the International Workshop on Feedback Control Implementation and Design in Computing Systems and Networks (FeBID'09)*.

[67] Geoff Werner-Allen, Konrad Lorincz, Jeff Johnson, Jonathan Lees, and Matt Welsh. 2006. Fidelity and yield in a volcano monitoring sensor network. In *Proceedings of the 7th Symposium on Operating Systems Design and Implementation (OSDI'06)*. USENIX Association, Berkeley, CA, 381–396. DOI: http://dl.acm.org/citation.cfm?id=1298455.1298491

[68] You Xu, Abusayeed Saifullah, Yixin Chen, Chenyang Lu, and Sangeeta Bhattacharya. 2010. Near optimal multi-application allocation in shared sensor networks. In *Proceedings of the 11th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'10)*. ACM, New York, NY, 181–190. DOI: https://doi.org/10.1145/1860093.1860118

[69] Yanwei Zhang, Yefu Wang, and Xiaorui Wang. 2011. Capping the electricity cost of cloud-scale data centers with impacts on power markets. In *Proceedings of the 20th International Symposium on High Performance Distributed Computing (HPDC'11)*. ACM, 271–272. DOI: https://doi.org/10.1145/1996130.1996170

[70] Xia Zhou, Zengbin Zhang, Yibo Zhu, Yubo Li, Saipriya Kumar, Amin Vahdat, Ben Y. Zhao, and Haitao Zheng. 2012. Mirror mirror on the ceiling: Flexible wireless links for data centers. In *Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM'12)*. ACM, New York, NY, 443–454. DOI: https://doi.org/10.1145/2342356.2342440

Received February 2018; revised September 2018; accepted September 2018