CapNet: A Wireless Management Network for Power Capping in Data Centers

Blind review

Abstract

Data center management (DCM) is increasingly becoming a significant challenge for enterprises hosting large scale online and cloud services. Machines need to be continuously monitored, and the scale of operations mandate high reliability and automated management. Reliability in existing wired-based solutions for DCM comes with high cost. In this paper, we propose a new, wireless sensor-based approach that is significantly cost-effective while satisfying the reliability and performance requirements of management functions. Our system implementation, called CapNet, can scale to 100s of 1000s of servers at $<10\%$ the cost compared to wired implementation. We show its feasibility for a time-critical DCM application – Power Capping – in a deployment of 80 machines in two different data centers, and also emulate implementation on 480 machines in an operational data center, using 6-month long power traces.

1. Introduction

The continuous, low-cost, and efficient operation of a datacenter heavily depends on its management network and system. A typical data center management (DCM) system handles physical layer functionality, 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 [12, 23], depend on DCM to work correctly. For this reason, the DCM has to be extremely dependable. It is expected to function even when the servers do not have a working OS or the data network is not configured correctly [9].

Today's DCM is typically designed in parallel to the production data network (in other words, *out of band*), with a combination of Ethernet and *serial connections* for increased redundancy. There is a cluster controller (or controller for short) for a rack or a group of racks, which are connected through Eithernet to a central management server. Within the clusters, each server has a motherboard microcontroller (BIOS) 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 paper presents a different approach to data center management network at the rack granularity, by replacing serial cable connections with low cost wireless links. Wireless networks have intrinsic advantages in this application, as we elaborate in Section 2. First, they are cheaper individually than wired alternatives and the cost scales linearly with the number of servers. For 100,000 servers, the hardware material cost of a wireless solution is only 5.4%-8% the cost of a conventional serial solution (Figure 1). Second, they can be integrated onto the motherboard and made physically small to save precious rack space. Third, they can be self-configuring and self repairing with the broadcast media to prevent human cabling error. Finally, they consume less power. With a small on-board battery, the DCM can continue to function 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.

- Wireless links within a rack may not work. The amount of sheet metals, electronics, and cables may completely shield RF signal propagation within racks.
- 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 a scenario that requires emergency situation handling. Today, data center operators commonly over-subscribe the power infrastructure by installing more servers to an electrical 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 that the aggregated power consumption from all servers behind a circuit breaker approaches the circuit's power capacity, some servers must be slowed down, through dynamic frequency and voltage scaling (DVFS) or CPU time throttling, to prevent the circuit breaker from tripping. Power capping is a critical event that must be handled in real time. A power controller must detect power surge events and send capping commands to corresponding servers. Several of these round trips must be finished within the trip time of a circuit breaker, usually within a few seconds.

This paper studies 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 an event-driven real-time control protocol, called CapNet, for power capping over wireless DCM. 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 each node, and then issues power capping commands to a subset of the servers. We deployed a real wireless DCM on 80 wireless nodes to evaluate its real-time performance. We further use real data center power traces to emulate a 480 servers deployment and show that CapNet can perform as reliably and timely as wired DCM with a fraction of the cost.

2. The Case for Wireless DCM (CapNet)

Typical wired DCM solutions in data centers scale 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. 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 [1]) and additional management network switches ($3000/48port on average [2]). 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 the motherboard, we assume $5 per server for the RF chip and antenna as the motherboard controller is already in place [8].

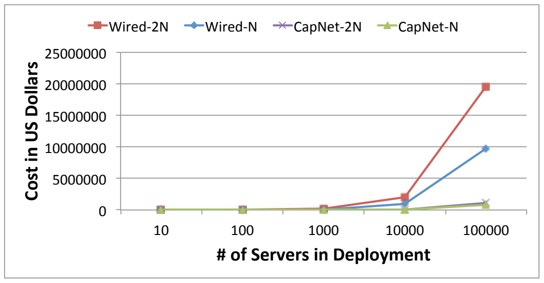


Figure 1. System cost comparison and scalability

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 (in order to capture how cost scales with increase in 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 1 shows the cost comparison across these solutions. We see that a wired N-redundant DCM solution (Wired-N) for 100,000 servers is $12.5\times$ the cost of a wireless N-redundant DCM solution (CapNet-N). When we increase the redundancy of the management network to 2N, the cost of a wired solutions (between Wired-2N and Wired-N) doubles, yet wireless solutions only increase by 36% (mainly due to 2N controllers and 2N switches at the top level). The resulting cost of Wired-2N is $18.4\times$ that of CapNet-2N. Given the significant cost difference between the wired DCM and CapNet, we are motivated to explore whether the technology (IEEE 802.15.4) is indeed feasible for communication within the rack in the next section.

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 (BIOS) microcontroller can remain simple. Although we do not rule out other wireless technologies, we chose to prototype with ZigBee in this paper.

2.3 Radio Environment inside Racks

One of the first questions to answer is whether networks like IEEE 802.15.4 can work *within a rack*, between a management controller and a stack of servers separated by sheet metal boxes. The sheet metals inside the enclosure are known to weaken radio signal strength, giving a harsh environment for radio propagation. RACNet [25] studied wireless characteristics in data centers, but only across racks when all radios are mounted at the top of the rack. We did not find any previous study that evaluated the signal strength within the racks through servers and sheet metal. Therefore, we first perform an in-depth 802.15.4 link layer measurement study based on in-rack radio propagation in real data-center environments.

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 node (also called a *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 2(a) shows the placement of 8 motes inside a bottom sled (which is open in the figure but was closed during the experiment). While measuring the 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 packet reception rate (PRR).

Results. Figure 2(b) shows the cumulative distribution function (CDF) of Received Signal Strength Indicator (RSSI) values at a receiver inside the bottom sled for 1000 transmissions from the node on ToR for different transmission (Tx) power using IEEE 802.15.4 channel 26. For -10dBm 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 [33] 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 2(c) shows the CDF of RSSI values at the same receiver for 1000 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 96%. We observed similar results in all other setups of the measurement study, and omit those results here due to space limit.

The measurement study reveals that low-power wireless, such as IEEE 802.15.4, is viable for communication within data center racks and that the link reliability can be sufficient for telemetry purpose. However, a DCM network must also support real-time control scenarios such as power capping, with high data reliability and timeliness of delivery. In the rest of the paper, we focus on the power capping scenario and design a network protocol for real-time control.

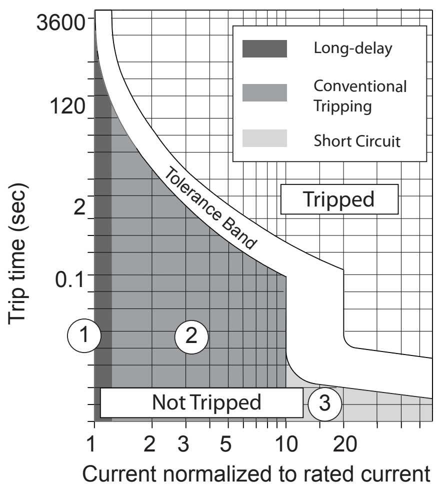


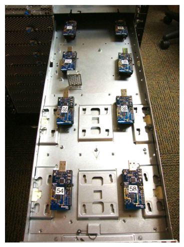
Figure 3. The trip curve of Rockwell Allen-Bradley 1489-A circuit breaker at 40°C [11]. X-axis indicates the magnitudes of oversubscription. Y-axis shows corresponding trip times.

3. CapNet Design Overview

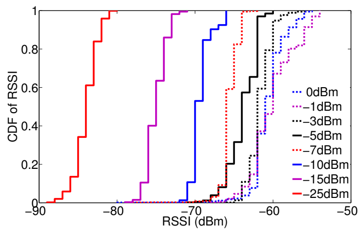
Power over-subscription and power capping are common practice to improve data center utilization. Power capping demands high reliability and low latency from the DCM network. This section gives an overview of the CapNet design.

3.1 The Power Capping Problem

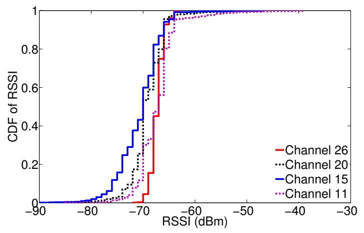
Power infrastructure bears huge capital investment for a data center, up to 40% of the total cost of a large data center that can cost hundreds of millions of US Dollars [22] to build. Hence, it is desirable to use the provisioned infrastructure to its maximum rated capacity. The capacity of a branch circuit is provisioned during design time, based on upstream transformer capacity during normal operation or UPS/Generator capacity when running on backup power. When we do over-subscription [18, 21, 26, 28], we allocate servers in a circuit exceeding the rated capacity that the circuit can support, since not all servers reach their maximum power consumption at the same time. Power capping is he mechanism to make sure that the circuit capacity limit is not exceeded during power surges and prevent power loss or potential damage to expensive equipment. Hence, there is a circuit breaker (CB) that trips to protect the expensive equipment. The peak power consumption above the power cap has a specified time limit, called a *trip time*, depending on the magnitude of over-subscription (as shown in Figure 3 for Rockwell Allen-Bradley 1489-A circuit breaker). If the over-subscription continues for longer than the trip time, the circuit breaker will trip and cause undesired server shutdowns and power outages disrupting data center operation. 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 [11].



(a) Placement of 8 motes in bottom sled



(b) CDF of RSSI at a receiver in bottom sled under different Tx power



(c) CDF of RSSI at a receiver in bottom sled under different channels

Figure 2. Measuring downward signal strength

To enable power capping for a rack or collection of racks (cluster), an aggregation manager[1] collects all servers' power consumption and determines the cluster-level aggregated power consumption. If the aggregate consumption is over the cap, 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. In addition, in some graceful throttling policies, the control messages are delivered by the BIOS to the host OS or VMs, which introduce additional latency due to OS stack [15, 27]. To avoid abrupt changes to application performance, the aggregation manager 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 [26, 35]. These control policies have been studied extensively by previous work and are out of the scope of this paper.

3.2 Power Capping over Wireless DCM

Using wireless DCM does not fundamentally change the power capping architecture. All servers in a cluster [2] incorporate a wireless transceiver that connects to the BIOS microcontroller. The servers are capable of measuring its own power consumption. 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. When the aggregated power consumption approaches the circuit power capacity, the 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 the order of communication to achieve real-time message delivery.

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 can run in parallel. Channel allocation can be done using existing protocols that minimize inter-cluster interference(e.g. [31]), and is not the focus of our paper. For protocol design, we focus on a single cluster of servers.

3.3 A Naive Periodic Protocol

A naive approach for a fine-grained power capping policy is to periodically collect power consumption readings from individual servers. The period must be smaller than an upper bound of the total aggregation latency. The manager periodically computes the aggregate power. Whenever the aggregate power exceeds the cap, it generates a control message and broadcasts the message after suspending all other communications. To handle broadcast failures, it can repeat the broadcast γ times. If the control algorithm requires η-iteration to reach the settling time, then after the first round capping control command is executed, the controller will again run the aggregation cycle to collect all readings from servers, and reconfirm that capping was done correctly. The procedure continues up to $(\eta - 1)$ more iterations. Upon finishing the control, it resumes the periodic aggregation again.

3.4 Event-Driven CapNet

Since power capping is a rare event, the naive periodic protocol is an overkill as it can saturate the wireless media by always preparing for the worst case. Other delay-tolerant telemetry messages cannot get enough network resources. 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.

CapNet's event-driven capping protocol runs in 3-phases: *distributed event detection*, *power aggregation*, and *control*. The event detection phase generates an alarm implying a potential power surge that may require power capping. The event-detection technique can take advantage of usually synchronized peak power characteristics in real data centers. Only if this phase generates an alarm, CapNet runs the second phase which invokes the power aggregation protocol. If

[1] We use the terms 'controller' and 'aggregation manager' interchangeably in the CapNet design.

[2] A *Cluster* is a logical notion indicating a power management unit consisting of one or more racks.

aggregated power exceeds a predefined threshold, the control algorithm on the controller broadcasts control messages requesting a subset of the servers to be capped. The servers react to the capping messages by DVFS or CPU throttling. The details of the protocol is explained in the next section.

4. Power Capping Protocol

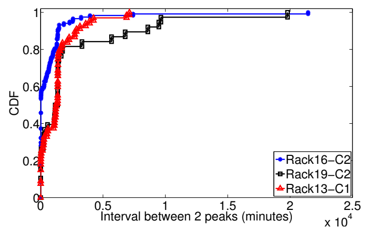
Data center power capping is a rare event. Oversubscribing data centers require to cap only in 1% to 5% of all cases [15]. Therefore, an ideal protocol should generate significant traffic only when a power surge occurs that may require power capping. This motivates the design of an event-driven protocol. We describe the design of the CapNet protocol with insights based on characterization of actual power data from real data centers in the following section.

4.1 Power Peak Analysis of Data Centers

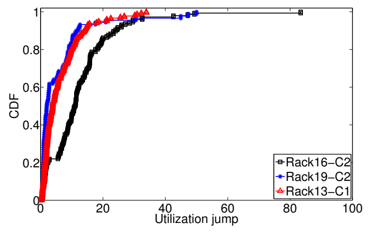
While a global detection is possible by just monitoring at the branch circuit level, it cannot support fine-grained and flexible power capping policies such as those based on individual server-priority or allocating power caps to individual servers based on their power consumption. We design an event detection approach based on local power surges. When a server observes a local power surge based on its own power reading, it triggers 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, the system initiates a power capping action.

The event detection policy is designed based on the characteristics of the power usage in real data centers. In the following, we first analyze the power usage patterns based on data traces collected from multiple data centers, before describing the event-driven protocol in the next subsection.

Data Sets. We use workload demand traces from multiple geo-distributed data centers run by a global corporation over a period of six consecutive months. We specially give results here for data pertaining to 2 representative server clusters for clarity. 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 several other online cloud applications, catering to millions of users around the world. Each cluster uses homogeneous hardware, though there could be differences across clusters. We name the 2 clusters C1 and C2. In both clusters each individual server has CPU utilization data of 6 consecutive months in every 2 minutes 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 [16, 17, 19, 30]. Hence, we use server's CPU utilization as a proxy for power consumption in all trace analyses and experiments. Any rack-level or cluster-level aggregate power is determined as the average utilization over all servers under that rack or cluster. For



(a) Interval between 2 peaks

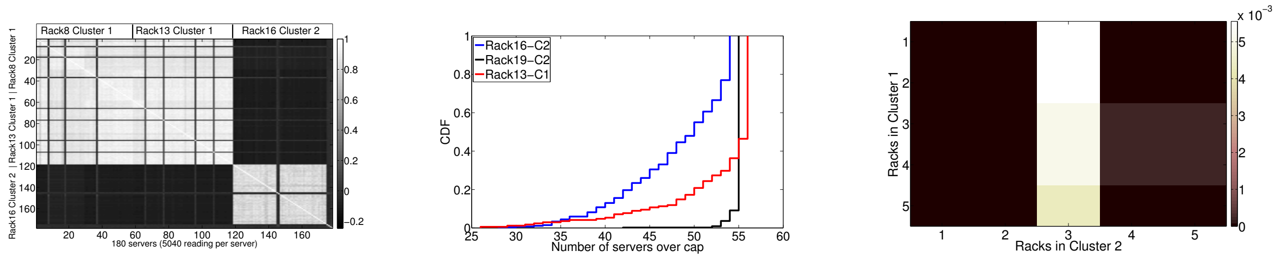


(b) Utilization jump

Figure 5. Power characteristics (2 month data)

brevity, we present the trace analysis results of 3 racks (Rack 13 from C1, and Racks 16, 19 from C2) from 2 clusters. For each rack, we use the 95-th percentile of aggregate utilization over 2 consecutive months as the power cap.

Figure 4 illustrates the correlations across 180 servers from different racks and clusters using their raw utilization 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 i-th and the j-th server. 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. This usually happens because the servers in the same cluster hosts similar workloads leading to synchronous power characteristics [16]. Figure 4(b) shows that in 80% cases when the rack level aggregate utilization is over the cap, the numbers of servers (among 60 servers per rack) that are also over the cap are 43, 55, and 50 for Rack 16, 19, and 13, respectively. The strong *intra-cluster synchrony* in power surge suggests the feasibility of detecting a cluster-level power surge based on local server-level measurements. Figure 4(c) shows probabilities of different racks in 2 clusters to be at peak simultaneously. The entry indexed at $[i, j]$ in this 2D matrix is the probability that the i-th rack in cluster 1 and the j-th 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 upon detecting an event) significantly minimizes inter cluster interference caused by transmissions generated by the even-driven CapNet in different clusters.



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

(b) Total number (out of 60) of servers over cap

(c) Probability of simultaneous peak between two different clusters

Figure 4. Correlations among servers, racks, and clusters

The synchrony of the power peaks of many servers also implies that a simple event-driven design would not work well. Specifically, if each server tries to send its power reading to the controller when it locally exceeds the power cap, many servers would attempt to transmit simultaneously. A standard CSMA/CA protocol can suffer significant packet loss due to excessive contention and collisions. Figure 5 shows that the intervals between two consecutive peaks can range between few minutes to several hundred hours suggesting that TDMA (Time Division Multiple Access) based protocols do not fit well for our problem since they need to have a predefined communication schedule for all nodes.

We next explore the feasibility of predicting an upcoming power peak based on historical readings. We define *utilization jump* as the difference between the utilization that exceeds the cap and the preceding measurement that is below the cap. As Figure 5(b) shows that utilization jumps can vary between 0 to 80%, it is impractical to dynamically schedule the next round of data collection for power capping based on the current value of power consumption. This observation leads us to avoid a predictive protocol that proactively schedule data collection based on historical power readings.

The major implication of all the above analysis is that when some servers' power consumption exceed the cap, it indicates a potential power capping event. However, such policy may generate false positive result in cases (since there are cases when a server exceeds the cap, but the aggregate utilization is still under the cap). The next subsection explains how CapNet avoids all false positives and false negatives through a three-phase design.

4.2 Event-Driven Protocol

The event-driven protocol consists of 3 phases as illustrated in Figure 6: **detection**, **aggregation**, and **control**. Detection phase determines if some server in the rack is over the threshold (cap) and, if so, a server generates an alarm. There may be situations where some server is over the cap, but the aggregate utilization is still under the cap. Such false positive cases are corrected through the aggregation phase, where the controller determines the aggregate power consumption. The control phase is executed only if the controller gets

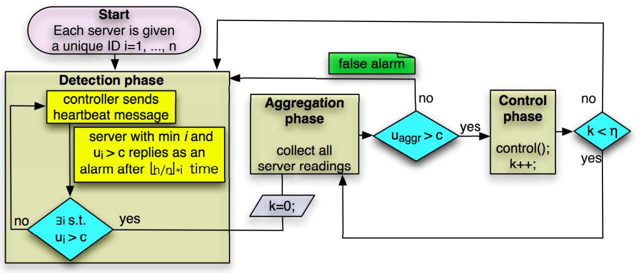

Start
Each server is given a unique ID i=1, ..., n
Detection phase
controller sends heartbeat message
server with min i and $u_i > c$ replies as an alarm after $\lfloor h/n \rfloor * i$ time
no
$\exists i$ s.t. $u_i > c$
yes
k=0;
Aggregation phase
collect all server readings
false alarm
no
$u_{aggr} > c$
yes
Control phase
control(); k++;
no
$k < \eta$
yes

Figure 6. CapNet's event-driven protocol flow diagram

confirmation that it was not a false alarm. The impact of a false positive case is that the system runs into the aggregation phase which incurs additional wireless traffic. No power capping control will be performed if the power capping manager determines the aggregate power consumption of the cluster has not exceeded the power cap after the aggregation phase.

Our event based protocol works as follows. Each server is assigned a unique ID i, where $i = 1, 2, \cdots, n$. The Controller broadcasts a *heartbeat* packet at every h time units called *detection interval*. The detection interval of length h is slotted among n slots, with each slot length being $\left\lfloor \frac{h}{n} \right\rfloor$. 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.

Detection phase. Each node i, $1 \leq i \leq n$, takes its sample at the i-th slot i.e. at time $\left\lfloor \frac{h}{n} \right\rfloor i$ in the detection phase. If its reading is over the cap, it generates an alarm and sends the reading to the controller as an acknowledgement of the heartbeat message. Otherwise, it ignores the heartbeat message, and does nothing. If no server sends an alarm in the detection phase (or the sent alarm is missed due to transmission failure), the controller resumes the next phase when the current phase is over. That is, at most one server can make the transmission in the detection phase which is also the alarm. If no servers are over the peak, there is no (alarm) transmission in the detection phase. If the controller receives any alarm, it generates a control message (repeated up to γ times for reliability that informs the servers that the detection phase is over. It then starts an aggregation phase and determines the aggregate power consumption, denoted by u_{aggr}, using the aggregation protocol presented as follows.

Aggregation phase. To minimize aggregation latency, CapNet adopts a sliding window based protocol to determine aggregate power. 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). Upon 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, 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.

Control phase. Upon finishing the aggregation phase, if $u_{aggr} > c$, where c is the cap, it starts the control phase. The control phase generates a capping control command using a control algorithm, and then the controller broadcasts the message. To handle broadcast failures, it repeats the broadcast γ times (since the broadcast is not acknowledged). If the control algorithm requires η-iteration, then after the capping control command is executed in the first round, the controller will again need to run the aggregation phase, and reconfirm that capping was done correctly. The procedure iterates up to $(\eta - 1)$ more iterations. Upon finishing the control, or after the aggregation phase that started upon a false alarm, it resumes the detection phase again.

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 (see Section 4.1). In absence of intra-cluster synchrony in power peaks, CapNet will not cause unnecessary power capping control or more wireless traffic than a periodic protocol.

4.3 Latency analysis

Given the latency bound corresponding to the trip time of a cluster's circuit breaker, it is important for CapNet to achieve bounded latency. In this section, we provide an analytical latency upper bound for CapNet. Note that the analysis provides safe but conservative latency bounds. 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_{aggr} 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 $(n \bmod \omega + \tau_d)$ time to accommodate the remaining $(n \bmod \omega)$ servers. Hence,

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

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

$$L_{aggr} \leq \Big((\tau_u\omega + \tau_d)\left\lfloor \frac{n}{\omega} \right\rfloor + (n \bmod \omega + \tau_d)\Big)\gamma \quad (1)$$

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}, and is given by

$$L_{det} \leq h \quad (2)$$

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. Let the operating system 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

$$L_{cap} \leq L_{det} + L_{aggr} + \tau_d\gamma + L_{os} + L_{hw}$$

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

$$L_{cap} \leq L_{det} + (L_{aggr} + \tau_c\gamma + L_{os} + L_{hw})\eta \quad (3)$$

5. Experiments

In this section, we present the experimental results of CapNet. The objective of the experiments is to evaluate the effectiveness and robustness of CapNet in meeting the real-time requirements of power capping under data center realistic settings. We first explain the implementation of CapNet, and then the experimental setups followed by the detailed results.

5.1 Implementation

The wireless communication side of CapNet is implemented in NesC on TinyOS [10] platform. To comply with realistic data center praxis, we have implemented the information processing and control management at the aggregation manager side. In our current implementation, wireless devices are plugged to the servers directly through their serial interface.

5.2 Experimental Setup

5.2.1 Experimental Methodology

We experiment with CapNet using TelosB motes for wireless communication. We use a total of 81 motes (1 for controller, 80 for servers). The motes communicate on behalf of the servers. 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. We experiment the network part of CapNet, and emulate the control part. We place all 80 motes in racks. The controller 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 controller. We use workload demand traces from an enterprise data center over a period of six consecutive months, which we explained in Section 4.1. Using these server readings 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 this traces at the same time stamp. While the data traces are of 6-month long, our experiment does not run for actual 6-month. 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.

5.2.2 Oversubscription and Trip Time

We use the trip times from Figure 3 as the basis, in order to determine the different caps required in various experiments. X-axis shows the ratio of current draw to the rated current, and Y-axis shows the corresponding trip time. 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 2 curves is the area where it is non-deterministic if the circuit breaker will trip. In our data traces, server utilization is proportional to current draw. The *magnitude of oversubscription* is defined as the ratio of aggregate utilization to the cap. Therefore, in the trip curve the values in X-axis indicate the oversubscription magnitude. LB trip time is a very conservative bound. In our experiments we use both LB and UB to verify the robustness of CapNet. The circuit breaker has three types of trip time behaviors that are shown as different regions below the tolerance band. Region 1 is the long-delay tripping zone with the oversubscription magnitude between 1 and 1.35, and trip time between minutes to hours. Region 2 is the conventional tripping zone with magnitude between 1.35 and 10. Region 3 is the instantaneous tripping zone ($ratio > 10$) that is designed to handle short-circuits.

5.2.3 CapNet Parameters

For all experiments, we use channel 26 and Tx power of -3 dBm. The payload size of each packet sent from the nodes that emulate servers is 8 bytes. This size of payload is enough since these nodes need to send only server's power consumption reading. The maximum payload size of each packet sent from the controller is 29 bytes which is the maximum default size in IEEE 802.15.4 radio stack for TelosB motes. This payload size is set large since those packets need to contain the schedules as well as 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 controller'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 controller 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 two wireless devices is in the range of several milliseconds. But in our design the controller node is connected through its serial interface to a PC. The TelosB's serial interface does not always incur a fixed communication between PC and the mote through serial. Upon experimenting and observing a wide variation of this time, we have set τ_d and τ_u to 25ms.

5.2.4 Control Emulation

In our experiments, we only emulate control. 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. Upon 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 (frequency 2.10GHz, 8 cores with hyper-threading), each running Windows Server 2008 R2 [15]. The ranges of OS level and hardware level latencies are in the range of 10-50ms and 100-300ms, respectively [15]. We generate OS and hardware level latencies in a single iteration using a uniform distribution in this range values taken by real physical servers.

5.3 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. 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 the corresponding message from the controller to the servers informing the suspension of current detection phase (when an aggregation phase is in order). Setting a larger value reduces the number of cycles of detection phase, but reduces the granularity of monitoring. In the results, ***slack*** is defined as the difference between the trip time and the total latency required for power capping. That is, a negative value of slack implies a deadlines[3] miss. An oversubscription is associated with both a lower bound (LB) and an upper bound (UB) trip time (Figure 3). Hence, 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.

5.3.1 Performance Comparison with Base Lines

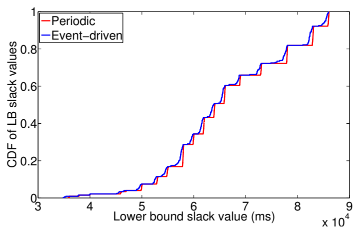
Figure 7 presents the results using 60 servers on one rack for single-iteration control loop. We used 4 weeks long data traces for this rack. We set the 95-th percentile of all aggregate utilization values of all data points in every 2-minute interval as its cap. In running the protocols using these traces, the protocols observe all peaks. The upper bound of L_{aggr} given in (1) was set as the period of the periodic protocol. Figure 7(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 oversubscription was above 1.5 for better resolution as the slack was too big for a smaller magnitude of oversubscription (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 using non-stop communications (i.e., the periodic protocol) does not benefit in practice.

While the slacks in event-driven protocol are shorter than those in the periodic protocol because the latter 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 7(b) in 90% cases among all power capping events the alarms were generated in the first slot (i.e., by the first server in the sequence used by the protocol in the detection phase) of the detection cycle. This result actually validates our trace analysis results and detection policy design in Section 4 that when the aggregate power is over the cap, a server has very high probability to generate an alarm. Therefore, the first server (i.e., the server that is assigned slot 1 of the detection cycles) in most cases generates the alarm. Only in 10% cases, the alarm was generated after the first slot of the detection phase. In 100% cases of the capping events the alarms were generated within the 6-th slot, although the phase had a total of 60 slots (for 60 servers, one slot per server). 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 3.

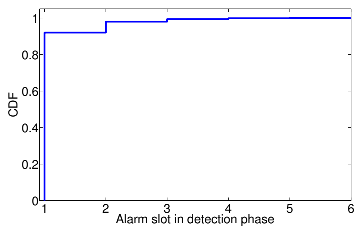
We also evaluate the performance when BoxMAC (a CSMA/CA based protocol) is used for power capping communication under default settings in TinyOS [10] for up to first 6 capping events in the data traces. Figure 7(c) shows that it experiences packet loss rate over 74% while performing communication for 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 comparison with CSMA/CA as it does not fit for power capping.

CapNet's event-driven protocol may generate false alarms. In this experiment, among the total alarms, only 45.56% alarms are false while the remaining 54.44% alarms were real power capping events. Note that this value is very small compared to the whole time window since power capping happens in at most 5% cases. Therefore alarms are also generated rarely. The biggest advantage is that in the rest of the times detection phases simply encounter no alarms, i.e., no server makes any transmission. Thus all those cycles go without any transmission from the servers. 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 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.

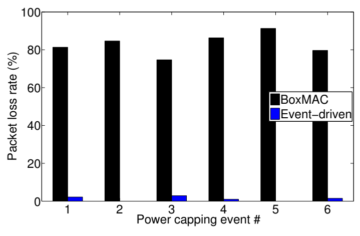
[3] The terms 'deadline' and circuit breaker 'trip time' are interchangeably used.



(a) CDF of lower bound slack



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



(c) Packet Loss Rate

Figure 7. Performance of Event-Driven protocol on 60 servers (4 weeks)

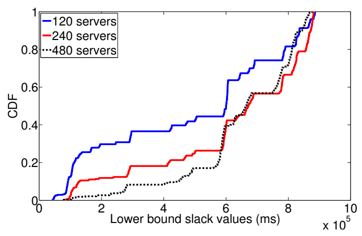


Figure 8. CDF of lower bound slack under event-driven protocol for various numbers of servers (4 weeks)

5.3.2 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 shown in Section 4). For sake of experimentation time, in all subsequent experiments we set cap at 98-th percentile (that would result in a smaller number of capping events). The lower bound slack distribution are shown in Figure 8 for 120, 240, and 480 servers by merging 2, 4, and 8 racks, respectively (for single iteration capping). The results show that for single iteration the deadlines are easily met for even 480 servers (since in each setup, 100% of all slack values are positive). Hence we move to more complicated cases, and in the next experiments we increase the number of iterations.

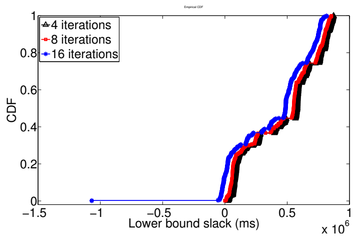
5.3.3 Scalability in Number of Control Iterations

Now we consider a complicated case where multiple iterations of control loop are required to settle to a sustained power level [15, 26, 35]. The number of iterations required for the rack-level loop as experimented in [35] can be up to 16 in the worst case. Hence, we now conduct experiments considering multiple numbers of control iterations (up to 16 as the worst case consideration). We plot the results in Figure 9 for various numbers of servers under various number of iterations. As shown in Figure 9(a), for 120 servers under 16-iteration case, we have 13% cases with negative slack meaning that the LB trip times were missed. However, the UB trip times were easily met. Note that we have considered a quite pessimistic set up here because using 16-iteration 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 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-iteration, 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 (Figures 9(b), 9(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-iteration and 13.94% cases under 16-iteration.

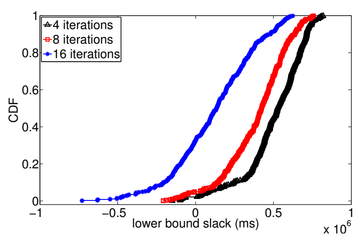
For all cases we met all upper bound trip times, while the lower bound trip times are pretty conservative. Again, 16-iteration is a very pessimistic consideration for power capping control. Hence, the above results show that, even for 480 servers in a cluster, the latencies incurred in CapNet for power capping remain within even the conservative latency requirements in most cases.

5.3.4 Experiments under Varying Caps

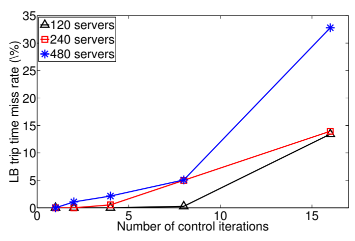
In all experiments we have performed so far, CapNet was able to meet UB trip times. Now we make some setup changes to encounter some scenario where UB trip times can be smaller, by making oversubscription magnitude higher. For this purpose, we now decrease the cap to decrease the trip times so as to make scenarios to miss upper bound trip times to see the robustness of the protocol. Now again we set the 95-th percentile of aggregate utilization 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 10 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. These results demonstrate the robustness for larger magnitude of oversubscription in that even when we use 16-iteration only 1.02% upper bound trip times are missed.



(a) LB slack for 120 servers

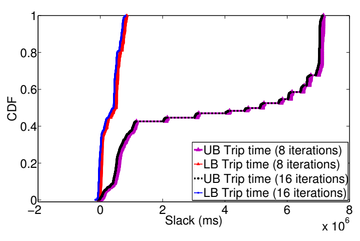


(b) LB slack for 480 servers

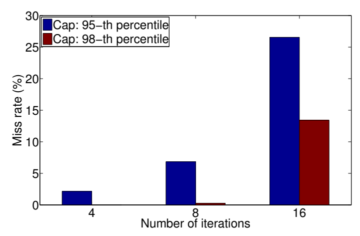


(c) Miss rate (LB trip time)

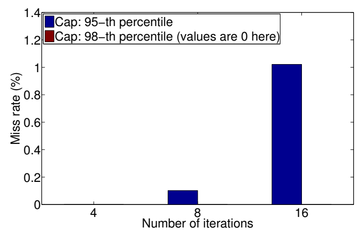
Figure 9. Multi-iteration capping under event-driven protocol (4 weeks)



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

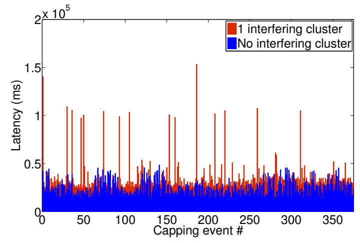


(b) LB trip time miss rate

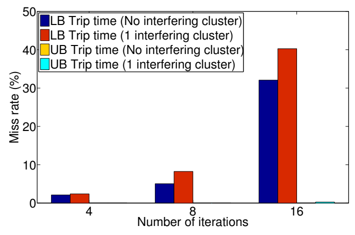


(c) UB trip time miss rate

Figure 10. Capping under different caps on 120 servers (4 weeks)



(a) Capping latency



(b) Miss rate

Figure 11. Capping for 480 servers under interfering cluster

5.3.5 Experiments in Presence of Multiple Clusters

We have already shown through data center trace analysis in Figure 4(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 the event-driven protocol under an interfering cluster.

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 another inside the rack. We set their transmission power at maximum (0dBm). The mote at the ToR represents its controller and carries on a pattern of communication like a real controller to control 480 servers. The mote inside the rack responds as if it were connected to each of 480 servers. Specifically, the controller executes a detection phase of $100 * 480$ms, and the receiver node in the rack randomly selects a slot between 1 and 480. On the selected slot, it generates an alarm with probability 5% since capping happens in no more than 5% cases. Whenever the controller 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 weeks data traces, and plot the results in Figure 11. Figure 11(a) shows the latencies for different capping events in 4 weeks 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% cases the increase was less than 15.089s. We noticed that such big increase happened because of the loss of detection message. At the beginning of capping if a detection message is lost, then the next cycle starts after 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 repeatations, 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-iteration, we missed the upper bound of trip time in 40.27% cases and lower bound of trip times in 32.08% cases. However 16-iteration is very pessimistic case. For 4 iteration miss rate was 5.06% and 8.26% only. And for 2-iteration 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. Discussions and Future Work

CapNet may face a number of engineering challenges in real world deployments. While our paper addresses feasibility, protocol design and implementation, challenges like security, EMI and fault tolerance needs to be addressed.

Fault Tolerance. One important challenge is handling the failure of power capping manager in a cluster. To address this challenge, power capping managers can be connected among themselves either through a different band or through a wired backbone. As a result, when some controller node fails, a nearby controller can take over its servers. This paper focuses on communication within a single cluster. DCM fault detection, isolation, and node migration need to be studied in future work.

Security. Another challenge is the security of the management system itself. Since the system relies on wireless control, it is theoretically possible that someone might be able to maliciously tap into the wireless network and take control of the data center. 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 which would be known to the server machines. Second, it is possible to encrypt wireless messages, for example, using MoteAODV (+AES) [13].

EMI & 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. [7]). Previous work has also used WiFi and ZigBee in live data centers for monitoring purposes [25].

7. Related Work

In order to reduce the capital spending on data centers, enterprise data centers use an over-subscription approach as studied in [18, 21, 26, 28], which is similar to over-booking in airline reservations. Server vendors and data center solutions providers have started to offer power capping solutions [5, 6]. Power capping using online self-tuning methodology [32] and feedback control algorithms [36] has been studied for individual servers. In contrast, the study of this paper concentrates to coordinated power capping which is more desirable in data centers as it allows servers to exploit power left unused by other servers. While such power capping has been studied before [20, 24, 26, 29, 34, 35, 38], all existing solutions rely on wired network for controller-server communication. In contrast, we focus on wireless networking for power capping. We have outlined the advantages of wireless management in Section 2.

Previous work on using wireless network in data centers exists on applications to high bandwidth (e.g. with 60GHz radio) production data network [39]. In contrast, CapNet is targeted at data management functions that have much lower bandwidth requirement while demanding real-time communication through racks. RACNet [25] 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. Our solution enables active control and requires communication through racks and server enclosures, and hence encounters fundamentally different challenges. RACNet also does not have real-time features. In contrast, CapNet is designed to meet the real-time requirements in power capping and provides analytical latency bounds.

8. Conclusion

Power capping in data center is a time-critical management operation for data centers that commonly oversubscribe power infrastructure for cost savings. In this paper, we have designed CapNet, a low-cost, real-time wireless sensor network for data Centers and validated its feasibility for power capping. We deploy and evaluate CapNet in two data centers. Using 6-month long data traces of the real servers of a data center we emulate 480 servers in a cluster, and show that CapNet can meet the latency requirements of power capping. CapNet represents a promising step towards applying wireless sensor networks to time-critical, close-loop control in DCM. 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 [14]. Since White Spaces are available everywhere specially in indoor environments [37] and have transmission range of several kilometers, we shall be able to cover all power capping managers within a single-hop network for any existing data center. We believe that there is significant potential in introducing wireless techniques to cover critical functions in data centers hosting future online and cloud services.

References

[1] . http://www.cdwg.com/shop/search/Cables/Networking-Cables/Category-5-TP-Cables-Ethernet/result.aspx?w=B25&pCurrent=1&ctlgfilter=&key=rj45+cable&searchscope=All&sr=1&x=0&y=0#Brand.

[2] . http://www.cdwg.com/shop/search/Networking-Products/Ethernet-Switches/Fixed-Managed-Switches/result.aspx?w=N11&MaxRecords=25&SortBy=TopSellers.

[3] . http://www.cdwg.com/shop/products/Digi-Passport-48-console-server/1317701.aspx.

[4] . http://www.cdwg.com/shop/search/Servers-Server-Management/Servers/x86-Based-Servers/result.aspx?w=S62&pCurrent=1&p=200008&a1520=002200.

[5] HP power capping and HP dynamic power capping for ProLiant servers. Technology brief, 2nd edition. http://h20000.www2.hp.com/bc/docs/support/SupportManual/c01549455/c01549455.pdf.

[6] Intelligent power optimization for higher server density racks a baidu case study with intel intelligent power technology. http://www.intel.com/content/dam/doc/case-study/data-center-efficiency-xeon-baidu-case-study.pdf.

[7] http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en535967.

[8] http://www.digikey.com/us/en/techzone/wireless/resources/articles/comparing-low-power-wireless.html.

[9] Private communication with data center operators.

[10] TinyOS Community Forum. http://www.tinyos.net/.

[11] Rockwell Automation Inc Bulletin 1489 circuit breakers selection guide, 2010. http://literature.rockwellautomation.com/idc/groups/literature/documents/sg/1489-sg001_-en-p.pdf.

[12] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. In *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, SIGCOMM '08, pages 63–74, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-175-0. . URL http://doi.acm.org/10.1145/1402958.1402967.

[13] W. Backes and J. Cordasco. Moteaodv &#8211; an aodv implementation for tinyos 2.0. In *Proceedings of the 4th IFIP WG 11.2 international conference on Information Security Theory and Practices: security and Privacy of Pervasive Systems and Smart Devices*, WISTP '10, 2010.

[14] P. Bahl, R. Chandra, T. Moscibroda, R. Murty, and M. Welsh. White space networking with wi-fi like connectivity. In *SIGCOMM '09*.

[15] A. A. Bhattacharya, D. Culler, A. Kansal, S. Govindan, and S. Sankar. The need for speed and stability in data center power capping. In *IGCC '12*.

[16] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao, and F. Zhao. 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, 2008.

[17] J. Choi, S. Govindan, B. Urgaonkar, and A. Sivasubramaniam. Profiling, prediction, and capping of power consumption in consolidated environments. In *MASCOTS '08*.

[18] X. Fan, W.-D. Weber, and L. A. Barroso. Power provisioning for a warehouse-sized computer. In *ISCA '07*.

[19] X. Fan, W.-D. Weber, and L. A. Barroso. Power provisioning for a warehouse-sized computer. In *ISCA '07*, 2007.

[20] M. E. Femal and V. W. Freeh. Boosting data center performance through non-uniform power allocation. In *ICAC '05*.

[21] X. Fu, X. Wang, and C. Lefurgy. How much power oversubscription is safe and allowed in data centers? In *ICAC '11*.

[22] Hamilton. Cost of power in large-scale data centers, 2008. http://perspectives.mvdirona.com.

[23] M. Isard. Autopilot: automatic data center management. *Operating Systems Review*, 41:60–67, 2007. .

[24] V. Kontorinisy, L. E. Zhangy, B. Aksanliy, and J. Sampson. Managing distributed UPS energy for effective power capping in data centers. In *ISCA '12*.

[25] C.-J. M. Liang, J. Liu, L. Luo, A. Terzis, and F. Zhao. RACNet: a high-fidelity data center sensing network. In *SenSys '09*, 2009.

[26] H. Lim, A. Kansal, and J. Liu. Power budgeting for virtualized data centers. In *USENIXATC '11*.

[27] H. Lim, A. Kansal, and J. Liu. Power budgeting for virtualized data centers. In *In USENIX Annual Technical Conference (USENIX ATC '11)*, 2011.

[28] S. Pelley, D. Meisner, P. Zandevakili, T. F. Wenisch, and J. Underwood. Power routing: Dynamic power provisioning in the data center. In *ASPLOS '10*.

[29] R. Raghavendra, P. Ranganathan, V. Talwar, Z. Wang, and X. Zhu. No power struggles: coordinated multi-level power management for the data center. In *ASPLOS '08*.

[30] P. Ranganathan, P. Leech, D. Irwin, J. Chase, and H. Packard. Ensemble-level power management for dense blade servers. In *ISCA '06*.

[31] A. Saifullah, Y. Xu, C. Lu, and Y. Chen. Distributed channel allocation protocols for wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*.

[32] M. Saravana and S. Govidan. Using on-line power modeling for server power capping. In *Workshop on Energy-Efficient Design 2009*.

[33] K. Srinivasan and P. Levis. RSSI is under appreciated. In *EmNets '06*.

[34] X. Wang and Y. Wang. Coordinating power control and performance management for virtualized server clusters. *IEEE Transactions on Parallel and Distributed Systems*, 99, 2010.

[35] X. Wang, M. Chen, C. Lefurgy, and T. Keller. SHIP: A scalable hierarchical power control architecture for large-scale

data centers. *IEEE Transactions on Parallel and Distributed Systems*, 23, 2012.

[36] Z. Wang, C. McCarthy, X. Zhu, P. Ranganathan, and V. Talwar. Feedback control algorithm for power management of servers. In *ASPLOS '08*.

[37] X. Ying, J. Zhang, L. Yan, G. Zhang, M. Chen, and R. Chandra. Exploring indoor white spaces in metropolises. In *MobiCom '13*.

[38] Y. Zhang, Y. Wang, and X. Wang. Capping the electricity cost of cloud-scale data centers with impacts on power markets. In *HPDC '11*.

[39] X. Zhou, Z. Zhang, Y. Zhu, Y. Li, S. Kumar, A. Vahdat, B. Y. Zhao, and H. Zheng. 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, pages 443–454, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1419-0. . URL `http://doi.acm.org/10.1145/2342356.2342440`.