RACNet: Reliable ACquisition Network for High-Fidelity Data Center Sensing

Chieh-Jan Mike Liang[1], Jie Liu[2], Liqian Luo[2], Andreas Terzis[1]
[1]*Computer Science Dept., Johns Hopkins University, {cliang4, terzis}@cs.jhu.edu*
[2]*Microsoft Research, Redmond, WA, {liuj, liqian}@microsoft.com*

Abstract

RACNet is a monitoring sensor network to provide high-fidelity visibility, in terms of spatial and temporal resolution, of data center environmental conditions for energy efficiency improvement. RACNet overcomes the high node density and harsh RF environment challenges in data centers to achieve over 99% reliable data yield and short data collection latency. It does so through a novel network architecture that decouples data collection from the construction of the routing tree. This design, coupled with the use of different frequencies along neighboring data collection trees, enables RACNet to support large-scale, dense networks while maintaining perfect data reliability. Results from simulations, testbed, and real world deployments indicate that RACNet outperforms previous data collection systems, especially as node density increases.

1 Introduction

Data center energy consumption has attracted global attention due to the fast growth of the IT industry and increasing concerns about carbon footprints and global climate change. While advances in component design continue to decrease the compute servers' power consumption, the energy consumed by the hosting facilities cannot be overlooked. In a typical data center, only 30% to 60% of the total energy consumption is used to power its IT equipment, such as servers and networking devices. The rest is either lost during the power delivery and conversion process, or used by environmental control systems such as Computer Room Air Conditioning (CRAC) units, water chillers, and (de)humidifiers [1, 32]. A root cause for this low energy efficiency is the lack of visibility in the data center's operating conditions. Specifically, as conventional wisdom dictates that IT equipment need abundant cooling to operate reliably, the CRAC systems in many data centers use very low setpoints, to reduce the danger of creating any potential hotspots. Furthermore, when servers issue thermal alarms, data center operators have limited means to diagnose the problem and to make informed decisions. Thus, they tend to further decrease the CRAC's temperature settings.

It should be evident, based on this example, that historical and up-to-date information about the fine-grain environmental conditions inside a data center is invaluable to operators. They can be used to troubleshoot thermal alarms, make intelligent decisions on rack layout and server deployments, and innovate on facility management. More importantly, such data can be particularly useful as data centers start to employ sophisticated cooling controls to accommodate environmental and workload changes [23]. For example, air-side economizers bring in outside air for cooling, while dynamic server provisioning strategies turn on or shut down a large number of servers following load fluctuations [4]. While both techniques can reduce a data center's total energy consumption, the variations in the resulting spatial and temporal heat distributions can also cause thermal instability, leading to server shutdowns and catastrophic failures.

Traditional solutions for dense (i.e., rack-level) environmental monitoring use wired sensors, connected through 1-wire [25], or Ethernet interfaces [23]. However, those approaches suffer from high installation and configuration costs. Furthermore, sensor locations are constrained by the availability of network connections. The alternative of using the motherboards' temperature sensors to estimate data center environmental conditions is also challenging, because these sensors are affected by the servers' CPU and disk activities.

On the other hand, wireless sensor network (WSN) technology offers many advantages for these monitoring and control tasks. It is low-cost, non-intrusive, can provide wide coverage, and can be easily re-purposed. Furthermore, WSNs require no additional network and facility infrastructure, simplifying their deployment in the already complex data center IT environment.

At the same time, the data center monitoring application imposes severe constraints in terms of data delivery reliability and latency, which we will further elaborate in section 2. High data reliability has been a long standing challenge for wireless sensor networks. Past real world WSN deployments exhibited data yields of $20 - 60\%$ [9, 30, 34], which are unacceptable for data center control and scheduling purposes. The size and density of data center sensor networks also make current data gathering protocols such as CTP [8] and Koala [22] not suitable, as the results from Section 4 indicate.

This paper presents our design and implementation of *RACNet*, a large-scale sensor network for high-fidelity data center environmental monitoring. By high-fidelity, we expect RACNet to reliably deliver environmental data at rack-level spatial resolution and sub-minute-level time resolution.

RACNet uses the custom-made *Genomote* sensor node and utilizes a combination of wired and wireless communications to scale. While our previous work focused on the problem description and the hardware we built [17], this paper focuses on *rDCP*, the reliable data collection protocol that RACNet uses. To tackle the network density and scalability challenges, rDCP uses multiple wireless channels concurrently[1]. Although a Genomote can be on a single channel at any particular time, rDCP dynamically balances the number of nodes on each channel to adapt to link quality changes. Furthermore, to achieve reliable and real-time data collection, rDCP separates the tasks of tree generation, performed by a distributed protocol in the network, and data retrieval, initiated by the roots of those trees. This separation achieves low internode contention, even in dense networks.

Contrary to the common belief that WSN cannot maintain high data yield, results from our simulations and a 696 Genomote deployment at a production data center (including 174 wireless nodes), show that rDCP can achieve over 99% data reliability, while delivering over 90% of the data under the current soft real-time requirement of 30 seconds, in a challenging RF environment.

The technical contributions of this paper are on the design and implementation of rDCP, including:

- *Data center RF environment characterization* We quantify the distinct challenges that the data center environment brings to sensor networks, through measurements of RF signal strength, signal quality, packet loss rate, and communication range in actual data centers.
- *Distributed, multi-channel and multi-hop topology control.* We present an efficient and distributed protocol for creating and maintaining multi-hop bi-directional data collection trees leveraging multiple frequency channels. Membership in individual trees dynamically adapts to the number of gateways in the network as well as link qualities on each channel.
- *Coordinated data retrieval.* We show that using the gateways to coordinate the sequence of data downloads, significantly improves data yields in dense sensor networks that generate large number of measurements.

Although the RACNet design is driven by the concrete data center monitoring requirements, the resulting protocol and experiences should apply broadly to WSN applications in scientific, industrial, and environmental domains.

In the remainder of the paper, we first present the high-level requirements for data center monitoring and outline the challenges of using IEEE 802.15.4 wireless communications in these environments through a site survey in Section 2. Section 3 elaborates on the design of the RACNet reliable data collection system. We present our evaluation results in Section 4, while Section 5 outlines results from a production data center deployment of RACNet. Section 6 reviews related work and we conclude in Section 7 with a summary and discussion about future work.

2 System Design Challenges

A data center monitoring and control system requires a low cost data acquisition system that offers wide coverage and is easy to install and maintain, which dictates the use of wireless sensors. Furthermore, wireless sensor networks can be easily re-purposed and are not constrained by network availability or administrative domains. To achieve low cost and ease of maintenance, we chose a radio based on the IEEE 802.15.4 standard [10], rather than Bluetooth or WiFi. Although they use the same 2.4GHz ISM band, IEEE 802.15.4 radios consume less power, have simpler network stacks, and require fewer processing cycles, thereby reducing the overall hardware cost. On the other hand, 802.15.4 radios can support data rates of only up to 250 Kbps (effective data rates are usually much less due to MAC overhead and multi-hop transmissions) and their lower transmission power[2] can lead to higher bit error rates, especially in RF-challenging environments such as data centers (see Section 2.2). The combined constraints of low data rates and high loss rates underlie the technical challenges that RACNet must resolve.

2.1 Application Requirements

In order to support cooling control and dynamic workload distribution, RACNet must provide data yields of 95%, or higher. Furthermore, the sampling frequency depends on the rate of change of the underlying physical phenomena. In the case of temperature, we recorded temperature changes of 10^oC within 5 minutes in response to server and CRAC operations. Based on these

[1] The IEEE 802.15.4 standard defines 16 channels in the 2.4 GHz frequency range.

[2] The CC2420 802.15.4 radio we use, transmits at 0 dBm, or 1 mW [31].

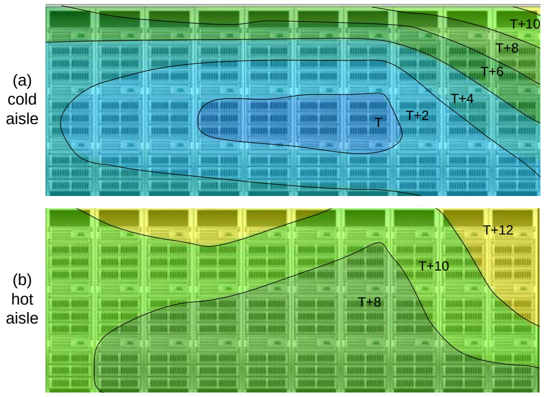


Figure 1: Temperature distribution over the front (cold aisle) and back (hot aisle) of a row of 10 racks. Significant spacial variation demands dense sensor networks.

observations, we use 30 second sampling rate in RACNet in order to promptly detect abnormal thermal conditions. The sampling rate can be even higher when troubleshooting hot spots or when sampling different sensors (e.g., monitoring the server's power consumption). Ideally, these measurements need to be collected before the next samples are generated. When this is not achievable, we still want to collect the data for archiving and long term decision making purpose.

The challenge of reliable and prompt data delivery is exacerbated by RACNet's expected scale and density. For a sense of this scale, consider that a data center can have several adjacent server colocation rooms (*colos* for short). Each colo can have several hundred server racks. Moreover, because there is significant spacial variations in air temperature (as much as 5^oC over 10 feet), illustrated in Figure 1, we need at least 12 sensors to be deployed over each side (air intake or exhaust) of a row of 10 racks to create a heat map of acceptable granularity [18]. This implies that thousands of sensing points are required within a colo.

Some of these challenges are partially addressed by RACNet's hardware design. Figure 2 presents a pair of Genomotes, which are sensor devices we specifically designed for RACNet [18]. One wireless master node (left) and several wired sensors (right) form a daisy chain to cover one side of a rack and collect data at different heights. This design increases sensing coverage and reduces the number of contending radios in the same space, without sacrificing deployment flexibility. However, even with the chain design, there are easily several hundred wireless master nodes in a colo.

The master node also has a flash memory chip that caches data locally to mitigate temporary connectivity variations. The whole chain is powered by a USB port connected to a server or a wall charger. Using a USB connection to power the whole mote chain means that unlike many previous sensor networks, power availability is not a critical concern in RACNet. On the other hand, using that USB port to carry measurements is not an option because it requires the installation of additional software on the servers, something that is not administratively possible in our environment. For the rest of the paper, we only consider the network among the wireless master nodes, treating the whole chain as a single node with multiple sensors.

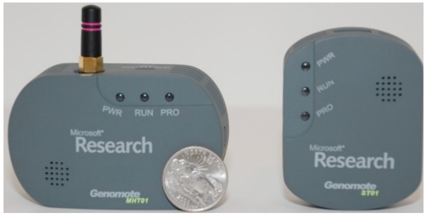


Figure 2: Two types of Genomotes designed for RACNet. The wireless node (on the left) controls several wired nodes (on the right) to reduce the number of wireless sensors within the same broadcast domain.

2.2 Data Center RF Environment Survey

Data centers present a radio environment that is different from the ones that past sensor network deployments faced. This is intuitively true as metals are the dominant materials in a data center. In addition to switches, servers, racks, and cables, other metallic obstacles include the cooling ducts, power distribution systems, and cable rails. Given the departure from RF environments studied in the past (e.g., [26, 38]), characterizing the data center environment is crucial to understanding the challenges it poses to reliable data collection protocols.

For this reason we performed a site survey by distributing 32 wireless Genomotes in a 12,000 square-foot colo at a production data center. The motes were placed at the top of the racks, following a regular grid pattern, with adjacent nodes approximately 35 feet from each other. During the experiment, nodes took turns broadcasting 200 128-byte packets with an inter-packet interval of 50 msec. Upon the successful reception of each packet, receivers logged the Receive Signal Strength Indicator (RSSI), the Link Quality Indicator (LQI), and a sequence number.

Our findings from this survey are summarized below:

Neighborhood Size. The average inbound and outbound node degrees in the network were 26. In other words, 82% of the nodes are within a node's communication range, which potentially leads to high interference among nodes. The number of neighboring nodes is likely to increase significantly as production deployments consist of a couple hundreds of wireless nodes in the same space. Thereby, data collection protocols must coordinate node transmissions to avoid packets losses due to

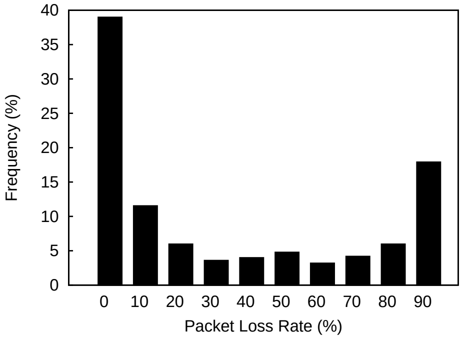


Figure 3: Average link packet loss rate from a 32-node data center site survey.

interference.

Packet Loss Rate. Figure 3 illustrates the distribution of link packet loss rates, which indicates that while the majority of the network links have low loss rate (i.e., $< 10\%$), there is still a significant percentage of links experiencing high number of losses. This further suggests that in order to build low loss end-to-end paths, routing protocols need to estimate the packet delivery ratios of available links and select the low loss ones.

Link Quality. Both RSSI and LQI have been used to estimate link quality [27, 33]. RSSI is an estimate of the signal power for received packets, while LQI can be viewed as the chip error rate over the packet's first 8 bits (802.15.4 radios use a Direct Sequence Spread Spectrum encoding scheme). Indeed, the results shown in Figure 4 indicate that there is a correlation between RSSI/LQI and packet reception rates. Based on these results, rDCP uses an RSSI threshold of -75 dBm as a first filter of potential weak links. Selecting this conservative threshold removes a large number of links. Fortunately, since each node has many neighbors some of which have low loss rates, the network remains connected.

3 Reliable Data Collection Protocol

We present rDCP, a *Reliable Data Collection Protocol* that dynamically constructs spanning trees rooted at one or more network gateways. These trees form the paths along which data from the nodes arrive at the gateways.

The architectural design of the protocol centers around the balance between distributed and centralized decisions, especially when performing routing and data retrieval operations. At one extreme, one can implement both inside the network. In this case, motes implement a distributed protocol that constructs a common routing tree and independently forward data as soon as they generate them [8, 35]. This approach adapts well to network and node failures, due to its distributed nature. On the other hand, the lack of coordination can lead to channel contention and eventually packet losses, especially towards the root of the tree. At the other extreme lies the centralized approach, in which the gateway controls the entire network operation [22, 28], leveraging its ample computational resources and knowledge of the complete network topology. Nodes simply report their local channel conditions to the gateways, which in turn determine the routing trees and request data downloads. This approach usually achieves high reliability and flexibility as gateways orchestrate the downloads. On the other hand, collecting neighborhood information scales with the number of network links, which for dense networks can grow with the square of the number of nodes.

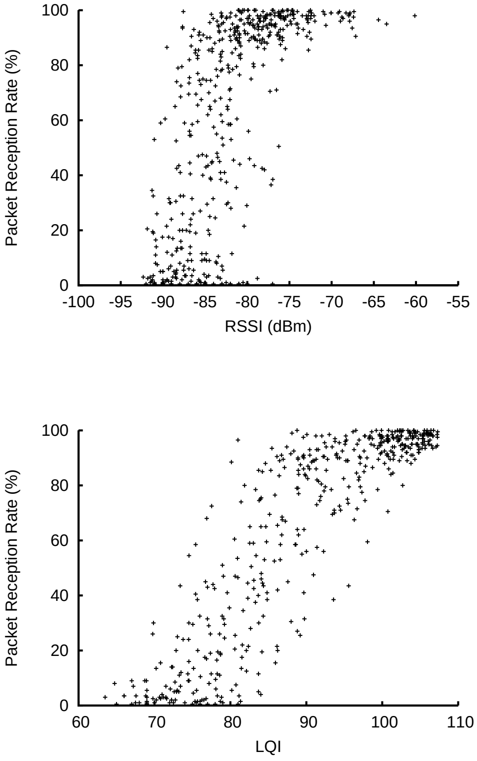


Figure 4: Packet reception rates of directional node pairs for different RSSI and LQI.

Based on these observations, rDCP employs a hybrid approach in which nodes cooperatively determine the routing topology while the gateways initiate data downloads from individual nodes. This division of responsibilities ensures timely and reliable delivery of data, while reducing contention during data downloads.

Pictorially shown in Figure 5, rDCP has two layers: the topology control layer (TCL), which builds bidirectional collection tree in the network and the data

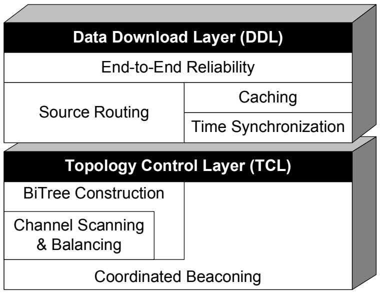


Figure 5: Architecture of rDCP, including the mechanisms to generate and maintain the tree topology and the mechanisms to reliably download data from individual Genomotes.

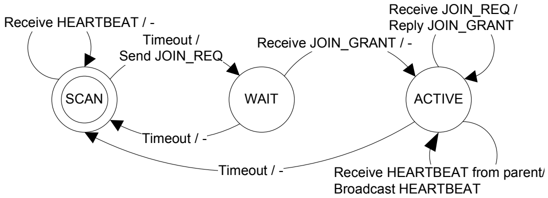


Figure 6: State transition diagram of a sensor node. State transitions are marked using the condition/action notation in which a transition occurs when a condition is met and results in an action (or no action in case of "-").

download layer, which reliably retrieves data from individual nodes. Next, we elaborate on each of rDCP's components.

3.1 Topology Control

Topology control maintains robust data collection trees rooted at the network's gateways. The mechanism's distributed nature allows nodes to independently react to network changes, including degraded link qualities and node failures. Although there are multiple examples of tree-building algorithms in the sensor networks literature (e.g., [8, 35]), most of them focus on delivering data in the upstream direction, whereas rDCP builds bidirectional trees (BiTrees).

Basic Protocol

Gateways initiate BiTree construction by broadcasting HEARTBEAT messages. Upon receiving a HEARTBEAT message, non-tree nodes compete to join the tree. Once on the tree they generate their own HEARTBEAT messages to recruit more nodes. HEARTBEATs include fields that represent the node's status, including its hop distance from the root, its parent node ID, and a children list. This children list is used to coordinate the transmission of HEARTBEAT messages as explained below.

Figure 6 represents a node's state transition diagram. A non-tree node stays initially in the SCAN state and actively listens for HEARTBEAT messages from tree nodes. The node then selects a parent based on the following process: first, the node makes sure that the incoming HEARTBEAT has a Receive Signal Strength Indicator (RSSI) above a threshold. Section 2.2 shows the relationship between the link packet delivery rate and RSSI. Then, the node checks the children list and determines whether the potential parent has already reached its maximum number of children. If not, it evaluates the *path* quality to the gateway via this upstream node by computing the *expected total transmission count* (ETTC) as follows:

$$ETTC_j = \sum_{l \in P} \frac{1}{ELDR_l} = ETTC_i + \frac{1}{ELDR_{i,j}}$$

where j is the current node; i is its potential parent; P is the path from j to the gateway via i, and $ELDR_l$ is the estimated link delivery ratio of link l. To compute $ETTC_j$ recursively, the $ETTC_i$ is included in the HEARTBEAT message. However, estimating $ELDR_{i,j}$ directly from HEARTBEATs would require multiple message rounds. To reduce control message overhead, we take advantage of the Link Quality Indicator (LQI) available from modern radio chips such as TI/Chipcon CC2420 [31]. We use a piece-wise linear approximation to estimate a link's ELDR based on its LQI, similar to the approximation used in [3].

The ETTC metric essentially represents the expected number of radio transmissions necessary to reliably deliver one message along a specific end-to-end path. We choose this metric to align with the system goal of reliable and timely data collection.

At the end of the SCAN state, the node selects the upstream node with the smallest ETTC as its *potential* parent and initiates a tree join request. The parent also estimates the link quality from this potential child in the upstream direction before replying with a JOIN_GRANT message. Otherwise, the tree join operation times out. This two-way handshake brings about two benefits. First, it serves as an explicit agreement between the parent and the child node that both have the resources to relay messages for each other. Second, since we require a BiTree for data downloading, it is important to ensure the link quality in both directions, as wireless links are sometimes asymmetric [38].

Note that the gateway sends periodic HEARTBEAT messages. A node that has just joined the tree waits for the second HEARTBEAT message from its parent, which triggers the node to broadcast its own HEARTBEAT message. The HEARTBEAT message is an explicit indication to a node's parent and children that the node is still alive. If a node stops hearing from its parent for too long (i.e., three times the HEARTBEAT interval), it assumes that the parent is unavailable and transits back to the SCAN state. Similarly, the parent abandons a child if the timer expires.

Coordinated Beaconing

As described above, nodes broadcast HEARTBEAT messages to construct and maintain the BiTree. It is therefore desirable to transmit multiple HEARTBEATs in a short amount of time, to accelerate the tree construction process. However, in large and dense networks, this can lead to broadcast storms and severe collisions, eventually affecting the quality and stability of the resulting tree. Thereby, efficiently broadcasting HEARTBEAT messages is a crucial requirement.

A simple and low-maintenance approach would be to adopt a contention-based approach, in which nodes contend for the radio medium. However, this approach is unsuitable for dense networks because the large number of HEARTBEATs would most likely cause collisions and unbounded delays. Instead, a TDMA-based protocol that assigns exclusive time slots to each node within the same interference range could be used. However, maintaining such TDMA schedules is cumbersome as it requires tight time synchronization and control traffic to set up the schedule.

For this reason, we propose a hybrid mechanism for broadcasting HEARTBEATs Specifically, a fixed-length time frame T is assigned to a node's children. As mentioned earlier, a HEARTBEAT message carries the node's complete list of its children. This list serves as a local TDMA schedule. The i^{th} child uses time slot $[\frac{T}{n_{max}}i, \frac{T}{n_{max}}(i+1))$, where n_{max} represents the maximum number of children that a node can have. The remaining time slots in T are used by non-tree nodes to initiate the two-way handshake.

Grand children start sending HEARTBEATs after T expires, thereby ensuring that nodes of different generations receive exclusive time slots. While this mechanism reduces contention, it does not guarantee a collision-free network. Specifically, we do not coordinate among nodes within the same broadcast domain that connect to different parents. Instead, we avoid collisions by having nodes randomly pick a time within their allocated time slots and let them contend for the radio.

Because tree nodes initiate their HEARTBEATs only after they hear from their parents, it is possible for the gateway to pace the generation of HEARTBEATs depending on the number of tree nodes.

3.2 Channel Diversity

A RACNet may consist of thousands of nodes within one data center. One way to increase data throughput, and thus reduce data latency, is by using multiple gateways at different locations. Furthermore, we take advantage of channel diversity to build multiple BiTrees rooted at different gateways, each on a different RF frequency. Previous work has shown that simultaneous communications over different 802.15.4 channels do not interfere with each other [36]. This section addresses the challenges of building multi-hop BiTrees over multiple channels in a distributed way.

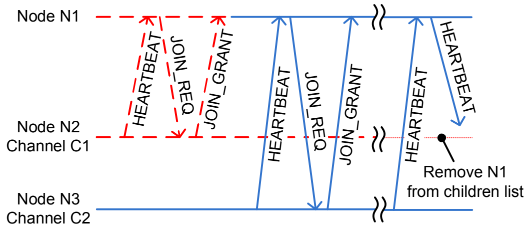


Figure 7: Tree construction with channel diversity. During the channel scanning phase, node $N1$ joined two trees. Finally it decided to stay at channel $C2$. Its old parent $N2$ at channel $C1$ eventually removed $N1$ from the children list.

Construction of Multiple BiTrees

In RACNet, every gateway has a fixed channel assigned by the operator. Non-gateway nodes start by scanning channels sequentially and looking for trees to join. Since gateways periodically initiate new rounds of HEARTBEAT messages, a node can bound its wait time on each channel to (little over) one HEARTBEAT time interval. A node joins the first tree using the two-way handshake mechanism described above. However for subsequent channels, the node joins the tree only if the estimated quality of the new path is better than the one on the current tree.

rDCP follows a transaction model when constructing BiTrees across different channels. As Figure 7 suggests, it is possible that a node (temporarily) joins multiple trees. However, nodes that actively scan channels do not broadcast HEARTBEAT messages to recruit children. This is to bound further disturbance in the candidate trees that the node decides not to join. When the scanning phase ends, the node switches to the last tree it joined. The candidate parents (other than the last one) eventually time out and remove the node from their children lists.

Nodes can significantly reduce their channel scanning time with the gateways' help. Specifically, gateways maintain the list of all channels they collectively occupy and include this information in their HEARTBEAT messages. Therefore, after receiving one HEARTBEAT message, nodes immediately know all available channels.

Balancing Multiple BiTrees

As nodes join and leave the network, the sizes of the network's BiTrees change. Since gateways coordinate data collections, the number of nodes does not significantly affect the download time from individual nodes. However, it implies that the sum of hops, or the total distance from each node to the gateway, can become unbalanced across channels. As we will show in Section 4, the sum of hops largely determines the overall time Δ to finish one round of data collection from all the nodes, thereby affecting application-level performance.

rDCP implements a distributed algorithm for balancing BiTrees. Once a gateway observes large differences across the Δ's of different trees, it initiates the channel-balancing process by sending a START_BAL message that propagates through the tree. To avoid network instability, rDCP utilizes two mechanisms: (i) it restricts the channel-balancing process to the gateway with the largest data collection delay, and (ii) it tolerates certain amount of imbalance in Δ. Let $\Delta_{avg} = \frac{\sum_{b \in B} \Delta_b}{|B|}$ be the average delay among all trees. A gateway b^* starts the channel-balancing process only if the following condition is met:

$$\Delta_{b^*} - \Delta_{avg} > \delta, \text{and}$$
$$b^* = argmax_{b \in B}(\Delta_b)$$

where B is the set of all gateways and δ is a threshold parameter that controls the amount of tolerable imbalance. The START_BAL message contains the probabilities for switching to each of the different channels. Switching probabilities are defined to be higher for more under-utilized channels.

Specifically, a node connected to the tree rooted at b^* will decide to switch to the tree rooted at gateway b_i with probability $P_i = \frac{\Delta_{avg}}{\Delta_{b^*}}$. The probability P_i to switch to the channel of basestation $B_i \neq b^*$ is calculated as follows:

$$P_i = 0, \text{ if } \Delta_i > \Delta_{avg}$$
$$P_i = \frac{\Delta_{b^*} - \Delta_{avg}}{\Delta_{b^*}} \cdot \frac{\Delta_{avg} - \Delta_i}{\sum\limits_{b \in B \text{ and } \Delta_b < \Delta_{avg}} (\Delta_{avg} - \Delta_b)},$$
$$\text{if } \Delta_i < \Delta_{avg}$$

Intuitively, we attempt to migrate the extra nodes at gateway b^* to underloaded gateways, based on their degrees of under-utilization. After receiving a START_BAL message, a node calculates the target channel based on the switching probabilities. More nodes will attempt to join the tree with fewer nodes. Finally, if the node can not find a parent in the target channel, it returns to its original channel.

3.3 Data Download

This section discusses the Data Download Layer, which reliably collects data to RACNet gateways along the BiTrees. Rather than having nodes initiate data upload asynchronously, which as we later show causes severe collisions and hence low data reliability in dense networks, rDCP uses a centralized pull-based approach. Gateways in this model sequentially poll each node in the constructed tree to download the measurements the node has collected since the last download. Note that downloading data from a single network path at any point in time is not inefficient for data center networks, due to their dense nature that limits the benefits of spatial reuse (cf. Sec.2.2).

Downstream Route Construction

As we mentioned earlier, BiTree nodes know only their parent and their children on the tree. However, in order for the gateway to iterate over all tree nodes, it needs to have an end-to-end route to every node. Gateways construct these downstream routes by merging all the children lists. List collection is a recursive process. A gateway first queries sequentially its own children. These nodes respond with their own children lists. The gateway then queries the nodes from the newly received children lists. This process continues until the gateway traverses all tree nodes. Since HEARTBEATs trigger BiTree topology changes, to minimize the overhead of route construction, gateways can cache the routes by reducing the HEARTBEAT frequency.

Self-Paced Data Streaming

After constructing downstream routes, the base station initiates data streaming by sequentially sending requests to each tree node. Each request carries a source route to the node the gateway wants to download data from and the range of data the gateway wants to download. The target node responds by streaming the requested data along the reverse path.

To stream efficiently, the source node must determine the inter-packet transmission interval that minimizes self-interference and end-to-end delay as these packets traverse the multi-hop path toward the gateway. rDCP adapts a technique proposed in [11] to estimate the required delay. A nodes estimates the inter-packet delay by overhearing the children list messages it sends to the gateway. Specifically, each node measures the time from the moment the message leaves until the last time it can overhear the same message from nodes upstream. To take into account the whole path, such local estimates are relayed downstream via the data request message. Each node updates its local inter-packet value to the maximum of the previous local value and the one carried in the message. The node also updated the inter-packet value before forwarding the message downstream.

Data Reliability and Integrity

To achieve end-to-end reliability, rDCP employs both link-level and end-to-end retransmissions. Modern radios, such as the TI/Chipcon CC2420, can automatically generate 802.15.4 acknowledgments for each successfully received packet. However, such hardware acknowledgments can cause false positives because the system can drop acknowledged packets before they reach the application. TinyOS 2.x, the software platform that rDCP is developed on, has a feature called software ACKs that allows instead the application to trigger 802.15.4 acknowledgments. Therefore, software ACKs achieve equivalent functionality but with higher confidence. In addition to using hop-by-hop software ACKs, rDCP also implements end-to-end negative acknowledgments. Specifically, after a node finishes streaming the requested data block, the gateway scans the received data stream for missing records and sends retransmission requests.

To ensure data integrity, rDCP performs Cyclic Redundancy Checks (CRC) at both the packet and the application level. We decided to add a CRC check on application payloads, because the 16-bit CRC used in 802.15.4 provides limited protection against corrupted packets. A data message is discarded if either CRC fails.

Data Time Stamping

RACNet relies on a large number of sensors to perform high fidelity data center sensing. To better correlate the measurements at different locations and generate heat maps, nodes must be synchronized and sample their sensors at the same time. rDCP synchronizes the nodes' clocks through a mechanism that adapts techniques proposed in the Flooding Time-Synchronization Protocol (FTSP) [21].

In more detail, rDCP assumes that the gateways maintain globally synchronized clocks. This is a reasonable assumption as protocols such as the Network Time Protocol (NTP) are readily available in the data center. Gateways timestamp each HEARTBEAT message with the current global time immediately before they transmit them. Upon receiving a HEARTBEAT message, a node creates a synchronization point (i.e., a pair of global and local timestamps). If clock frequencies and drifts were identical across nodes, a single synchronization point would suffice to translate local to global timestamp. However, [21] points out that this is not true in practice. Instead, rDCP takes multiple synchronization points and applies a linear interpolation to find the relation between the local and the global clock.

4 Protocol Design Evaluation

This section evaluates the design of rDCP using results from simulations and a prototype implementation in TinyOS 2.x [15], deployed to a lab testbed. We first present micro-benchmarks of the different rDCP components and then we compare rDCP to CTP [8], using data latency and reliability as two application-driven metrics.

All simulations use TOSSIM [16], a discrete event-based simulator for TinyOS. We selected TOSSIM because it uses the same application codebase as the one running on the motes, thereby reducing any discrepancies introduced from running different versions of the code. While TOSSIM can simulate arbitrary topologies, it requires the user to supply the signal attenuation levels for every network link. We use the Log Distance Path Loss model to calculate these attenuations [24]. Specifically, $\overline{PL}(d)[dB] = PL(d_0)[dB] + 10n\log(d/d_0)$, where d is the distance between transmitted and receiver, n is the path loss exponent, and $PL(d_0)$ is the attenuation at reference distance d_0. A path loss exponent of $n = 2$ corresponds to free space propagation, while $n = 3, 4$ correspond to environments with reflections and refractions. All of our simulations use n = 4.

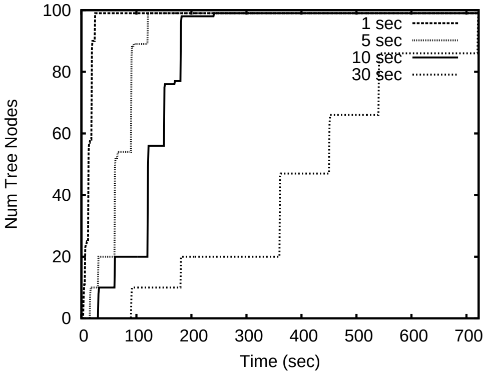


Figure 8: Tree settling time for a simulated 10×10 grid topology as a function of the HEARTBEAT interval.

While simulations allow us to vary experiment parameters, such as the network density, the lab testbed provides realistic radio channels. Furthermore, the heavily instrumented lab testbed allows the collection of latency and radio quality statistics through a wired back channel. The testbed network consists of 50 nodes, arranged in a grid topology that closely resembles the data center's physical layout. Nodes are divided into four columns to simulate the hot/cold server aisles and most nodes are within the same broadcast domain to simulate the data center RF environment (see Sec.2.2).

4.1 Micro-Benchmarks

Topology Maintenance

A node needs to join a BiTree before gateways can download data from it. Since nodes can dynamically join and leave BiTrees, especially during the network initialization and channel-balancing phases, the nodes' settling time, defined as the time necessary for a node to select its stable parent, affects data latency. In turn, the settling time depends on the HEARTBEAT frequency and the local TDMA time slot size T. We vary these two parameters in two simulated networks deployed over a $100ft \times 100ft$ grid.

Figure 8 presents the tree settling time, defined as the time until all the nodes join the tree, as a function of the HEARTBEAT beaconing intervals. The stair-like patterns correspond to nodes at different tree depths joining the tree. The figure also validates the intuition that

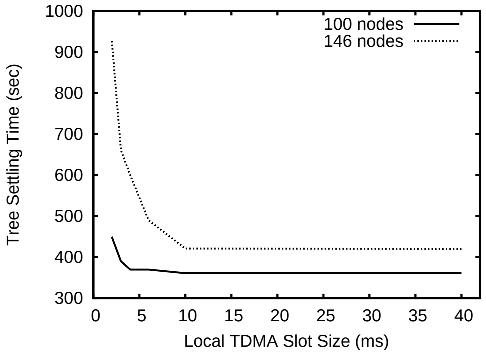


Figure 9: Tree settling time for two simulated grid topologies as a function of the local TDMA time frames.

increasing the HEARTBEAT frequency reduces the tree settling time.

Figure 9 presents the settling time as a function of the TDMA slot size. Since the slot size relates to the HEARTBEAT beaconing back-off time, the smaller it is, the shorter tree nodes wait on average before broadcasting their HEARTBEAT messages. However, because local TDMA schedules do not guarantee that transmissions from nodes in two adjacent tree branches do not collide, a smaller slot size increases the probability of collisions. Figure 9 validates this intuition, showing that the tree settling time starts to degrade when $T < 10$ msec. This effect is more visible in dense networks. On the other hand, increasing T does not necessarily increase the tree settling time because non-tree nodes try to survey different potential parents before committing to one. In other words, as long as the HEARTBEAT message arrives before a node's scanning timer expires (see Fig.6), the tree settling time will not be affected.

While Figures 8 and 9 show the time necessary to construct the whole tree from scratch, RACNet will incur this overhead only infrequently. Instead, once the network stabilizes, the time for a new node to join a BiTree is $N_c \times T_s$, where N_c is equal to the number of gateway channels and T_s is the time necessary to scan a single channel. The same formula provides the time necessary for a node to switch to a new parent after the existing parent becomes unfavorable or leaves the network.

Data Collection

Gateways sequentially collect data from every node in their BiTrees, so the worst case data latency is equal to the time necessary to finish one data collection round.

To estimate the duration of the data collection round, we ran rDCP on the lab testbed, while varying the number of nodes from 10 to 50. Each node sampled its sensors and generated 32-byte log entries once every 30 seconds. Figure 10 illustrates the network size, defined as sum of all tree hops, and the average data collection time for a single data collection round across the whole network. Because the average collection time does not exceed the sampling interval of 30 seconds, the gateway is able to keep up with the sampling rate. The other observation from Figure 10 is that the collection time closely follows the sum of tree hops. This result validates our decision to use the sum of hops as the metric for balancing the different BiTrees in the network.

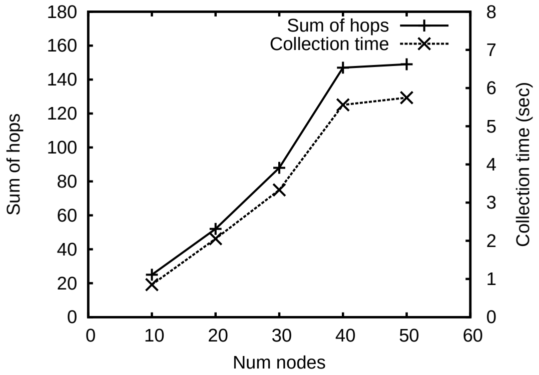


Figure 10: Average one-round collection time and the collection tree's sum of hops as a function of the testbed network size.

4.2 Application-Level Performance

Data center monitoring and control impose data yield and latency requirements on the data collection process. In this section, we evaluate how well rDCP meets these requirements. Furthermore, we use the Collection Tree Protocol (CTP) [8], the standard data collection protocol in TinyOS, as a comparison base line. To do so, we have implemented a sensing application on top of rDCP and CTP that periodically samples two on-board sensors. All comparisons use a single frequency channel because CTP does not have a channel balancing capability. In the case of rDCP we use at most three hop-by-hop and three end-to-end retransmissions, while for CTP we use the default number (30) of hop-by-hop retransmissions. Note that CTP does not support end-to-end retransmissions as it provides only upstream network paths towards the network's gateway(s).

We experiment with different network densities in TOSSIM and study data yield and latency on the lab testbed under different sampling frequencies.

Network Densities

Depending on the application requirements, nodes can be deployed under different network densities. To test

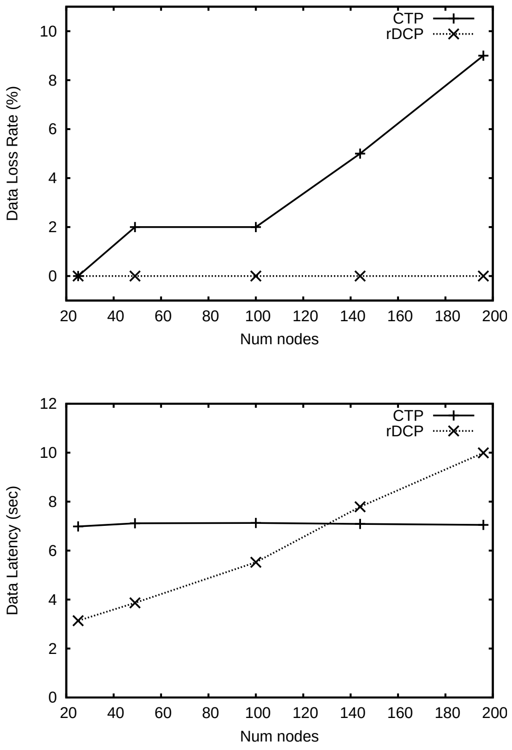


Figure 11: Loss rate and sensor data latency for simulated grids of $N \times N$ nodes deployed over an $100ft \times 100ft$ area. $N = 5, 7, 10, 12$, and 14, while the sampling interval is set to 30 seconds.

the effect of density on application performance we simulated networks of different size deployed over the same $100ft \times 100ft$ area. Specifically, nodes were arranged in a $N \times N$ grid, where $N = 5, 7, 10, 12$, and 14. Since TOSSIM does not simulate all aspects of the environment such as the processing delay, we do not intend to directly compare simulation and testbed results, but rather to observe growth trends.

It is evident from Figure 11 that rDCP is immune to increases in network density, while the loss rate for CTP increases as the network becomes more dense. On the other hand, the latency of the data that CTP delivers is lower than rDCP for smaller network densities. However, the latency of rDCP stays effectively constant (and equal to the data collection round), while CTP's latency deteriorates as network congestion levels increase.

Sampling Intervals

Different phenomena require different sampling intervals to capture changes in the state of the underlying environment. For example, server energy meters must be sampled more frequently than temperature sensors. Data collection protocols however should offer consistent performance across different sampling rates.

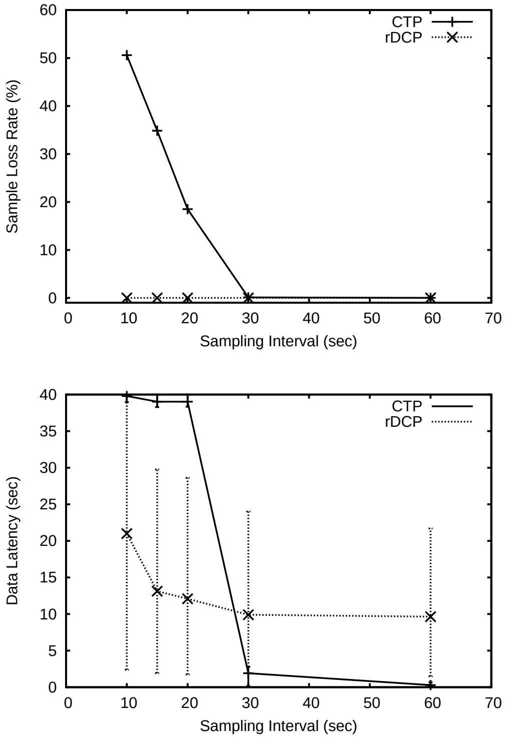


Figure 12: Data latency and loss rate on the 50-node lab testbed with different sampling intervals.

To evaluate how rDCP and CTP react to sampling rate changes, we ran one-hour experiments on the 50-node lab testbed, setting the sampling interval to 10, 15, 20, 30, and 60 seconds, and measured both data latency and yield. Figure 12 presents the behavior of CTP and rDCP as the sampling interval changes. Compared to CTP, rDCP maintains negligible data loss rates across all sampling intervals. Furthermore, although CTP offers lower latency when the network is not congested, latency and loss increase dramatically as network load increases. Finally, the higher data latency variation in the case of rDCP can be attributed to the decoupling of the times that data are collected by a node and retrieved by the gateway.

5 Data Center Deployment Results

To further evaluate rDCP under realistic conditions, we present results from a 100-node experimental deployment and then a 174-node production deployment in data centers. We present comparison results between rDCP and CTP and describe data yield and latency results from our production network.

5.1 Data Center Experiments

We further compare rDCP and CTP in a data center experimental deployment, using 100 Genomotes evenly distributed over a 12,000 sq-ft server room. Each mote

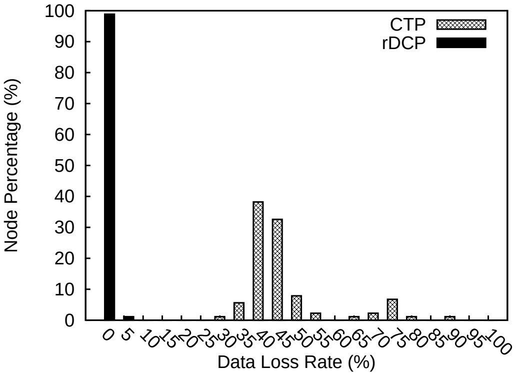


Figure 13: Distribution of data yield on the 100-node data center experimental deployment. Each node generates five samples every 30 seconds.

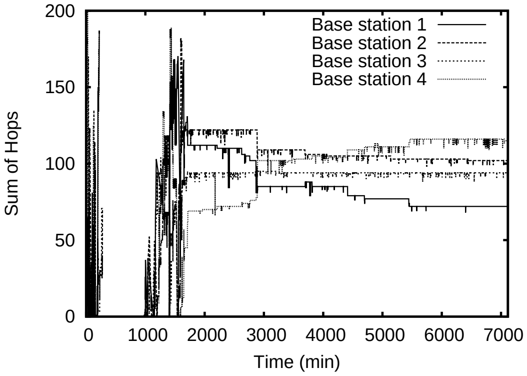


Figure 14: Sum of hops of four BiTrees as a function of time in the first four days of a production deployment at data center.

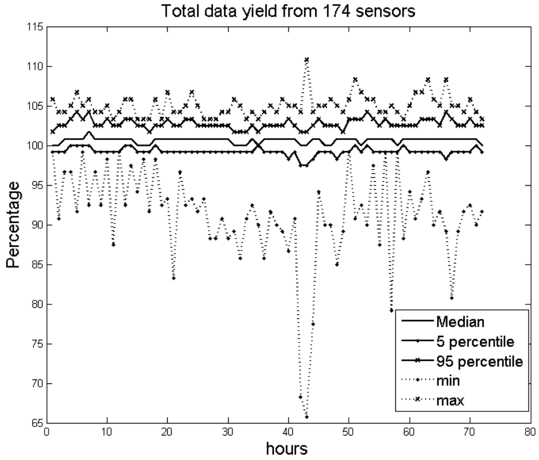


Figure 15: Total data yield over a 72-hour period from the 694 sensors in the production deployment.

generates five samples every 30 seconds to emulate the conditions where each wireless node is also responsible for relaying the measurements of the other nodes in its sensor chain.

Figure 13 presents the data yield distribution for rDCP and CTP. Specifically, rDCP achieves both higher data yields and lower variation across the network's nodes. On the other hand, CTP losses are not evenly distributed among nodes, complicating the analysis of the environmental results. There are several causes for the low data yields in CTP. First, nodes produce five packets bursts during each sampling period, temporarily congesting the network. Second, because the 802.15.4 radio is faster than the serial port, the gateway dropped packets due to overflows in its serial queue. rDCP solves both problems by coordinating node transmissions and introducing end-to-end flow and rate control.

5.2 Production Deployment Results

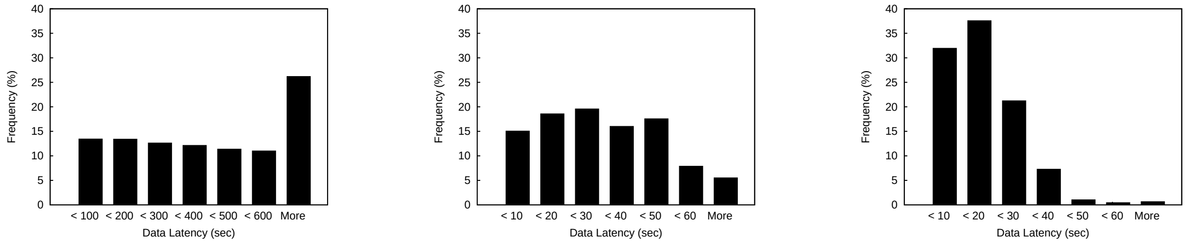
Results presented in this section are from a production RACNet deployment consisting of 694 Genomotes, including 174 wireless master nodes, in a 12,000 sq-ft colo. The network uses up to four wireless channels. The system has been running for more than 3 months, collecting more than 2.5 million measurement records per day, consisting of four sensor types: two temperature sensors, one humidity sensor, and the USB power status.

Channel balancing

Figure 14 illustrates rDCP's channel-balancing behavior, using the sum of hops metric during the first five days of the deployment. The early part of the figure ($t < 1,000$ min) shows significant fluctuations as the network is incrementally deployed and tested. The sudden drop in hop count is due to a controlled shutdown of all the gateways after all the nodes were deployed. The second part of the figure ($1,000 < t < 3,000$ min) corresponds to the phase during which the gateways balance the load across all four available channels. This phase ends when the difference between the expected load across all channels and the actual load on each channel is within 20% (cf. Sec.3.2). The last part of the figure ($t > 3,000$ min) shows that the network quickly reaches an equilibrium.

Total Data Yield

Figure 15 shows the per-sensor data yield distribution over a period of 72 hours in the production network. The percentage on the Y-axis is the ratio between the actual data received at the gateway during that hour and the expected amount of data, based on the sampling rate (i.e., 120 records per hour). The graph plots the min, 5^{th}-percentile, median, 95^{th}-percentile, and max data yields.



(a) Single wireless channel. Over 95% of the data exceed the 30 second deadline.

(b) Two wireless channels.

(c) Three wireless channels. Over 90% data are collected by the 30 second deadline.

Figure 16: Data collection delay distribution for production deployment.

One can see that more than 95% of the sensors have data yields of 98% or higher. Note that the $> 100\%$ data yields are an artifact of network time re-synchronization. Specifically, nodes occasionally get out of sync. When they subsequently re-synchronize to the global time, they may have to turn their local clocks back in time, resulting in a seemingly higher number of samples during that hour.

Data Collection Latency

To evaluate how the number of channels that rDCP uses affect data collection latency, we varied the number of frequency channels in the production network from one to three. We then computed the end-to-end latency as the difference between the time the data were timestamped by the node and the time they were inserted into the back-end database.

Figure 16 presents the distribution of data collection delay of 10,000 records after the network stabilizes. As the number of channels increase, the data collection latency drastically decreases from an average of 385 seconds when a single channel is used, to an average of 30 seconds for 2 channels, and an average of 16 seconds for three channels. Despite the difference in data collection latency, rDCP was able to maintain close to 100% data yield by caching sensor data on the external flash.

6 Related Work

Data Gathering Sensor Networks

Sensor networks have been used in several data gathering applications, including environmental [9, 34], habitat [19, 30], and structural monitoring [12, 37], just to name a few. However, most prior work focuses on outdoor deployments, in which sensors are sparsely deployed and power is the primary concern. On the other hand, RACNet, as a sensor network for a data center, has distinctly different trade-offs. First, power consumption is no longer a determining factor. Instead, performance issues such as delivery ratios and scalability are critical. Second, to monitor large data centers at fine spatial granularities, a large and dense network is necessary. In turn, this dramatic increase in scale leads to solutions that are qualitatively different from those employed in previous small-scale, sparse deployments.

Despite previous wired sensors for data center monitoring, LiveImaging from SynapSense is a similar wireless sensor network for measuring temperature, humidity, and air pressure inside data centers [29]. However, no quantitative results about the data yield of LiveImaging are publicly available. Furthermore, to the best of our knowledge, LiveImaging supports only five minute sampling intervals (i.e. ten times slower data rate) and does not support multiple frequency channels.

Data Collection Protocols

Data collection has been addressed at length in the sensor network literature. A large portion of the existing work focuses on the power aspect of the problem, aiming at minimizing energy consumption through data aggregation (e.g., [20]), ultra-low duty cycles (e.g., [2, 22]), or optimal sensor placement (e.g., [7]). In general, these systems are designed for low data rate applications with no delay requirements. On the other hand, rDCP faces new challenges from the large and dense network configuration and the stringent reliability requirements. The work closest to ours is the Collection Tree Protocol [8]. However, as the results in Section 4 suggest, CTP's best-effort nature prevents it from addressing the requirements of data center sensing.

A number of multi-channel protocols have been proposed to address the challenges associated with high densities in sensor networks. First, several general multi-channel MAC protocols [14, 39] assign nearby nodes to different channels to improve spatial reuse. The frequent channel switching required in such node-based channel assignment protocols usually entails large overhead. Considering the data collection traffic pattern in our application, we decide to adopt the more lightweight alternative: tree-based channel assignment. Instead of assigning different channels to individual nodes, we assign one channel to each spanning tree rooted at a sink node. Channel switches occur only occasionally when new sink nodes join the network.

Recent work from Le et al. [13] and Wu et al. [36], uses channel assignment strategies that are similar to ours. However, one relies on a centralized algorithm to assign channels [36], while the other achieves load balance among different trees based on a control theory approach [13]. Both mechanisms do not offer reliable data delivery. Comparatively, our approach is both distributed and reliable. Finally, Dust Networks Incincorporates a frequency-hopping protocol called Time Synchronized Mesh Protocol (TSMP) [6] for data gathering. TSMP nodes that belong to the same network are time-synchronized and share the same frequency-hopping sequence. Unfortunately, no results on the performance of TSMP are publicly available.

7 Conclusions and Future Work

The RACNet presented in this paper is among the first attempts to provide high-fidelity visibility into data center cooling behaviors, which is of increasing importance as cooling contributes to roughly half of total data center energy consumptions. This practical application challenges existing sensor network technologies in terms of reliability and scalability. The rDCP protocol tackles these challenges with three key ideas: channel diversity, bi-directional collection trees, and coordinated data downloading.

Comparing to existing data collection protocols such as CTP, rDCP is more flexible and scalable. When the network is congested, Genomotes store data locally for future retrieval. When more communication channels are available, the system adapts to it and the total throughput improves. As empirical results from a deployment of 174 sensors in a real data center suggest, rDCP achieves over 99% data yield for over 95% of the sensors, despite limiting factors including temporary disconnections and a harsh RF environment.

The combination of BiTrees and coordinated data retrieval mechanisms is powerful. Since the gateway dictates the data downloading order, it is now possible to achieve multi-resolution sensing or model-driven acquisition [5] over large networks. The gateway can prioritize which sensors to contact to create a coarse view of the environment first, and then selectively refine the views when resources allow. In data centers, since temperatures and humidities can be highly correlated over space and time, multi-resolution sensing may help reduce data redundancy and improve acquisition latency.

RACNet is a core component of the Data Center Genome project at Microsoft Research. Collecting cooling data is a first step towards understanding the conditions and resource usage in data centers. To reduce the total data center energy consumption without sacrificing user performance or device lifetime, we need a holistic understanding of key operation and performance parameters, such as power consumptions, device utilizations, network traffic, and application behaviors at fine granularities. With that knowledge, we will be able to close the loop between physical resources and application performance.

References

[1] Belady, C. L. In the data center, power and cooling costs more than the it equipment it supports. *ElectronicsCooling magazine 3*, 1 (February 2007).

[2] Burri, N., von Rickenbach, P., and Wattenhofer, R. Dozer: ultra-low power data gathering in sensor networks. In *Proceedings of the 6th IPSN Conference* (2007).

[3] Chen, B.-R., Muniswamy-Reddy, K.-K., and Welsh, M. Ad-hoc multicast routing on resource-limited sensor nodes. In *REALMAN '06* (2006).

[4] Chen, G., He, W., Liu, J., Nath, S., Rigas, L., Xiao, L., and Zhao, F. Energy-aware server provisioning and load dispatching for connection-intensive internet services. In *NSDI '08* (2008).

[5] Deshpande, A., Guestrin, C., Madden, S., Hellerstein, J., and Hong, W. Model-driven data acquisition in sensor networks. In *30th International Conference on Very Large Data Bases (VLDB 2004)* (Toronto, Canada, August 2004).

[6] Time Synchronized Mesh Protocol. Available at http://www.dustnetworks.com/docs/TSMP_Whitepaper.pdf, 2006.

[7] Ganesan, D., Cristescu, R., and Beferull-Lozano, B. Power-efficient sensor placement and transmission structure for data gathering under distortion constraints. In *IPSN '04* (2004).

[8] Gnawali, O., Fonseca, R., Jamieson, K., and Levis, P. Robust and efficient collection through control and data plane integration. Technical Report SING-08-02, 2008.

[9] Hartung, C., Han, R., Seielstad, C., and Holbrook, S. Firewxnet: a multi-tiered portable wireless system for monitoring weather conditions in wildland fire environments. In *MobiSys '06* (2006).

[10] IEEE Standard for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks. Specific requirements – Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs). Available at http://www.ieee802.org/15/pub/TG4.html, May 2003.

[11] Kim, S., Fonseca, R., Dutta, P., Tavakoli, A., Culler, D., Levis, P., Shenker, S., and Stoica, I. Flush: a reliable bulk transport protocol for multihop wireless networks. In *SenSys '07* (2007).

[12] Kim, S., Pakzad, S., Culler, D., Demmel, J., Fenves, G., Glaser, S., and Turon, M. Wireless sensor networks for structural health monitoring. In *SenSys '06* (2006).

[13] Le, H. K., Henriksson, D., and Abdelzaher, T. A control theory approach to throughput optimization in multi-channel collection sensor networks. In *IPSN '07* (2007).

[14] Le, H. K., Henriksson, D., and Abdelzaher, T. A practical multi-channel medium access control protocol for wireless sensor networks. In *IPSN '08* (2008).

[15] Levis, P., Gay, D., Handziski, V., Hauer, J.-H., Greenstein, B., Turon, M., Hui, J., Klues, K., Cory Sharp, R. S., Polastre, J., Buonadonna, P., Nachman, L., Tolle, G., Culler, D., and Wolisz, A. T2: A Second Generation OS For Embedded Sensor Networks. Tech. Rep. TKN-05-007, Telecommunication Networks Group, Technische Universitat Berlin, 2005.

[16] Levis, P., Lee, N., Welsh, M., and Culler, D. Tossim: accurate and scalable simulation of entire tinyos applications. In *SenSys '03* (2003).

[17] Liu, J., Priyantha, B., Zhao, F., Liang, C.-J. M., Wang, Q., and James, S. Towards fine-grained data center cooling monitoring using racnet. In *HotEmNets'08* (2008).

[18] Liu, J., Priyantha, B., Zhao, F., Liang, C.-J. M., Wang, Q., and James, S. Towards fine-grained data center cooling monitoring using racnet. In *HotEmNets'08: Proceedings of the 5th Workshop on Embedded Networked Sensors* (2008).

[19] Liu, T., Sadler, C. M., Zhang, P., and Martonosi, M. Implementing software on resource-constrained mobile sensors: experiences with impala and zebranet. In *MobiSys '04* (2004).

[20] Madden, S., Franklin, M. J., Hellerstein, J. M., and Hong, W. Tag: a tiny aggregation service for ad-hoc sensor networks. In *OSDI '02* (2002).

[21] Maróti, M., Kusy, B., Simon, G., and Lédeczi, A. The Flooding Time Synchronization Protocol. In *SenSys '04* (2004).

[22] Musaloiu-E., R., Liang, C.-J., and Terzis, A. Koala: Ultra-low power data retrieval in wireless sensor networks. In *IPSN '08* (2008).

[23] Patel, C. D., Bash, C. E., Sharma, R., Beitelmal, M., and Friedrich, R. Smart cooling of data centers. In *Proceedings of International Electronic Packaging Technical Conference and Exhibition* (Maui, Hawaii, June 2003).

[24] Rappaport, T. S. *Wireless Communications: Principles & Practices*. Prentice Hall, 1996.

[25] Smart Works. http://www.smart-works.com.

[26] Srinivasan, K., Dutta, P., Tavakoli, A., and Levis, P. Some implications of low power wireless to ip networking. In *Proceedings of the Fifth Workshop on Hot Topics in Networks (HotNets-V)* (Nov. 2006).

[27] Srinivasan, K., and Levis, P. RSSI is Under Appreciated. In *Proceedings of the 3^{rd} Workshop on Embedded Networked Sensors (EmNets)* (May 2006).

[28] Stathopoulos, T., Girod, L., Heidemann, J., and Estrin, D. Mote herding for tiered wireless sensor networks. Tech. Rep. CENS-TR-58, University of California, Los Angeles, Center for Embedded Networked Computing, December 2005.

[29] SynapSense Corporation. LiveImaging: Wireless Instrumentation Solutions. Available from: `http://www.synapsense.com/`, 2008.

[30] Szewczyk, R., Mainwaring, A., Polastre, J., Anderson, J., and Culler, D. An analysis of a large scale habitat monitoring application. In *SenSys '04* (2004).

[31] Texas Instruments. 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver. Available at `http://www.chipcon.com/files/CC2420_Data_Sheet_1_3.pdf`, 2006.

[32] The Green Grid. The green grid data center power efficiency metrics: PUE and DCiE. Available at `http://www.thegreengrid.org/gg_content/TGG_Data_Center_Power_Efficiency_Metrics_PUE_and_DCiE.pdf`, 2007.

[33] TinyOS. MultiHopLQI. Available from: `http://www.tinyos.net/tinyos-1.x/tos/lib/MultiHopLQI`, 2004.

[34] Werner-Allen, G., Lorincz, K., Johnson, J., Lees, J., and Welsh, M. Fidelity and yield in a volcano monitoring sensor network. In *OSDI '06* (2006).

[35] Woo, A., Tong, T., and Culler, D. Taming the underlying challenges of reliable multihop routing in sensor networks. In *SenSys '03* (2003).

[36] Wu, Y., Stankovic, J., He, T., and Lin, S. Realistic and efficient multi-channel communications in dense sensor networks. *INFOCOM 2008* (April 2008).

[37] Xu, N., Rangwala, S., Chintalapudi, K. K., Ganesan, D., Broad, A., Govindan, R., and Estrin, D. A wireless sensor network for structural monitoring. In *SenSys '04* (2004).

[38] Zhao, J., and Govindan, R. Understanding Packet Delivery Performance In Dense Wireless Sensor Networks. In *Proceedings of ACM Sensys* (Nov. 2003).

[39] Zhou, G., Huang, C., Yan, T., He, T., Stankovic, J. A., and Abdelzaher, T. F. Mmsn: Multi-frequency media access control for wireless sensor networks. *INFOCOM 2006* (April 2006).