2012 IEEE International Conference on Green Computing and Communications, Conference on Internet of Things, and Conference on Cyber, Physical and Social Computing

Improving Datacenter Operations Management using Wireless Sensor Networks

Panagiotis Garefalakis
Institute of Computer Science
Foundation for Research and Technology-Hellas
Heraklion GR-70013, Greece
pgaref@ics.forth.gr

Kostas Magoutis
Institute of Computer Science
Foundation for Research and Technology-Hellas
Heraklion GR-70013, Greece
magoutis@ics.forth.gr

Abstract— **Increasingly larger datacenters constructed to serve Internet-scale enterprise and Cloud computing workloads are creating significant challenges for the management systems designed to operate them. In this paper we present the design and implementation of a management system that uses server-attached wireless sensors to create an auto-configuring wireless-only monitoring network that can be used to address a number of challenges in the area of datacenter operations management. We exhibit the use of this wireless network in tracking the physical location of servers and in troubleshooting connectivity problems in the wired datacenter network infrastructure. Our system is implemented as an extension to the Nagios distributed monitoring and management system and demonstrated to provide continuous awareness of the physical location of the server infrastructure as well as insight into wired network partitions under switch failures.**

Keywords- ***Wireless sensor networks; Datacenter management***

I. INTRODUCTION

Datacenter management tools are critical for the administration of large, enterprise-scale datacenters. The high complexity of configuration management in such centers requires significant expertise by large teams of human administrators. Running a datacenter is fraught with significant difficulties that have to do with managing complex configurations, the provisioning of hardware and software, change management, etc. Several systems such as Autopilot [9], SmartFrog [8], and others have been proposed with the goal to tame the complexity of operating datacenters. However many data center management problems are still not fully solved, thus keeping the complexity and cost of running a datacenter high.

While the spectrum of existing challenges is very broad, in this paper we aim at providing a novel solution to two important such problems: First, to automatically determine the physical locations of servers over time and help administrators be aware of their positions as well as be notified of any changes. Second, automatically ascertaining the status of servers (whether they are alive or not) when transient failures in the wired network infrastructure prohibit the use of standard methods such as the exchange of heartbeat/ping messages. Our technical solution to both problems relies on the use of an auto-configuring wireless sensor network (WSN) based on the IEEE 802.15.4 (Zigbee) protocol that we integrated with a popular open-source distributed monitoring and management system.

The use of distributed monitoring and management systems such as Nagios [1] and Ganglia [13] offers the ability to monitor and collect information about a variety of services and is an essential component for reducing the complexity of IT operations management in datacenters. Distributed monitoring and management systems typically run periodic checks on user-specified datacenter resources and services. Resources that can be monitored include memory usage, disk usage, CPU load, and the number of currently running processes. In addition many such systems support user-specified extensions (add-ons or plugins as they are typically referred to) that can further simplify the configuration and management of large scale systems. A user-friendly Web-based graphical user interface is often provided. Systems such as Nagios and Ganglia are available in open source form and have thus become popular due to their extensibility and ease of use.

In this paper we propose extending the functionality of an open-source monitoring and management system (Nagios) by integrating it with an auto-configuring Wireless Sensor Network. Events such as temperature rise, airflow, energy consumption etc can be correlated with the physical location of the affected resources based on the signal strength of sensors (Received Signal Strength Indicator or RSSI) [6]. While positioning solutions using Satellite based global Position Systems (GPS) have been previously proposed, Zigbee devices offer a more viable positioning method using existing infrastructure, in closed spaces, and without a large impact on operating expenses. Taking full advantage of Nagios and Zigbee capabilities can give us critical information about server location and status, which could not be fully determined before. We have built an auto-configuration capability into our IEEE 802.15.4 Zigbee WSN to ensure a low-impact overall solution to datacenter operations management.

The remainder of our paper is structured as follows: Section II describes the design and the main components of the system. Section III introduces terminology and describes our implementation in more detail. Section IV discusses related work. Finally we present an evaluation of our prototype in Section V and our conclusions in Section VI.

978-0-7695-4865-4/12 $26.00 © 2012 IEEE
DOI 10.1109/GreenCom.2012.38

II. Design

We assume a typical datacenter architecture consisting of racks of server blades interconnected via a wired network infrastructure [4][5] as depicted in Figure 1**.** A key component of the management system presented in this paper is a wireless sensor network (WSN) interconnecting sensors (we use IEEE 802.15.4 Zigbees) physically mounted on each server. A second component of the system is a management server equipped with a wireless sensor and using a database repository for storing system configuration information. Our design allows the use of multiple management servers to achieve wide coverage of large datacenter floor spaces; for simplicity however, in this paper we focus on the use of a single management server. Our software architecture consists of an extended version of the Nagios management system (hosted on the management server) that controls a number of WSN agents hosted on all managed servers in the datacenter as shown in Figure 1.

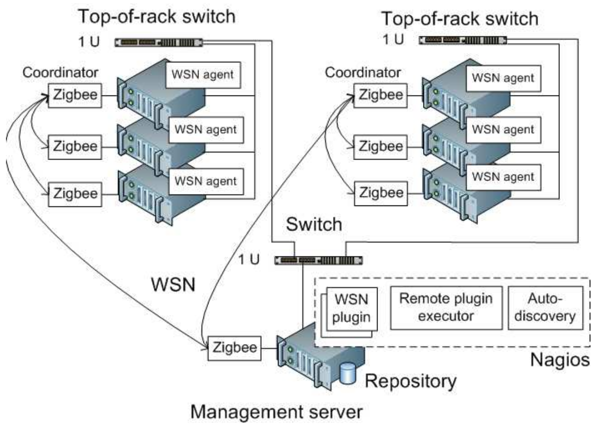


Figure 1: System architecture.

Nagios monitors datacenter resources (such as processes running on servers). Such resources are modeled as *services* that are associated with management information and a current status. Nagios monitors datacenter resources through the use of *plugins*. Each plugin contains expert information about how to gauge the status of a service and relies on script execution to mine the necessary information. Remote execution of scripts is supported via the *remote plug-in executor* (RPE). Nagios periodically invokes known plugins, which report on the status of datacenter resources. Whenever a service changes state, a handler can be triggered to notify an administrator or to take corrective action (such as restarting a non-responding service) via script execution. State changes and other information detected by plugins are also stored in log files.

Our management system effectively provides Nagios with a wireless communication path operating in parallel to the standard wired-network path that all standard Nagios plugins typically use. Our system consists of two main software components: The *WSN agent* and the *WSN plugin* that reside on managed systems and on the management server respectively (Figure 1), jointly referred to as the *WSN service*. The system goes through an initial phase of auto-configuring the WSN, periodically performing re-configuration actions to adapt to changes in the underlying infrastructure. Following the initial auto-configuration phase the system goes into a periodic schedule of data collection activities.

A. Auto-configuration phase

A WSN agent is deployed across each managed server with the responsibility to interface and control its locally-attached wireless sensor device. It is notified on the arrival of WSN control messages, which it processes taking specific actions. A straightforward method to ensure all servers contain the WSN agent is to deploy the agent at server boot time (or at provisioning time) along with the operating system (OS) image and application binaries. Systems such as Autopilot [9] and IBM Tivoli Provisioning Manager [11] support this method by storing OS images in a repository and deploying them on new servers on demand.

The next step in the auto-configuration process is to configure the wireless sensor attached to each server. This process is initiated by the Nagios management server (specifically, by the *WSN plugin* shown in Figure 1) by broadcasting an initialization message (called `REPORT RACKS`) aimed at all servers. The aim of this message is to discover all wireless devices along with the rack they are part of. We assume that a server can determine its rack automatically by using methods such as network subnetting (assuming each rack is assigned to a different IP subnet as is typically the case in datacenters), or by invoking special management interfaces. Upon receipt of a `REPORT RACKS` message, each wireless device obtains the rack identifier from its attached server and reports it via a unicast response message to the management server. The management server stores each wireless device's MAC address and rack ID to a local repository.

After collecting information about wireless devices across the entire system, the WSN plugin proceeds to assign devices within the same rack to a unique IEEE 802.15.4 personal-area network (PAN) and communication channel (Zigbee supports up to 65536 PANs with 16 channels each). The management server first chooses a specific wireless device for the role of coordinator within a rack and communicates that decision to it via a unicast message called `START COORDINATOR`. That message includes the addresses of the wireless devices in the same rack (referred to as *slave devices*) that will be controlled by that coordinator. The coordinator is responsible for creating the PAN and for inviting slave devices to join it. By grouping wireless sensors on a per-rack basis we avoid overdrawing the wireless bandwidth limit (up to 250Kbit/sec per

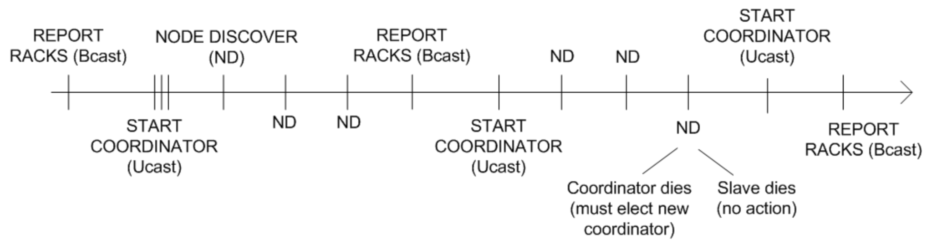

REPORT RACKS (Bcast)
NODE DISCOVER (ND)
REPORT RACKS (Bcast)
ND
ND
START COORDINATOR (Ucast)
START COORDINATOR (Ucast)
ND
ND
START COORDINATOR (Ucast)
ND
REPORT RACKS (Bcast)
Coordinator dies (must elect new coordinator)
Slave dies (no action)

Figure 2: Auto-configuration message sequence.

PAN/channel for Zigbee) and as a consequence extend the total number of sensors that can be used in a datacenter. Typical racks consist of around 20-40 servers, which is a sizeable amount. Moreover the per-rack grouping makes it easier to organize and collect management data.

After starting all coordinators, the management server periodically uses a wireless-device failure detector (such as the Zigbee Node Discover (ND) command or explicit heartbeats) to detect failures of either a coordinator or a slave device. In case of failure of a coordinator device, the management server elects a new coordinator in that rack picking one of the surviving slaves for that role. It then notifies that wireless device of its upgrade within the existing PAN of all wireless devices in its rack. If any node crashes due to a server reboot, it loses its state and thus tries to re-join the WSN when the server is back up again. The management server periodically re-broadcasts the REPORT RACKS message to discover newly appearing devices, as shown in Figure 2.

B. Wireless data collection

During the data collection phase the system uses unicast wireless-only communication between (i) the management server and the coordinator of each rack; and (ii) between the coordinator of each rack and its slave devices. Each coordinator is responsible for collecting management data from its subordinates and forwarding them wirelessly to the Nagios server when asked to do so.

The management server periodically (at configurable intervals) selects specific types of queries and sends them via the WSN plugin to coordinators across all racks. This function is similar to that of the standard Nagios RPE plugin except for operating over the wireless rather than the wired infrastructure. The coordinators in turn propagate command queries wirelessly to all slave sensor devices in their PAN, collect their responses, and return the aggregate information to the WSN plugin. The plugin processes the information storing status codes to the Nagios repository and log files.

Applications using the management system follow an event-driven style and are written as handlers invoked after a change in the status of a datacenter resource monitored by Nagios. We have so far experimented with two applications using the above mechanisms. The first application periodically tracks the physical location of wireless sensors, optionally notifying an administrator in case of change. A second application is a network partition detector implemented as a handler invoked when an IP connectivity probe service reports a failure (such as when a switch on the network path to a server fails). The handler correlates this information with the status of the WSN service on the same server. If the WSN service appears to be operating normally, the handler concludes that there could be a switch failure. If the WSN service appears to be down as well, the handler concludes that the server must be dead.

III. IMPLEMENTATION

We first provide a brief background on the communication modes offered by the Zigbee protocol. We will hereafter refer to the wireless devices as XBee modules. In general XBee modules exchange two types of messages: one used for querying and setting configuration parameters (known as AT commands) and another used for more general communication needs. XBee modules operate under three communication modes: transparent, application programming interface (API), and command mode (used only for setting configuration settings). Transparent mode is the simpler, more convenient way to communicate since the modules require no configuration. Transmission of a byte requires it to be sent to the local XBee module. After a certain timeout period the module packetizes any bytes in its buffer and sends them to all remote XBees within range. Bytes communicated in transparent mode may be lost since the link layer under that mode does not support reliable data transfer. The transparent mode supports broadcast communication only (no unicast messaging).

The API mode of communication (the one we are mainly using in our system) allows greater control of the link as well as the ability to send packets to an explicit address (unicast communication) in addition to unreliable broadcasts. Data formatting must be explicitly performed by the application. Received API-mode packets contain the source address of transmitting radio, a checksum for data integrity, and the RSSI (signal strength) value. Reliable data transfer is implemented at the link layer: when sending a unicast message to a specific host, the receiving radio will

REPORT RACKS request

REPORT RACKS response	MAC	RACK ID

START COORDINATOR	PAN ID	CHN ID	MAC addrs of slave devices

Table 1: Structure of auto-configuration messages (top-down): Report racks request, response, and start device as a coordinator.

send an acknowledgement (ACK) if the packet was successfully received. If the transmitting radio does not receive the ACK it retransmits the packet.

Our WSN-agent and WSN-plugin modules interact with their attached XBee radios using an open-source Java library implementing basic communication primitives in API mode [7]. The library provides functions for sending and receiving API-mode packets in broadcast and unicast mode and for configuring the locally-attached and (in the case of a coordinator) remote XBee modules. The library relies on another open-source component (RXTX) [15] to transfer packets to and from the locally-attached XBee module over a serial port (USB). To address security concerns over the use of wireless network infrastructure in monitoring business-critical systems we leverage standard Zigbee hardware support for 128-bit symmetric key encryption (AES) over the IEEE 802.15.4 protocol [16]. We minimize the complexity of key management by utilizing a single encryption key shared across all devices.

In our current implementation we assume the WSN agent is already deployed on all managed servers. If a wired network infrastructure is available however, we additionally support agent deployment via Nagios remote plugin execution (using a remotely executed script to pull the agent binary into the server). We can further simplify the deployment process by automatically discovering all managed servers via the use of a Nagios plugin that scans the IP network infrastructure over pre-specified subnets.

Our WSN agent service initially configures its attached XBee device to join a specific PAN and channel (initially the same for all devices) and executes a loop waiting for broadcast or unicast messages of the formats described in Table 1. After receiving a `REPORT RACKS` request message (Table 1), the WSN agent responds to the management server via a unicast message carrying its module's unique 64-bit MAC address and rack identifier. Since message broadcast is unreliable, the `REPORT RACKS` message may not be received by all wireless devices. Repeated periodic transmissions of that message however reduce the likelihood that any wireless device will not report itself for long periods of time. This method has worked well in practice as we have never seen delivery errors in our experiments.

Following this initial broadcast step, all communication takes place using reliable unicast messaging. XBee modules that receive a unicast message of type `START COORDINATOR` (Table 1) know that they have been elected coordinators. Our XBee modules support messages of up to 100 bytes, thus with each MAC address occupying 64 bits we can fit up to 12 slave devices in each `START COORDINATOR` message. If the number of slave devices exceeds that number we may need additional messages. A coordinator's WSN agent configures its XBee module for the specified PAN ID, channel, and sets the coordinator-enable option. It also configures all slave XBee modules accordingly via remote configuration commands. At the end of this process the wireless sensor network is ready for use.

The WSN plugin we developed (in Python) is a general-purpose service that periodically sends specific requests as API-mode packets to coordinators. The WSN plugin follows standard Nagios practice in returning a service status code and description (Table 2). Our current implementation of the WSN plugin requests on-board state of an XBee module (e.g., power, temperature) and uses the RSSI value reported on the response message to calculate physical position; it is however easily extensible to support other types of data collection. For example, it may ask the XBee module to collect readings (also known as IOSamples) of analog or digital pins from its attached sensors via I/O operations. XBee modules may also have access to a multitude of server-mounted sensors through invocation (via the WSN agent) of intelligent platform management interfaces such as IPMI [10]. Figure 3 depicts an event handler using the plugin's reported status to detect wired-network partitions.

```
# Script called when ping service returns failure on
# server <host address>.
#
# Input arguments are:
# - service state, in (OK, Warning, Critical, Unknown)
# - service type, in (SOFT, HARD)
# - attempt # (service-type switches from SOFT
#         to HARD after a certain number of attempts)
# - host address

If (state is "Critical") AND
        (type is "SOFT") AND (attempts > 3) :

WirelessSensor =  Repository.WSN_Status(hostaddress)

        if (WirelessSensor is "UP"):
          Nagios.log <- "<host address> unreachable"
        else
          Nagios.log <- "<host address> is down"
```

Figure 3: Handler called when the ping service on <host address> fails. It correlates with the status of the wireless device to determine whether the host is unreachable over the wired network or down.

Based on previous work on node positioning [6] we estimate distances through *trilateration*, which is a method of determining the relative position of objects using the geometry of triangles in a similar fashion to triangulation. Unlike triangulation which uses angle measurements to calculate the subject's location, trilateration uses the known locations of two or more reference points and the measured distance between the subject and each reference point. To accurately and uniquely determine the relative location of a point using trilateration we need at least three reference points on a 2D plane as shown in Figure 4. The above estimate requires that the node must be within the intersection of three other nodes whose locations are known to the system. This assumption is not expected to be a problem in practice as in large datacenters one can easily pinpoint three static nodes whose coordinates are fixed and a-priori known to our system. To achieve maximum range coverage and precision in our system we place the reference points on two corners and the center of the datacenter room.

IV. RELATED WORK

Zigbee [16] is an open standard that has been widely adopted in the design and implementation of wireless sensor networks and is rapidly becoming a key technology for ambient intelligence environments. Zigbee is based on the IEEE 802.15.4 standard [17] extended with higher-level network and application layers. Its features include low radio emission, low power consumption, long standby duration, support for large number of nodes, flexible topology, and ease of configuration. Zigbee applications today extend into the domains of home care, home control, security, and location tracking.

Location-tracking by wireless sensor networks is an active area of research [6][14][20][21]. A variety of location-tracking techniques have been proposed to identify the physical location of large-scale system resources. Many such techniques utilize a number of static nodes (three or more) as reference points and the value of Received Signal Strength Indicator (RSSI) provided by Zigbees for distance measurement. The approaches differ on how they calculate their localization estimates. We implemented the approach described in [6] using a centralized algorithm to achieve better estimation precision.

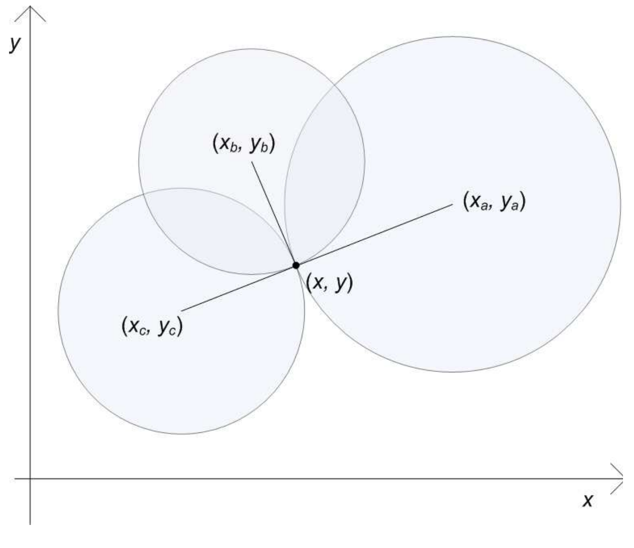


Figure 4: Intersection of three spheres.

Status code	**Explanation and status message**
OK	The plugin was able to check the service and it appeared to be functioning properly : "Signal-Fine Distance + *distance* (m)"
Warning	The plugin was able to check the service, but it appeared to violate a warning threshold or not working properly : "Signal-Low Distance + *distance* (m)" or "Sensor Changed Position + *distance* (m)"
Critical	The plugin detected that either the service was not running or it was violating a critical threshold: "Sensor Disconnected!"
Unknown	Invalid command line arguments were supplied to the plugin or low-level failures internal to the plugin (such as unable to fork or to open a TCP socket) that prevent it from performing the specified operation. "Unknown State!"

Table 2: WSN plugin return codes and messages.

Datacenters are increasingly growing to support the needs of Internet-scale electronic services and of Cloud computing users. New ways to efficiently build very large-scale datacenters have been a topic of intense interest recently [4][5]. A challenge facing enterprise-scale datacenters today is to tame the need for more administrative staff for day-to-day operations along with the associated complexity and operational cost [2]. This challenge is addressed to some extent with distributed monitoring and management tools such as Nagios [1], Ganglia [13], Autopilot [9], and SmartFrog [8]. Open source monitoring tools such as Nagios and Ganglia have recently received significant attention by developers and datacenter managers. Both systems are designed to be extensible with a large number of plugins (extensions) being contributed to them by an active development community.

The use of sensors in collecting datacenter information for management purposes is not a new concept. Previous work by HP [19] and Intel [20] proposed using RFID and Wireless USB (WUSB) technologies respectively for asset tracking and location-based services. The HP system is based on passive RFID tags attached to each server and requires one radio frequency identification (RFID) reader per server which communicates with a rack-level data collector. The RFID reader is connected with a controller that learns its location by interacting with a configuration server. The Intel system uses WUSB technology projected to be widely available in the future along with knowledge about data center geometry to arrive to a more cost-effective solution. Their work focuses on the efficiency of localization, whereas ours focuses primarily on integrating such a mechanism with a data center management system.

Hewlett Packard recently (2012) announced the commercial availability of servers equipped with a plethora of sensors and the ability to pinpoint the server's physical location [3]. While details of HP's management system (to the best of the authors' knowledge) have not been openly disclosed, marketing literature seems to indicate that the location-tracking facility depends on information drawn from smart racks (presumably entered by administrators) and is thus prone to errors over time. In the area of efficient datacenter design, the use of sensors to detect airflow issues in tightly packed datacenter server containers has been proposed in the past [12].

Besides the problem of physical-location aware visualization of datacenter resources, reasoning about datacenter connectivity is another important problem that we target with significant implications for scalable software systems. Specifically, being able to distinguish between server and link failures can greatly improve the accuracy of failure detectors and thus the robustness of distributed systems software deployed in modern datacenters.

V. EVALUATION

Our experimental testbed consists of server-grade PCs with wireless sensors attached to them over USB. We use XBee Pro Series 1 chips with 60mW output power combined with XBee explorer dongles directly plugged into each server's USB port. The Nagios management server is hosted by one of the servers. At startup, the Nagios auto-discovery option locates all servers connected to the datacenter subnets. Next the system discovers the wireless sensors attached to the servers, sets a PAN identity for them, and elects the coordinator devices. The auto-configuration service can create a valid sensor network in less than 30 seconds. When the Nagios configuration is successfully completed the Nagios user interface displays information returned by the WSN plugin such as status of XBee module along with sensor measurements such as physical location, power, etc.

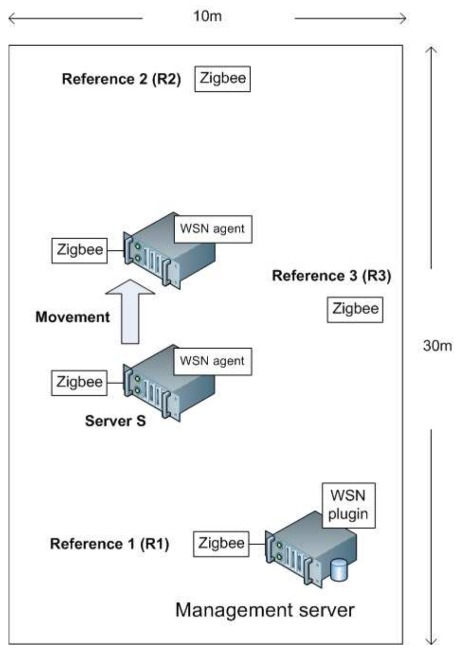


Figure 5: Experimental testbed used for tracking the physical location of servers (excerpt shown).

To demonstrate the accuracy of our server location-tracking methodology we set up an experiment in a 10x30m office (Figure 5) used by graduate students and software engineers. The office contains about 30 server PCs and associated network equipment where all but one of the machines in the room are stationary over the time of our experiment. Figure 5 depicts our three reference points (abbreviated as R*x*, *x*=1, 2, 3), R1 being the Nagios management server itself, and S being the server being moved. The management server continuously evaluates the RSSI of messages received from the other two reference points (R2, R3) and from S. RSSI measurements are taken on a per-minute basis for periods of two hours. As we move S over a 2m distance we compare the means of the three RSSI time series (R2, R3, S) before and after the movement using the unpaired Student t-test statistic. We find that the means of the R2 and R3 time series do not change over time (thus no movement is detected) whereas the mean of the S time series has a statistically reliable shift (P value $< .0001$). Thus we conclude that the server-movement event thrown by our WSN plugin is accurate and has a low probability of triggering a false alert. The warning message sent to the administrator when the sensor distance has changed is depicted in Figure 9. Moreover a critical message is sent to the administrator when a sensor disappears.

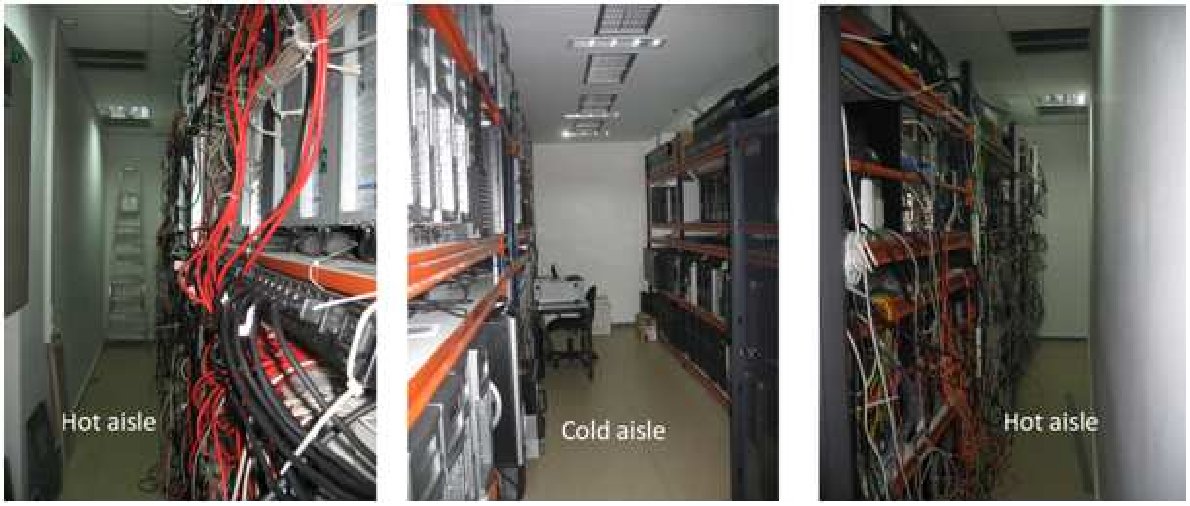


Figure 6: Data center used in localization experiment.

To test the accuracy of our methodology in tracking server locations in a data center environment (where the presence of metallic enclosures and electromagnetic interference introduce noise), we deployed our management system in the data center pictured in Figure 6 whose topology (view from top) is depicted in Figure 7. The data center consists of two rows, each row housing three vertical layers of server PCs. The front sides of servers are facing a cold aisle (HVAC equipment injecting cold air from the ceiling). Two hot aisles are at the back side of each row. The setup of our management system is as follows: All servers have Zigbee devices connected to their USB ports (back side). A group of 12 servers (4 servers per layer, 3 layers) reports to a single coordinator. We place the management server at the middle of the cold aisle. Two of the coordinators and the management server serve as the three reference points. The management server continuously evaluates the RSSI of messages received from all coordinators. As we move a coordinator over a 1.5m distance we compare the means of the RSSI time series (two references and the moving coordinator) before and after the movement using the unpaired Student t-test. We find that the means of all time series have a statistically significant shift, although the shift is small (<1.5dB) for the reference points and larger (4.5dB) for the moved coordinator. Larger data centers typically pack servers even more densely, introducing additional levels of noise. Our system can leverage known techniques, such as signal filtering [18], to increase the accuracy of indoor location tracking using the link quality indicator (LQI) available from the Zigbee physical layer.

Another use-case of interest is detecting disconnections in the wired network infrastructure. To demonstrate this use case we inject into the platform a failure of our top-of-the-rack switch connecting the servers. While the IP-based service probes (HTTP, PING, SSH) report a critical problem (Figure 8) that may be interpreted either as connectivity or server-related issue, the WSN service reports that the servers are in good health and thus pinpoints to the actual cause of the problem. The Nagios monitoring system supports monitoring the status of network switches and routers using the SNMP protocol and some extra plugins. However many commodity unmanaged switches and hubs do not take an IP address and are essentially invisible, so there is no way to monitor their status. In contrast, our service is a comprehensive solution for discovering connectivity issues in a large datacenter.

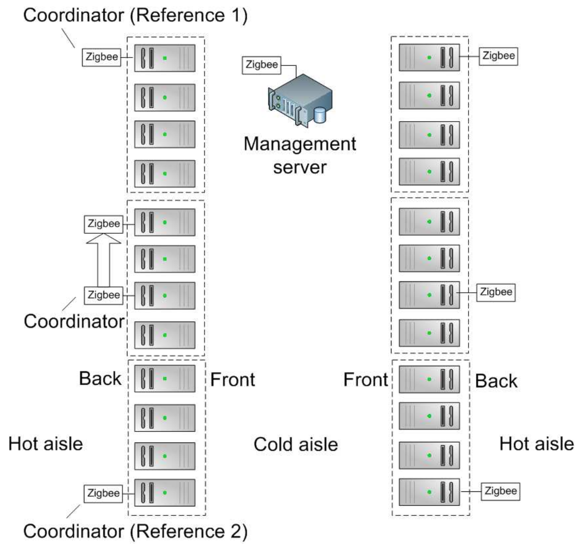
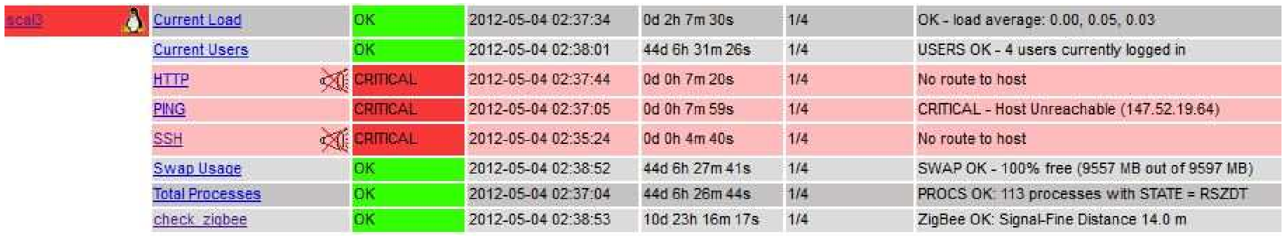
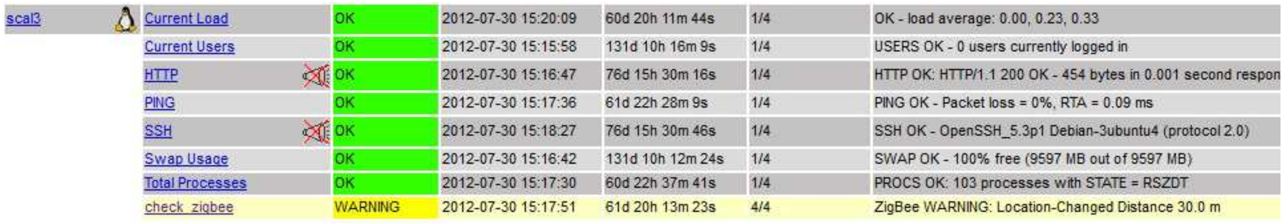


Figure 7: Data center topology. Only the top out of three vertical layers of servers is shown. Groups of servers sharing a coordinator are shown in dashed boxes. Slave Zigbees are omitted from the figure.



scal3	Current Load	OK	2012-05-04 02:37:34	0d 2h 7m 30s	1/4	OK - load average: 0.00, 0.05, 0.03
	Current Users	OK	2012-05-04 02:38:01	44d 6h 31m 26s	1/4	USERS OK - 4 users currently logged in
	HTTP	CRITICAL	2012-05-04 02:37:44	0d 0h 7m 20s	1/4	No route to host
	PING	CRITICAL	2012-05-04 02:37:05	0d 0h 7m 59s	1/4	CRITICAL - Host Unreachable (147.52.19.64)
	SSH	CRITICAL	2012-05-04 02:35:24	0d 0h 4m 40s	1/4	No route to host
	Swap Usage	OK	2012-05-04 02:38:52	44d 6h 27m 41s	1/4	SWAP OK - 100% free (9557 MB out of 9597 MB)
	Total Processes	OK	2012-05-04 02:37:04	44d 6h 26m 44s	1/4	PROCS OK: 113 processes with STATE = RSZDT
	check_zigbee	OK	2012-05-04 02:38:53	10d 23h 16m 17s	1/4	ZigBee OK: Signal-Fine Distance 14.0 m

Figure 8. Nagios state when a network problem occurs but Zigbee is reachable.



scal3	Current Load	OK	2012-07-30 15:20:09	60d 20h 11m 44s	1/4	OK - load average: 0.00, 0.23, 0.33
	Current Users	OK	2012-07-30 15:15:58	131d 10h 16m 9s	1/4	USERS OK - 0 users currently logged in
	HTTP	OK	2012-07-30 15:16:47	76d 15h 30m 16s	1/4	HTTP OK: HTTP/1.1 200 OK - 454 bytes in 0.001 second respon
	PING	OK	2012-07-30 15:17:36	61d 22h 28m 9s	1/4	PING OK - Packet loss = 0%, RTA = 0.09 ms
	SSH	OK	2012-07-30 15:18:27	76d 15h 30m 46s	1/4	SSH OK - OpenSSH_5.3p1 Debian-3ubuntu4 (protocol 2.0)
	Swap Usage	OK	2012-07-30 15:16:42	131d 10h 12m 24s	1/4	SWAP OK - 100% free (9597 MB out of 9597 MB)
	Total Processes	OK	2012-07-30 15:17:30	60d 22h 37m 41s	1/4	PROCS OK: 103 processes with STATE = RSZDT
	check_zigbee	WARNING	2012-07-30 15:17:51	61d 20h 13m 23s	4/4	ZigBee WARNING: Location-Changed Distance 30.0 m

Figure 9. Nagios state when server location changes.

VI. CONCLUSIONS

In this paper we have presented an extension of the Nagios large-scale monitoring and management system to take advantage of an auto-configuring wireless sensor network that collects and can periodically be asked to report information and status about managed servers. Our system requires minimal effort to deploy and use, has low capital and operational costs, and can significantly improve the efficiency of datacenter management operations by helping administrators pinpoint the physical locations of servers as well as alert them in case of any changes in their location. Additionally the system enables sophisticated correlations of wireless sensor-reported state with other datacenter system state reported by other agents, such as in the case of our network partition detector. Our work highlights the potential of leveraging wireless sensors for datacenter management operations and for addressing a wide spectrum of challenges in this space.

VII. ACKNOWLEDGMENTS

We thankfully acknowledge the support of the CumuloNimbo (FP7-257993) and PaaSage (FP7-317715) EU projects.

REFERENCES

[1] Nagios, http://www.nagios.org/

[2] M. Armbrust et al. Above the Clouds: A Berkeley View of Cloud Computing. Technical Report UCB/EECS-2009-28, UC, Berkeley, Feb 2009.

[3] Q. Hardy, “HP servers are a play against the Cloud”, in NY Times, Business of technology section, February 2012.

[4] M. Mitchell Waldrop, "Data Center In a Box", Scientific American, August 2007.

[5] G. Hamilton, “An Architecture for Modular Data Centers”, in Proceedings of 3rd Biennial Conference on Innovative Data Systems Research (CIDR), Asilomar, CA, January 2007.

[6] R, Mardeni and Othman, Shaifull Nizman. “Node positioning in Zigbee network using trilateration method based on the received signal strength indicator (RSSI)”, European Journal of Scientific Research, 46(1): 048-061, 2010.

[7] Xbee-api, http://code.google.com/p/xbee-api/

[8] P. Goldsack, “The SmartFrog Configuration Management Framework”, ACM SIGOPS Operating Systems Review, 43(1):16-25, 2009.

[9] M. Isard, “Autopilot: Automatic Data Center Management”, ACM SIGOPS Operating Systems Review, 41(2):60-67, 2007.

[10] Intelligent Platform Management Interface (IPMI) version 2.0, Intel, http://www.intel.com/design/servers/ipmi/

[11] IBM Tivoli Provisioning Manager, http://www-01.ibm.com/software/tivoli/products/prov-mgr/

[12] C. Thaker, “Rethinking data centers”, Networking Seminar given at Stanford Univesity, October 2007.

[13] M. Massie, B. Chun, and D. Culler, “The Ganglia Distributed Monitoring System: Design, Implementation, and Experience”, Parallel Computing 30(7):817–840, 2004.

[14] C.-Y. Liu and G.-J. Yu, “Location Tracking by ZigBee”, Cheng Shiu University (CSU), Technical Report. http://ir.csu.edu.tw/bitstream/987654321/1288/1/496.pdf

[15] RXTX library, http://rxtx.qbang.org

[16] Zigbee Standard, http://www.zigbee.org/

[17] IEEE 802.15.4 standard, http://www.ieee802.org/15/pub/TG4.html

[18] S. Halder, J.-G. Park, and W. Kim. “Adaptive Filtering for Indoor Localization using Zigbee RSSI and LQI Measurement”, Adaptive Filtering Applications, Lino Garcia (Ed.), InTech Publishing, ISBN 978-953-307-306-4, 2011.

[19] C. Brignone, et al.: “Real Time Asset Tracking in the Data Center”, Distributed and Parallel Databases, 21(2-3):145–165, 2007.

[20] N. Udar, K. Kant, and R. Viswanathan. “Asset localization in data centers using WUSB radios”, In Proceedings of the Seventh International Networking Conference on Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet (NETWORKING 2008), Singapore, May 2008, pp. 756-767.

[21] Liu, H., Darabi, H., Banerjee, P., Liu, J.: Survey of Wireless Indoor Positioning Techniques and Systems. IEEE Transactions on Systems, Man & Cybernetics 37(6), 1067–1080 (2007).