HOLISTIC PERFORMANCE CONTROL FOR MISSION-CRITICAL CYBER-PHYSICAL SYSTEMS

By

Jinzhu Chen

A DISSERTATION

Submitted
to Michigan State University
in partial fulfillment of the requirements
for the degree of

Computer Science – Doctor of Philosophy

2014

ABSTRACT

HOLISTIC PERFORMANCE CONTROL FOR MISSION-CRITICAL CYBER-PHYSICAL SYSTEMS

By

Jinzhu Chen

Recent years have seen the growing deployments of Cyber-Physical Systems (CPSs) in many mission-critical applications such as security, civil infrastructure, and transportation. These applications often impose stringent performance requirements on system *sensing fidelity*, *timeliness*, *energy efficiency* and *reliability*. However, existing approaches treat these concerns in isolation and hence are not suitable for CPSs where the system performances are dependent of each other because of the tight integration of computational and physical processes. In this dissertation, we investigate the dependencies between these performances and propose the holistic performance control approaches for two typical mission-critical CPSs, which are Wireless Cyber-phyiscal Surveillance (WCS) systems and data centers. We first propose a holistic approach called *Fidelity-Aware Utilization Controller* (FAUC) for WCS systems that combine low-end sensors with cameras for large-scale *ad hoc* surveillance in unplanned environments. By integrating data fusion with feedback control, FAUC enforces a CPU utilization upper bound to ensure the system's real-time schedulability under dynamic CPU workloads at runtime because of stochastic detection results. At the same time, FAUC optimizes system fidelity and adjusts the control objective of CPU utilization adaptively in the presence of variations of target/noise characteristics. The testbed experiments and the trace-driven simulations show that FAUC can achieve robust fidelity and real-time guarantees in dynamic environments.

We then present a proactive thermal and energy control approach for data centers to improve the energy efficiency while ensuring the data center reliability. It consists of a high-fidelity real-time temperature prediction system and a predictive thermal and energy

control (PTEC) system. The prediction system integrates Computational Fluid Dynamics (CFD) modeling, *in situ* wireless sensing and real-time data-driven prediction. To ensure the forecasting fidelity, we leverage the realistic physical thermodynamic models of CFD to generate transient temperature distribution and calibrate it using sensor feedback. Both simulated temperature distribution and sensor measurements are then used to train a real-time prediction algorithm. Based on the temperature prediction system, we propose the PTEC system, which leverages the server built-in sensors and monitoring utilities, as well as a network of wireless sensors to monitor the thermal and power conditions of a data center. It predicts the server inlet temperatures in real-time, and optimizes temperature setpoints and cold air supply rates of cooling systems, as well as the speeds of server internal fans, to minimize their overall energy consumption. To ensure the data center reliability, PTEC enforces a set of thermal safety requirements including the upper bounds on server inlet temperatures and their variations, to prevent server overheating and reduce server hardware failure rate. A partition-based approach is proposed to solve the control problem efficiently for large-scale data centers. Extensive testbed experiments and trace-driven CFD simulations show that PTEC can safely reduce substantial cooling and circulation energy consumption compared with traditional approaches, and can adapt to the realistic and dynamic data center workload.

To my beloved family.

ACKNOWLEDGEMENTS

There would be no possibility for me to finish this dissertation without the guidance from my dissertation guidance committee, the help from my colleagues and friends, as well as the support from my family members. I sincerely own my gratitudes to all these people who make this dissertation possible.

I would like to express my deepest gratitude to my advisor, Dr. Guoliang Xing, for his guidance and generous support throughout my entire doctoral study. He provided me the opportunity to work with excellent team members and guided me to build up my research vision and capability. Every discussion with him comprised a wonderful journey where I was guided to be a mature and independent researcher. This rewarding experience will benefit my entire life. Besides my advisor, I would like to thank the rest of my guidance committee members: Dr. William F. Punch, Dr. Subir Biswas and Dr. Li Xiao, for their strong encouragement, critical comments, and helpful suggestions. In particular, I thank Dr. Punch for supporting my experiments in HPCC. I also thank Dr. Juyang Weng for his guidance in early stage of my dissertation.

Special thanks go to Dr. Rui Tan, who had greatly supported all my projects. I cannot express how much I have learned from him. We worked together through many late nights and deadlines. To me, he is a sincere friend, an enthusiastic collaborator and an inspiring mentor. I also thank Dr. Xiaorui Wang for his professional comments that steered me in the right research direction. My fellow colleagues, Xiaodong Wang and Xing Fu, from Dr. Wang's team, had helped me through various technical difficulties. I thank Dr. Dirk Colbry and Jim Leikert for helping me access the HPCC server room and the server data. I thank Kelly Climer for donating us several computing servers, as well as Dr. Eric Torng, Linda Moore, Norma Teague, Debbie Krunch and Cathy Davison for their administrative support. I am also thankful for the continuous funding support from National Science Foundation. In

addition, I specially thank my supervisor Dr. Fan Bai at General Motors, who offered the maximum flexibility for me to finish the dissertation.

I sincerely thank my fellow lab-mates in eLANS: Ruogu Zhou, Dennis Philips, Mohammad Moazzami, Yu Wang, Tian Hao, Jun Huang, Yuanteng (Jeff) Pei, Pei Huang, Fernando Cintron-Gonzalez, Souror Soltani, Sayeed Choudhary, Kanthakumar Pongaliur. They helped me progress and our friendship made my journey complete. In particular, I thank Yu for helping me carry the burden of constructing the testbed, and Tian for helping with my every single request. I thank Ruogu and Dennis for their advices about the hardware implementation, without which I could not complete my experimental systems. I also thank Mohammad for helping me understand several hard concepts through various pleasant discussions.

My sincere gratitude also goes to my dear parents. Their unconditional love, sacrifice, support and encouragement from the beginning of my life accompany my every step. Their hardworking hands allow me to quest the knowledge freely. I deeply thank my grandmother, who gives all her love to me no matter where I am. I still remember her tears at the time when I was leaving for my study. I am very grateful to my extended family (in-laws), who always financially and emotionally support my wife and me with their best. I also thank my parents and mother-in-law for taking care of our baby and our family in the lonely foreign country. Besides, I would like to thank my little dog for the warmest welcome everyday.

How lucky I am to be with my beloved beautiful wife through the toughest study. No words can express how much you have sacrificed for me. Thank you for loving me, waiting for me, being with me, and understanding me. Thank you for giving birth to our lovely daughter, for all your company when I was working late, for all your tough days when I was on foot cast and for all your delicious cooking. Finally, I thank my lovely daughter, for your coming, crying, laughing, and every kissing, which make me a father and make the hard study a fascinating journey.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Embedded systems have been increasingly deployed to improve people's daily life and can be found in diverse domains including automotive, manufacturing, transportation, entertainment, etc. However, traditional embedded systems are mostly standalone, close-loop systems. In recent years, Cyber-Physical System (CPS) [1] has emerged as a new class of embedded systems which tightly integrate computational and physical processes. The CPS usually consists of a networked embedded systems with sensing and actuation capabilities to interact with the physical process. Its operation is highly dependent on the coordination and synergy between the computational and physical components of the system. This dissertation focuses on a set of mission-critical CPSs that have stringent system performance requirements. For example, data centers are complex CPSs which consist of hundreds of thousands of networked servers and several cooling systems. The thermal characteristics of the data centers are inherently affected by both physical (e.g., complex airflows and server deployment layout) and cyber (dynamic server workloads) factors. In particular, the dynamic workload and other server activities, e.g., disk and network access, generate significant heat over time. The heat is dissipated by complex airflow and removed by the cooling systems. However, the recirculated heat may increase the room temperature and cause potential hot spot. The overheated servers in the hot spot may experience heat-induced protective shutdown, causing data center service outages. Therefore, the data centers usually consume excessive energy to remove the heat in order to prevent the server overheating. Due to the increasing data center scales, electricity price and demands of reliable server services, the data centers are required to meet stringent energy-efficiency and reliability requirements.

1.1 Performance of CPS

Recent years have seen the growing deployments of CPSs in many mission-critical applications such as security, civil infrastructure, computing infrastructure, transportation and safety. The failure of these CPSs will result in security/safety breach, significant economical loss or even catastrophic consequences. Therefore, these applications often impose stringent performance requirements to the system design including *fidelity*, *timeliness*, *reliability* and *energy efficiency*.

Fidelity: Fidelity refers to a system's capability of maintaining its performance to an acceptable level when noise and errors exist in its input and/or its components. A mission-critical CPS requires high-fidelity sensing and actuation resulted from the computational process. However, the computational process usually depends on the dynamic physical process that imposes significant challenges for achieving high-fidelity actuation. For example, a surveillance system [2] needs to reach accurate detection results based on the physical light and acoustic signals collected by distributed sensors. The target detection results largely depend on the sensor measurements, which are commonly inaccurate due to several physical uncertainties, such as noises, hardware biases and dynamics from monitored physical process. Despite the physical uncertainties, the mission-critical CPSs are still required to achieve high-fidelity sensing and actuation.

Timeliness: The mission-critical CPSs are usually real-time systems that must handle the computational and physical dynamics in a timely fashion. The computational tasks usually need to be completed within certain timing constraints to avoid undesirable or even catastrophic consequences. For example, a CPS for earthquake detection minimizes the communication and computational overhead in seismic data collection and earthquake event detection to meet the stringent deadline (e.g., less that one second delay [3]). The violation of timeliness requirement for such mission-critical CPSs will largely reduce the system effectiveness and may result in significant economical and life loss.

Reliability: In many CPS applications, the system consists of a set of wirelessly networked devices that interact with the physical world. The reliability of each individual hardware and the communication link of the network substantially affect the performance of the entire system. However, these systems are usually deployed in harsh environments [3] where the natural conditions, e.g., weather, temperature and humidity adversely impact the reliability of wireless links and the electronic devices. In addition, the mobility of the CPSs [4] also introduces considerable wireless link dynamics. For instance, in data centers, the abnormal physical server temperatures will cause potential heat-induced server shutdown, resulting in computing and storage resource outages and substantially degrading the data center reliability.

Energy efficiency: Energy efficiency is an intrinsic requirement for CPSs. First, many CPSs require an *ad hoc* deployment of networked sensor nodes in the environment without infrastructure support. With limited power resources, the energy efficiency is a key factor that affects the lifetime of the system. Second, some large-scale CPSs such as data centers consist of hundreds of thousands of networked devices which generate significant amount of heat. The data centers usually consume excessive energy to remove the heat due to inefficient operation of their cooling systems (e.g., Computer Room Air Conditioning (CRAC)), which can account for up to half of their energy consumption. Therefore, energy efficiency is critical to their sustainability and operational costs.

Traditionally, these requirements of computing systems have been addressed by different approaches separately. The fidelity of a CPS is mainly affected by three factors: measurement noises, hardware biases, and physical dynamics. As CPSs are deeply integrated in physical environments, noises are inevitably captured by sensors as part of data measurements. Advanced signal processing techniques such as data fusion [5] have been widely employed by existing sensor systems to mitigate the impact of noisy data. Moreover, the hardware of a CPS often has systematic biases due to imperfect manufacturing process and operational conditions. Various calibration methods [6][7][8] have been developed to correct hardware

biases, which usually map the output of a hardware component to the ground truth of known stimulus. To deal with physical dynamics such as the unpredictable evolution of physical processes of interest, a CPS is often manually re-tasked based on human input. Timing constraints of computing and communication systems can be handled by real-time scheduling algorithms that schedule tasks based on their worst case execution times. The reliability of the systems can be maintained by adding redundant nodes and communication links, or increasing the safety margins (e.g., overcooling the data centers). In addition, the energy efficiency can be improved through hardware improvement, power states optimization [9] and various energy-efficient task scheduling algorithms [10][11][12].

Unfortunately, simply combining or extending these approaches is inadequate for simultaneously addressing those requirements of CPSs. First, these requirements are highly dependent on each other. As a result, addressing one requirement in isolation inevitably affects the other. For example, when the system fidelity is controlled in an online manner, it results in resource contention with other real-time tasks and hence affects the system timeliness. In [2], the detection fidelity of a cyber-physical surveillance system is maximized under several physical dynamics. However, it does not account for the impact on system timeliness. Moreover, various uncertainties such as unpredictable environmental variations and noise not only affect the fidelity of a CPS, but also lead to significantly variable computation and communication workload making it difficult for meeting timing constraints of real-time tasks. For instance, scheduling CPU tasks triggered by noise or biased sensors not only wastes computing resources but also hurts the system$\tilde{\text{O}}$s real-time performance. However, overshooting fidelity by aggressive system calibration or signal processing should also be avoided due to the potential resource contention with real-time tasks in the system. In addition, the workloads of servers are distributed to achieve even hot exhaust temperatures at the back of the server racks [13]. This allows the air conditioner systems to reduce safety margin by increasing the room temperature, achieving significant energy saving. However, it doesn't explicitly account for the overheating prevention and the unpredictable tempera-

ture variations may overheat the servers. Therefore, without a joint consideration on these performance requirements, the CPSs may not achieve globally optimized performance. This dissertation focuses on the *holistic* performance control for the following two mission-critical CPSs by jointly considering those performance requirements.

1.2 Wireless Cyber-Physical Surveillance System

A typical Wireless Cyber-physical Surveillance (WCS) system consists of battery-powered cameras, sensors, and embedded computers that communicate through wireless networks [2]. Without the reliance on wired power and communication infrastructure, WCS systems can be rapidly deployed in an *ad hoc* manner for large-scale surveillance in *unplanned* environments for detecting the events of interest. This is a key advantage for many critical domains such as security, transportation, and natural/physical hazard monitoring. For instance, in 2008, a number of wirelessly connected cameras were deployed for real-time and high-fidelity surveillance over a 26-mile course of the Boston Marathon which attracted over 20,000 runners and more than one million spectators [14]. In other scenarios like border security, WCS systems need to provide surveillance and intruder detection during an extended period of time up to several years. Because of the tight budget on power resources and network bandwidth, WCS systems often operate in an on-demand fashion where low-end (e.g., acoustic/infrared/magnetic) sensors serve as "sentinels" that wake up high-quality but power consuming sensors (e.g., pan-tilt-zoom cameras) once a possible target is detected. High-quality sensing results (e.g, images) are then transmitted to an embedded computing device for high-fidelity object detection and recognition [2].

Both *fidelity* and *timeliness* are essential requirements of the WCS systems described above. As an example, users may require any target of interest to be detected "at high fidelity (both missing and false alarm rates lower than 1%) and in real time (delay within five seconds)". However, a key challenge is that the timeliness and fidelity of a WCS system are tightly dependent of each other. First, the performance of low-end sensors is extremely

sensitive to dynamics in the physical environment. It is shown in [15] that individual dual-axis magnetometers on Mica2 motes [16] can exhibit up to 60% false alarm and missing rates. As low-end sensors trigger image capture and processing, their poor fidelity can significantly affect the workload and real-time performance of the system. For instance, the false alarms from low-end sensors not only lead to energy waste of cameras but also generate extra computation workload for image processing. On the other hand, reducing CPU workload and camera activity unnecessarily may lead to the increased target missing rate to degrade the fidelity of the WCS systems.

Numerous real-time scheduling algorithms have been proposed to achieve real-time guarantees for computing systems. However, many of them require detailed knowledge of CPU workload while WCS systems are subject to stochastic workload because of the impact of physical dynamics. Several recent approaches [17, 18, 19] can handle variable system workload. However, they are incognizant of system fidelity requirements. On the other hand, although sensor calibration [8] and signal processing [5] techniques are available to improve the fidelity of a sensing system, they do not account for the impact on system timeliness. For instance, minimizing target missing rate often leads to a high false alarm rate [5], which in turn poses undesirable CPU workload for a WCS system as discussed earlier.

1.3 Data Center

Data centers have become a critical computing infrastructure in the era of cloud computing. As complex Cyber-Physical Systems, the data centers are tightly coupled with their environment and inherently affected by a tight integration of computational and physical processes. With the rapid increasing demands and usages on computation, data storage and networking tasks, the scales and the power densities of the data centers increase significantly. In a 2007 report to the US Congress [20], the Environmental Protection Agency (EPA) estimated that the annual data center energy consumption in the US will grow to over 100 billion kWh at a cost of $7.4 billion by 2011. In addition to excessive energy consumption

and adverse environmental implications, the high heat dissipation caused by improper data center design and thermal management may lead to significant hardware malfunction and heat-induced server self-protective shutdown, causing the service outage of the entire data center. Research has shown that more than 23% of data center outages are caused by servers' self-protective shutdowns because of overheating [21]. A total of 2.8 million hours of server downtime worldwide has been estimated for 2011, which costs as much as 426 billion dollars a year [22]. Therefore, the *energy efficiency* and *reliability* of a sustainable data center are two fundamental performance requirements in data center design and management.

Various efforts have been made to improve data center energy efficiency. New green data center technologies have proven their effectiveness in a few latest industrial scale data centers. For instance, the new Google data centers reduce the non-computing energy ratios down to about 10% [23]. However, these technologies require a clean slate redesign and hence are cost prohibitive to apply in existing data centers. In addition to data center redesign, a variety of thermal control schemes have been recently proposed to prevent thermal emergencies in a data center while reducing the energy costs. The existing approaches either optimize a single thermal variable (e.g., server workload, CRAC setpoint, or fan speed, etc.) [24][25] or a combination of them [26][27] to minimize the energy costs. However, most of these approaches are based on a *reactive* scheme, which reactively controls the cooling systems to eliminate detected hot spots. Unfortunately, this approach often cannot achieve desirable energy efficiency, primarily due to the complex thermodynamics of data centers. For instance, the heat generated by increased server workload takes substantial delays to be recirculated to the server inlets. To react to the detected hot spots at the server inlets, the CRAC systems need to adopt sufficiently low temperature setpoints, which, however, significantly downgrade their energy efficiency [25].

1.4 Contributions

Different from the previous researches addressing the performance requirements of these Cyber-Physical Systems (CPSs) separately, this dissertation focuses on the holistic performance control that jointly addressing these performance requirements under various uncertainties, such as stochastic system workload and dynamic environmental noises. In particular, this dissertation makes the following contributions:

- We propose a novel fidelity-aware utilization control problem formulation to jointly enforce the fidelity and timeliness requirements of a WCS system. Then, It develops Fidelity-Aware Utilization Controller (FAUC) that adaptively controls the WCS system to bound the CPU utilization for timeliness and schedulability while minimizing the system detection error rate. FAUC has been implemented on a small-scale WCS testbed and evaluated under real dynamics and extensive simulations based on real acoustic data traces.

- We propose a novel cyber-physical system approach for predicting temperature distribution of data centers. This approach integrates Computational Fluid Dynamics (CFD) modeling and real-time data-driven prediction to achieve high fidelity temperature forecasting in various thermal conditions of data centers, including rare but critical thermal emergency situations like AC failures.

- We propose a proactive thermal and energy control approach based on the proposed real-time temperature prediction. By proactively controlling the cooling system and server internal fans, it significantly reduces the total energy consumption that comprises cooling and air circulation energy consumption while ensuring the thermal safety of the data center. Moreover, we propose a two-stage partition-based approach to solve the control problem efficiently for large-scale data centers.

1.5 Dissertation Organization

Chapter 2 discusses the related work. Chapter 3 investigates the problem of addressing both fidelity and timeliness requirements of a Wireless Cyber-physical Surveillance (WCS) system. A Fidelity-Aware Utilization Controller (FAUC) is proposed to adaptively controls the WCS system to bound the CPU utilization for timeliness and schedulability while minimizing the system detection error rate. Chapter 4 and Chapter 5 discuss a system for predictive thermal and energy control in data centers. In particular, Chapter 4 describes a high-fidelity real-time temperature prediction approach and discuss its performance with both a single-rack testbed and a 5-rack section of a production data center. Based on this temperature prediction approach, Chapter 5 designs a predictive thermal and energy control system that accounts for energy efficiency of both data center cooling systems and server internal fans. Chapter 6 concludes the dissertation.

CHAPTER 2

RELATED WORK

2.1 Fidelity and Timelinesss Control

Data fusion [5] is an effective signal processing technique that improves the fidelity of sensing systems by mitigating the impact of noise. Most previous studies [5] focus on analyzing the optimal fusion strategy of a given sensing system. In [28, 29], authors study the impact of data fusion on spatial and temporal coverage of large-scale sensor networks. Sensor calibration can also improve system fidelity by correcting sensor biases. In [7], the biases of light sensors are estimated by solving the equations that correlate their measurements. Similarly, in [8], the parameters of ranging sensors are estimated based on pair-wise range measurements. In [30], sensors are jointly calibrated to improve the system-level performance of fusion-based sensor networks. The above approaches calibrate sensors according to known ground truth inputs and hence work in an open-loop fashion. In [2], authros develop a feedback-based calibration algorithm that maintains system sensing fidelity in the presence of environmental dynamics. However, data fusion and sensor calibration are not concerned with meeting other requirements, such as energy efficiency and timing constraints.

Feedback control techniques have shown great promise in providing real-time guarantees for CPSs by adapting to workload variations based on dynamic feedback. For instance, feedback-based CPU utilization control [17, 18, 19] has been demonstrated to be an effective way of meeting the end-to-end deadlines for real-time systems. However, most of these algorithms rely on task rate adaptation and hence cannot handle unpredictable task rate variations that may be caused by low system fidelity. Different from these studies, we aim to jointly address the requirements on system fidelity and timeliness.

2.2 Reliability and Energy Control for Data Centers

2.2.1 Data Center Thermal Modeling

Several researches have focused on the modeling and prediction of temperature distribution to prevent thermal emergencies. In [31][32], the temperature distribution of a single server is emulated based on simplified thermodynamic laws, CPU temperature/utilization, and airflow velocity. In [33], a heat flow model is proposed to characterize the heat recirculation and predict the temperature distribution. In [34], artificial neural network is employed to learn and predict the steady-state temperature distribution under static workload assignment. However, these approaches rely on steady-state thermal models, which cannot well model the temperature evolution when the heat dissipation from servers is dynamic. They also require a controlled training procedure which is usually intrusive or even infeasible to a production data center. Moreover, such data-driven approaches often suffer low prediction fidelity due to insufficient training data, especially for rare but critical thermal emergency conditions like cooling system failures. In [35], a CFD model is constructed to enable the offline temperature prediction and analysis for critical thermal emergencies, however, without online perdition capability. In [36], a forecasting model, called ThermoCast, predicts the temperature distribution in the near future based on a simplified thermodynamic model. However, the model relies on several specific assumptions on the airflow dynamics, which may not hold in diverse data center environments. For instance, it assumes that the cold air runs vertically from raised floor tiles. This does not hold in many data centers where the cooling equipment is placed in the row of the racks [37][38] or near the racks [39], which generates significant side-to-side airflow. A recent work [40] predicts the temperature as weighted average of contributing temperatures of each heat source, where the model coefficients are determined via offline CFD simulations. Thus this approach does not leverage the real temperature measurements in a data center to ensure the prediction accuracy.

2.2.2 Thermal and Energy Control

Existing data center thermal and energy control approaches can be broadly divided into two categories. The first category of approaches minimizes the energy consumption or a cost function by controlling a single type of thermal variables, e.g., server workloads [25, 41], CRAC setpoints [24], fan speeds [42], CPU frequencies [43], or the number of virtual machines [44]. Thermal-aware load balancing (e.g., [25]) can prevent hotspots and help increase room temperature for energy saving. In [24], CRAC systems are controlled to maintain the temperatures of selected locations at their setpoints. In [42], fan speeds of blade servers are controlled to minimize the total fan power consumption subject to an upper bound on CPU temperatures. In [43], the sum of CPU frequencies is maximized under a certain power budget. All the above approaches focus on controlling one type of thermal variables to reduce the power consumption.

The second category of approaches controls multiple types of thermal variables to reduce the overall power consumption. In [45], server fan speeds, CPU power states and workload migration are controlled to minimize the overall power consumption within a blade server enclosure. In [32], server thermal management policies are selected based on predicted temperatures. However, these two approaches [32][45] do not consider the cooling energy consumption. In [26], computing jobs are assigned to the servers with the highest cooling efficiencies. The CRAC setpoints are then calculated based on the job assignment to ensure thermal safety for servers. The approach in [46] minimizes the cost of electricity and quality of service degradation while maintaining the server temperature within a predefined range. In [27], the CRAC setpoints are first determined based on data center utilization levels, and then a feedback server fan control approach is applied to achieve a trade-off between server leakage power and fan power. However, it does not minimize the overall energy consumption of CRAC systems and server fans. The approach in [47] minimizes a holistic cost metric, accounting for the availabilities and prices of the traditional/renewable electricity sources, multiple cooling options, and data center workload status. However, this approach accounts

for neither the variability of fan power, nor the complex dynamics of server temperatures. In [48], floor-mounted adaptive vent tiles and CRAC cooling provision are controlled to reduce the cooling cost. However, the server fan power consumption is not considered. Moreover, their approach cannot be readily applied to other cooling structures such as in-row cooling that is commonly adopted in data centers.

2.2.3 Data Center Thermal Monitoring

Several sensor systems have been developed for temperature monitoring in data centers [49][50]. RACNet [49] is designed for reliable data collection in large-scale data centers, where each node is connected with multiple daisy-chained temperature sensors. In [51], a fusion-based approach is developed to detect hot spots in data centers using measurements of multiple sensors. Robotic systems have also been designed to roam inside the data centers for plotting thermal map [52] and energy management [53]. In [54], thermoelectric coolers on server processors are used to remove the heat directly from hot processors. However, these studies are not concerned with the real-time temperature prediction or could not be used for legacy data centers, where the servers are not equipped with thermoelectric coolers.

CHAPTER 3

FIDELITY-AWARE REAL-TIME UTILIZATION CONTROL

3.1 Introduction

This chapter investigates the problem of addressing both fidelity and timeliness requirements of Wireless Cyber-physical Surveillance (WCS) systems. The fidelity is defined as a system's capability of reaching correct conclusions even when the sensing results from the dynamic physical environment are noisy. In addition to fidelity, timeliness is another fundamental requirement as many computational tasks in a CPS must complete within tight deadlines in order to avoid undesirable or even catastrophic consequences. The fidelity and real-time concerns of WCS systems must be *jointly* addressed because of the tight integration of system computational and physical components.

Numerous real-time scheduling algorithms have been proposed to achieve real-time guarantees for computing systems. However, many of them require detailed knowledge of CPU workload while WCS systems are subject to stochastic workload because of the impact of physical dynamics. Several recent approaches [17, 18, 19] can handle variable system workload. However, they are incognizant of system fidelity requirements. On the other hand, although sensor calibration [8] and signal processing [5] techniques are available to improve the fidelity of a sensing system, they do not account for the impact on system timeliness. For instance, minimizing target missing rate often leads to a high false alarm rate [5], which in turn poses undesirable CPU workload for a WCS system as discussed earlier.

Unlike previous literatures, this chapter proposes a novel approach to holistically address the fidelity and timeliness requirements of WCS systems. This approach integrates multi-sensor data fusion [5] with feedback control to achieve adaptive fidelity and real-time guarantees for WCS systems operating in dynamic environments. Specifically, this chapter

makes the following major contributions:

1. We propose a novel problem formulation for the fidelity-aware utilization control problem where a given upper bound on the CPU utilization is enforced while system detection error rate is minimized. The formulation is based on rigorous performance models that characterize the fusion-based detection performance and the expected CPU utilization induced by processing stochastic detection results.

2. We develop Fidelity-Aware Utilization Controller (FAUC) that adaptively adjusts the data fusion threshold to bound the CPU utilization according to user requirement. At the same time, FAUC minimizes the system detection error rate while ensuring real-time schedulability.

3. FAUC has been implemented and tested on a small-scale WCS testbed consisting of TelosB motes, Iris motes, and cameras. The extensive experiments on light and acoustic target detection show that FAUC can achieve robust fidelity and utilization control in the presence of significant physical dynamics and unreliable wireless links.

4. We conduct extensive simulations based on real acoustic data traces collected in a military vehicle surveillance experiment. The simulations evaluate the performance of FAUC under a wide range of settings of network size and signal-to-noise ratio.

The rest of this chapter is organized as follows. Section 3.2 presents the background on sensing and data fusion models. Section 3.3 describes our problem and provides an overview of our approach. Section 3.4 and 3.5 model system performance and present the design of FAUC, respectively. Section 3.6 and 3.7 present the testbed experiment results and trace-driven simulation results, respectively. Section 3.8 concludes this chapter.

3.2 Preliminaries

3.2.1 Sensor Measurement Model

Assuming that sensors measure the energy of received signals for event detection, let s_i denote the signal energy received by sensor i, which is affected by several factors and varies for different sensors. First, each sensor may have its hardware bias. Second, the measurement value is stochastic as it inevitably contains environmental noise. Third, the signal path loss between the event and sensor varies with distance and terrain. Depending on the hypothesis that the target is absent (H_0) or present (H_1), the measurement of sensor i, denoted by y_i, is given by

$$H_0 : y_i = n_i, \qquad H_1 : y_i = s_i + n_i,$$

where n_i is the energy of noise in sensor i's measurement. Assume that the noises of sensors are independent and follow the normal distributions, i.e., $n_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$, where μ_i and σ_i^2 are the mean and variance of n_i, respectively. The sensor measurement model described above has been widely adopted in the literature of event detection [55] and also have been empirically verified [56, 57]. However, many previous studies assume that the parameters of the above model, i.e., s_i, μ_i and σ_i, are known *a priori*. Unfortunately, this assumption often does not hold in reality because of the stochastic nature of sensing and these parameters are assumed to be unknown in this work.

3.2.2 Data Fusion Model

Data fusion [5] has been proposed as an effective signal processing technique to improve the system performance of sensing systems. A system based on data fusion is usually organized into multiple clusters. Each cluster has a cluster head that gathers information from member sensors and makes the system decision regarding the presence of the target. This work adopts a simple data fusion model where the system decision is made by comparing the sum of member sensors' measurements against a threshold T, which is referred to as

the *fusion threshold* hereafter. The cluster head makes a positive decision if the sum of measurements exceeds the threshold. Such a model has been adopted by several previous studies[28, 2]. Suppose there are N sensors in a cluster. The sum of measurements, denoted by Y, is given by $Y = \sum_{i=1}^{N} y_i$. Let $\widetilde{H}_0$ and $\widetilde{H}_1$ represent the detection decisions that the target is absent and present, respectively. Denote $S = \sum_{i=1}^{N} s_i$, $\mu = \sum_{i=1}^{N} \mu_i$ and $\sigma^2 = \sum_{i=1}^{N} \sigma_i^2$. Depending on whether the target is present, the sum of sensor measurements follows the normal distribution, i.e., $Y|H_0 \sim \mathcal{N}(\mu, \sigma^2)$ or $Y|H_1 = \sum_{i=1}^{N} \sim \mathcal{N}(\mu + S, \sigma^2)$. A target is detected only if the sum of sensor measurements is greater than the threshold T, i.e., $Y > T$. The detection of a target is inherently stochastic because of the random noises in sensor measurements. The *system* detection performance is characterized by two metrics, namely, the false alarm rate (denoted by P_F) and missing probability (denoted by P_M). P_F is the probability of deciding $\widetilde{H}_1$ when *no* target is present, and P_M is the probability of deciding $\widetilde{H}_0$ when a target is present. P_F and P_M can be computed by $P_F = Q\left(\frac{T-\mu}{\sigma}\right)$ and $P_M = Q\left(-\frac{T-\mu-S}{\sigma}\right)$, where $Q(x)$ is the Q-function of the standard normal distribution, i.e., $Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^{+\infty} \exp\left(-\frac{t^2}{2}\right) \mathrm{d}t$.

3.3 Problem Statement

This section describes the problem of fidelity-aware utilization control. It first discusses the system model in Section 3.3.1. It then formulates the problem and provide a brief overview of the approach in Section 3.3.2.

3.3.1 System Model

Assume that a Wireless Cyber-physical Surveillance (WCS) system consists of a base station and multiple sensor clusters. Each cluster is composed of *low-end* and *high-quality* sensors. The low-end sensors (e.g., acoustic and infrared sensors) usually have a low manufacturing cost and low energy consumption. As a result, their sensing capability is often

limited. As discussed in Section 3.2.2, the system performance can be improved by employing data fusion on the measurements of low-end sensors. The high-quality sensors (e.g., pan-tilt-zoom cameras [58] and active radars [59]) can provide high-accuracy sensing and detection at the price of higher manufacturing cost and energy consumption. In this work, it is assumed that there is only one high-quality sensor in each cluster. However, this approach can be easily extended to the case of multiple high-quality sensors, where they may fuse their measurements to yield a detection result.

In accordance with the heterogeneous system architecture, The WCS adopts a *two-phase* target detection process. Initially, the low-end sensors periodically sense the environment while the cameras remain asleep because their power consumption is typically several orders of magnitude higher than low-end sensors [16]. The cluster head fuses data from low-end sensors and makes a decision according to the threshold T. If it is a positive decision, the cluster head then activates the camera to capture an image of the surveillance region, and sends the image to the base station. Finally, a target recognition algorithm is executed by the base station to process the images and detect whether a target of interest is present. The key advantage of such a two-phase target detection scheme is that the system power consumption can be significantly reduced without sacrificing the detection performance. In particular, most false alarms can be filtered out by the data fusion of low-end sensors and hence the high-quality sensors (i.e., cameras) can sleep for most of the time and be switched on only when the probability of target presence is high. As a result, the system can achieve high-fidelity surveillance for extended lifetime in unplanned environments without wired power infrastructure. It is worth noting that, this approach is not restricted to the particular WCS system architecture described above. It can be applied to similar heterogeneous system architectures where computation intensive tasks are triggered by the sensing results of low-power sensors.

3.3.2 Problem Formulation

The objective of this work is to achieve satisfactory timeliness and fidelity of WCS systems. To guarantee the end-to-end timeliness required by a WCS system, the delay of each stage of the entire process of sensing, communication, and computing must be carefully considered. In previous studies [60], achieving real-time sensor sampling, data fusion, and wireless communications in surveillance systems has been extensively studied. This work focuses on providing the real-time guarantee on target detection in base station that must run computation-intensive tasks to process high-quality sensor data such as images. Specifically, the system controls the CPU utilization to enforce appropriate schedulable utilization bounds, e.g., the Liu and Layland bound for rate-monotonic scheduling [61], despite significant uncertainties in system workloads. In the meantime, utilization control can also enhance system survivability by providing overload protection against workload fluctuation [62]. Our approach can be integrated with previous solutions [60] to ensure the end-to-end timeliness of a WCS system. For instance, the deadline of target detection can be ensured by enforcing sub-deadlines for sensing, communication, and computing separately. Particularly, the delay of computation is often significant when complex sensing data such images need to be processed.

Several challenges must be addressed to satisfy both timeliness and fidelity requirements simultaneously for WCS systems. First, the timeliness and fidelity performance of a system are highly dependent of each other. For instance, although the false alarms of low-end sensors can be dealt with by turning on the camera more frequently, it inevitably increases CPU workload and impedes system timeliness. On the other hand, reducing CPU workload and camera activity aggressively may lead to an increased target missing rate. Second, system CPU workloads are highly variable because of several factors such as uncertain image processing time and stochastic detection performance of low-end sensors. The probability that the camera is activated and an image needs to be processed is highly dependent on the data fusion results, which in turn are affected by time-varying noise and target characteristics

in dynamic environments.

This proposes a control-theoretic solution called Fidelity-Aware Utilization Controller (FAUC) to address these challenges. FAUC employs a feedback controller to enforce the specified upper bound on CPU utilization of base station while minimizing the overall system detection error rate. By taking advantage of the adaptivity of the controller, FAUC allows a WCS system to achieve robust assurance on timeliness and fidelity in dynamic environments. We formally formulate our problem as follows.

Definition 1 (Fidelity-Aware Utilization Control)**.** *To find a stable and converging control algorithm for the fusion threshold T at the cluster head based on the feedback of the base station, such that the expected CPU utilization $\mathbb{E}[u]$ is upper bounded by u_s while the detection error rate P_e is minimized, where u_s is a constant that ensures system's real-time schedulability.*

In the above formulation, the CPU utilization bound u_s is a predefined user input to the controller, and we focus on the utilization control of only one cluster. When there exist multiple clusters in the system, their CPU utilization bounds can be determined by schedulability analysis [61] and then separately enforced by multiple FAUC controllers. The detection error rate P_e in Problem 1 is the probability that the system makes a wrong detection decision, which jointly accounts for false alarms and misses. Such a metric is widely adopted in the literature of sensing systems [5]. We choose fusion threshold T as the control input as it affects both the system detection performance and timeliness. Specifically, when T is lower, the missing rate is lower while more false alarms may be triggered by noise leading to higher system workload. On the other hand, a higher T reduces both false alarm rate and system workload while a target is more likely to be missed.

Figure 3.1 illustrates the architecture of FAUC and a WCS system. We now describe the three main components in the system architecture. (1) *Two-phase fusion-based target detection* (Section 3.4). The measurements of low-end sensors are first fused by the cluster head. If a positive decision is made, the camera is activated and the captured image is then

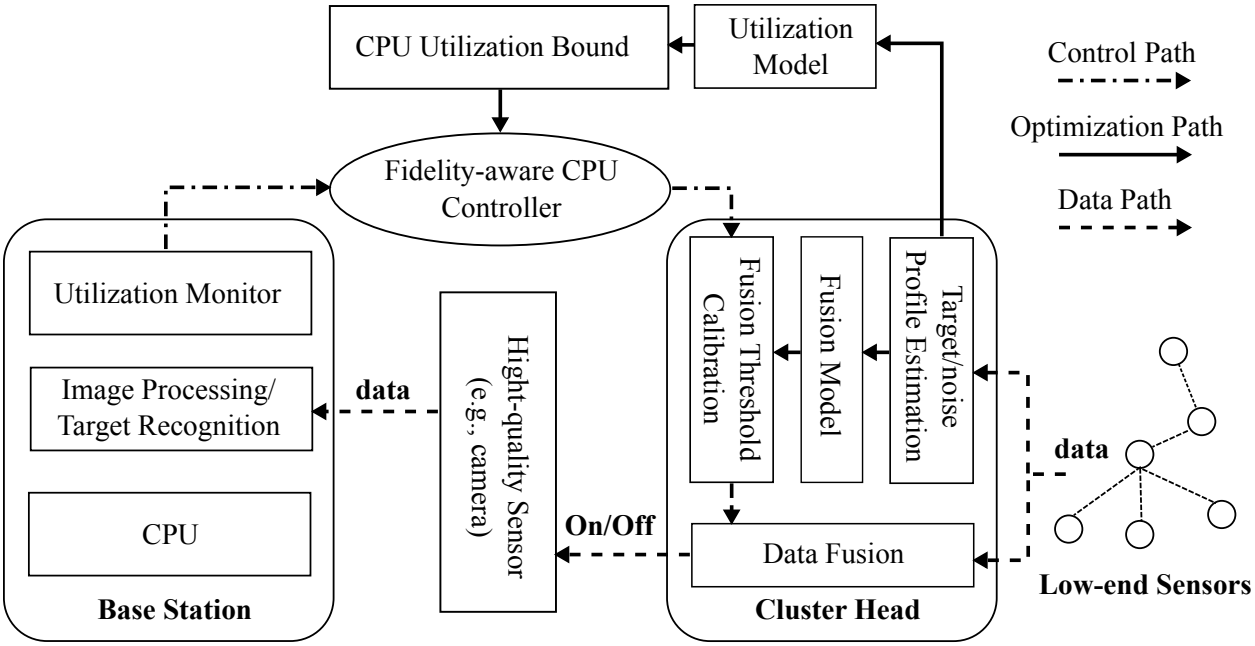


Figure 3.1 The architecture of FAUC controller.

processed by the base station for target recognition. (2) *Utilization feedback control loop* (Section 3.5.1 and 3.5.2). The FAUC utilization controller adaptively calibrates the fusion threshold T to enforce a user-specified utilization upper bound when the system workloads vary significantly as a result of stochastic camera activations and uncertain image processing time. The utilization monitor measures the average CPU utilization and provides feedback for calibrating the fusion threshold. The controller design is based on analytical models of CPU utilization and system fidelity. (3) *Detection performance optimization loop* (Section 3.5.3 and 3.5.4). FAUC employs the k-means clustering algorithm [63] to periodically estimate system parameters (e.g., target and noise models) from sensor measurements. The results are used to optimize the fusion threshold and adjust the control objective of CPU utilization. In the presence of physical dynamics (e.g., variations of target/noise energy), such an optimization mechanism adapts utilization control objectives and maintains satisfactory system fidelity.

3.4 Performance Modeling

In this section, we formally model the performance of WCS systems. The results provide a foundation for the design of FAUC controller in Section 3.5. We first model the system detection error rate in Section 3.4.1. The impact of communication packet loss is analyzed in Section 3.4.2. Finally, we model system CPU utilization in Section 3.4.3.

3.4.1 System Detection Performance

Before formally modeling the system detection performance, we first make the following assumptions. First, the probability that a target is present at any time instance is P_a, which is unknown but can be estimated from detection history. Second, the false alarm rate and missing probability of the high-quality sensor, denoted by P_{FH} and P_{MH}, are known. P_{FH} and P_{MH} can often be measured via offline experiments. Because of the high accuracy of the high-quality sensor, both P_{FH} and P_{MH} are close to zero. In addition, we let P_{FL} and P_{ML} denote the false alarm rate and missing probability of low-end sensors. The system detection error rate P_e is the weighted sum of the joint false alarm rate and missing probability of high-quality and low-end sensors. Specifically,

$$P_e = (1-P_a)\cdot P_{FL}\cdot P_{FH} + P_a\cdot(P_{ML}+(1-P_{ML})\cdot P_{MH})\,, \tag{3.1}$$

where $P_{FL}\cdot P_{FH}$ corresponds to the case that both the low-end sensors and the camera raise a false alarm, and $P_{ML}+(1-P_{ML})\cdot P_{MH}$ corresponds to the case that the low-end sensors make a correct detection but the activated camera misses the event. We now discuss how to achieve the minimal P_e. In Section 3.2.2, we have derived the expressions for the false alarm rate and missing probability of low-end sensors, i.e., $P_{FL} = Q\left(\frac{T-\mu}{\sigma}\right)$ and $P_{ML} = Q\left(-\frac{T-\mu-S}{\sigma}\right)$. By replacing P_{FL} and P_{ML} in Eqn. (3.1) with these expressions and solving the condition for minimal P_e, i.e., $\frac{\partial P_e}{\partial T} = 0$, the optimal fusion threshold T_{opt} that minimizes P_e is given by

$$T_{\text{opt}} = \frac{\delta\cdot\sigma^2}{2S} + \mu + \frac{S}{2},$$

where $\delta = 2\ln\left(\frac{1-P_a}{P_a} \cdot \frac{P_{FH}}{1-P_{MH}}\right)$. Note that the performance modeling above is based on the assumption that all the packets sent by low-end sensors are correctly received.

3.4.2 Impact of Packet Loss

Packet loss due to unreliable wireless communication can cause the system detection performance of low-end sensors to deviate from the theoretical results derived in Section 3.4.1. We propose to address the impact of packet loss by exploiting *temporal sampling*, where the number of samples that each low-end sensor transmits to the cluster head is determined by the quality of communication links. In this section, the quality of a communication link from a sensor to the cluster head is characterized by the end-to-end packet reception rate. Suppose sensor i samples m times in a detection process. Let y_{ij} denote the jth noisy measurement of sensor i in a detection, p_i denote the end-to-end packet reception rate of the path from sensor i to the cluster head, and $u_{ij} \in \{0, 1\}$ denote the packet delivery state of measurement y_{ij}. Hence, u_{ij} is a Bernoulli random variable with success probability of p_i. In the temporal sampling scheme, the cluster head fuses all measurement received during a detection process to make a decision, where the fusion statistic Y is given by $Y = \sum_{i=1}^{N}\sum_{j=1}^{m} u_{ij} \cdot y_{ij}$. In the absence of target, the mean and variance of Y are given by:

$$\mathbb{E}[Y|H_0] = m \cdot \sum_{i=1}^{N} p_i \cdot \mu_i,$$

$$\mathrm{Var}[Y|H_0] = m \cdot \sum_{i=1}^{N} p_i \cdot \sigma_i^2 + \mu_i^2 \cdot (p_i - p_i^2).$$

In the presence of target, the mean and variance of Y are given by

$$\mathbb{E}[Y|H_1] = m \cdot \sum_{i=1}^{N} p_i \cdot (s_i + \mu_i),$$

$$\mathrm{Var}[Y|H_1] = m \cdot \sum_{i=1}^{N} p_i \cdot \sigma_i^2 + (s_i + \mu_i)^2 \cdot (p_i - p_i^2).$$

The detailed derivations of the above equations are in Appendix A. As Y is the sum of $N \cdot m$ independent random variables, according to the Central Limit Theorem (CLT), Y

follows the normal distribution when $N \cdot m$ is large enough. Although N is often limited in practice, we can increase the number of samples during a detection to satisfy the condition of CLT. Denote $\mu' = m \cdot \sum_{i=1}^{N} p_i \cdot \mu_i$, $S' = m \cdot \sum_{i=1}^{N} p_i \cdot s_i$, $\sigma'_{H_0} = \sqrt{\text{Var}[Y|H_0]}$ and $\sigma'_{H_1} = \sqrt{\text{Var}[Y|H_1]}$. The system false alarm rate and missing probability of low-end sensors are given by $P_{FL} = Q\left(\frac{T-\mu'}{\sigma_{H_0}}\right)$ and $P_{ML} = Q\left(-\frac{T-\mu'-S'}{\sigma_{H_1}}\right)$, respectively. Therefore, when the impact of packet loss cannot be ignored, we need to separately estimate the variances for the cases of target absence and presence, respectively. By replacing P_{FL} and P_{ML} in Eqn. (3.1), the equation $\frac{\partial P_e}{\partial T} = 0$ has two roots, which are given by

$$\frac{\sigma^2_{H_0} S' - \sigma^2_{H_1}\mu' + \sigma^2_{H_0}\mu' \pm \sigma_{H_1}\sigma_{H_0}\sqrt{\delta'\sigma^2_{H_1} - \delta'\sigma^2_{H_0} + S'^2}}{\sigma^2_{H_0} - \sigma^2_{H_1}},$$

where $\delta' = 2\ln\frac{(1-P_a)P_{FH}\sigma_{H_1}}{P_a(1-P_{MH})\sigma_{H_0}}$.[1] The optimal fusion threshold T_{opt} that minimizes P_e is one of the above roots that gives the smaller P_e.

3.4.3 System CPU Utilization

To guarantee the real-time schedulability (e.g., by rate-monotonic scheduling [61]), the CPU utilization of each task at the base station shall be maintained at a certain level. In this section, we derive the CPU utilization model. The CPU workload of the base station is mainly generated by processing the images captured by the camera. As the camera is activated by the stochastic decisions of data fusion, the CPU workload is hence subject to change over time. We define that a *control cycle* consists of m *detections*. In each detection, low-end sensors send their measurements to the cluster head for data fusion. We now derive the expected CPU utilization in m detections of a control cycle, denoted by $\mathbb{E}[u]$, by accounting for the workload generated by both correct decisions and false alarms.

[1] We assume $\delta'\sigma^2_{H_1} - \delta'\sigma^2_{H_0} + S'^2 \geq 0$ such that there exists optimal fusion threshold minimizing P_e; otherwise, P_e will be monotonically decreasing function of T and there is no optimal fusion threshold.

We define the following notations subject to a control cycle: 1) n_{f1} and n_{d1} are the numbers of false alarms and correct detections made by the cluster head, respectively. Note that they are unknown to the system; 2) n_{f2} and n_{d2} are the numbers of positive decisions made by the cluster head but regarded as false alarms and correct detections by the camera, respectively. These two numbers can be counted by the base station after processing the images of the camera. We have the following relationships:

$$n_{f1} + n_{d1} = n_{f2} + n_{d2}, \tag{3.2}$$

$$n_{f2} = n_{f1} \cdot (1 - P_{FH}) + n_{d1} \cdot P_{MH}, \tag{3.3}$$

$$n_{d2} = n_{f1} \cdot P_{FH} + n_{d1} \cdot (1 - P_{MH}). \tag{3.4}$$

Eqn. (3.2) holds because either a correct decision or false alarm from data fusion would trigger an image processing task at the base station. The result of image processing can then again be classified as correct decision or false alarm. In Eqn. (3.3), $n_{f1} \cdot (1 - P_{FH})$ represents the number of false alarms that are correctly identified by the high-quality sensor, and $n_{d1} \cdot P_{MH}$ represents the number of correct detections that are incorrectly decided as false alarms. In (3.4), $n_{f1} \cdot P_{FH}$ represents the number of false alarms that are incorrectly decided as correct detections, and $n_{d1} \cdot (1 - P_{MH})$ represents the number of detections that are correctly identified. From (3.3) and (3.4), the unknown n_{f1} and n_{d1} can be estimated as

$$n_{f1} = \frac{n_{f2}(1 - P_{MH}) - n_{d2}P_{MH}}{1 - P_{FH} - P_{MH}},$$
$$n_{d1} = \frac{n_{d2}(1 - P_{FH}) - n_{f2}P_{FH}}{1 - P_{FH} - P_{MH}}.$$

Therefore, the estimates of P_{FL} and P_{ML}, denoted by $\widetilde{P}_{FL}$ and $\widetilde{P}_{ML}$, respectively, are given by

$$\widetilde{P}_{FL} = \frac{n_{f1}}{m - m \cdot P_a}, \quad \widetilde{P}_{ML} = \frac{m \cdot P_a - n_{d1}}{m \cdot P_a}. \tag{3.5}$$

The high-quality sensor sends the data to the base station for image processing, which consumes the CPU resource. We assume that the average CPU execution time with or without processing an image is e or e', respectively. Note that e' may equal zero if no

processing is required when the data fusion produces a negative result. Let T_d represent the duration of a control cycle and u represent the CPU utilization of the base station. The *expected* CPU utilization, denoted by $\mathbb{E}[u]$, is given by

$$\mathbb{E}[u] = (n_{f1} + n_{d1}) \cdot \frac{e}{T_d} + (m - n_{f1} - n_{d1}) \cdot \frac{e'}{T_d}. \tag{3.6}$$

By replacing n_{f1} and n_{d1} in Eqn. (3.6) with Eqn. (3.5), we have

$$\begin{aligned}\mathbb{E}[u] &= \frac{m \cdot e'}{T_d} + \frac{m \cdot (e - e')}{T_d} \left((1 - P_a) \cdot \widetilde{P}_{FL} + P_a \cdot (1 - \widetilde{P}_{ML}) \right) \\ &\simeq K_1 + K_2 \left((1 - P_a) Q\left(\frac{T - \mu}{\sigma} \right) + P_a Q\left(\frac{T - \mu - S}{\sigma} \right) \right),\end{aligned} \tag{3.7}$$

where $K_1 = \frac{m \cdot e'}{T_d}$ and $K_2 = \frac{m \cdot (e - e')}{T_d}$. When the impact of packet loss cannot be ignored, $\widetilde{P}_{FL}$ and $\widetilde{P}_{ML}$ should be replaced by $P_{FL} = Q\left(\frac{T - \mu'}{\sigma_{H_0}} \right)$ and $P_{ML} = Q\left(-\frac{T - \mu' - S'}{\sigma_{H_1}} \right)$, respectively, where S', μ', σ_{H_0} and σ_{H_1} are derived in Section 3.4.2. From Eqn. (3.7), the expected CPU utilization monotonically decreases with T because both P_{FL} and $1 - P_{ML}$ decrease with T.

3.5 Fidelity-Aware Utilization Controller

Based on the performance modeling presented in Section 3.4, we first design the fidelity-aware CPU controller in Section 3.5.1 and discuss its stability and convergency in section 3.5.2. After that, we discuss the estimation of system parameters in Section 3.5.3 and the approach to optimizing the detection error rate in Section 3.5.4.

3.5.1 The Design of FAUC

The objective of Problem 1 is to ensure $\mathbb{E}[u] \leq u_s$ while minimizing system detection error rate P_e, where $\mathbb{E}[u]$ is a function of the fusion threshold T given by Eqn. (3.7) and u_s is a user-specified utilization bound. As the threshold T is calibrated every control cycle, Problem 1 is a typical discrete-time control problem, in which u_s is the *reference*, T is *control*

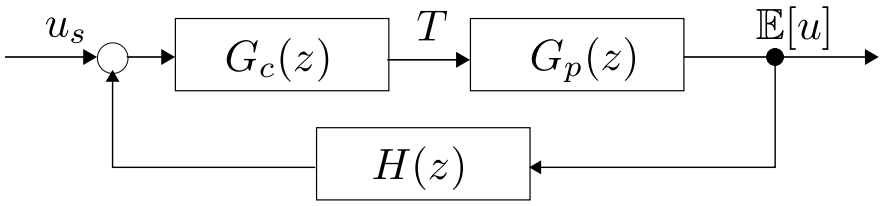


Figure 3.2 The closed-loop system to control the fusion threshold according to the CPU utilization feedback.

input and $\mathbb{E}[u]$ is the *controlled variable*. In the following, we present the design of Fidelity-Aware Utilization Controller (FAUC). We first discuss how FAUC ensures the utilization bound, i.e., $\mathbb{E}[u] \leq u_s$. In Section 3.5.4, we discuss how to minimize the system detection error rate P_e under the given utilization bound.

The block diagram of the FAUC feedback control loop is shown in Figure 3.2. The key challenge of deriving the transfer function $G_p(z)$ is that $Q(x)$ in Eqn. (3.7) is a nonlinear function. In this work, we adopt the linear approximation of Eqn. (3.7), which is given by $\mathbb{E}[u](T) \simeq \mathbb{E}[u](T_0) + \frac{\partial \mathbb{E}[u]}{\partial T}\Big|_{T_0} \cdot (T - T_0)$, where the derivative $\frac{\partial \mathbb{E}[u]}{\partial T}$ is given by

$$\frac{\partial \mathbb{E}[u]}{\partial T} = -\frac{K_2}{\sigma\sqrt{2\pi}} \left((1-P_a) \cdot \mathrm{e}^{-\frac{(T-\mu)^2}{2\sigma^2}} + P_a \cdot \mathrm{e}^{-\frac{(T-\mu-S)^2}{2\sigma^2}} \right). \tag{3.8}$$

T_0 is referred to as the *operating point*, which greatly affects the control performance. We will discuss how to choose T_0 in Section 3.5.3. Hereafter, we denote $F(T_0) = \frac{\partial \mathbb{E}[u]}{\partial T}\Big|_{T_0}$. By taking z-transform to the linear approximation, we have $G_p(z) = F(T_0)$. Therefore, the system to be controlled is a zero-order system. We can estimate $\mathbb{E}[u]$ based on the samples of CPU utilization at the base station in a control cycle. This estimate is then fed back to compare with the reference u_s. As the estimation takes one control cycle, $H(z) = z^{-1}$ which represents a delay of one cycle. We now design $G_c(z)$ to solve Problem 1. As $G_p(z)$ is zero-order, a first-order controller is sufficient to meet stability and convergence requirements. Therefore, we let $G_c(z)$ be

$$G_c(z) = \frac{a}{1 - b \cdot z^{-1}}, \tag{3.9}$$

where $a > 0$ and $b > 0$. The coefficients a and b should be chosen to ensure the system

stability and convergence. The conditions of system stability and convergence are analyzed in Section 3.5.2.

3.5.2 Stability and Convergence

We first analyze the system stability. The closed-loop transfer function, denoted by $T_c(z)$, is given by $T_c(z) = \frac{G_c(z)G_p(z)}{1+G_c(z)G_p(z)H(z)} = \frac{a \cdot F(T_0) \cdot z}{z-(b-a \cdot F(T_0))}$. The closed-loop system has a pole at $z = b - a \cdot F(T_0)$. From control theory [64], if the pole is within the unit circle centered at the origin, i.e., $|b - a \cdot F(T_0)| < 1$, the system is stable. As $F(T_0)$ is always negative, the sufficient condition for stability is $\frac{b+1}{F(T_0)} < a < \frac{b-1}{F(T_0)}$.

We then analyze the steady-state error of the system. The open-loop transfer function, denoted by $T_o(z)$, is given by $T_o(z) = G_c(z)G_p(z)H(z) = \frac{a \cdot F(T_0)}{z-b}$. By letting $b = 1$, the system is a type I system [64], in which the controlled variable $\mathbb{E}[u]$ can converge to the reference u_s provided that the system is stable. Hence, by replacing b with 1, the condition for both stability and convergence is $\frac{2}{F(T_0)} < a < 0$.

According to Figure 3.2, we have $G_c(z) = \frac{T(z)}{u_s - H(z)\mathbb{E}[u](z)}$. By replacing $G_c(z)$ with Eqn. (3.9) and letting $H(z) = z^{-1}$, we have the z-domain equation $T(z) = b \cdot T(z) \cdot z^{-1} + a \cdot (u_s - z^{-1} \cdot \mathbb{E}[u](z))$, which is equivalent to the equation in time-domain as $T[n] = b \cdot T[n-1] + a \cdot (u_s - \mathbb{E}[u][n-1])$, where $\mathbb{E}[u][n-1]$ is the average CPU utilization measured in the $(n-1)$th calibration cycle, $T[n]$ and $T[n-1]$ are the fusion thresholds for the nth and $(n-1)$th control cycle, respectively.

3.5.3 Online System Parameter Estimation

There exists a fundamental trade-off between the stability and transient response performance of a control system [64]. In our problem, the system converges faster for a larger a at the price of greater system oscillations. Therefore, the best setting for a is a value close to its lower bound $\frac{2}{F(T_0)}$. The stability can be enforced if we can online estimate the $F(T_0)$. We now discuss how to choose T_0 and estimate $F(T_0)$. According to Eqn. (3.8), in order to

compute $F(T_0)$, several parameters, i.e., e, e', μ, σ, and S, need to be estimated. Because e and e' are subject to change because of the stochastic task execution time, we calculate the average execution time for the tasks for the estimation. Assuming that μ, σ, and S can slowly change over time because of the uncertainty of the environment, we employ the k-means [63] clustering algorithm to estimate these parameters. Specifically, the k-means algorithm iteratively constructs two clusters of sensor measurements which correspond to the cases of target absence and presence, respectively. The noise mean μ is estimated by averaging the measurements in the cluster representing the noise. The target energy S can be calculated by subtracting μ from the average of measurements in the cluster representing the case of target presence. Note that if the impact of packet loss can be ignored, the variance σ^2 is estimated by averaging the variances from two clusters; otherwise, we use the separate variances from the two clusters. As the CPU utilization shall be controlled around u_s, the operating point T_0 of the linearization is obtained by solving $\mathbb{E}[u] = u_s$ where $\mathbb{E}[u]$ is given by (3.7) with the estimated μ, σ^2 and S. With the chosen T_0, we can compute $F(T_0)$.

3.5.4 Optimizing Detection Error Rate

So far, our discussion is only concerned with controlling the utilization bound while the impact on detection error rate is not considered. Although such a solution can meet the deadline once a target is detected, it may lead to low system fidelity such as excessive false alarms and consume unnecessary energy. In this section, we discuss how to optimize system detection performance without violating the utilization bound.

According to our performance modeling in Section 3.4.1, the detection performance is optimized if the fusion threshold T is set to T_{opt}. However, we cannot simply adjust the current threshold to T_{opt} without accounting for the impact on CPU utilization. FAUC addresses this issue by implementing a *dual-cycle* control strategy. In each *control* cycle, CPU utilization is enforced to the utilization bound u that is initially set to the user-specified constant u_s. Each *optimization* cycle consists of multiple control cycles. The system pa-

rameters are estimated every optimization cycle as discussed in Section 3.5.3 and then used to update T_{opt} and compute the expected utilization u^* according to our utilization model (Eqn. (3.7)). If the estimated utilization is lower than the initial utilization bound, i.e., $u^* < u_s$, it will be set as the new control objective for the following control cycles until the start of next optimization cycle. Therefore, the optimization process opportunistically lowers the utilization bound if the system detection performance can be optimized. In other words, the CPU utilization never exceeds the initial value specified by user. Hence, the real-time schedulability is always satisfied.

3.6 Testbed Experiments

To evaluate the performance of FAUC, we have conducted two testbed experiments for detecting light and acoustic targets, respectively. The results allow us to evaluate the effectiveness of our approach for different sensor modalities. Section 3.6.1 discusses the experimental methodology. Section 3.6.2 and 3.6.3 present the experimental results of detecting light and acoustic targets, respectively. Section 3.6.4 and Section 3.6.5 evaluate the impact of packet loss and number of sensor nodes to the system by using the light spot detection experiment. Section 3.6.6 shows a multi-cluster system that works with FAUC to have all the clusters' utilization well-controlled.

3.6.1 Experimental Methodology

We adopt a baseline approach referred to as *fixed-step closed-loop heuristic* for performance comparison. In this heuristic algorithm, the expected CPU utilization $\mathbb{E}[u]$ is fed back to compare with the reference u_s. As $\mathbb{E}[u]$ is a decreasing function of threshold T, if $\mathbb{E}[u] > u_s$, the new threshold $T[n]$ is calculated by adding a fixed step Δ_T to the previous threshold $T[n-1]$; otherwise, Δ_T is subtracted from previous $T[n-1]$ to calculate new threshold $T[n]$ if $\mathbb{E}[u] < u_s$. However, this approach does not consider system stability and

Figure 3.3 Testbed for light spot detection. 4 TelosB motes and a webcam detect the light spot. For interpretation of the references to color in this and all other figures, the reader is referred to the electronic version of this dissertation.

convergence. In the experiment, we employ different Δ_T for this approach to evaluate the impact of step size on the system performance.

Our evaluation focuses on two performance metrics: utilization error and average detection error rate. The utilization error is the error between the measured CPU utilization and the reference in each control cycle. In order to calculate the detection error rate, we log the ground truth information regarding the target appearance to compute the system false alarm and missing rates in each control cycle. The detection error rate is then computed as the weighted sum of the false alarm and missing rates, with weights $(1 - P_a)$ and P_a, respectively.

3.6.2 Light Spot Detection

In the light spot detection experiment, four TelosB motes [16] and a webcam are attached against the LCD screen of a desktop computer to detect a light spot that is randomly displayed on the screen (see Figure 3.3). The display of the light spot is controlled by a computer program, simulating the random presence of the target at a probability of $P_a = 50\%$ in each one-second time slot. Note that a similar method has been adopted by previous

work [2]. We vary the light intensity of each pixel on the LCD screen to simulate the changeable characteristics of noise and target. To create noise in sensor measurements, each pixel is assigned a small random light intensity value I_N with the mean of μ. I_N varies over time to simulate the changing environmental noise. To create the target, a constant light intensity value I_T is added to the noise for each pixel. Note that I_T decreases with the distance from the center of the light spot. Similarly, I_T varies to simulate the change of target profile. The TelosB motes measure the light intensity every 250 milliseconds via the on-board Hamamatsu S1087-01 light sensors [16] and transmit the measurements to the cluster head that is connected to a base station laptop. The cluster head fuses the readings received within every 250 milliseconds and detects the light spot. When the cluster head makes a positive decision, the webcam is triggered to take an image and compare the average intensity over all pixels with a threshold. The false alarm and missing rates of the webcam are 5.1% and 3.9%, respectively, which are estimated in a separate offline experiment. We evaluate the utilization control algorithms under a variety of settings. Moreover, each experiment consists of two phases. Specifically, in the first phase, the mean of I_N as well as I_T remain at their initial values. The second phase starts after 12 control cycles, where we vary the mean of I_N or I_T to simulate the changes of noise or target profile.

3.6.2.1 Sensor Measurement Performance

We first verify the Gaussian noise assumption made in Section 3.2. Figure 3.4(a) plots the Cumulative Distribution Function (CDF) of a light sensor's measurements in office environment. We can see from the figure that the sensor measurements well match the normal distribution. We then evaluate the performance of the k-means algorithm that is used to estimate the online noise and target profiles. Note that the FAUC adjusts the control reference based on the estimates given by the k-means algorithm. Therefore, the estimation accuracy can affect the performance of FAUC. Figure 3.4(b) plots the CDF of relative errors of the estimated μ, σ^2 and S with respect to their ground truths, respectively. We can see

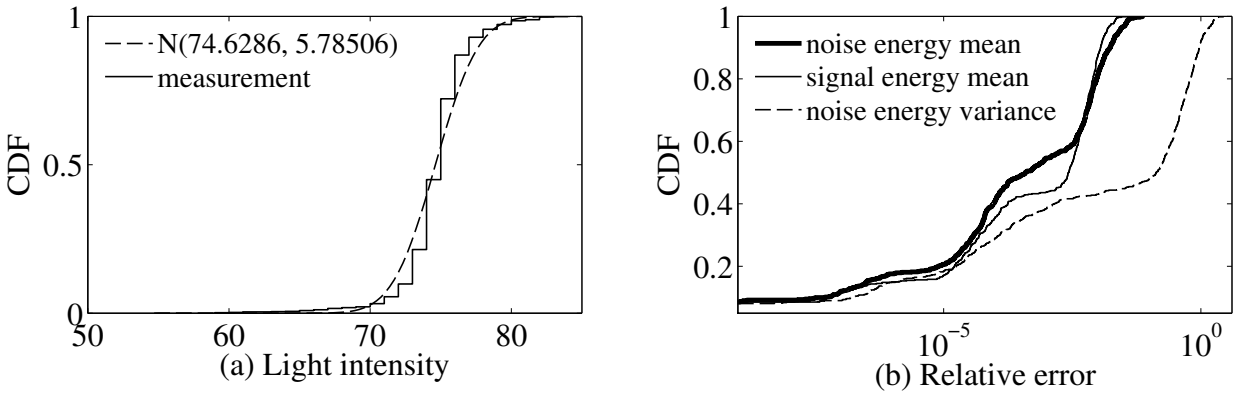


Figure 3.4 (a) The CDF of the light intensity measurements of a TelosB mote; (b) The CDF of estimation errors of noise and target profiles.

that all of them can be estimated accurately. For instance, about 40% of the estimated σ^2 has a near-zero error and the maximum error is only about 2%.

3.6.2.2 Stability and Convergence

We now evaluate the stability and convergence of FAUC in dynamic environments. Figure 3.5 shows the temporal evolution of the system when noise mean is increased at the 12th control cycle. Each optimization cycle comprises five control cycles. The initial CPU utilization reference is set to be 0.62. Based on this setting, FAUC computes an initial fusion threshold T, which is very low as shown in the top sub-figure in Figure 3.5. As a result, the noise occasionally exceeds the fusion threshold causing a false alarm rate of about 5%. At the end of the first optimization cycle, i.e., the 5th control cycle, FAUC computes the optimal fusion threshold T_{opt} based on the estimated target/noise parameters. As $T_{\text{opt}} > T$, there exists an opportunity to reduce system false alarms. FAUC thus computes a new utilization bound of 0.38 based on T_{opt}, which causes the controller to increase T. The bottom two sub-figures in Figure 3.5 show that the measured utilization quickly converges to the new reference and system error rate drops to zero. When the noise energy is increased at the 12th control cycle, the CPU utilization increases accordingly because false alarms

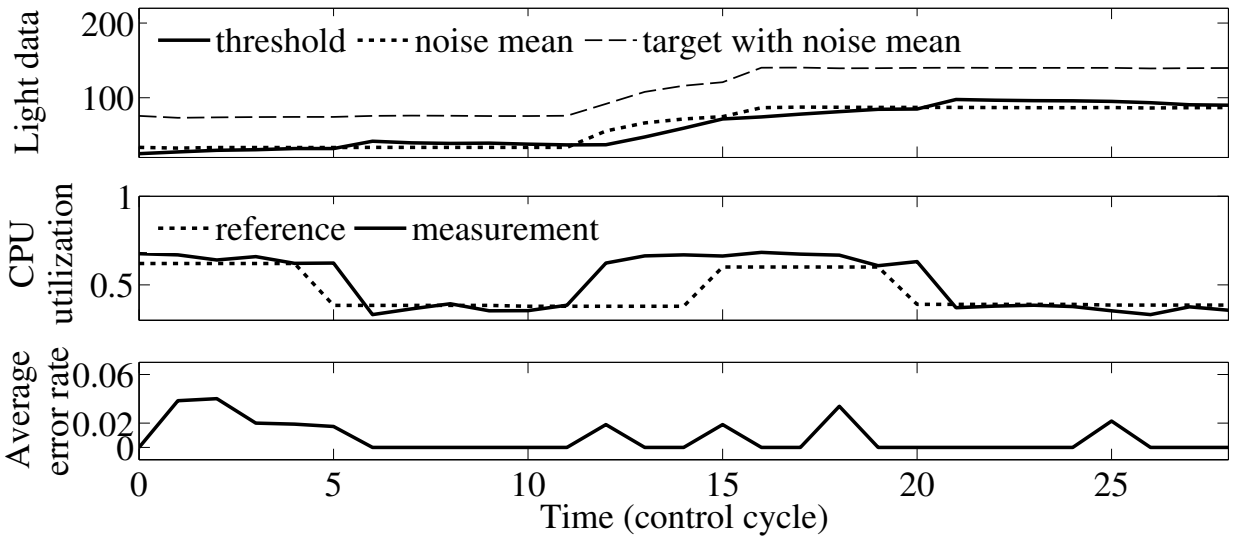


Figure 3.5 The temporal evolution of the light spot detection in dynamic environments.

from low-end sensors trigger extra image processing tasks. FAUC then attempts to lower the utilization by increasing T. At the next optimization cycle, i.e., the 15th control cycle, FAUC estimates the target/noise parameters and computes a new T_{opt} that is lower than T. As the utilization reference (about 0.6) that corresponds to the new T_{opt} is still lower than the initial bound 0.62, FAUC increases the reference to reduce false alarms, as discussed in Section 3.5.4. At the next optimization cycle, i.e., the 20th control cycle, T_{opt} exceeds the current T. FAUC then lowers the utilization reference again, which frees more CPU resources. The above results demonstrate several salient features of FAUC when it operates in dynamic environments. First, it yields satisfactory control performance as the CPU utilization quickly converges to the reference even when false alarms introduce unpredictable system workloads. Second, FAUC can effectively adapts the utilization reference to minimize system error rate.

3.6.2.3 Effectiveness

We now compare the performances of various approaches in seven groups of experiments with a variety of target and noise dynamics. In the first four groups of experiments, we

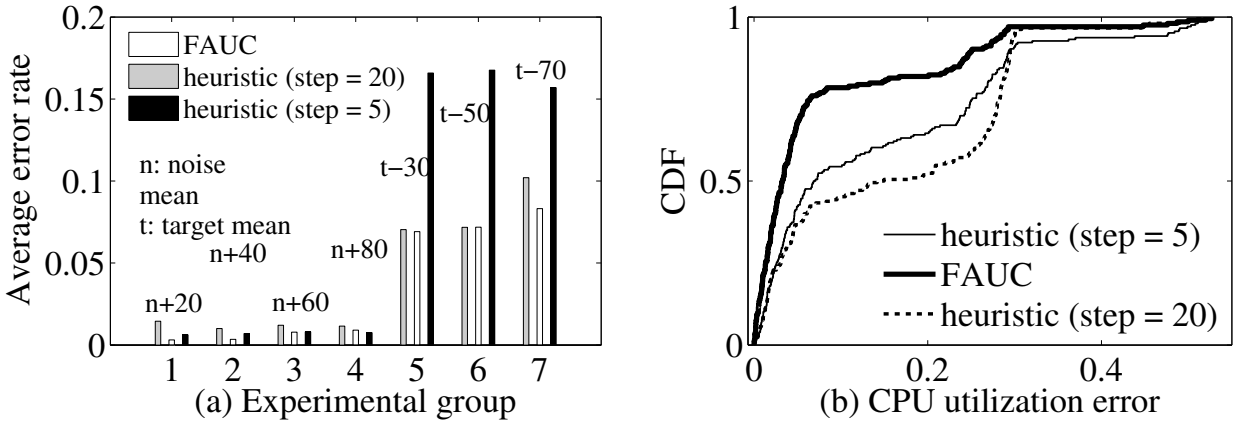


Figure 3.6 (a) The average detection error rate under different target/noise dynamics. (b) The CDF of the absolute CPU utilization error.

increase the noise mean from 20 to 80 with a step of 20. In the last three groups of experiments, we decrease the target signal by 30, 50 and 70, respectively. Figure 3.6 (a) shows the average detection error rate in different group of experiments. We can see that all three approaches can maintain small detection error rates when the noise increases. Although the increasing noise causes more false alarms for low-end sensors, the false alarms can be effectively filtered out by the image-based target detection. When the target signal decreases, all three approaches yield higher error rates. In particular, the heuristic algorithm with a step size of 5 misses about 16% targets. When the step size is 20, it has fewer misses because the controller settles faster. FAUC achieves the minimal error rate under all settings.

Figure 3.6 (b) plots the CDF of the utilization errors in the seven groups of experiments. We can see that FAUC significantly outperforms the heuristic algorithm with various settings. In particular, about 80% of errors of FAUC fall within 10%. In contrast, the heuristic algorithm does not exploit the relationship between the control input and the controlled variable and hence yields considerable steady-state errors. We can conclude from Figure 3.6 (a) and (b) that FAUC yields excellent control performance while maintaining satisfactory fidelity even when target/noise have dynamic characteristics.

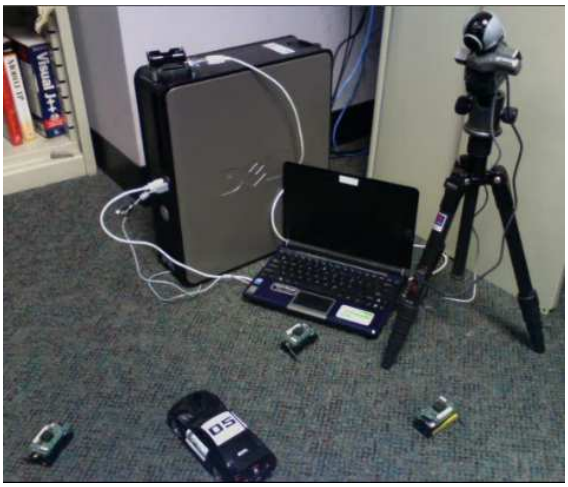


Figure 3.7 Testbed for acoustic target detection. 3 Iris motes and a webcam detect the moving toy car.

3.6.3 Acoustic Target Detection

In the second set of testbed experiments, we use three Iris motes [16] and a webcam to detect a radio-controlled toy car. Figure 3.7 illustrates the setup of the experiments. The cluster head is connected to the base station laptop. The microphone of MTS300 sensor board [16] on Iris mote samples at 100 Hz to detect acoustic signals from the toy car. Each mote calculates the acoustic signal energy every 50 samples and transmits to the cluster head every 500 milliseconds. The cluster head fuses the measurements received from the Iris motes and activates the webcam to capture an image if a positive decision is made. The base station compares the image with the stored background image to detect the target using the ImageMagick tool [65]. Note that another webcam connected to another computer records the ground truth information. It is challenging to detect the toy car using the low-cost Iris motes because of the significantly dynamical characteristics of the toy car. Specifically, as the toy car moves through the surveillance region quickly, the acoustic signal emitted by the car varies significantly with the car's location and speed.

Figure 3.8 shows the evolution of system performance over 16 control cycles. The CPU utilization reference is set to be 0.3. We intentionally adopt such a low reference to study the

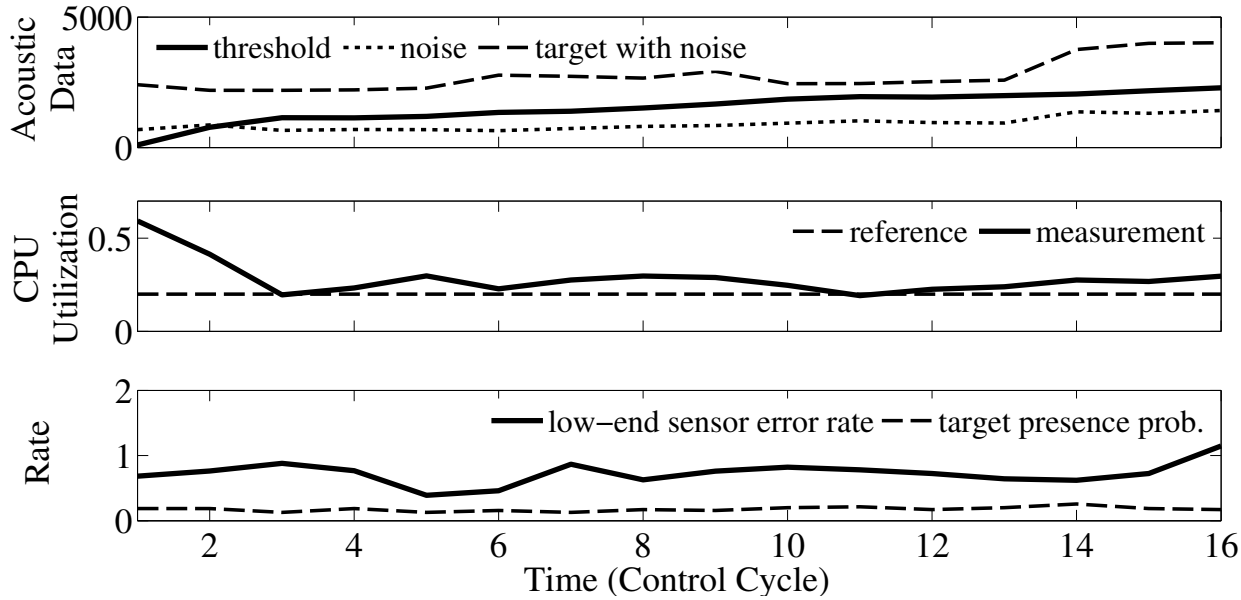


Figure 3.8 The temporal evolution of the acoustic target detection in dynamic environments.

trade-off between utilization and detection performance. Each optimization cycle comprises four control cycles and each control cycle comprises 70 detections. The toy car moves along a circular path that crosses the surveillance region. The target appearance probability varies during the experiment as shown in the bottom sub-figure of Figure 3.8. The fusion threshold is initially set to a low value, which leads to many false alarms and image processing tasks. This in turn causes higher CPU utilization which exceeds the required CPU utilization reference. In response, FAUC increases the fusion threshold and hence the CPU utilization quickly drops to the reference at the second control cycle. Afterward, the CPU utilization is well controlled at the reference. The slightly changing target appearance probability leads to slightly changing CPU utilization and fusion threshold. In addition, the car's acoustic signal happens to decrease at the 13th cycle, which causes a higher false alarm rate and may increase the CPU utilization. However, the target appearance probability is slightly reduced at that time so that the CPU utilization does not increase substantially. To adapt to this change, the threshold also slightly decreases. Overall, the results of this experiment show that the system is robust to the variations of target energy and appearance probability.

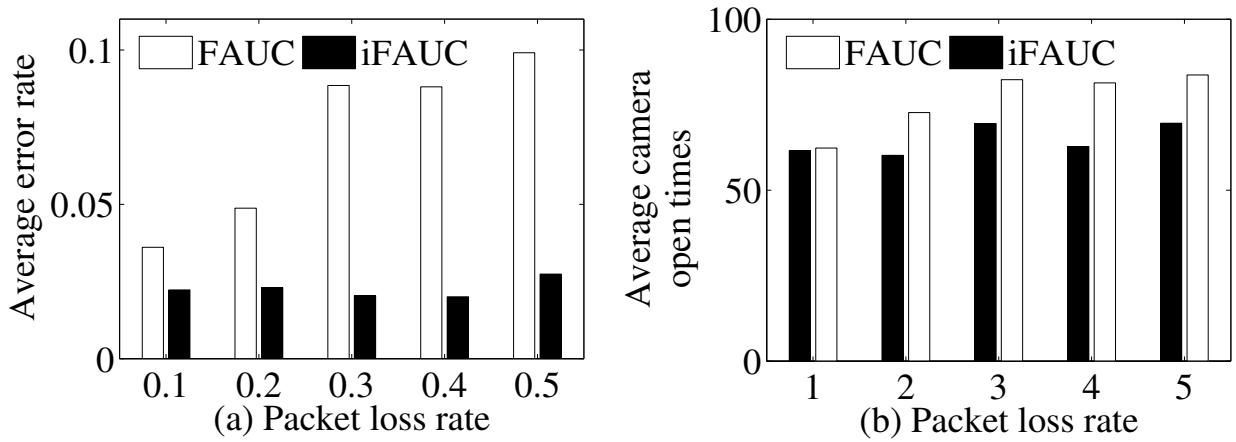


Figure 3.9 (a) Average detection error;(b) Average camera switched-on times.

3.6.4 Impact of Packet Loss

As shown by the analysis of impact of packet loss on FAUC in Section 3.4.2, we can improve the performance by increasing the number of sensor nodes or number of samplings per detection in the condition of the packet loss. Based on this result, we design an improved FAUC (iFAUC) and evaluate it in light detection experiment. In FAUC, each sensor takes one sample every 250 milliseconds and sends to the cluster head for data fusion. On the contrary, each sensor in iFAUC samples every 25 milliseconds and transmits to the cluster head once every 10 samples. Taking each sample as a packet, we apply different packet loss rates to iFAUC and FAUC for evaluation. The experiments are conducted the same way as described in the Section 3.6.2. Figure 3.9 (a) shows the average detection error rate comparison between iFAUC and FAUC under the different packet loss rates. Although both methods achieve low average detection error rate, iFAUC significantly outperforms FAUC under all the packet loss rates. Due to the better accuracy, iFAUC has more chances to reduce the CPU utilization reference, thus releasing more CPU resources. Comparison of camera switched-on times between iFAUC and FAUC shows iFAUC always switch the camera on less frequently than FAUC in all packet loss rates (Figure 3.9 (b)). As the camera switched-on times is proportional to the CPU utilization in Eqn. (3.6), iFAUC outperforms FAUC again

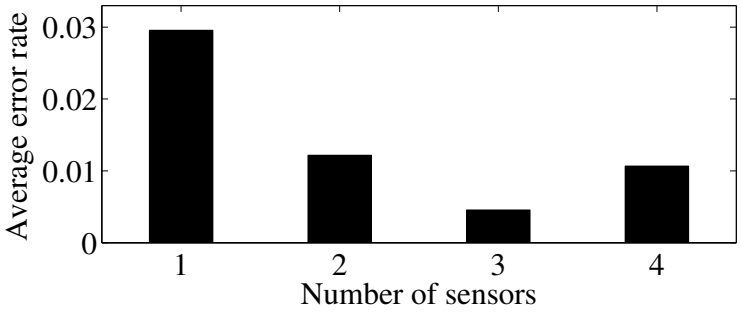


Figure 3.10 Average error rate under different number of sensor nodes.

regarding average CPU utilization.

3.6.5 Impact of the Number of Sensor Nodes

In CPSs that adopt sensor data fusion, the number of sensors may affect the performance of the system. Based on the light detection experiment, we conducted experiments with different number of sensors to evaluate the resulted performance. We vary the number of sensor from one to four and conduct experiment described in Section 3.6.2. From Figure 3.10, the average error rate doubles when there is only one sensor compared to two sensors. We can see from Figure 3.10 that the error rate of four sensors is slightly higher than that of three sensors. In the experiment, the fourth sensor receives the lowest signal-to-noise ratio (SNR). As the data fusion rule adopted in this work treats all sensors with equal weight, fusing the reading from a sensor with low SNR may not be beneficial. We note that if we adopt more effective data fusion rule that accounts for sensors' SNRs [5], system detection performance would not degrade even if low-SNR readings are fused.

3.6.6 Multi-cluster CPSs

We now extend our evaluation to the case of multiple clusters that may exist for large CPSs. In order to assure the real-time performance of the base station whose resources are shared by all clusters, the CPU utilization of the base station shall be carefully maintained

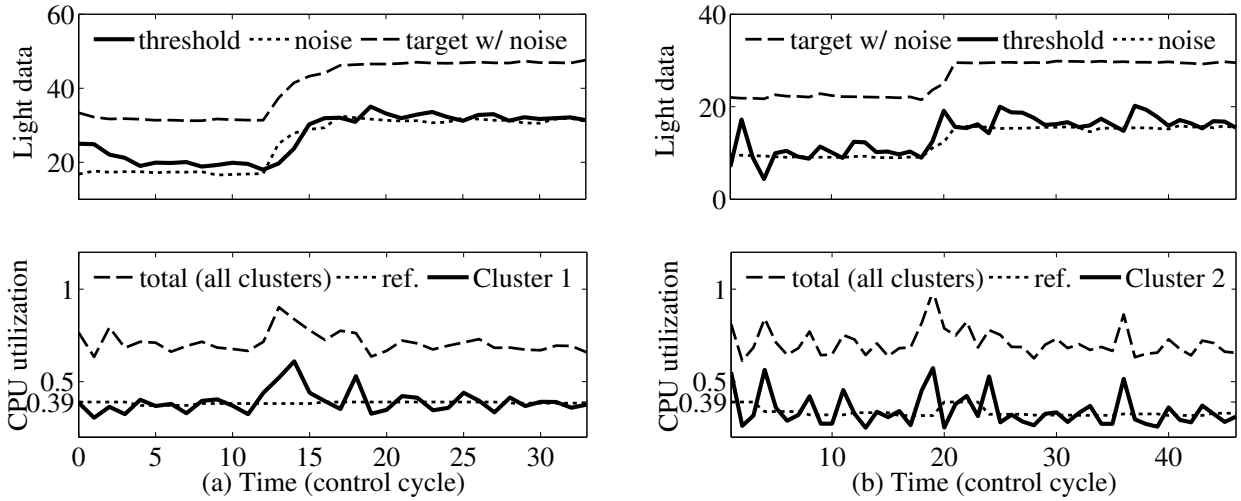


Figure 3.11 The temporal evolution of the light target detection in dynamic environments, where the CPU utilization reference is set to 39%. (a) Cluster 1; (b) Cluster 2.

according to the real-time scheduling strategy. In this experiment, we deploy two clusters, each of which is used to monitor a light spot. Two light spots are generated by computer program, separately, with a target present probability of 50%. One high-quality camera is used for each cluster to confirm the presence of the light spots, respectively. However, we deploy different number of sensor nodes and apply the different control and estimation cycle length for the two clusters. Cluster 1 contains two sensor nodes and comprises 100 samples per control cycle with 400 samples per estimation cycle. On the other hand, Cluster 2 contains only one sensor nodes and comprises 80 samples per control cycle with 320 samples per estimation cycle. We conduct the experiment with changeable noise for both light spots. Based on the real-time scheduling strategy, we set the base station CPU utilization's upper bound to be 78% and then assign 39% utilization for each cluster. Figure 3.11 (a) shows the time evolution for Cluster 1 in light spot detection. It performs similarly as in the previous light spot detection experiment. In the first 12 control cycles, CPU utilization is controlled around 39%. We can see the system tries to reduce the CPU utilization reference for Cluster 1 at the 4th cycle. At the 12th cycle, the noise increases, hence the CPU utilization consumed by Cluster 1 increases due to the higher false alarm rate of low-end sensors. This change on the CPU utilization of Cluster 1 is also reflected in the total CPU utilization. Then the

system quickly adapts the threshold to reduce the CPU utilization of Cluster 1 to the set point within three cycles. Accordingly the total CPU utilization of base station reduces to a lower level than the reference 78%. At the 16th cycle, the estimated T_{opt} requires higher CPU utilization. However this higher utilization is still lower than the original user setpiont for Cluster 1. As a result, the system increases the CPU utilization reference accordingly. The system then controls the CPU utilization of Cluster 1 on this reference for the rest of time. Similarly, Figure 3.11 (b) shows the time evolution for Cluster 2. As there is only one sensor node in the cluster, it is more likely to have higher false alarm rate and missing probability. However, from the 5th, 20th, 25th and 37th cycle, we can see the system adapts to the CPU utilization changes very quickly by adjusting the threshold under the environment dynamics. From the time evolutions of the two clusters, we can see that the individual CPU utilization of each cluster is well controlled at the reference, and the average CPU utilization of the base station is always quickly controlled to around 70%, which meets the real-time requirement.

3.7 Trace-Driven Simulations

In addition to the testbed experiments, we also conduct trace-driven simulations to extensively evaluate the performance of FAUC under a wide range of settings including network size and signal-to-noise ratio (SNR).

3.7.1 Simulation Methodology and Settings

The data traces used in the simulations were collected in the DARPA SensIT military vehicle localization experiment [66], where 75 WINS NG 2.0 nodes are deployed to localize Amphibious Assault Vehicles (AAVs) driving through a three-way intersection. We refer to [66] for detailed setup of the experiment. The dataset used in our simulations includes the ground truth data and the acoustic time series recorded by 17 nodes at a frequency of 4960 Hz. The ground truth data include the positions of sensors and the trajectories of AAV

runs recorded by GPS devices. However, as the vehicle localization experiment [66] only comprises a limited number of AAV runs, the dataset cannot be readily used to evaluate the detection performance of FAUC. We note that, in order to evaluate the average error rate, the system needs to conduct enough detections in both cases of target presence and absence. To address this issue, we reuse the data traces as follows. In the simulations, we assume that the target appears at a fixed location within the deployment region. For each detection, an AAV run is randomly selected. In the presence of target, a sensor's measurement is set to be the sum of random noise and the real measurement when the AAV in the selected run is closest to the fixed location. In the absence of target, the sensor's measurement is only set to the random noise. Such a simulation methodology accounts for many realistic affecting factors such as terrain and the dynamical characteristics of the vehicle. Moreover, we can evaluate the performance of FAUC under different SNRs by changing the parameters of the noise generator. The target appearance probability P_a is set to be 50%. As there is no extra high-quality sensor such as camera in the SensIT experiment [66], we use a pseudo camera in the simulations, which generates random detection results based on the ground truth. The pseudo camera's false alarm rate and missing probability, i.e., P_{FH} and P_{MH} are set to be 2%. If the pseudo camera is activated in a detection, the CPU utilization in the detection follows a normal distribution with mean of 40% and standard deviation of 10%. The utilization bound u_s is set to be 35%.

3.7.2 Simulation Results

We first evaluate the temporal evolution of FAUC in the case of changing noise. The change of noise mean over time is shown in Figure 3.12. The standard deviation of noise is initially set to be 0.1 and increased by 0.1 at the 200th and 350th control cycle. As we select a relatively high utilization upper bound, the low-end sensors are allowed to raise more false alarms and the system can minimize the error rate. As a result, from Figure 3.12, we can see that the calibrated threshold is around the noise mean. From the figure, we can also

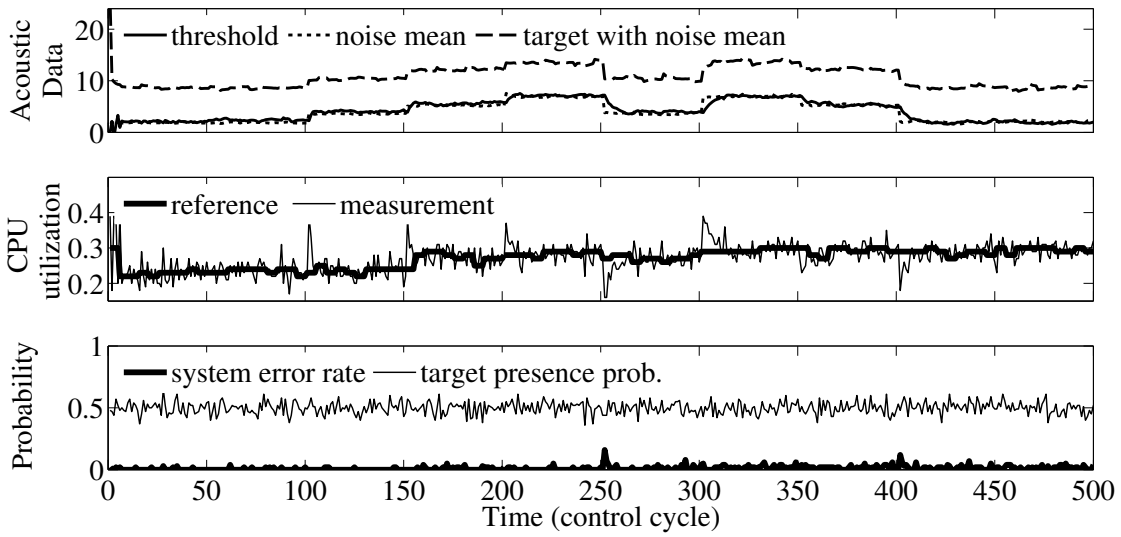


Figure 3.12 The temporal evolution of the vehicle detection in the presence of changing noise (17 sensors).

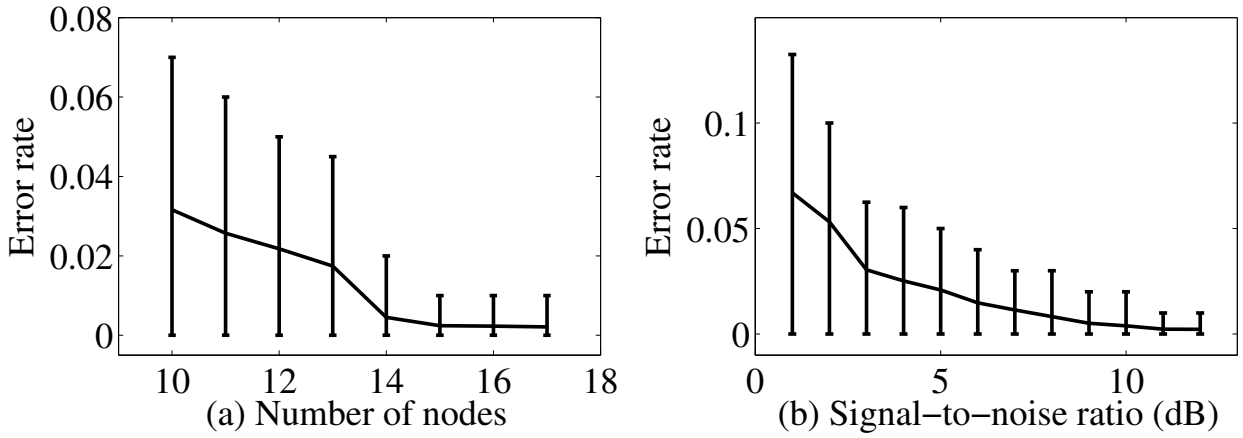


Figure 3.13 (a) System error rate versus the number of sensor nodes; (b) System error rate versus SNR (17 sensors). The error bars are plotted with 5% and 95% quantiles.

see spikes in utilization and error rate, which are caused by the changes of noise profile. However, the system can adapt to these changes within 20 control cycles.

In the second set of simulations, we evaluate the impacts of the number of sensors and SNR on the performance of FAUC. Figure 3.13(a) shows the system error rate versus the number of sensors used in the simulations. Other settings are the same as in Figure 3.12. We can see from the figure that the system detection performance increases with the number of

sensors. Figure 3.13(b) shows the system error rate versus SNR. Note that SNR is defined as $10 \log_{10} S/\sigma$. We can see that the system detection performance increases with SNR. For both Figure 3.13(a) and (b), the CPU utilization can be controlled to meet the upper bound u_s. Therefore, FAUC can work in wide range of settings of network size and SNR.

3.8 Conclusion

In this chapter, we propose a holistic approach called *Fidelity-Aware Utilization Controller* (FAUC) to address both fidelity and timeliness requirements of Wireless Cyber-physical Surveillance (WCS) systems. FAUC integrates data fusion with feedback control to enforce CPU utilization upper bound although system workloads vary significantly at runtime because of stochastic detection results. FAUC also optimizes system fidelity and adjusts the control objective of CPU utilization adaptively in dynamic environments. We have implemented FAUC on a small-scale WCS testbed consisting of TelosB/Iris motes and cameras. Our experiments on light and acoustic target detection show that FAUC can achieve robust fidelity and enforce desired utilization bounds in the presence of significant variations of target/noise characteristics and unreliable wireless links. Moreover, extensive simulations based on real acoustic data traces collected in a vehicle surveillance experiment show that FAUC can work in a wide range of settings including network size and SNR.

CHAPTER 4

HIGH-FIDELITY TEMPERATURE PREDICTION FOR DATA CENTERS

4.1 Introduction

Data centers have become a critical computing infrastructure in the era of cloud computing. Research has shown that more than 23% of data center outages are caused by servers' self-protective shutdowns because of overheating [21]. For instance, Wikipedia, a popular online encyclopedia, went down on March 24th, 2010 because of server overheating [67]. Currently, the common practice to prevent overheating is to overcool the server rooms. Due to such a conservative strategy, the cooling systems consume excessive power, which takes up to 50% of the total energy consumption in many data centers [20].

Various thermal management schemes for improving the energy efficiency of data centers rely on real-time and high-fidelity ambient temperature monitoring [24][68][25][41]. However, the existing data centers typically have limited temperature monitoring capability. For instance, the servers in many legacy data centers are equipped with motherboard temperature sensors only, which cannot accurately measure the ambient temperature [49]. Recently, Wireless Sensor Networks (WSN) has been identified as an ideal enabling technology for thermal monitoring in data centers due to several of its salient advantages, including sufficient coverage and no reliance on additional network and facility infrastructure in already complicated data center environments. However, the precise temperature monitoring alone may not be sufficient for preventing unexpected server shutdowns, because various thermal emergencies may quickly cause overheating. Therefore, it is important to design temperature prediction systems to forecast potential overheating events such that the thermal actuators (e.g., the cooling systems) have enough time to react. Moreover, the prediction system can also send alarm messages to the data center administrators for human intervention if nec-

essary. In addition to the overheating alarming, proactive thermal control systems for data centers can be built upon the real-time temperature prediction to improve the energy efficiency and hardware reliability. Specifically, with accurately predicted temperature evolution in the near future, the cooling systems can safely increase the temperature setpoints without leading to server overheating and thus improving energy efficiency in data centers [24]. Moreover, the proactive control can reduce the transient temperature variation, which is shown to contribute to the hardware failure rates in data centers [69].

However, several major challenges must be addressed in designing a real-time and high-fidelity temperature prediction system. First, data centers are complex Cyber-Physical Systems (CPS) whose thermal characteristics are inherently affected by both physical (e.g., complex airflows and server deployment layout) and cyber (dynamic server workloads) factors. Therefore, prediction algorithms designed based on simplified physical and cyber models would not yield satisfactory performance. Second, the number of locations where the temperatures are of particular interest (e.g., the inlets and outlets of all servers) is often large, making it prohibitively expensive to deploy a sensor at each of such locations. It is challenging to reduce the number of sensors while maintaining satisfactory temperature prediction accuracy. Third, it is desirable to decouple the prediction system and the computing resources of the monitored data center. This design not only avoids the potential interruptions to the prediction system due to unexpected server shutdowns, but also improves the system portability. To this end, the prediction system must operate on limited computing resources while maintaining high prediction fidelity.

To address the above challenges, we propose a novel cyber-physical approach that integrates *in situ* wireless sensors, transient Computational Fluid Dynamics (CFD) modeling [70] and real-time data-driven prediction algorithms. CFD is a widely adopted numerical tool that can simulate the future evolution of temperature distribution of data centers. However, without accounting for runtime behaviors of the data center, CFD has highly variable accuracy, poor scalability, and prohibitive computational complexity, which make it ill-suited

for high-fidelity online prediction. To overcome these limitations, our approach leverages the realistic physical thermodynamic models provided by CFD to generate simulated temperature distribution, which is then integrated with the real sensor measurements to train the real-time prediction algorithm. Moreover, unlike traditional thermal management solutions where CFD is used in an open-loop fashion, our approach utilizes real sensor feedback to calibrate the CFD simulation results. Our approach has the following advantages.

First, by leveraging transient CFD modeling, our approach ensures the fidelity of predicting many rare but critical thermal emergencies (e.g., cooling system failures) that may not be captured by real sensors in operational data centers. Second, by integrating realistic physical CFD models and real sensor measurements, our approach only requires prediction algorithms with low computational complexity. This enables the development of portable thermal management systems that do not rely on the infrastructure of the monitored data center. Finally, as CFD can simulate the temperature at the uninstrumented locations, our approach significantly reduces the number of sensors without leading to substantial degradation of prediction fidelity.

We implemented our temperature prediction system using 36 wireless motes equipped with temperature and airflow sensors. We deployed our system in a single-rack testbed, composed of 15 running servers, and a small-scale production data center testbed composed of five racks and 229 servers. The extensive evaluation shows that our system can predict the temperature evolution of servers with highly dynamic workloads at an average error of 0.52 °C, within a duration up to 10 minutes.

4.2 Problem Statement and Approach Overview

4.2.1 Problem Statement

The temperatures at the inlet and outlet of a server are critical thermal conditions for the operation of the server. The inlet temperature is often defined as the server's operating

ambient temperature, which is required to be within a small range (e.g., 15°C to 27°C [71]). The outlet temperature characterizes the amount of heat that needs to be removed by the air conditioners (ACs) to avoid overheating. Therefore, in this work, we aim to predict the temperatures at the inlets and outlets of the servers of interest. The set of these temperatures is referred to as *temperature distribution*. The accurate prediction of the temporal evolution of temperature distribution is challenging because of the complex thermal and air dynamics in data centers. Specifically, the dynamic workload and other server activities, e.g., disk and network access, generate different amount of heat over time. The heat is dissipated by the extremely complex airflows, which are driven by server internal fans and ACs. Moreover, the heat dissipation is highly dependent on the racks and other physical structures in a data center.

Our temperature distribution forecasting system is designed to meet the following objectives. (1) ***High fidelity***. We aim to achieve high prediction fidelity with about 1°C error bound. This requirement ensures that the predicted temperature will not trigger excessive false alarms of overheating or miss real overheating events. Moreover, as shown in [25], an 1°C increase of the maximum server inlet temperature can lead to 10% higher cooling costs. Therefore, high prediction fidelity allows servers to operate with less conservative temperature setpoints, improving the energy efficiency of data centers. (2) ***Long prediction horizon***. The system should achieve satisfactory prediction accuracy in a considerably long time duration (e.g., 10 minutes), referred to as *prediction horizon*, into the future. We focus on providing a prediction horizon in the order of minutes. This is motivated by the fact that it usually takes up to several minutes to reach the overheating temperature [72] in the thermal emergencies (e.g., excessive server overload or AC failure). This provides enough time for the thermal actuators (e.g., ACs) to prevent overheating as well as for the data center administrators to take necessary intervention. However, a longer prediction horizon often requires the sacrifice of prediction fidelity. (3) ***Full coverage of thermal conditions***. Our approach is designed to predict the temperature distribution under normal working condi-

tions of the data center, in which overheating is mainly due to high workload, as well as the abnormal and emergency situations (e.g., AC failures) that can lead to catastrophic consequences. (4) ***Timeliness and low overhead***. To enable prompt actuation, the prediction should be performed in an online fashion with tight real-time requirement. The overhead of the prediction system should be affordable to low-end servers, desktop computers or even embedded computing devices, such that the system can be easily deployed and operated in a noninvasive fashion, without relying on the infrastructure of the monitored data center.

Computational Fluid Dynamics (CFD) [70] is a widely used tool to guide the data center layout and cooling system design. The predictive nature of CFD also allows the user to simulate the future evolution of temperature distribution. However, CFD has the following two major limitations. First, the accuracy of CFD highly depends on how well the adopted thermodynamic models reflect the realities. Considerable expertise and labor-intensive fine tuning are often required in the modeling process, which makes fine-grained CFD simulation intractable for medium- to large-scale data centers. Moreover, CFD often has considerable temperature modeling errors that range from 2°C to 5°C [51, 73]. This often leads to highly conservative temperature setpoints, resulting in excessively high power consumption of cooling systems in a data center. Second, CFD has high computational complexity that prohibits it from temperature prediction at run time. For instance, it can take 5 minutes on a high-end 12-core server to simulate 5 seconds of the temperature evolution of a rack equipped with 15 servers. As a result, CFD alone is not sufficient for high-fidelity and real-time temperature prediction in data centers.

4.2.2 Approach Overview

Our approach integrates *in situ* wireless/on-board sensors, transient CFD simulation, and real-time prediction modeling to achieve high-fidelity temperature prediction in data centers. The sensors collect environment temperature and airflow velocity data at various physical locations (e.g., server inlets, outlets, fans, raised floor tiles, AC cold air inlets

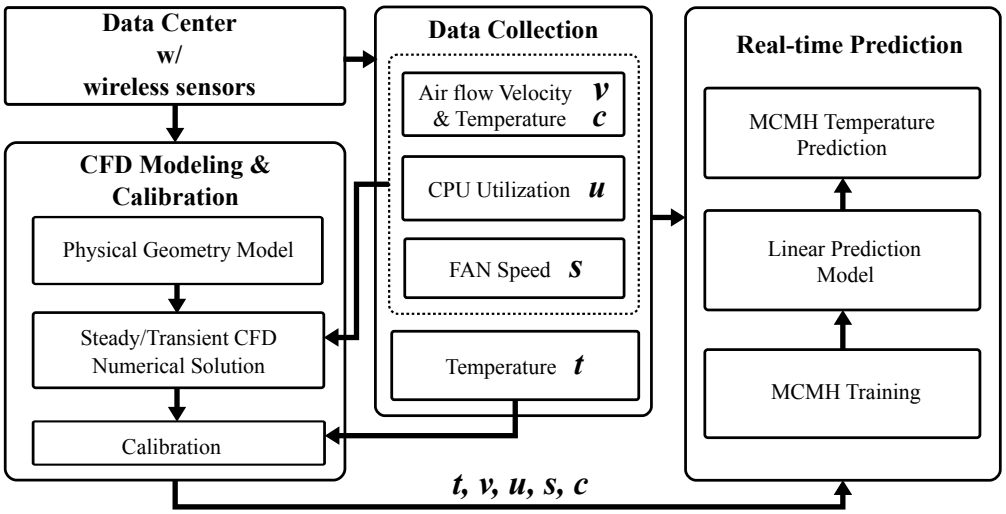


Figure 4.1 Prediction system architecture.

and hot air outlets). The collected data are then used to train the time series prediction models. To ensure the modeling fidelity, multiple models are used to capture the normal and various abnormal working states such as the failure of different AC units. A challenge of such training-based approach is to collect sufficient data sets that cover various thermal conditions, especially for the abnormal and emergency situations that rarely happen, but have catastrophe consequences. The controlled experiments for generating these situations are often intrusive or even harmful in operational data centers. To address this issue, we leverage transient CFD simulation, which is capable of simulating any overheating condition, to generate additional training data for the prediction models. This approach avoids running the computationally intensive CFD in an online fashion, yet preserves realistic physical characteristics of the training data. The CFD simulation results are also calibrated by runtime sensor measurements. As a result, our approach only requires moderately accurate CFD modeling, significantly reducing the efforts of CFD model tuning. Another advantage of our approach is that, by integrating CFD simulation and runtime sensor measurements, the number of required sensors is significantly reduced, leading to lower deployment costs and less intrusiveness to the production data centers.

Figure 4.1 illustrates the architecture of our prediction system. The system consists of

three major components. (1) ***Data collection***. This component periodically collects the measurements of CPU utilization and server fan speed through on-board sensors, while using a wireless sensor network to collect the measurements of temperatures and airflow velocities. The historical measurements are used to calibrate the CFD modeling and train the prediction models, while the runtime sensor measurements are fed to the real-time prediction component to predict temperatures. (2) ***CFD modeling and calibration***. In addition to the sensor measurements, a key feature of our system is to leverage the transient CFD simulation to compute the fine-grained temperature evolution, which assists the training of the time series prediction models. Our system uses *in situ* sensor measurements to calibrate the transient CFD simulations, and generate calibrated temperature time series data for normal and various abnormal thermal conditions, such AC failures. These results are then fed as the training data to the real-time prediction component. (3) ***Real-time multi-channel and multi-horizon temperature prediction***. The real-time prediction component constructs time series prediction models with training data from both historical measurements and CFD simulations, and outputs the runtime temperature predictions. Although complex non-linear models may achieve good prediction accuracy, they often have high complexity. Our solution uses multiple simple linear models to approximate the complex non-linear thermodynamic laws. For each different major thermal condition (hereafter referred to as *channel*), such as the failure of different AC units, multiple prediction models with different prediction horizons are constructed. Different prediction horizons of our system give administrators more flexibility to implement thermal actuators, e.g., taking appropriate measures in an incremental fashion.

4.3 CFD Modeling and Calibration

In this section, we first briefly introduce the Computational Fluid Dynamics (CFD) and then present a case study of modeling a rack of servers using CFD. The case study helps us understand the major limitations of CFD. We then present an approach to calibrating CFD

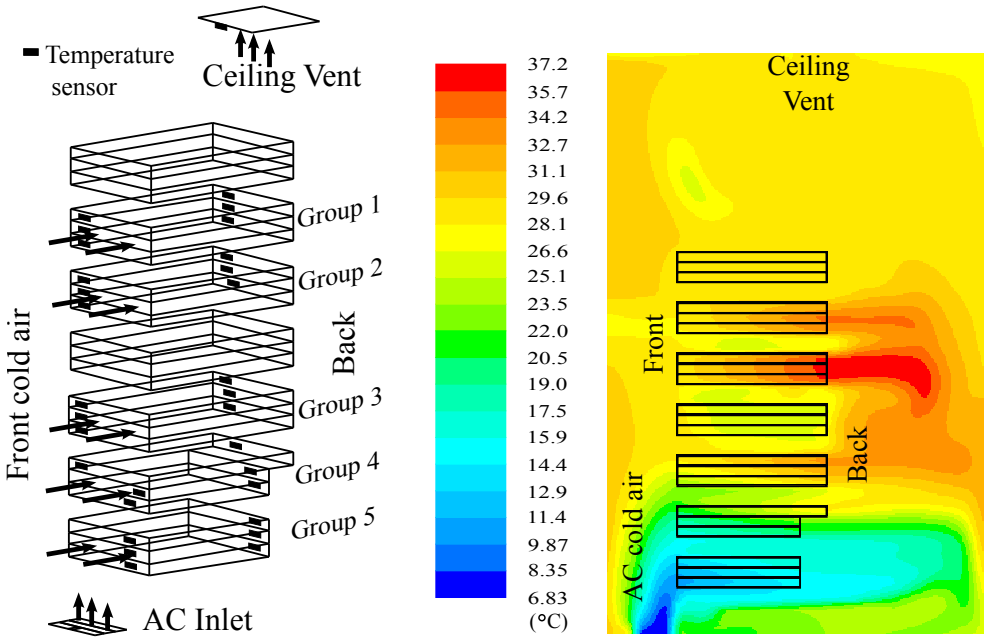
using real sensor measurements.

4.3.1 Background on CFD

CFD is a widely adopted numerical tool to simulate the future temperature evolution. It iteratively solves a system of fluid and heat transfer equations in the form of non-linear partial differential equations under the constraints of mass, momentum and energy conservation. The non-linear nature of these equations and the complex boundary conditions in data center environment (e.g., the physical structures) usually make it impossible to solve these equations analytically. Therefore, CFD typically solves these equations using numerical approaches. Specifically, by dividing the continuous fluid field into small cells, CFD solves the fluid and heat transfer equations in each cell with significantly simplified boundary conditions. The global optimal solution is found iteratively where all cells meet the convergence requirements, giving the steady-state temperature distribution. For the transient simulation, the model is also discretized into small time steps in time domain. At each time step, CFD iteratively finds the global optimal solution, giving the transient temperature distribution. The boundary conditions such as AC airflow temperature, velocity, and power consumption of servers can also be set for each time step with user-defined values (e.g., sensor measurements) such that CFD can simulate any normal or abnormal thermal situations. Therefore, the accuracy of the transient CFD modeling is particularly important for achieving high prediction fidelity.

4.3.2 A Case Study

We now present a case study of using CFD to model a testbed of rack servers, which helps us understand the performance limitations of CFD. Figure 4.2(a) shows the physical geometry of rack server testbed. A total of 15 servers on the rack are grouped into 5 server groups. The detailed settings of the server rack can be found in Section 4.5. For CFD modeling, we use wireless sensors to measure the boundary conditions including the temperatures and velocities of the air discharged by the AC and exhausted by the ceiling



(a) (b)

Figure 4.2 (a) Server geometries with temperature sensor locations; (b) Side view of the steady-state temperature map when the servers in Group 1 and Group 2 are running with full utilization.

vent. A total of 30 sensors are deployed to measure the temperature distribution around the rack. Node 1 to 15 are installed at the outlets of the servers and Node 16 to 30 are installed at the inlets of the servers.

Figure 4.2(b) shows the steady-state temperature map calculated by CFD software (Fluent) when servers in Group 1 and Group 2 are running with full utilization (referred to as Case 1). We can see that the cold air is mostly drawn by the lower servers and the two groups of servers running with full utilization have much higher exhaust air temperatures than other servers. Figure 4.3 plots the sensor readings as well as the temperatures calculated by CFD. We can see that, for Case 1, CFD can accurately predict the steady-state temperature distribution. The root-mean-square error (RMSE) across all sensors is only 0.7°C. The result in Figure 4.3 is achieved by extensively tuning CFD with the help from an expert with 20 years of experience in CFD modeling. For instance, the exhaust airflow of servers, the cooling airflow of AC and the corresponding sensor locations in the CFD physical

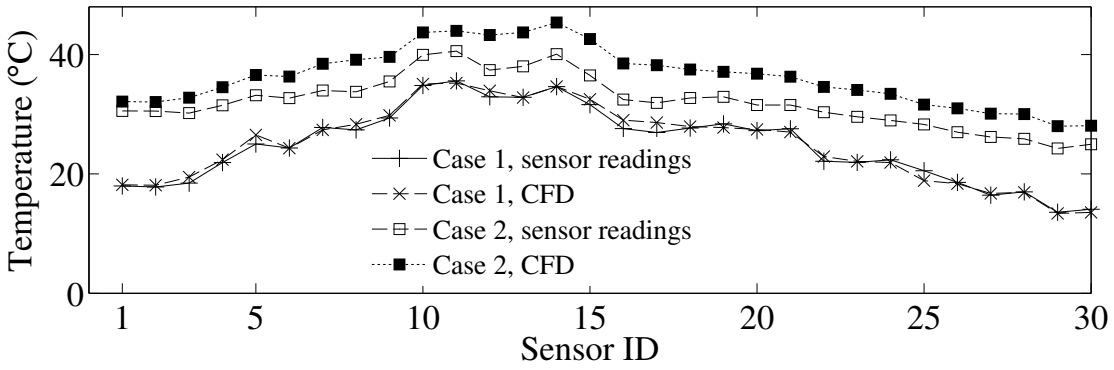


Figure 4.3 Real sensor readings and CFD prediction. Case 1: servers in Group 1 and Group 2 run with full utilization; Case 2: AC failure.

model were carefully adjusted in a number of iterations. We note that such an extensive tuning process is a common practice for constructing CFD for real data centers. We then use the well-tuned CFD to predict the steady-state temperature distribution for the case of AC failure (referred to as Case 2). Figure 4.3 shows that the CFD exhibits considerable errors (RMSE of 4.4°C) in case of AC failure. In addition to the steady-state prediction, we also examine the accuracy of CFD in transient simulation, which is critical for the performance of real-time prediciton. Figure 4.4 shows the temporal evolution at the location of sensor 1 computed by CFD, as well as the real readings from sensor 1. During this period, the CPU utilizations of servers are varied, resulting in highly dynamic temperature at this sensor location. It can be clearly seen that CFD result contains significant biases with respect to the real sensor readings. The major reason of those errors is that CFD does not exactly model the true data center environment and all the important system parameters (e.g., material properties). In practice, it is extremely difficult and labor-intensive to construct a CFD model that is accurate in all thermal conditions. Therefore, to make CFD practical in our prediction system, we discuss in Section 4.3.3 how to calibrate the temperature data simulated by CFD using real sensor measurements collected in the data center. Such calibration significantly reduces the dependency of prediction performance on CFD modeling.

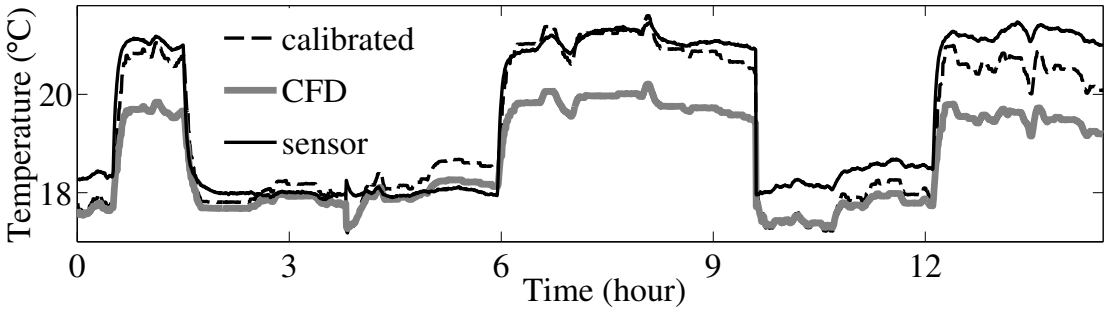


Figure 4.4 Transient temperatures at the outlet of the lowest server (*sensor*: real sensor readings; *CFD*: transient simulation result of CFD; *calibrated*: calibrated transient simulation result of CFD).

4.3.3 CFD Calibration

The results from Section 4.3.2 show that the CFD simulation exhibits considerable errors, particularly in transient-state simulations. To address this limitation, we propose to calibrate the CFD simulation results using runtime sensor measurements. By denoting x_i and y_i as the temperature calculated by CFD and the calibrated temperature at the position of sensor i, the calibration function is given by $y_i = \sum_{k=0}^{K} a_{i,k} \cdot x_i^k$, where K is the order of the calibration function and $a_{i,k}$'s are the coefficients to be learned from training data. By providing real sensor data as y_i, the coefficients $a_{i,k}$'s can be learned based on least square criterion. For each sensor, a calibration function is constructed as long as there are sufficient real sensor measurements collected. As an example, we use the first 3 hours of data in Figure 4.4 to construct the calibration function for each sensor and then use all the data for testing. Figure 4.4 also shows an example of calibrated CFD results with $K = 1$.

4.4 Real-Time Temperature Prediction

This section first presents our approach of predicting the temperature distributions using a linear prediction model, and then discusses the training of the prediction model.

4.4.1 Real-Time Prediction Model

Suppose that wireless temperature sensors are deployed at the inlets and outlets of a total of N monitored servers. The temperature distribution is defined as $\mathbf{T} = [t^1_{\text{in}}; t^1_{\text{out}}; \ldots; t^N_{\text{in}}; t^N_{\text{out}}] \in \mathbb{R}^{2N\times 1}$, where t^n_{in} and t^n_{out} denote the temperatures at the inlet and outlet of the nth server. The prediction model should include the observable variables that significantly affect $\mathbf{T}$ to achieve the accurate prediction of $\mathbf{T}$. In this work, our prediction model accounts for the temperatures (denoted by $\mathbf{C}$) and velocities (denoted by $\mathbf{V}$) of the cold airflow distributed by the ACs, CPU utilization (denoted by $\mathbf{U}$), and internal fan speeds (denoted by $\mathbf{S}$) of all monitored servers. Moreover, the historical temperature distributions also largely affect the temperature distributions in the near future. Therefore, we define the *state* of the monitored servers at a time instance, denoted by $\mathbf{P}$, as the concatenation of $\mathbf{T}$, $\mathbf{C}$, $\mathbf{V}$, $\mathbf{U}$ and $\mathbf{S}$. Specifically, $\mathbf{P} = [\mathbf{T}; \mathbf{C}; \mathbf{V}; \mathbf{U}; \mathbf{S}]$. Our approach can be easily extended to include other observable variables to address various kinds of servers. For instance, hard disc access rates can play an important role in the temperature distribution of file servers.

We assume that each variable in $\mathbf{P}$ can be measured periodically and synchronously by multiple sensors. In the rest of this chapter, the period of data collection is referred to as *time step*. Intuitively, the most recent states significantly affect the current and the future states. In our approach, we predict the temperature distribution at time step $(t+k)$ based on the most recent R states, where $t \in \mathbb{Z}$ denotes current time step and $k \in \mathbb{Z}$ is referred to as *prediction horizon*. For a given k, we assume that the predicted temperature distribution[1] at time step $(t+k)$ is given by $\hat{\mathbf{T}}(t+k) = f_k(\mathbf{P}(t), \mathbf{P}(t-1), \ldots, \mathbf{P}(t-R+1))$, where $f_k(\cdot)$ is the function characterizing the physical law governing the thermodynamic process. However, $f_k(\cdot)$ is often difficult to find in practice due to the high complexity of data center environment. In this work, we propose a linear prediction model to approximate $f_k(\cdot)$, which allows the online real-time prediction at low overhead. Suppose $\mathbf{P} = [p_1; p_2; \ldots]$, define $\mathbf{P}^s = [p^s_1; p^s_2; \ldots]$ where $s \in \mathbb{Z}$. Moreover, we define $\mathbf{q}(t) = [\mathbf{P}(t); \mathbf{P}^2(t); \ldots; \mathbf{P}^s(t)]$

[1]For clarity of presentation, we let $\hat{x}$ denote the *predicted* value of x.

and $\mathbf{x}(t) = [\mathbf{q}(t); \mathbf{q}(t-1); \ldots; \mathbf{q}(t-R+1)]$. According to the Taylor's theorem, the high order Taylor polynomial can well approximate a function. The sth order Taylor polynomial of $f_k(\cdot)$ is given by the linear combination of all the combinatorial terms of the elements in $\mathbf{x}(t)$, which however results in exponential complexity with respect to N. Therefore, we ignore all the cross terms in the Taylor polynomial and adopt the following linear prediction model:

$$\hat{\mathbf{T}}(t+k) = \mathbf{A}_k \cdot \mathbf{x}(t) \tag{4.1}$$

where $\mathbf{A}_k \in \mathbb{R}^{2N \times M}$ and M is the length of $\mathbf{x}(t)$.

Since only the arithmetic calculations are involved in Eq. (4.1), the prediction can be efficiently computed even on low-power embedded platforms. Note that $\mathbf{A}_k$ is different for each prediction horizon k. By setting increasing prediction horizons, Eq. (4.1) predicts the temporal evolution of the temperature distribution. Intuitively, because the correlation between $\mathbf{T}$ and $\mathbf{P}$ decreases over time in a dynamic environment, the prediction with a larger k becomes less accurate.

4.4.2 Model Training

During the normal running state of the data center, the training data are collected from the wireless sensors (e.g., temperature and airflow velocity) or server on-board sensors (e.g., CPU utilization and fan speed). In addition to the sensor data, CFD data are generated for normal and abnormal running states by manually giving different boundary conditions to the CFD transient simulations. For example, different ACs can be shutdown during the CFD transient simulation. Suppose a data set with time step index from 1 to L is collected after system deployment or generated by CFD to train the linear model $\mathbf{A}_k$ for any given k.

We adopt the least square criterion to train $\mathbf{A}_k$. Specifically,

$$\begin{aligned}\mathbf{A}_k &= \arg\min_{\mathbf{A}_k} \sum_{t=R}^{L-k} \|\mathbf{T}(t+k) - \hat{\mathbf{T}}(t+k)\|_{\ell_2}^2 \\ &= \arg\min_{\mathbf{A}_k} \sum_{t=R}^{L-k} \|\mathbf{T}(t+k) - \mathbf{A}_k \cdot \mathbf{x}(t)\|_{\ell_2}^2,\end{aligned}$$

where $\|\cdot\|_{\ell_2}$ represents the Euclidean norm. A desirable property of the above formulation is that the problem can be decomposed to the sub-problems of finding the rows of $\mathbf{A}_k$ separately. The separation can significantly reduce the computation complexity in training. By denoting $\mathbf{a}_j$ as the jth row of $\mathbf{A}_k$ and $t_j(t+k)$ as the jth element in $\mathbf{t}(t+k)$, the sub-problem is

$$\mathbf{a}_j = \arg\min_{\mathbf{a}_j} \sum_{t=R}^{L-k} \big(t_j(t+k) - \mathbf{a}_j \cdot \mathbf{x}(t)\big)^2. \tag{4.2}$$

The closed-form solution of $\mathbf{a}_j$ is $\mathbf{a}_j = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{t}_j$, where $\mathbf{X} = [\mathbf{x}(R), \mathbf{x}(R+1), \ldots, \mathbf{x}(L-k)]^\top$ and $\mathbf{t}_j = [t_j(R+k); t_j(R+k+1); \ldots; t_j(L)]$. The matrix $\mathbf{A}_k$ can be constructed once all its rows are computed.

We now discuss two practical issues related to model training.

4.4.2.1 Regularized Regression

As the variables in the state $\mathbf{P}$ may be affected by the same thermal conditions, they may be correlated with each other. This is called *multicollinearity* in regression analysis. Multicollinearity can lead to inflation of the estimated coefficients in $\mathbf{a}_j$, and hence adversely affects the performance of model training. A common approach to deal with multicollinearity problem is to use regularized regression[74]. Specifically,

$$\mathbf{a}_j = \arg\min_{\mathbf{a}_j} \sum_{t=R}^{L-k} \big(t_j(t+k) - \mathbf{a}_j \cdot \mathbf{x}(t)\big)^2 + \lambda \|\mathbf{a}_j\|^2,$$

where λ is the *regulation factor*. By including the norm of $\mathbf{a}_j$ in the minimization, the inflation of the coefficients in $\mathbf{a}_j$ can be effectively restricted. The closed-form solution of $\mathbf{a}_j$

is $\mathbf{a}_j = (\mathbf{X}^\top \mathbf{X} - \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{t}_j$. As shown in Section 4.6.1.3, with proper settings for λ, the regulation can improve the performance of the model training.

4.4.2.2 Training Data Generation using CFD

A practical issue about training data generation using CFD is how to generate sufficient training data to ensure that the trained model well captures the underlying thermal dynamics and hence delivers accurate predictions. A naive solution is to generate training data to fully cover all possible thermal conditions. For instance, to generate training data for the channel that addresses AC failure, the naive solution needs to simulate the AC failure under all possible initial states. However, a major challenge here is that the state $\mathbf{P}$ has combinatorial complexity. Due to the high dimensionality of $\mathbf{P}$, enumerating all possible initial states in the generated training data will incur extremely high computation overhead. Intuitively, a certain amount of simulated data traces with initial states that sparsely span in the state space may be sufficient to train the linear regression model. Based on this intuition, we generate data traces with random initial state $\mathbf{P}$ that is uniformly distributed within its possible range. The number of generated data traces should be large enough to prevent overfitting. In Section 4.6.1.6, our experiments show that 10 transient data traces generated by CFD are sufficient for training the model characterizing AC failure for a rack of 15 servers.

4.4.3 Dimension Reduction

A monitored temperature instance (i.e., an element of the temperature distribution $\mathbf{T}$) may not be strongly correlated with every other variable in the state $\mathbf{P}$. For small-scale deployments such as a rack, the dimension of the state $\mathbf{P}$ is limited. With sufficient training data, the regression result can accurately characterize the correlations between the monitored temperature and every other variable in $\mathbf{P}$. However, for large-scale deployments such as a server room with many racks, the dimension of $\mathbf{P}$ is high. For instance, on our small-scale production data center testbed (cf. Section 4.6.2), the dimension of $\mathbf{P}$ is 229. As a result, the

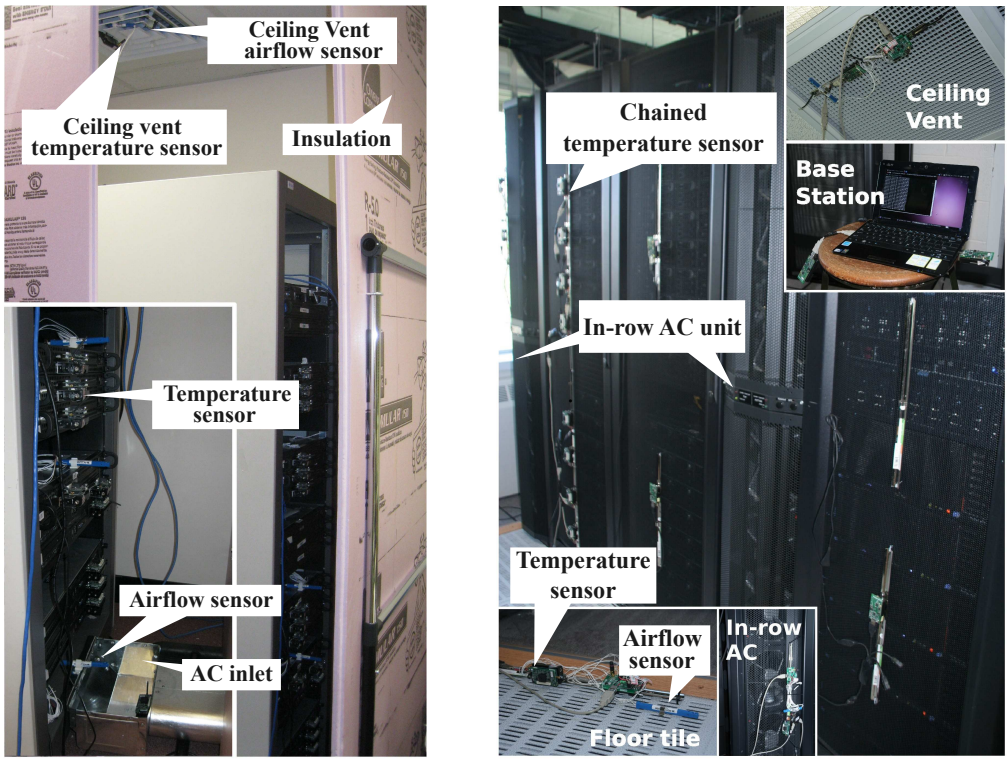
regression is prone to overfitting, leading to poor prediction performance. Therefore, before constructing the prediction model in Eq. (4.1), it is desirable to choose a subset of variables in $\mathbf{P}$ that are most correlated to the monitored temperature to increase the prediction accuracy. In this work, we present a dimension reduction approach, which adopts the *partial correlation* metric [75] to rank the variables in $\mathbf{P}$ regarding their correlation with the monitored temperature. We note that other approaches may also be applicable to the dimension reduction problem [76]. Our approach performs dimension reduction based on the variable ranking for each monitored temperature separately. Specifically, let n denote the dimension of state $\mathbf{P}$, and p_i denote the ith element of $\mathbf{P}$. To compute the partial correlation of the monitored temperature t_j and a variable p_i, we first construct the linear regression models for predicting t_j and p_i based on the remaining variables in $\mathbf{P}$, i.e., $\{p_1, \cdots, p_{i-1}, p_{i+1}, \cdots, p_n\}$, using the approach described in Section 4.4.2. Let $\hat{t}_j$ and $\hat{p}_i$ denote the predictions based on the remaining variables in $\mathbf{P}$. The partial correlation, denoted by $\rho(t_j, p_i)$, is given by the correlation coefficient of the errors in predicting t_j and p_i using remaining variables, i.e.,

$$\rho(t_j, p_i) = \mathrm{r}(t_j - \hat{t}_j, p_i - \hat{p}_i),$$

where $\mathrm{r}(\cdot)$ measures the correlation coefficient. Therefore, the $\rho(t_j, p_i)$ measures the partial correlation between t_j and p_i at a particular horizon while the effects from all other input variables are removed. After calculating the partial correlations $\{\rho(t_j, p_i) | i \in [1, n]\}$, we choose a subset of variables in $\mathbf{P}$ with the *highest* partial determination coefficients (i.e., $\rho^2(t_j, p_i)$). With the chosen variables, we construct the linear regression model in Eq. (4.2), where the elements of $\mathbf{a}_j$ corresponding to the unselected variables are set to be zeros. Note that the dimension reduction result varies with prediction horizon k.

4.5 System Implementation and Deployment

We have implemented the proposed system and deployed it on two testbeds. Now we first describe the set-up of the two testbeds, and then discuss the system implementation.



(a) (b)

Figure 4.5 Testbeds. (a) Single-rack testbed; (b) Prodution testbed (HPCC)

4.5.1 Testbeds and Sensor Deployment

Our first single-rack testbed, shown in Figure 4.5(a), consists of a rack of 15 1U [2] servers in a 5 × 6 square feet room insulated by foam boards. Two types of servers (4 DELL PowerEdge 850 nodes and 11 Western Scientific nodes), are placed on the rack. The rack is placed directly under an infrastructure ceiling vent that exhausts the hot air out of the room. A portable AC made by Tripp Lite, Inc. (model SRCOOL12K) is placed out of the room. It delivers cold air through the AC inlet located at the bottom of the room in front of the rack, which is consistent with the cooling airflow of popular raised floor cooling design. On the rack, the 15 servers are grouped every three servers with a 2U distance between every adjacent two groups. A total of 15 Iris [16] temperature sensors are mounted with brackets at the inlets of the 5 group of servers, and another 15 temperature sensors (8 Iris

[2]U is the unit of the height of a server, which is 1.75 inches.

and 7 TelosB [16]) are mounted with brackets at the outlets of these servers. At the ceiling vent, a temperature sensor (TelosB) is mounted with bracket and a F333 airflow velocity sensor [77] is taped to face the exhausting airflow. To monitor the AC cold airflow, we place a temperature sensor (Iris) in the AC inlet register and tape a same airflow velocity sensor in front of the register. This small testbed allows us to study the fine-grained thermal dynamics of a single rack. Moreover, by controlling the AC system, the testbed can emulate various thermal emergency scenarios.

Figure 4.5(b) shows the second testbed in a server room of High Performance Computer Center (HPCC) at Michigan State University. The testbed consists of 229 servers with 2016 CPU cores on five server racks. Those racks are arranged in two rows with a cold aisle between them. One row of racks is shown in Figure 4.5(b). In addition to the raised floor cooling system which blows cold air vertically from the floor tile into the cold aisle, two in-row AC cooling units are installed between the racks for each row, which produce major cold air at different heights and generate significant side-to-side airflow. To prevent the major hot air recirculation, two pieces of glass wall are installed at the end of the cold aisle. We chain the sensors and mount them at both the front and rear doors of the server racks to monitor the inlets and outlets temperatures, respectively. For one rack, we deploy 8 sensors evenly to monitor the server inlets and 8 sensors to monitor the server outlets, respectively. For other racks, we mount one or two sensors to monitor the server inlets and outlets at different heights. We monitor two out of four in-row AC units by mounting a bundle of temperature sensor and airflow sensor at cold air inlets. Another two bundles are fixed at the floor tile and the ceiling vent. The details of sensor deployment can be found from Figure 4.16.

4.5.2 Implementation of the Sensor Network

Wireless Sensors: In each of our testbed implementations, we use a single-hop network architecture where the base station sends the data collection requests to sensors sequentially and each sensor transmits the measurements. Every 5 seconds, the base station performs

a round of sequential data collection from all sensors. We note that a multi-hop network topology can be employed when more server racks need to be monitored. As this collection scheme works in a time-division fashion, the system does not generate many collisions between the data transmissions of different sensors. TelosB [16] and Iris [16] motes are used for collecting temperature data. To collect the airflow velocity data, we connect the Senshoc mote, an implementation of the open design of TelosB, to a standalone air velocity sensor [77] via I^2C interface. The programs on these motes are implemented in TinyOS 2.1 [78].

On-board Sensors: CPU utilization and fan speed are two important thermal variables that the system needs to collect from the on-board sensors of each server. Data centers typically run various server monitoring utility tools (e.g., `atop`, `ganglia`) that can collect on-board sensor information. These tools are used to implement the data collection of CPU utilization and fan speed for our production testbed. In our single-rack testbed, we implement a simple program to control and measure the CPU utilization, and report fan speed from either `lm-sensors` or `ipmimonitoring` utilities, which are commonly available in `GNU/Linux` distributions. Similar to the wireless sensor data collection, the base station requests the CPU utilization and fan speed from each server sequentially. However, instead of using wireless links, the base station takes advantage of the existing Ethernet infrastructure to collect these on-board sensor data.

4.5.3 Discussion

We now discuss the costs and benefits of deploying our temperature prediction system in data centers. Our prediction system comprises low-cost wireless sensor nodes and PC-class base stations. Moreover, it is even more cost-effective in newer data centers where the servers may have already been equipped with inlet temperature sensors. In addition to the costs of the wireless sensor network, our system requires compute-intensive CFD simulations to assist the temperature prediction, where the major costs are due to the computing resources, labor of model construction and CFD software license. Since our system only requires off-

line CFD simulations, it can leverage the existing data center computing resources during off-peak hours for training the prediction models. Once the models are constructed, our system will no longer consume any computing resources in the data center. In addition, many data centers have already incorporated the CFD analysis in the thermal design, which can be reused to reduce the extra labor and license costs for CFD.

The benefits of deploying our prediction system are two folds. First, heat-induced server shutdown contributes more than 23% of server outages in data centers [21]. It equals 0.2 million US dollars loss per year for a data center on average [79]. With accurate multi-horizon temperature prediction, the data center administrators can be alerted of the potential thermal emergencies, e.g., server overheating, allowing more time for necessary actuations such as migrating server workload to prevent these emergencies. Second, the real-time temperature prediction could enable the proactive thermal control at a higher room temperature in data centers without causing the server overheating, achieving as much as 30% of total energy saving as we discuss in Chapter 5. For instance, the proactive thermal control system can dynamically adjust the cold air temperatures and velocities of the cooling system based on the predicted future temperatures instead of the measured temperatures.

4.6 Performance Evaluation

To evaluate the performance of our prediction system, we conduct extensive experiments on the single-rack testbed and the small-scale production testbed. On the single-rack testbed, we can conduct controlled experiments such as simulating AC failures to extensively evaluate our system. The production testbed allows us to evaluate our system under realistic, long-term computation workloads.

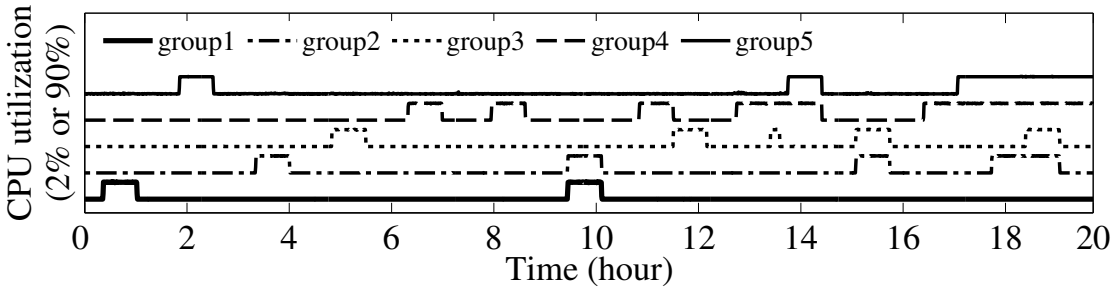


Figure 4.6 CPU utilization of the training data.

4.6.1 Single-Rack Testbed Experiments

Figure 4.2(a) shows the server groups and the temperature sensor locations on the rack of single-rack testbed. Five server groups, denoted as Group 1 to Group 5, are controlled to run in either idle state (about 2% CPU utilization) or full utilization (about 90% CPU utilization). These settings are consistent with many data centers where servers running computational-intensive batch jobs tend to use all available CPUs [25]. We conduct various controlled experiments by adjusting servers' CPU utilization to simulate the normal running state of data centers, as well as turning off the cooling function of the AC to simulate a thermal emergency.

4.6.1.1 Predication under Dynamic Workloads

The first experiment evaluates the performance of our system in response to the CPU utilization changes. A total of 25 hours of data were collected during 6 days. As the infrastructure ceiling vent is regularly shut down every night, we concatenate the data collected in different days when the ceiling vent is running. We use the first 20 hours of data as training data and the remaining 5 hours of data for testing. The settings of the prediction model include $R = 1$ and $k = 10$ min. Figure 4.6 shows CPU utilization of the training data and Figure 4.7 shows the CPU utilization and temperature prediction at both inlets and outlets. We can see that our system can accurately predict the temperatures. From the middle graph of Figure 4.7, at about the 30th minute from the start, the temperature reached equilibrium

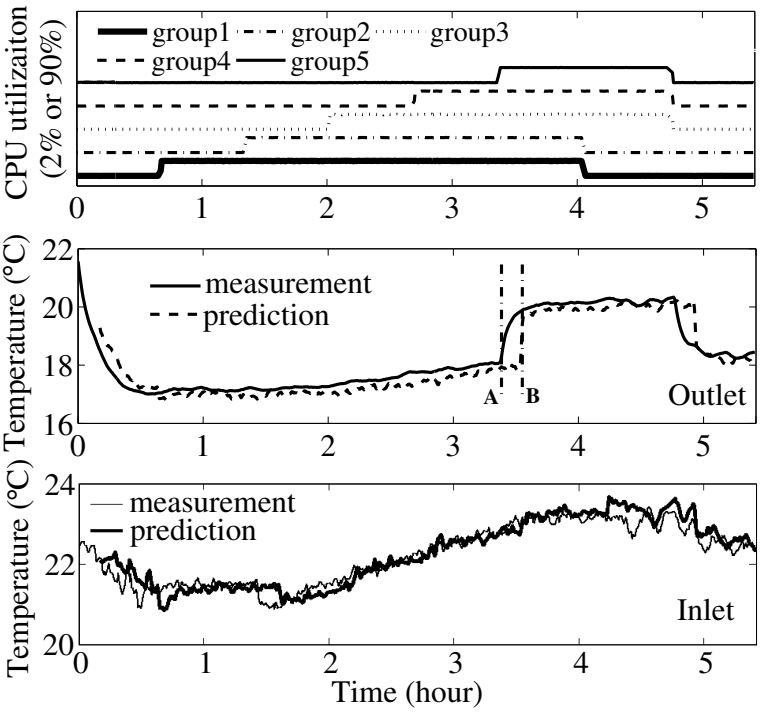


Figure 4.7 Top: CPU utilization of test data. Middle: temperature measurements and predictions at an outlet of Group 5 with 10 minutes prediction horizon. Bottom: temperature measurements and predictions at an inlet of Group 3 with 10 minutes prediction horizon.

as the start of our experiment. In the first 3 hours, as only Group 1 to Group 4 changed their running states, the measurements of Sensor 2 at an outlet of Group 5 did not change significantly. A small temperature rise during this period was caused by the complex airflow at the back of the rack. When the servers in Group 5 changed to full utilization during the 4th hour, a significant temperature rise is observed. With a 10-minute prediction horizon, each point on dashed curve is calculated using measurements of all sensors 10 minutes before. We can see that the temperature at the future time instant B is accurately predicted at the actual time instant A when the system observes the CPU utilization change. While the prediction results well match the sensor measurements during the first 3 hours, however, we observe a considerable gap between the predicted temperatures and sensor measurements in a duration of 10 minutes (i.e., between A and B shown in the figure) after the CPU utilization change of Group 5. This is due to the fact that the system is not aware of the state change of Group 5 at time instance A. In this work, this type of error is referred to

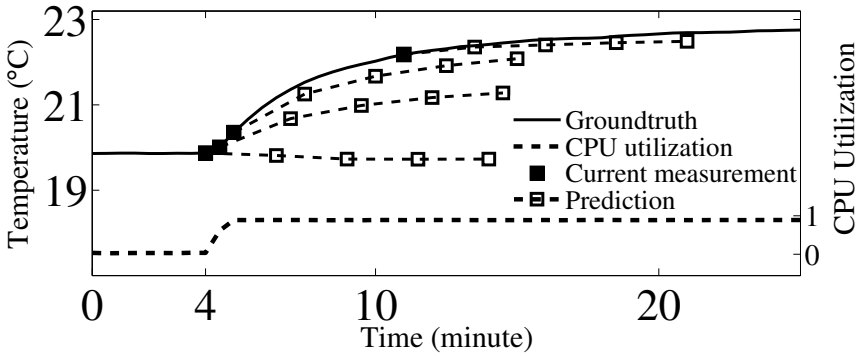


Figure 4.8 Temperature evolution prediction. Each solid rectangle represents the temperature measurement at current time instance and the white rectangles are the predicted temperatures at four different prediction horizons (0.5, 2.5, 5 and 7.5 minutes).

as *horizon-induced* prediction error. According to the multi-horizon prediction scheme discussed in Section 4.6.1.2, the duration that suffer horizon-induced prediction error can be shortened by setting a smaller prediction horizon. This hypothesis will be verified in Section 4.6.1.6. Different from the temperature at the outlets, the temperature at the inlets is mainly affected by the complex heat recirculation. The bottom graph shows that our system can also accurately predict the temperature at the inlet. During the 5-hour testing period, the average absolute prediction error over all sensors is only 0.3°C.

4.6.1.2 Multi-Horizon Prediction

In our prediction system, by training models with different k in Eq. (4.1), we can build multiple models to predict the evolution of temperature in the future. Figure 4.8 shows the results of different prediction horizons of 0.5, 2.5, 5 and 7.5 minutes. At about the 4th minute when the CPU utilization just started to increase, the predicted temperatures at different prediction horizons are similar to the current measurement. After the system evolved to the second solid rectangle where the CPU utilization had increased significantly from 2% to 90%, the system predicts an increasing trend of temperature evolution for the following four horizons. From the time instance of the 3rd solid rectangle, the predicted temperature evolution starts to match the ground truth. Figure 4.9 shows the root-mean-square error

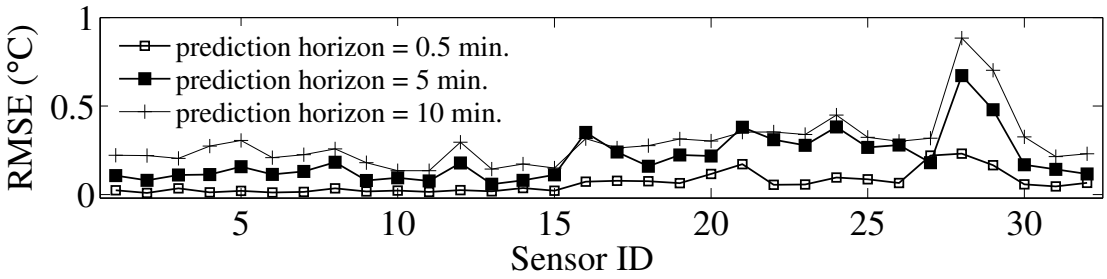


Figure 4.9 Root-mean-square error (RMSE) of multi-horizon temperature prediction.

(RMSE) of multi-horizon predictions for each sensor. We can see that the RMSE generally increases with the prediction horizon. This conforms to the intuition that the temperature at a farther time instance in the future is less correlated with historical measurements in a dynamic environment. The RMSEs are less than 0.5°C for most sensor locations. Slightly larger RMSEs are observed at sensor 28 and 29. We found that this is caused by the slight displacement of the two sensors during the experiment. Nevertheless, the RMSEs are still less than 1°C.

4.6.1.3 Effectiveness of Regularized Regression

In this experiment, we conduct cross-validation based on the 14-hour data set used in Section 4.6.1.2 to evaluate the effectiveness of the regularized regression discussed in Section 4.4.2.1. Performance evaluation of model training depends on how the data set is partitioned into training and test data. Ten-fold cross-validation [80, p. 435] is commonly used to mitigate the impact of data set partitioning in evaluating model training performance. Specifically, in each *fold*, we choose 1.4 hours of continuous data for testing and the remaining data for training. For each fold, given a regulation factor λ, we conduct model trainings and predictions with 5 different horizons from 0.5 minutes to 10 minutes. The average RMSE of prediction over all horizons is used to represent the average prediction error given λ. The above process is repeated for 10 times (i.e., 10 folds) with different partitions of training and test data sets. Figure 4.10 shows the average RMSE for three individual

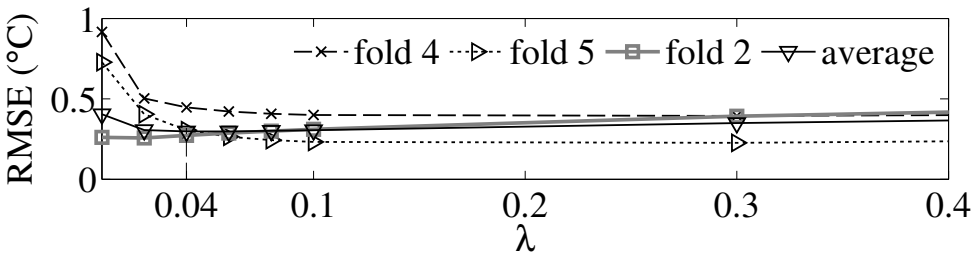


Figure 4.10 RMSE of prediction versus λ in cross-validation experiments.

folds versus λ. In the experiment of fold 4, the average RMSE of prediction is as high as 1°C when the training is not regularized (i.e., $\lambda = 0$). The average RMSE decreases with λ when $\lambda < 0.3$ and exhibits a slight increase after that. The experiment of fold 5 shows similar trend with much lower RMSE. In the experiment of fold 2, the average RMSE always increases with λ. Figure 4.10 also shows the average RMSE over all folds, which characterizes the expected performance given any training/test data partition. From the average RMSE over all folds shown in Figure 4.10, we can see that in a large range of λ (i.e., from 0 to 0.4), the regularized regression outperforms the unregularized version (i.e., $\lambda = 0$). Note that the typical setting of λ is no greater than one. Moreover, when $\lambda = 0.04$, the average RMSE is minimized and reduced by 25% with respect to the unregularized result. Under this setting, the trained models yield good prediction accuracy across all folds. Therefore, 0.04 is a desirable setting for λ for our single-rack testbed.

4.6.1.4 Performance under Noisy Sensor Measurements

In this section, we evaluate the prediction performance under noisy sensor measurements with dynamic CPU utilizations. In particular, we manually add the Gaussian white noises to the sensor measurements. The standard deviation of the noises are proportional to the range of its readings in the data traces. Figure 4.11 shows the RMSE of predictions versus the increasing noise standard deviation for each thermal variable under different prediction horizons. Consistent with intuition, the RMSE increases with the noise level. However, even with noise standard deviation up to 15% of the sensor reading range, the RMSEs are still

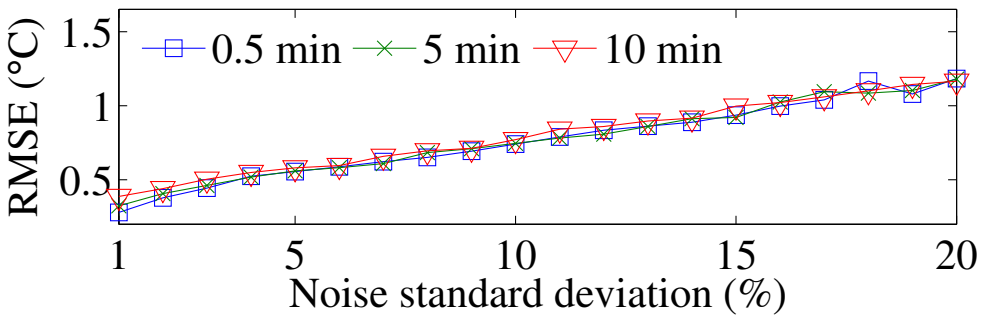


Figure 4.11 RMSE under noisy data

within 1°C. In practice, various noise suppression techniques, e.g., moving window average, can be employed to mitigate the sensor measurement errors and improve the prediction accuracy.

4.6.1.5 Multi-Channel Prediction

In this experiment, we evaluate the accuracy of prediction in multiple thermal conditions (i.e., channels). As the AC malfunction is a major cause of server overheating in data centers, we conducted a controlled experiment to simulate the AC failure on our single-rack testbed. We construct two channels corresponding to the normal running state and the AC failure, respectively. A total of 10 hours of data were collected when the servers run in normal state with different CPU utilization combinations. These data are used to train the normal channel of the prediction system. The prediction horizon is set to be 5 minutes. Then, another 14 hours of data, which contain both normal running state and AC failure, were collected. A transient CFD simulation is conducted using the sensor data (after excluding the temperature measurements at server inlets/outlets) collected during this 14-hour experiment. The CFD-simulated training data, together with the 10 hours of real measurements in normal running state, are then used to train the channel of AC failure. In real data centers, it is often infeasible to collect training data for the scenario of AC failure. Therefore, to ensure the realism of our experiments, we did not use the sensor measurements during AC failure to calibrate the CFD.

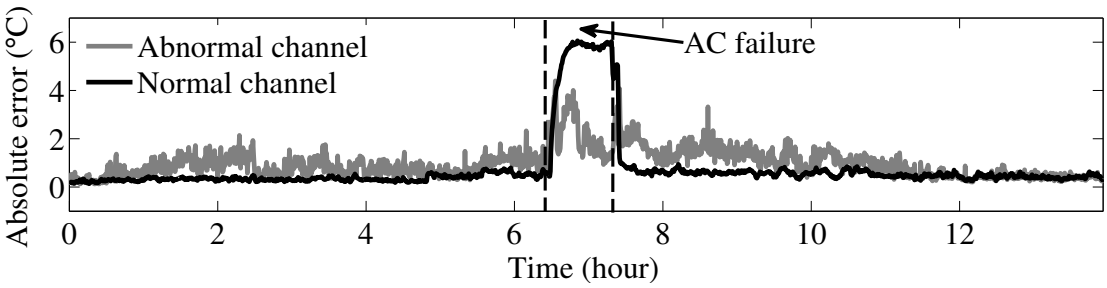


Figure 4.12 Average absolute temperature prediction error (prediction horizon = 5 minutes)

Figure 4.12 shows the absolute prediction errors of the two channels with respect to the ground truth sensor measurements. The system exhibits very small absolute error in normal state, while it suffers up to 6°C absolute error during the AC failure. This is because the training data for the normal channel do not capture this abnormal situation. On the contrary, AC failure channel exhibits slightly higher absolute error than the normal channel during the normal state, while it has significantly lower absolute error during the AC failure. From this result, we can see that the simulated training data generated by CFD can help the real-time prediction model capture various thermal emergencies. In practice, several different abnormal channels can be constructed with CFD according to the possible cooling system failure situations. The detection results from different channels can further be fused by existing data fusion techniques [5].

4.6.1.6 Sufficiency of Training Data from CFD

In this set of experiments, we evaluate the training data generation approach described in Section 4.4.2.2. These experiments focus on the thermal emergency of AC failure. We first explain how we generate the training data traces. Figure 4.13 shows the temperature trace of a server inlet in one of the transient simulations. The simulation starts with all servers in idle state with 0% CPU utilization, followed by a change of CPU utilization at about the 4th minute. Following the random approach described in Section 4.4.2.2, the new CPU utilization of a server group is randomly drawn from a uniform distribution over $[0\%, 100\%]$.

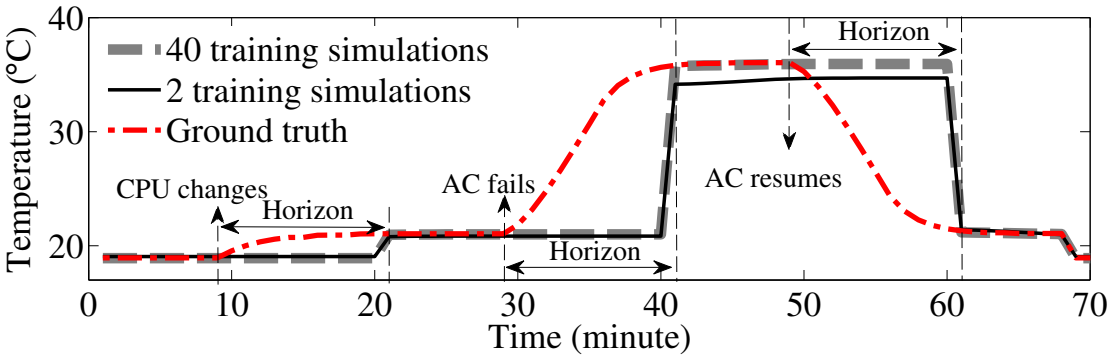


Figure 4.13 Example of training data generated from CFD for AC failure emergency. The simulation starts with all the servers in idle status, followed by a uniform random change on CPU utilization at about the 4th minute. Then, the AC fails at about the 14th minute and resumes at about 24th minute.

The inlet temperature shown in Figure 4.13 starts to rise because of the air recirculation. The simulated AC failure occurs at about the 14th minute and the AC recovers from failure at about the 24th minute. We conduct 50 transient simulations to generate the training data, where the CPU utilization of each server group is drawn from the uniform distribution. Moreover, we conduct two transient simulations with extreme conditions, that is, the CPU utilization of all server groups is either 100% or 0%.

As discussed in Section 4.4.2.2, the number of generated data traces should be large enough to prevent over-fitting. We design an experiment to evaluate the impact of the amount of training data on the performance of model training. We choose 10 out of 50 transient simulations as the test data. Initially, only the two transient simulations with extreme conditions are included in the training data set. We then incrementally add a simulation that is chosen from the unused simulations to the training data set. For each training data set, we evaluate the prediction performance of the trained model using the test data of 10 transient simulations. When we compute the RMSE to characterize the prediction error, we carefully choose the testing results to exclude the horizon-induced prediction error, which is explained in Section 4.6.1.1. For instance, we exclude the durations labeled by "Horizon" in Figure 4.13, which suffer horizon-induced errors. The root cause of this type of

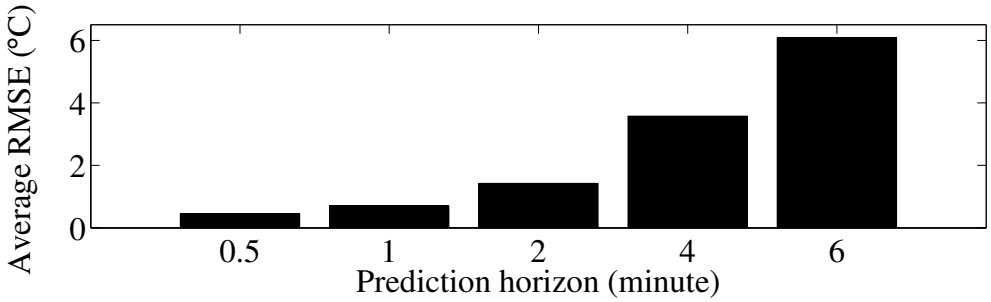


Figure 4.14 Horizon-induced prediction error versus horizon.

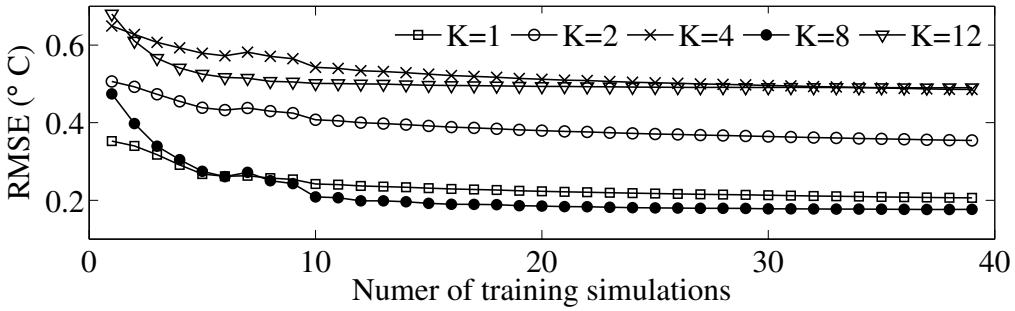


Figure 4.15 Prediction errors with incremental training samples.

errors (as shown in Figure 4.14) is horizon, rather than the insufficiency of training data.

Figure 4.13 shows two traces of prediction when 2 and 40 transient simulations are used as training data, respectively. Figure 4.15 shows the prediction error versus the size of training data set (i.e., the number of transient simulations), under various settings of prediction horizon. We can see that the prediction error generally decreases with the size of training data set. In particular, when more than 10 transient simulations are used to train the model, the prediction error becomes flat. This result shows that, by our training data generation approach, a small number of transient simulations can be sufficient to train the model. Moreover, from Figure 4.15, we do not see strong correlation between the prediction error and horizon. This is because, after excluding the durations suffering horizon-induced prediction error, the remaining durations are mostly steady states, where horizon is not an major factor.

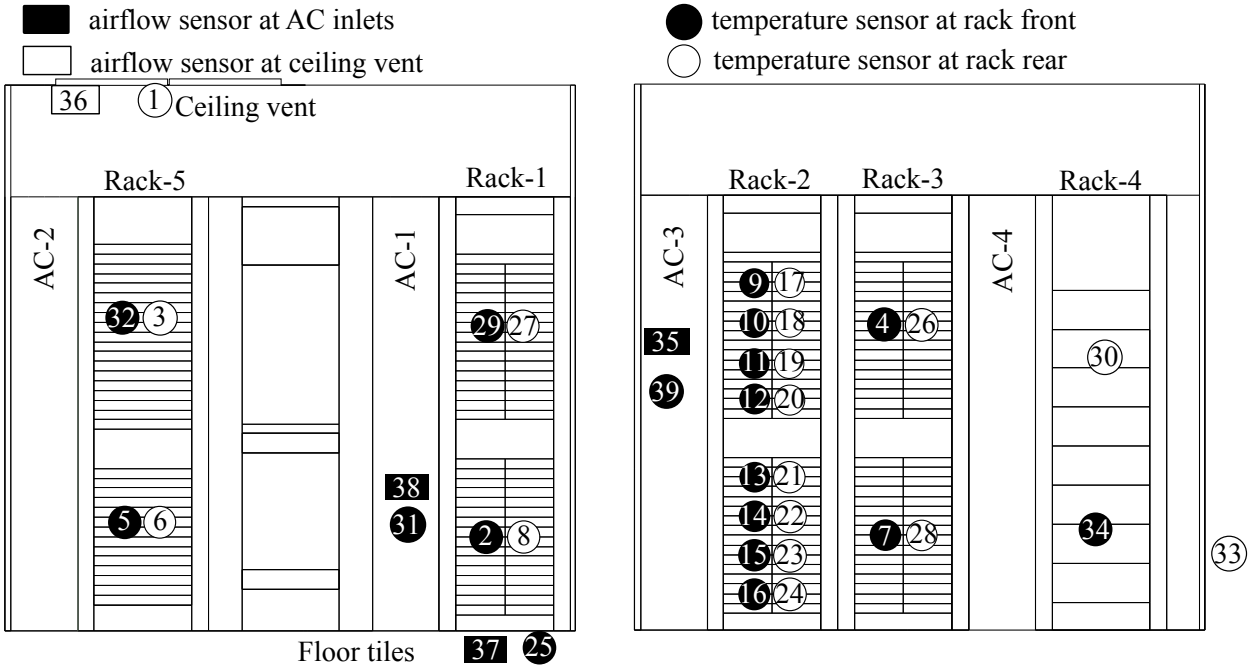


Figure 4.16 Front view of the two rows of racks, which face to each other in the server room.

4.6.2 Production Testbed Experiments

We also deployed and evaluated our system on a small-scale production testbed in a server room of High Performance Computer Center (HPCC) at Michigan State University. In this testbed, we deployed 35 temperature sensors and 4 airflow velocity sensors. Figure 4.16 shows the sensor deployment from the front view of the two rack rows. To evaluate the impact of sensor density, we deploy 16 sensors on one rack (Rack-2) while other racks are instrumented with 2 or 4 sensors. Different from the single-rack testbed whose CPU utilization is controlled, the CPU utilization in HPCC testbed is subject to real use and hence dynamic. Therefore, the accurate temperature prediction is more challenging. Figure 4.17 shows the average CPU utilization of the upper and lower section of the servers on Rack-2 in a 12-day period. We can observe that the lower section of servers usually has high CPU utilization, except on April 8, which is a Sunday. On the contrary, the upper section of servers has more variable CPU utilization. In this section, we evaluate our prediction approach in HPCC testbed using the data collected in 15 continuous days. The data from

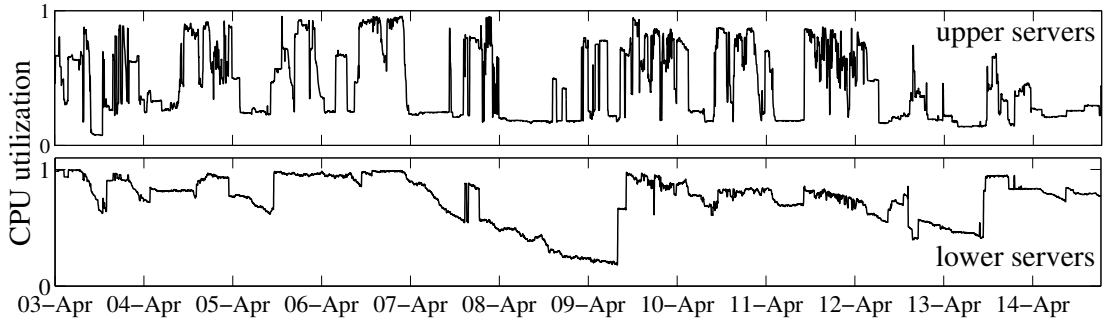


Figure 4.17 CPU utilization of servers on upper and lower levels of Rack-2

the first three days (March 31 to April 2, 2012) are used as training data, while the data of the following 12 days are used for prediction evaluation.

4.6.2.1 Dimension Reduction

On our single-rack testbed described in Section 4.6.1, the dimension of the state $\mathbf{P}$ is 64. However, on the production data center testbed, the dimension of the state $\mathbf{P}$ grows to several hundreds. As discussed in Section 4.4.3, it is desirable to perform dimension reduction to avoid overfitting. In addition to the *partial correlation* approach described in Section 4.4.3, we employ two baseline approaches. The first baseline approach, referred to as *random* approach, randomly selects variables. The second baseline approach, referred to as *location* approach, selects the variables that are geographically closest to the monitored location. In following experiments, we vary the percentage of selected variables over all available variables.

Figure 4.18(a) shows the average RMSE of the prediction results with prediction horizons of 5 and 10 minutes. For the partial correlation approach, RMSE drops from 3°C to about 0.6°C when the percentage of selected variables reduces from 100% to 7%. This result conforms to our motivation for dimension reduction discussed in Section 4.4.3. However, when the percentage continues to decrease, RMSE starts to increase since the number of variables is too small to contain enough information for a good prediction. When the per-

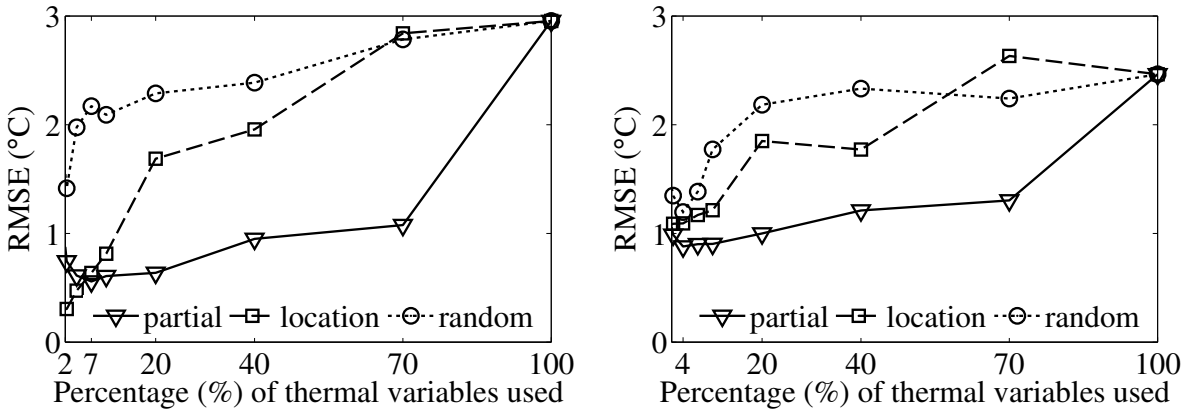


Figure 4.18 Prediction error versus the percentage of selected variables in dimension reduction. (a) All data are used for testing; (b) Only transient data are used for testing.

centage reduces from 100% to 7%, the RMSE of the location approach is larger than that of the partial correlation approach which, however, is outperformed by the location approach when the percentage is lower than 7%. With the location approach, the monitored temperature itself is assigned with a very high weight in $\mathbf{a}_j$ given by Eq. (4.2) when a small percentage of variables are chosen. Therefore, the resulted prediction models tend to follow an autoregression model. As a result, when the temperatures are in steady state, the autoregression prediction is highly accurate. As the test data used for Figure 4.18(a) includes lots of steady states, the location approach yields good performance when the percentage is low. However, such a autoregression prediction model is not helpful for predicting overheating in emergencies such as AC failures because the variables related to AC may not be chosen by the location approach. In practice, it is unlikely to create these thermal emergencies in the production data center for system training. As a compromise, we manually select transient states, which are mainly caused by the changes of CPU utilization, as the test data. With these transient test data, the partial correlation approach consistently outperforms the location approach, as shown in Figure 4.18(b), achieving the minimal RMSE when 4% of variables are selected. Moreover, from both Figure 4.18(a) and Figure 4.18(b), the partial correlation approach achieves low RMSE in a wide range of settings (2% to 70%), which

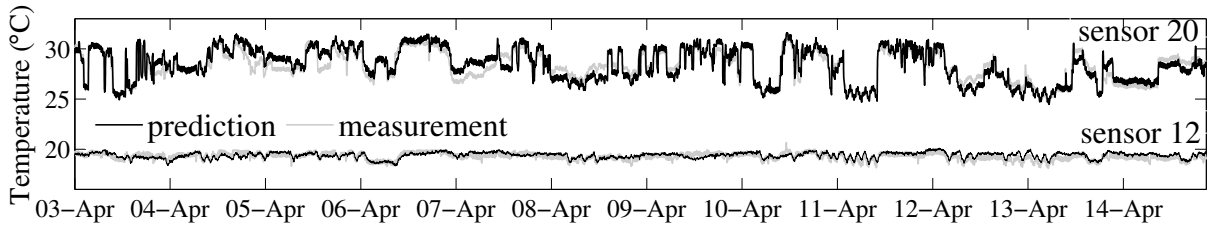


Figure 4.19 Long-term monitoring with 10 minutes prediction horizon. Sensor 20 and sensor 12 are located at server outlet and inlet, respectively.

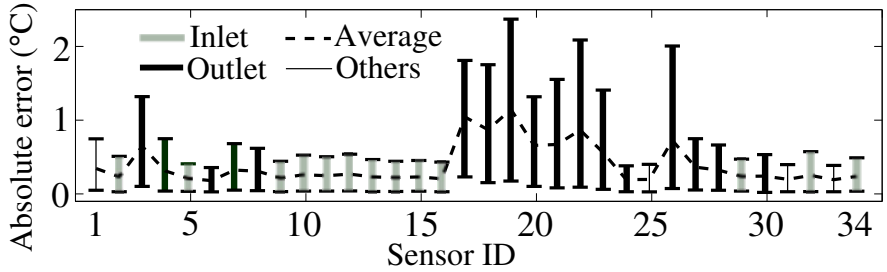
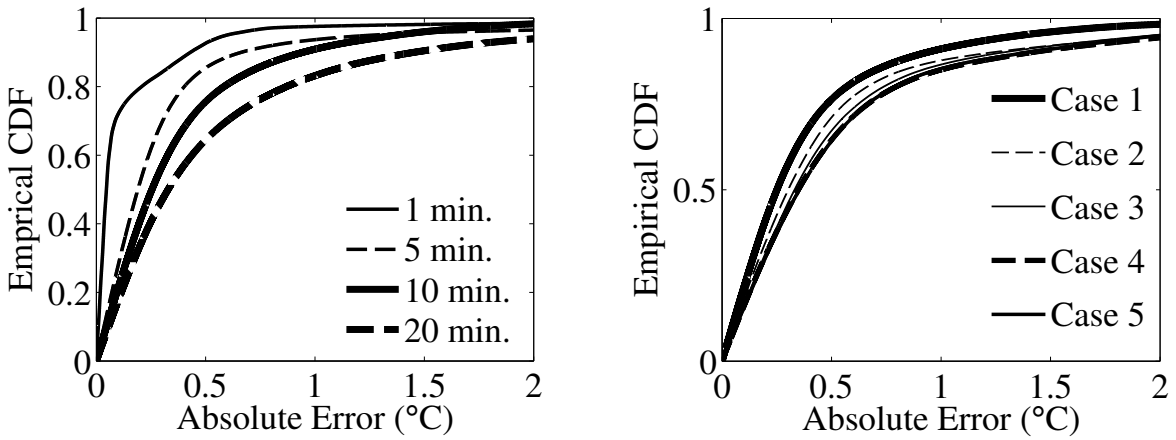


Figure 4.20 Absolute errors with the 90% error bound for each sensor with 10 minutes prediction horizon.

allows flexible setting without sacrificing the prediction performance substantially. For the experiments conducted in the rest of this chapter, we perform dimension reduction using partial correlation approach with the setting of 4%. From Figure 4.18(a) and Figure 4.18(b), the random approach consistently yields the worst performance.

4.6.2.2 Long-term Monitoring

Figure 4.19 shows the prediction results at two locations during 12 days. The prediction horizon is set to be 10 minutes. We can observe that our prediction results well match the groundtruth measurements of both server inlet and outlet sensors. Sensor 20, located at a server outlet, exhibits slightly larger prediction errors. This is because the server outlets suffer more influence from system workloads and hence have more dynamic thermal profiles. Figure 4.20 shows the average absolute prediction error and the 90% error bound for all sensor locations. We can observe that the prediction errors on outlet sensors are slightly higher than inlet sensors. Nevertheless, the average absolute error of outlet predictions is only around



(a) (b)

Figure 4.21 Empirical CDF of absolute error (a) for all sensors with different prediction horizons; (b) when different subsets of sensors are used in model training.

1°C and 90% of predictions have errors lower than 2°C. We also evaluate the prediction errors under different settings of prediction horizons. Figure 4.21(a) shows the empirical Cumulative Distribution Function (CDF) of prediction error over all sensor locations. Similar to the results in Figure 4.9, the prediction error increases with the prediction horizon.

4.6.2.3 CFD-Assisted Prediction

In this section, we evaluate the performance of using CFD to assist the temperature predictions. We focus on evaluating how effective the CFD modeling reduces the number of required sensors under the normal running state because no thermal emergencies were observed on the production testbed during the 15-day experimental period. Specifically, during the model training, we remove the measurements of some sensors and replace them with CFD transient simulation results. The removed temperature sensors will not be selected as thermal variables. However, with their CFD replacements, they can be used as output to train the temperature predictions at those locations. We use the first 3 days of boundary condition data (e.g., CPU utilization) to drive the CFD transient simulation. Then, the sensor data of the first day are used to construct the calibration functions discussed in

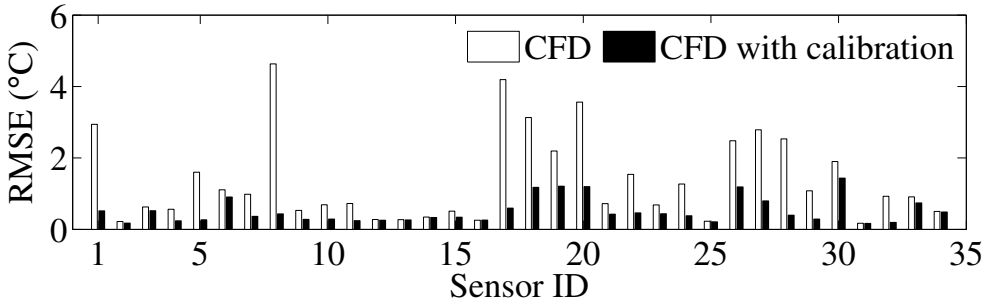


Figure 4.22 RMSE of CFD calibration in production testbed.

Section 4.3.3. After that, all the 3-day simulated training data are calibrated using the calibration functions. Figure 4.22 shows a significant accuracy improvement after the CFD calibration.

We evaluate the performance of the CFD-assisted prediction by gradually removing sensors in the model training. As shown in Table 4.1, we gradually replace measurements of some sensors with CFD simulation results. In Case 1, all total 39 sensors are used for training. Then, we replace sensor 10, 12, ..., 24 from Case 1 to generate Case 2, which uses 31 sensors in total for training. In Case 5, 26 sensors (i.e., 67% of all sensors) are replaced with data generated from CFD. The empirical CDF of absolute errors of different cases are plotted in Figure 4.21(b). The prediction horizon is 10 minutes. We can see that the prediction accuracy increases with the number of sensors. This is consistent with the intuition that the CFD data is not as accurate as the actual sensor measurements. However, it can be seen that, even when less than 40% of sensors are deployed in Case 5, 85% of predictions have absolute errors lower than 1°C. Overall, our approach can reduce 67% of sensors for monitoring the inlets and outlets of all servers while only increasing the average prediction error by 0.2°C. For all cases, the average RMSEs are less than 1°C, while the maximum RMSE is within 2°C. This result clearly demonstrates the advantage of integrating calibrated transient CFD modeling with real sensor measurements. Currently, the sensor reduction is performed empirically. The number of required sensors to achieve a certain prediction accuracy is highly affected by the physical properties in the data center. This is still an open issue and left for

Table 4.1 Evaluation scheme of replacing sensors with CFD

Case	Sensors removed	Total
1	none	39
2	10, 12, 14, 16, 18, 20, 22, 24	31
3	32, 3, 29, 27, 4, 26	25
4	9, 17, 13, 21, 36, 1	19
5	5, 6, 25, 37, 7, 28	13

our future work.

4.7 Conclusion

In this chapter, we describe the design and implementation of a novel cyber-physical system for predicting temperature distribution of data centers. Our approach integrates Computational Fluid Dynamics (CFD) modeling and real-time data-driven prediction to achieve high fidelity temperature forecasting in various thermal conditions of data centers, including rare but critical thermal emergency situations like AC failures. We have implemented the system on a single-rack testbed and a testbed of 5 racks and 229 servers in a production high performance computing center. Extensive experimental results show that our approach can accurately predict the temperatures up to 10 minutes into the future, even in the presence of highly dynamic server workloads.

CHAPTER 5

PREDICTIVE THERMAL AND ENERGY CONTROL IN DATA CENTERS

5.1 Introduction

Thermal and energy management has become a key challenge in the design and operation of data centers. A recent worldwide data center survey shows that the non-computing energy takes average 45% and up to 60% of total energy [81]. One of the key reasons for these data centers to have excessive energy consumption is the inefficient operation of Computer Room Air Conditioning (CRAC) systems. Because of the lack of visibility in the operating conditions, the CRAC systems often use unnecessarily low temperature setpoints to reduce the risk of server overheating. Due to such a conservative strategy, the CRAC systems can account for up to half of the energy consumption of a data center [20]. Moreover, data centers usually maintain unnecessarily high levels of air circulation by adopting static settings or simplistic control strategies for the circulation systems including server fans. As a result, the server fans can take up to 23% of server power consumption [42]. Thus, improving the efficiency of cooling and circulation systems plays an important role in reducing the total energy consumption of a data center.

Various efforts have been made to improve data center energy efficiency. New green data center technologies have proven their effectiveness in a few latest industrial scale data centers. For instance, the new Google data centers reduce the non-computing energy ratios down to about 10% [23]. However, these technologies require a clean slate redesign and hence are cost prohibitive to apply in existing data centers. A recent survey reveals that 85% of existing data centers have non-computing energy ratios higher than 40% [81]. Therefore, low-cost effective thermal management systems that can retrofit existing data centers with better energy efficiency are highly appealing. Data center operational guidelines have been revised recently to avoid overly conservative settings. For instance, the American So-

ciety of Heating, Refrigerating and Air Conditioning Engineers (ASHRAE) recommends to increase the environmental temperatures in data centers up to 27°C to reduce cooling energy consumption [71]. However, without a real-time thermal control system that ensures the thermal safety, the higher environmental temperature setpoints will increase the risk of server overheating. In fact, a survey in 2013 shows that 90% of data centers still operate under 24°C [82].

A variety of thermal control schemes have been recently proposed to prevent thermal emergencies in a data center while reducing the energy costs. The existing approaches either optimize a single thermal variable (e.g., server workload, CRAC setpoint, or fan speed, etc.) [24][25] or a combination of them [26][27] to minimize the energy costs. However, most of these approaches are based on a *reactive* scheme, which reactively controls the cooling systems to eliminate detected hot spots. Unfortunately, this approach often cannot achieve desirable energy efficiency, primarily due to the complex thermodynamics of data centers. For instance, the heat generated by increased server workload takes substantial delays to be recirculated to the server inlets. To react to the detected hot spots at the server inlets, the CRAC systems need to adopt sufficiently low temperature setpoints, which, however, significantly downgrade their energy efficiency [25].

In contrast to the existing reactive thermal control schemes, this chapter proposes a *proactive* approach to prevent potential future hot spots. Specifically, our approach predicts the energy consumption and thermal conditions resulted by each possible CRAC/circulation control action in real time, and then executes the best one. We need to address two major challenges to realize such a proactive control scheme. First, the thermal characteristics of a data center are inherently affected by both physical (e.g., complex airflows) and cyber (e.g., dynamic server workloads) factors. In particular, the temperature evolution, the energy consumption of CRAC/fan systems, and their control decisions are tightly coupled together. Moreover, the control decisions not only need to improve the energy efficiency, but also must account for servers' thermal safety requirements. Second, a large number of variables in a

data center may affect the temperatures and the total energy consumption, including the fan speed of each server and the temperature setpoint of each CRAC system. The global energy optimization based on all controllable variables often has prohibitive computational complexity. Moreover, to ensure system reliability even in certain thermal emergencies, the thermal control system should not resort to the computing resources of the monitored data center.

This chapter presents a real-time Predictive Thermal and Energy Control (PTEC) system that improves the energy efficiency of both the cooling and circulation systems of a data center, while meeting a set of thermal safety requirements. PTEC leverages the server built-in sensors and monitoring utilities, as well as an external easy-to-deploy wireless sensor network to monitor both the cyber and physical status of a data center, which includes CPU utilization, dynamic air flow, temperature distribution, and CRAC/fan settings. Based on these measurements, PTEC predicts the server inlet temperatures in real time and proactively controls the temperature setpoints and blower speeds of CRAC systems, and the speeds of server fans, to reduce their overall energy consumption. PTEC enforces a set of thermal safety requirements including the upper bounds on server inlet temperatures and their temporal variations. A high inlet temperature directly indicates server overheating and potential server shutdowns, while a high temporal variation of server temperature can significantly increase hardware failure rate [69]. To make the energy optimization problem tractable, PTEC adopts a *coordinated control* approach. Specifically, a novel dynamic fan speed control algorithm is first developed to automatically control the server fans based on the server CPU utilization and the inlet temperature. Then, PTEC searches for the temperature setpoints and blower speeds of CRAC systems to minimize the overall energy consumption of CRAC systems and server fans. Moreover, we propose a partition-based algorithm, which divides the CRAC systems to multiple regions based on their spatial thermal correlations with the servers, to reduce the computational overhead of PTEC for large-scale data centers.

We prototyped PTEC and deployed it on a hardware testbed consisting of 15 servers

and a total of 23 temperature and power sensors. The results show that PTEC can reduce the cooling and circulation energy consumption by up to 34% and 30%, compared with an overcooling strategy and a reactive control strategy, respectively. We also conducted trace-driven Computational Fluid Dynamics (CFD) simulations for an existing data center with 229 servers to validate the effectiveness and scalability of PTEC.

5.2 Problem Statement and Approach Overview

5.2.1 Problem Statement

The CRAC systems are the major source of energy consumption in many existing data centers [20]. Another thermal-related source of energy consumption is the server fans, which can take up to 23% of server power [42]. It has been shown that the cooling efficiency of a CRAC system increases with its temperature setpoint [25]. With a higher setpoint, a CRAC system can remove the same amount of heat with up to 40% less energy consumption[25]. However, a higher CRAC setpoint increases the likelihood of server overheating and heat-induced server shutdown. Moreover, it may adversely increase server fan speeds for removing the heat generated by the servers, resulting in higher overall energy consumption. This work aims to design a control system to reduce the overall energy consumption of CRAC systems and server fans, subject to a set of thermal requirements including upper bounds on server inlet temperatures and their temporal variations. Upper-bounding them can prevent server overheating and severe temperature fluctuations that can cause high hardware failure rates [69].

A CRAC and server fan control system has to address the following two fundamental issues. First, the data center has complex thermodynamics and tight thermal coupling among the servers, the CRAC systems, and the physical environment. It is challenging to accurately model the thermodynamics, which, however, is the base for designing an effective thermal control system. Second, to adapt to the unpredictable dynamics of the data center,

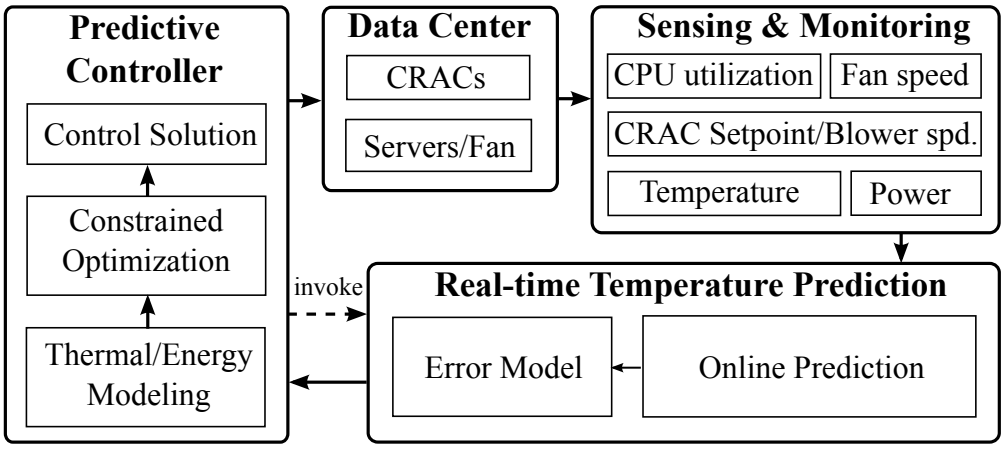


Figure 5.1 PTEC system architecture.

the control system must run in a real-time manner. However, the optimal control algorithm is computation-intensive due to the complex and non-linear relationships among control actions, energy efficiency, and thermal conditions.

This chapter designs a data center thermal and energy control system based on a *predictive* scheme. This design choice is based on the key observation that there are various time delays in the data center thermodynamics. For instance, the extra heat generated by suddenly increased server workload takes a considerable delay to be recirculated by the server fans and the CRAC systems to the server inlets. Moreover, due to limited cooling capacity, it typically takes a considerable delay for a CRAC system to reach the new temperature setpoint. Thus, it is desirable to *proactively* control the CRAC/fan systems in an energy-efficient manner to prevent potential future hot spots.

5.2.2 Approach Overview

Figure 5.1 illustrates the architecture of PTEC based on a predictive control scheme. It consists of three major components:

Sensing and monitoring: PTEC periodically collects the measurements of the variables that affect the temperatures in a data center, which include server status (CPU utilizations, system temperatures, fan speeds, and powers) and CRAC status (powers, temperature set-

points, and blower speeds) from the built-in sensors and monitoring utilities, and a few critical temperatures (e.g., server inlet and CRAC return hot air temperatures) from a small number of wireless sensors. These wireless sensors, powered by either onboard batteries or USB interface of servers, can self-organize into a network for data collection. Therefore, the overhead of sensor installation process is small. Figure 5.8 shows the sensor deployment on a single rack testbed.

Real-time temperature prediction: The system can rapidly predict the evolution of temperature distribution based on the collected sensor data and a candidate CRAC/fan control solution. Such a real-time prediction enables the system to assess a large number of candidate control solutions during runtime.

Predictive controller: We model the power consumption of server fans and CRAC systems. Based on the models, we formally formulate the problem of minimizing the predicted overall energy consumption of server fans and CRAC systems, subject to a set of thermal safety requirements. A predictive controller assesses the temperature evolution in the future for each candidate control solution and chooses the most energy-efficient one. The solution comprises the temperature setpoints, blower speeds of CRAC systems, and server fan speeds.

5.3 Power Consumption Models

5.3.1 Server Fan Power Consumption Model

Air circulation is critical for cooling servers in a data center. A server system is typically equipped with several internal fans for cooling different components, such as power supply unit and CPU. A server fan often regulates its speed according to the duty cycle of a pulse width modulation (PWM) signal. Most servers control the fan speed using a simple native algorithm, which linearly increases the PWM duty cycle based on the server inlet temperature [83]. In Figure 5.2, curve r_1 illustrates the input and output of the algorithm, which is parameterized by two thresholds V_1 and W_1. When the server inlet temperature is lower

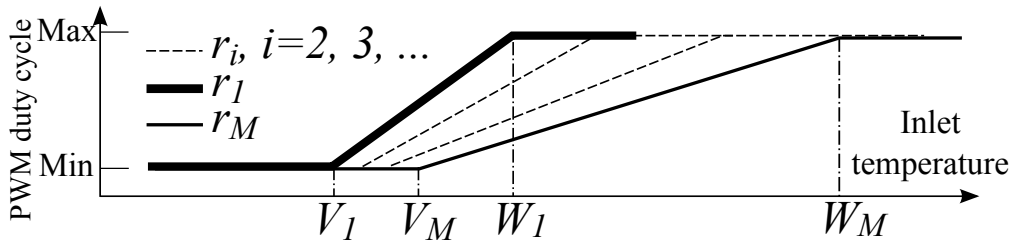


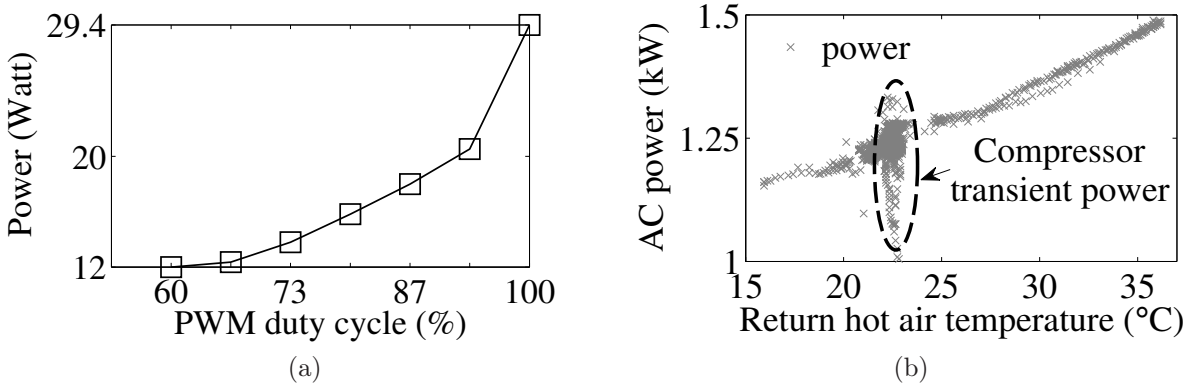
Figure 5.2 PWM duty cycle vs. server inlet temperature in fan speed control. Curve r_1 is the native fan speed control algorithm. Our new Dynamic Fan Speed Control algorithm consists of the curves $r_1, r_2, \ldots, r_M$ (cf. Section 5.4.3).

than V_1, the algorithm sets a minimal PWM duty cycle to maintain the lowest allowed fan speed. When the inlet temperature exceeds V_1, the algorithm increases the PWM duty cycle (and hence the fan speed) linearly with the inlet temperature. When the inlet temperature exceeds W_1, the algorithm sets the maximal PWM duty cycle, maintaining the full fan speed.

The relationship between the fan power and the PWM duty cycle can be estimated from either offline experiments or online measurements. We conduct an experiment to measure the power of a server fan (Delta Electronics BFB1012EH) under various PMW duty cycles. The results are shown in Figure 5.3(a). Let $D(T)$ denote the PWM duty cycle determined by the native control algorithm, where T is the inlet temperature. We adopt a discrete-time model with equal time steps. Let t_n denote the time instant of the nth time step. For the lth server, let $P_{Fl}(D(T_l(t_n))$ denote the instantaneous power of its fan at time instant t_n, where T_l is the inlet temperature. Therefore, with the predicted inlet temperature $T_l(t_{n+k})$ at time instant t_{n+k}, the predicted server fan instantaneous power is given by $P_{Fl}(D(T_l(t_{n+k}))$, which is abbreviated as $P_{Fl}(t_n, k)$ hereafter. The notation used in this chapter is summarized in Appendix B.

5.3.2 CRAC Power Consumption Model

The power of a CRAC system is mainly determined by its temperature setpoint, blower speed, and return hot air temperature. At time instant t_n, the power consumption of the jth CRAC system is denoted as $P_{Cj}(S_j(t_n), B_j(t_n), T_{Hj}(t_n))$, where $S_j(t_n)$, $B_j(t_n)$, and $T_{Hj}(t_n)$



(a) (b)

Figure 5.3 (a) Power of a server fan vs. PWM duty cycle; (b) AC power vs. return hot air temperature.

represent the temperature setpoint, the blower speed, and the return hot air temperature for this CRAC system. For a CRAC system with continuous setpoint and blower speed, the parameters of P_{Cj} can be obtained by interpolation based on offline measurements. With the predicted return hot air temperature $T_{Hj}(t_{n+k})$ at time instant t_{n+k}, the predicted instantaneous power of the jth CRAC system is given by $P_{Cj}(S_j(t_{n+k}), B_j(t_{n+k}), T_{Hj}(t_{n+k}))$, which is abbreviated as $P_{Cj}(t_n, k)$ hereafter.

As an example, we empirically study the power consumption model of a Tripp Lite SRCOOL12K air conditioner (AC), which is specially designed for data centers. The resulted model will be also used in our testbed experiments in Section 5.6. This AC has three selectable blower speeds and a maximum cooling power of 1.5 kW. Figure 5.3(b) shows that the power of this AC almost linearly increases with the return hot air temperature when the compressor is not in a transient state. When the compressor is off, its power is almost constant (not shown in Figure 5.3(b)) if the blower speed is fixed. Moreover, as different blower speeds result in less than 10 W power difference regardless of the compressor status, we round up the power consumption of the blower to its maximal value to simplify the model. When the compressor transits from off to on, there is a spike of transient power. However, this spike lasts for about 15 seconds only and takes about 0.5% of the total energy under normal settings. Therefore, it is neglected in our power consumption model. This AC does

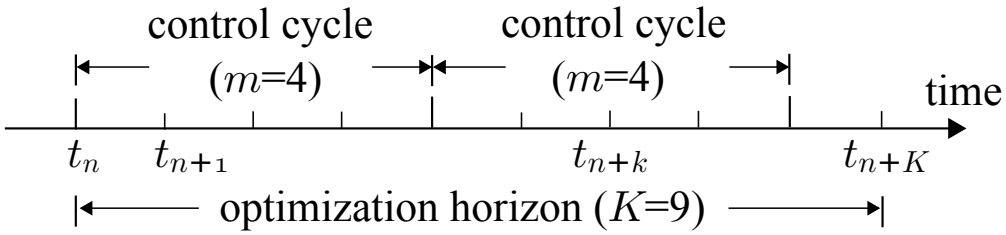


Figure 5.4 Predictive control scheme.

not allow us to program the temperature setpoint. Therefore, for this particular AC, we only control the on/off states of the blower and the compressor. As a result, the status of this AC can be represented by two binary variables, which are the compressor status $S \in \{0, 1\}$ and the blower status $B \in \{0, 1\}$, where 0 and 1 represent off and on states. The instantaneous power of this AC can be described by $P_C = B \cdot [S \cdot (\omega_1 T_H + \omega_0) + \omega_2]$, where the parameters ω_0, ω_1, and ω_2 can be estimated from the data shown in Figure 5.3(b).

5.4 Design of PTEC

5.4.1 Problem Formulation

PTEC adopts a predictive control scheme illustrated in Figure 5.4. Suppose the current time instant is t_n. The time interval between t_n and t_{n+1} is referred to as a *time step*, which is the period for sensor sampling and temperature prediction. Our system collects measurements from sensors at the beginning of every time step. A *control cycle* is defined as m consecutive time steps. At the beginning of each control cycle, PTEC determines the CRAC settings and the server fan speeds to minimize the predicted overall power consumption of the CRAC systems and server fans subject to a set of thermal requirements during the following K time steps (i.e., from t_n to t_{n+K}), where K is the *optimization horizon*. Based on this scheme, this section formulates the predictive control problem.

For a total of J CRAC systems and L servers, the predicted average power consumption

during the future K time steps is

$$\overline{P(t_n)} = \frac{1}{K}\sum_{k=1}^{K}\left(\sum_{j=1}^{J} P_{Cj}(t_n,k) + \sum_{l=1}^{L} P_{Fl}(t_n,k)\right), \tag{5.1}$$

where P_{Cj} is the power consumption of the jth CRAC system and P_{Fl} is the fan power consumption of the lth server. We note that the server temperatures also affect the leakage powers of server electronics. However, their temperature-induced changes are negligible compared to the power consumption of server fans [69]. Therefore, our objective function in Eq. (5.1) does not account for server leakage power.

PTEC enforces that the servers will not be overheated in the future K time steps. Let $T_{l,k}$ and T_U denote the inlet temperature of the lth server at time instant t_{n+k} and the *maximum allowed temperature* (MAT) at any server inlet, respectively. PTEC aims to ensure $T_{l,k} < T_U$, $\forall l \in [1, L]$ and $\forall k \in [1, K]$. A challenge in the design of any predictive control system is how to cope with prediction errors [84][40]. Let $\hat{T}_{l,k}$ denote the predicted inlet temperature for the lth server at the prediction horizon k. We assume that the prediction error (i.e., $\hat{T}_{l,k} - T_{l,k}$) follows the normal distribution $\mathcal{N}(\mu_{l,k}, \sigma^2_{l,k})$, which will be empirically verified in Section 5.4.2. Thus, the actual temperature $T_{l,k} \sim \mathcal{N}(\hat{T}_{l,k} - \mu_{l,k}, \sigma^2_{l,k})$. PTEC requires that the actual temperature $T_{l,k}$ is lower than T_U with a confidence level α, i.e., $\Pr(T_{l,k} < T_U) > \alpha$. Therefore, the upper bound for $\hat{T}_{l,k}$, denoted by $\widetilde{T}_{l,k}$, can be derived as

$$\widetilde{T}_{l,k} = T_U + \mu_{l,k} - \sigma_{l,k} Q^{-1}(1-\alpha), \tag{5.2}$$

where $Q(\cdot)$ is the Q-function of the standard normal distribution, i.e.,

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^{+\infty} \exp(-\frac{z^2}{2}) \mathrm{d}z.$$

Note that the parameters $\mu_{l,k}$ and $\sigma^2_{l,k}$ should be estimated for each server l and prediction horizon k. They can be continuously updated using the most recent historical measurements.

A data center should be prevented from significant temperature variation. The probability of a server outage can be doubled when the temporal temperature variation increases

by 50% [69]. PTEC computes the temperature variation over a moving window with size of $w \geq K$, from time instant $t_{n-(w-K)}$ to t_{n+K}. The values before the time instant t_n are historical measurements and the values after that are predicted temperatures. We use the *relative standard deviation* (RSD) to quantify the temperature variation, i.e., $\text{RSD} = \sigma_T/\mu_T$, where σ_T and μ_T are the standard deviation and mean of the temperatures in the moving window. We denote RSD_l the RSD for the lth server. PTEC requires that the RSD of each server is upper-bounded by a constant RSD_U specified by the data center operator. For instance, the setting $\text{RSD}_U = 0.04$ can maintain a satisfactorily low fluctuation-induced hardware error rate [69].

We now formally formulate the thermal and energy control problem that minimizes the power consumption of CRACs and server fans. For a total of J CRACs and L servers, let $\mathbf{D}$, $\mathbf{S}$, and $\mathbf{B}$ denote the vectors of server fan PWM duty cycles, CRAC temperature setpoints and blower speeds over the optimization horizon, respectively, i.e., $\mathbf{D} = [D_1, \ldots, D_L]$, $\mathbf{S} = [S_1, \ldots, S_J]$, and $\mathbf{B} = [B_1, \ldots, B_J]$. We formulate the problem as follows:

Problem 1. *To find* $\mathbf{D}$*,* $\mathbf{S}$*, and* $\mathbf{B}$ *to minimize the predicted average power consumption given by Eq. (5.1), subject to that,* $\forall l \in [1, L]$*,* $\forall k \in [1, K]$*,*

1. *the predicted inlet temperatures are lower than an upper bound:* $\hat{T}_{l,k} \leq \widetilde{T}_{l,k}$*, where* $\widetilde{T}_{l,k}$ *is given by Eq. (5.2); and*

2. *the* RSD*s of the inlet temperature are lower than an upper bound:* $\text{RSD}_l \leq \text{RSD}_U$*.*

PTEC solves Problem 1 at the beginning of each control cycle and controls the CRAC systems and server fans according to the solution. Problem 1 is a non-linear constrained optimization problem with prohibitive computational complexity due to the complex thermal interactions between CRAC systems and server fans. In particular, the exhaustive search has an exponential complexity with respect to the total number of CRAC systems and server fans. To make the problem tractable and achieve satisfactory real-time performance,

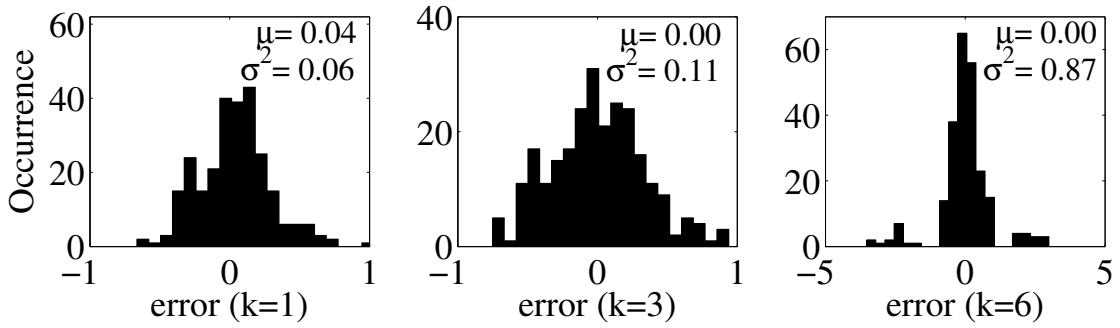


Figure 5.5 Histograms of prediction errors with $k = 1$, 3, and 6. (time step = 30s)

we propose a *coordinated control* approach. Specifically, the servers will control the fan speeds autonomously according to server inlet temperatures and CPU utilizations, by using an algorithm in Section 5.4.3. Thus, the variables of Problem 1 are reduced to $\mathbf{S}$ and $\mathbf{B}$ only. It is important to note that, when PTEC assesses each candidate solution $\langle \mathbf{S}, \mathbf{B} \rangle$, the resulted fan speeds under the autonomous control algorithm will be used to calculate the predicted average power consumption, inlet temperatures, and RSDs. Thus, this coordinated control approach accounts for the interdependence between server fans and CRAC systems. Therefore, it does not substantially degrade the solution quality.

5.4.2 Real-Time Temperature Prediction

PTEC integrates the real-time data-driven temperature prediction system proposed in Chapter 4, which predicts server inlet temperatures based on cyber and physical status of the data center. The input of the system includes the temperatures at a set of selected locations, e.g., server inlets and CRAC hot air return registers, measured by either a deployed wireless sensor network or server built-in sensors. For a total of D temperatures, we define the temperature distribution $\mathbf{T} = [T_1; T_2; \ldots; T_D] \in \mathbb{R}^{D \times 1}$, where T_d is the temperature at the dth location in the data center. The temperature distribution is predicted by a set of *thermal variable*s that significantly affect $\mathbf{T}$ and are monitored by sensors. They include the CRAC setpoints $\mathbf{S}$ and blower speeds $\mathbf{B}$, server fan speed control settings $\mathbf{R}$, and CPU utilizations $\mathbf{U}$. Moreover, the historical temperature distributions also largely

affect the temperature distributions in the near future. Therefore, we define the *state* of the monitored data center at a time instant, denoted by $\mathbf{p}$, as a collection of thermal variables, i.e., $\mathbf{p} = [\mathbf{S}; \mathbf{B}; \mathbf{R}; \mathbf{T}; \mathbf{U}]$. The state $\mathbf{p}$ is measured every time step. At the beginning of a control cycle, the temperature distribution at time instant t_{n+k}, denoted by $\mathbf{T}(t_{n+k})$, is predicted based on the most recent R states. By setting increasing prediction horizon k, the system predicts the temporal evolution of $\mathbf{T}$. This machine-learning-based prediction system, which is trained with data from real sensors and offline CFD simulations, achieves a desirable trade-off between prediction fidelity and computation overhead. Moreover, its high-speed feature allows PTEC to iteratively search for the best control solution (cf. Section 5.4.4).

Figure 5.5 shows the histograms of temperature prediction errors for a server inlet. The errors can be well characterized by normal distributions. Consistent with intuition, the error variance increases with the prediction horizon. The error means and variances for different prediction horizons are used in Eq. (5.2).

5.4.3 Dynamic Fan Speed Control

The main purpose of server fan is to prevent the internal electronic components, e.g., CPU, from overheating. A key drawback of the native fan speed control approach discussed in Section 5.3.1 is the neglect of the server status (e.g., CPU utilization) that also affects component temperatures. For instance, the fan may run at an unnecessarily high speed when the server is idle. Feedback fan control has been proposed to reduce the fan energy consumption [27]. However, the fan speeds under the feedback control are often unpredictable, making it difficult to model and minimize the total system energy consumption.

Our new fan speed control approach, called Dynamic Fan Speed Control (DFSC), jointly considers CPU utilization and inlet temperature. Figure 5.6 shows the minimal fan speeds (in PWM duty cycle) to meet two given upper bounds of CPU temperature (46°C and 40°C) versus server inlet temperature, under various CPU utilizations. We can observe that the minimal fan speed has a near-linear relationship with the inlet temperature. More

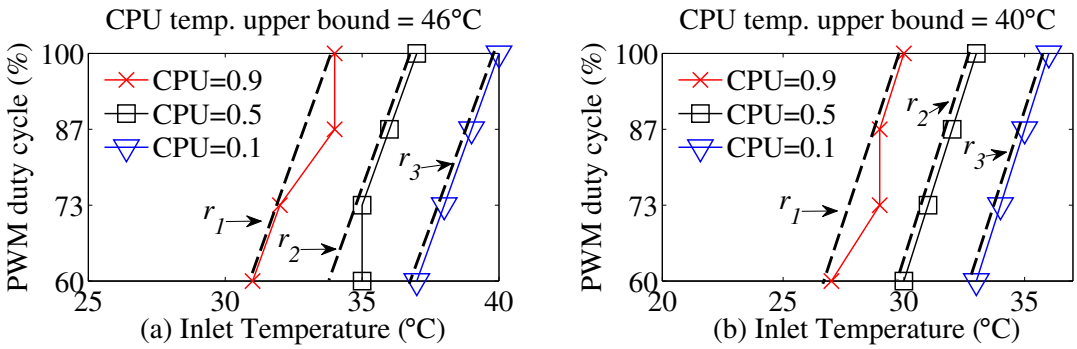


Figure 5.6 Minimal required PWM duty cycle (marked curve) vs. server inlet temperature under various CPU utilization. Sub-figures (a) and (b) are the results for different CPU temperature upper bounds (46°C and 40°C). A DFSC setting r_i comprises two endpoints of a dashed line.

importantly, such a relationship varies with the CPU utilization. Thus, DFSC reuses the native fan speed control algorithm but adjusts its setting in response to the CPU utilization changes while meeting a CPU temperature upper bound requirement. Specifically, DFSC discretizes the CPU utilization into M levels. The first level represents full utilization while the Mth level represents idle. Each level is mapped to a setting of two thresholds of the native control algorithm. As illustrated in Figure 5.2, the setting for the ith CPU utilization level is denoted by $r_i = \langle V_i, W_i \rangle$, where $i \in [1, M]$. When the CPU utilization is at the ith level, the native algorithm will be invoked with the setting r_i. Under the setting r_1, the CPU temperature upper bound should be met when the server is fully utilized. Similarly, under the setting r_M, the CPU temperature upper bound should be met when the server is idle. The settings $\{r_1, r_2, \ldots, r_M\}$ are hardware-dependent and can be empirically measured, e.g., through offline experiments, or provided by hardware manufacturers. The servers can also run an online feedback fan controller [27] for a certain period of time to measure these settings when the CPU temperature is stable. Once the settings are measured, the servers can start using DFSC. In Figure 5.6, the endpoints of dashed lines show the DFSC settings with $M = 3$.

5.4.4 Predictive Controller

In our current implementation of PTEC, we use the constrained simulated annealing (CSA) algorithm [85] to search for the CRAC setting (**S** and **B**) that minimizes Eq. (5.1) subject to the thermal safety requirements. The CSA algorithm is more efficient than the brute-force search and can asymptotically converge to the optimal solution [85]. For instance, for 229 servers and 4 CRAC units, each of which has 6 different states of setpoint and blower speed, it takes the CSA algorithm only 5 seconds on a 3.4GHz desktop computer to converge to a near-optimal solution with an optimization horizon of 8.

We now discuss the settings of control cycle m and optimization horizon K. First, intuitively, the system with a short control cycle can respond to the thermal condition changes quickly. However, a short control cycle allows less time for solving the optimization problem. Second, it is desirable to set a larger optimization horizon K such that the predictive controller accounts for a longer period of thermal dynamics into the future. However, its setting must also consider both the prediction accuracy and the controller's computational overhead. Therefore, the settings for m and K should achieve a satisfactory trade-off between control quality and computational overhead. In our testbed experiment with 6 servers and a portable AC, the control decision can be computed within a second when K is set to 6 to 9 minutes. Therefore, we set $m = 1$ to 3 minutes since the AC should not be switched frequently. Under these settings, nearly 90% of prediction errors are within 1°C.

5.4.5 Scalable Partition-Based Predictive Controller

Due to the non-convexity of Problem 1, the computational overhead of CSA increases exponentially with the number of CRAC systems. The resulted delay may jeopardize the real-time performance of PTEC for large-scale data centers. This section presents a partition-based algorithm that can significantly reduce the computational overhead while maintaining satisfactory solution quality. It consists of an *offline stage* and an *online stage*. The offline stage partitions the data center into several regions based on the thermal correlation index

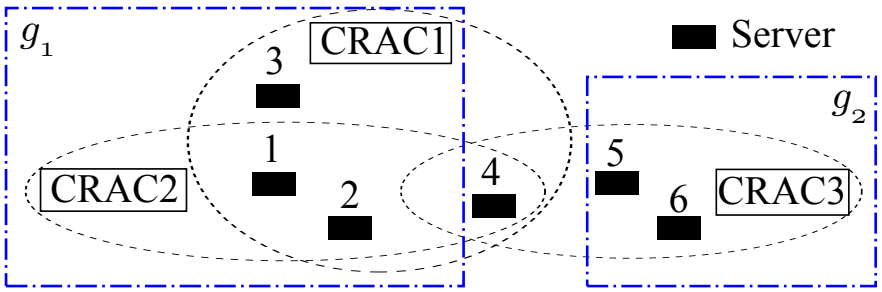


Figure 5.7 Example of partitioning. The servers within an oval are associated with the CRAC in the oval. Region g_2 contains CRAC3 only since Server5 and Server6 are associated with CRAC3 only. CRAC1 forms a region since Server3 is associated with CRAC1 only. No servers are associated with CRAC2 exclusively. Therefore, CRAC2 will be merged with CRAC1 to form region g_1.

(TCI) [24], which characterizes the cooling effectiveness for a location provided by a CRAC system. The online stage solves Problem 1 within each region, and iteratively revises the sub-solution for each region to meet the thermal requirements of the servers out of any regions.

The TCI for the lth server and the jth CRAC system is defined as $\gamma_{l,j} = \frac{\Delta T_l}{\Delta T_{Cj}}$, where ΔT_{Cj} denotes a step change of the supply air temperature of the jth CRAC system and ΔT_l is the resulted steady temperature change at the lth server inlet. A larger $\gamma_{l,j}$ indicates a stronger capability of the CRAC system to remove the heat from the server. TCI can be experimentally measured by perturbing CRAC setpoints or numerically obtained from Computational Fluid Dynamics simulations [24]. The lth server and the jth CRAC system are *associated* if $\gamma_{l,j} \geq \lambda$, where $\lambda \in (0, 1)$ is a threshold specified by the system operator. Thus, the temperature at the lth server's inlet is mainly affected by its associated CRAC systems. For instance, in Figure 5.7, Server5 and Server6 are associated with CRAC3 only, and their inlet temperatures are mainly affected by CRAC3.

The offline stage partitions the data center into a number of regions based on the TCIs. For a region denoted by g, let C_g denote a set of CRAC systems in this region and E_g denote a set of servers that are associated with the CRAC systems in C_g *exclusively*. Formally, $E_g = \{l|\gamma_{l,j} \geq \lambda, \exists j \in C_g, \forall l\} \cap \{l|\gamma_{l,j} < \lambda, \forall j \notin C_g, \forall l\}$. For example, the two dash-dotted rectangles in Figure 5.7 show two such regions, denoted as g_1 and g_2. In this example,

$C_{g_1} = \{\text{CRAC1}, \text{CRAC2}\}$, $C_{g_2} = \{\text{CRAC3}\}$, $E_{g_1} = \{\text{Server1}, \text{Server2}, \text{Server3}\}$, $E_{g_2} = \{\text{Sever5}, \text{Server6}\}$. We note that some servers that are associated with many CRAC systems may be out of any E_g, e.g., Server4 in Figure 5.7. These servers are called *ungrouped* servers. Let C and $\mathbb{G}$ denote the set of all CRAC systems and the set of all regions. The offline stage looks for a partition scheme to minimize $\max_{\forall g \in \mathbb{G}} |C_g|$ subject to 1) $\bigcup_{g \in \mathbb{G}} C_g = C$, 2) $C_g \cap C_h = \emptyset$, $\forall g \neq h$, and 3) $E_g \neq \emptyset$, $\forall g \in \mathbb{G}$. As we will solve Problem 1 within each region g in the online stage, the objective of minimizing the maximum number of CRAC systems in any region significantly reduces the computation overhead of the online stage. We develop a heuristic algorithm based on an existing Maximum-Weight Independent Set (MWIS) algorithm [86] to solve this partitioning problem. In Appendix C.1, we use an example to illustrate our algorithm.

Based on the regions partitioned by the offline stage, the online stage solves Problem 1. Algorithm 1 shows the pseudocode of the online stage algorithm. Initially, for each region g, Problem 1 is solved for C_g and E_g using the CSA algorithm. The solution for a region is called a sub-solution. The initial solution for the data center thus comprises all sub-solutions. However, this solution may not meet the thermal requirements for the ungrouped servers. The online stage iteratively updates the solution by solving Problem 1 for each region plus all these ungrouped servers. As an ungrouped server is cooled by the CRAC systems from multiple regions, its MAT constraint can be relaxed in each region where it is associated with a CRAC system. The relaxation in each iteration is as follows. For a prediction horizon k, the predicted inlet temperature $\hat{T}_{l,k}$ of the lth ungrouped server can be computed based on the solution in the previous iteration. If the predicted temperature $\hat{T}_{l,k}$ exceeds the upper bound $\widetilde{T}_{l,k}$ given by Eq. (5.2), the deviation $\hat{T}_{l,k} - \widetilde{T}_{l,k}$ characterizes the amount of extra heat that needs to be removed from the lth server inlet[87]. PTEC allocates this extra heat to each region g proportionally based on the total TCI of all CRAC systems in g. This is achieved by relaxing the MAT constraint for the lth ungrouped server in each region g as $\widetilde{T}^g_{l,k} = \hat{T}_{l,k} - \frac{\sum_{j \in C_g} \gamma_{l,j}}{\sum_{j \in C} \gamma_{l,j}} \cdot (\hat{T}_{l,k} - \widetilde{T}_{l,k})$. Problem 1 is solved for each region g with the relaxed

MAT constraint $\widetilde{T}^g_{l,k}$ for each ungrouped server l. The above process is repeated until all ungrouped servers' thermal requirements are satisfied. The pseudocode of the online stage and its convergence proof can be found in Appendix C.2.

In the partition-based predictive controller, Problem 1 is solved for each small region, resulting in significantly lower computation overhead compared with that of solving Problem 1 for the whole data center. In Section 5.6.4, we will evaluate the overhead and effectiveness of this approach.

5.5 Implementation

5.5.1 Testbed and Sensor Deployment

We implemented PTEC on a single-rack testbed shown in Figure 5.8. It consists of a rack of 15 1U [1] servers in a 5×6 square feet room insulated by foam boards. On the rack, 15 servers are grouped every three servers with a 2U distance between every adjacent two groups. Each server is equipped with 2 PWM-controlled fans (Delta Electronics BFB1012EH) to cool the internal components. Each fan consumes a maximum of 29.4 W input power and the two fans contribute up to 25% of total power consumption of a server. Each server also has three on-board temperature sensors to monitor the CPU and server inlet temperatures. A Tripp Lite portable AC (SRCOOL12K) with a rated power of 3.5 kW is placed aside the server rack within the room. To enable its automatic control, we connect it to several power relays, which can be remotely switched by a program. Its return hot air register faces the side of server outlets, drawing the hot air. It delivers cold air through a register located at the bottom of the room in front of the rack, which is consistent with the popular raised floor cooling design in production data centers. However, due to this AC's limited cooling capacity, up to 6 servers can be running at the same time in our experiments. A total of 15 Iris temperature sensors are mounted with brackets at the outlets of the 5 group of servers.

[1] U is the unit of the height of a server, which is 1.75 inches.

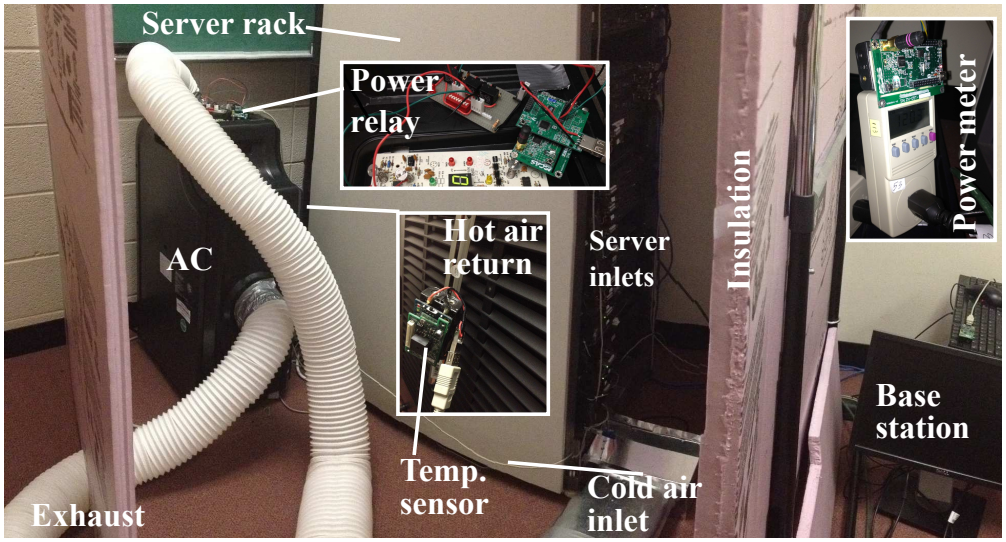


Figure 5.8 A single-rack testbed that consists of a base station, a portable AC, a rack of 15 servers, and a total of 23 temperature/power sensors.

To monitor the AC status, we place an Iris temperature sensor at the AC cold air register and another at the hot air return register. To measure the power consumption of the servers and AC, we attach a wireless power meter for each server and AC. This small testbed allows us to study the fine-grained performance of PTEC.

5.5.2 System Implementation

Our system consists of two data collection networks and a base station that runs the predictive controller. The base station collects the data from the wireless sensors connected via 802.15.4 wireless links and the internal server sensors (inlet temperature sensor and fan speed sensor) over the Ethernet. It also runs the optimization algorithm and sends the control commands to the AC. The data collection is implemented with JAVA on the base station, while the temperature prediction and the predictive controller are implemented with MATLAB.

Sensor network. We use a single-hop network architecture, where the base station sends the data collection requests to the sensors sequentially and each sensor transmits the measurements back. Every 30 seconds, the base station performs a round of sequential data

collection from all sensors. Note that a multi-hop network topology can be used when more server racks are monitored. As this collection scheme works in a time-division fashion, the system experiences few collisions between the data transmissions of different sensors. The programs on all wireless sensors are implemented in TinyOS 2.1.

Server network. CPU utilization, on-board temperatures, fan speeds and DFSC settings are important thermal variables from each server. Data centers typically run various server monitoring tools (e.g., `atop`, `ganglia`) that can collect on-board sensor information. We implement a program to control and measure the CPU utilization, and report on-board temperatures and fan speeds from the `lm-sensors` utilities. The base station leverages the existing Ethernet infrastructure to collect these on-board sensor readings and DFSC settings.

Fan speed and AC control. A `GNU BASH` script running on each server implements the DFSC algorithm. A separate wireless connection is established between the base station and a TelosB mote that connects to a power relay control circuit board of the AC. When the mote receives the control signal from the base station, it turns on/off the AC.

5.6 Performance Evaluation

We evaluate the performance of PTEC with testbed experiments in Section 5.6.1 to 5.6.3, and trace-driven Computational Fluid Dynamics simulations in Section 5.6.4.

5.6.1 Effectiveness of DFSC

DFSC adopts two settings, i.e., $r_1 = \langle 28°\text{C}, 34°\text{C} \rangle$ and $r_2 = \langle 30°\text{C}, 42°\text{C} \rangle$. We employ a baseline approach that controls the fan speed solely based on the setting r_1 to meet the MAT requirement when the server is fully utilized. This baseline is consistent with the static fan speed control scheme used in most servers. As both algorithms adopt r_1 when the CPU is fully utilized, we compare them when the server is idle. The two curves in Figure 5.9(a) show the server power consumption under the two approaches. The server

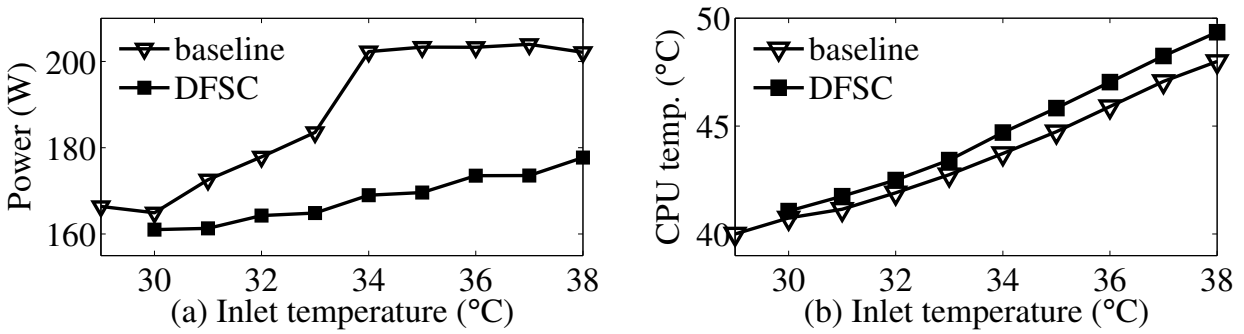


Figure 5.9 Server power and CPU temperatures under DFSC and the baseline approach when the server is idle.

power consumption of the baseline increases with inlet temperature much faster than DFSC. It reaches about 200 W when the temperature is 34°C. In comparison, DFSC consumes less than 180 W at the temperature of 34°C. Therefore, each individual idle server can save more than 20 W if T_U is 34°C. Moreover, as shown in Figure 5.9(b), the CPU temperatures under our DFSC approach are slightly higher than those under the baseline approach. However, they are still significantly lower than the maximum allowed temperature of the CPUs (69°C). This result shows that DFSC can save significant amount of energy on the idle or low utilized servers.

5.6.2 Effectiveness of Predictive Controller

In this experiment, we compare PTEC with two baselines. The first baseline, referred to as *max cooling*, sets a fixed low CRAC setpoint while the server fans maintain the full speed. This baseline provides the maximum cooling capability to prevent overheating. The second baseline, referred to as *reactive control*, applies DFSC to control server fans and a hysteresis principle to control the AC reactively, so that the inlet temperatures are maintained within a range. Let $R(T^U, T^B)$ denote a reactive control strategy specified by two parameters, i.e., the temperature upper bound T^U and the temperature band T^B. Specifically, when the inlet temperature exceeds T^U, the AC starts to work until the inlet temperature is reduced to $T^U - T^B$. Then, the AC is turned off.

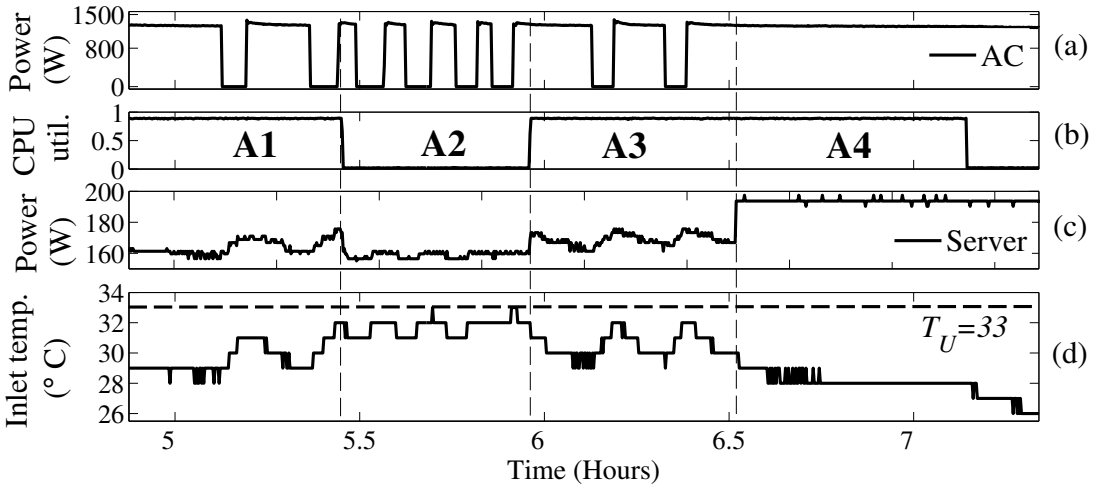


Figure 5.10 Evolution of PTEC and *max cooling* baseline on a server. (a) AC power; (b) CPU utilization; (c) Server power excluding non-idle CPU power; (d) Server inlet temperature.

Table 5.1 Average power consumption (Watt)

	A1	**A2**	**A3**	**A4**
AC power	903	639	931	1254
Server power	1065	1035	1077	1279

5.6.2.1 Comparison with max cooling

Figure 5.10 shows the comparison between PTEC and the max cooling scheme. The optimization horizon $K = 12$ (6 minutes), the control cycle length $m = 2$ (1 minute), and the MAT constraint $T_U = 33$°C. We intentionally use a large RSD_U to study the effect of MAT constraint. The entire experiment comprises four periods, i.e., A1 to A4. Periods A1 to A3 run PTEC and Period A4 runs the max cooling. In Period A1, the CPU is fully utilized (Figure 5.10(b)), which generates significant heat and may cause fast temperature rise. In response to this high CPU utilization, DFSC configures server fans to r_1, resulting in an average server power consumption of 170 W (Figure 5.10(c)). Note that, in Figure 5.10(c), CPU power consumption is not included and the power fluctuations are mostly caused by the fan speed changes. In Figure 5.10(d), we can see that the MAT constraint is satisfied. When the system enters Period A2, the server switches to idle state. With lower CPU utilization, DFSC configures server fans to r_2, resulting in a relatively low server power consumption

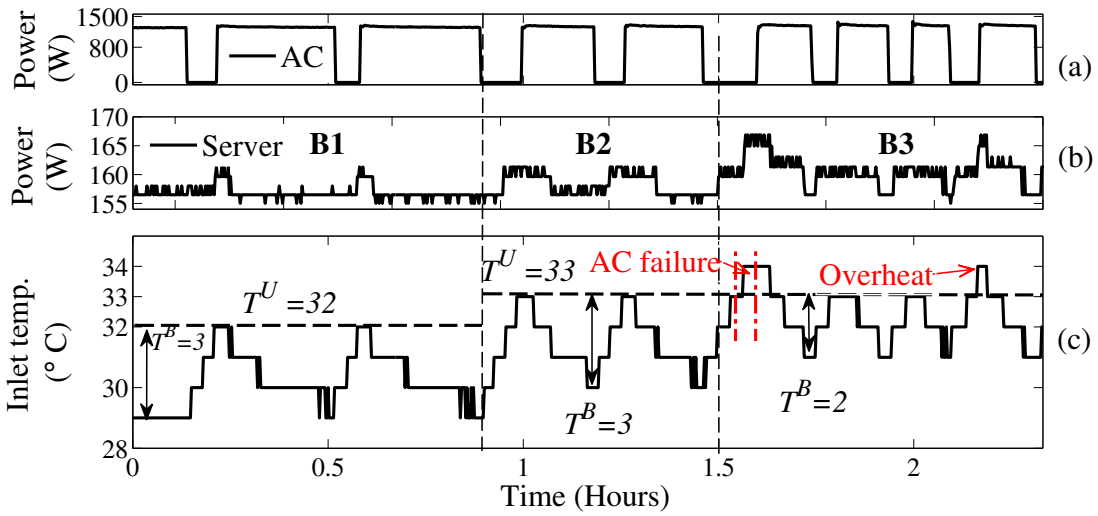


Figure 5.11 Reactive control approach when the servers are idle. (a) AC power; (b) Server power excluding non-idle CPU power; (c) Server inlet temperature. Three periods are marked from B1 to B3.

(Figure 5.10(c)). Since the CPU utilization is low in Period A2, the system maintains a higher inlet temperature without overheating the servers. Thus, the AC can be turned off more frequently to save energy. In Period A3, the server CPU utilization becomes high again and the inlet temperature is maintained at a lower level. In Period A4, we apply the max cooling approach. Table 5.1 shows the average AC and server power consumption during each period. PTEC in Period A1 (i.e., full CPU utilization) and A2 (i.e., server idle) reduces total power consumption by 22% and 34%, respectively, compared with the max cooling in Period A4.

5.6.2.2 Comparison with reactive control

Figure 5.11 shows the experiment results for the reactive control. We start the control with $T^U = 32$°C and $T^B = 3$°C. At about 50 minutes, we increase T^U to 33°C. Under both settings, the inlet temperature remains below T_U. After 1.5 hours, we set $T^B = 2$°C. An unexpected AC failure occurred in this experiment, during which the AC failed to respond to the turning-on request due to a wireless link disconnection. However, this failure lasts

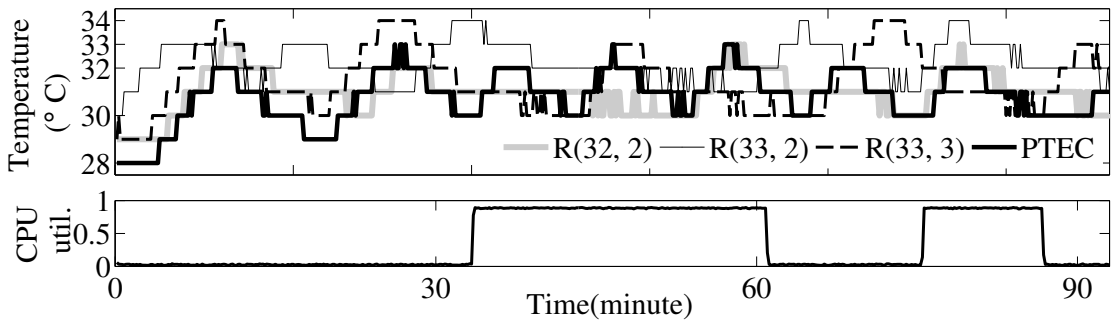


Figure 5.12 Inlet temperature under PTEC and reactive control. $R(T^U, T^B)$ denotes a reactive control baseline with settings T^U and T^B.

for 3 minutes only and does not affect the rest of the experiment. After about 2.3 hours, the temperature exceeds T_U for 2 minutes even if the AC has been turned on to react to overshooting T^U. Thus, for the reactive control approach to cope with the dynamic heat generated in a data center, T^U should be sufficiently low and T^B should be sufficiently large. Such conservative settings often lead to overcooling and excessive power consumption.

Figure 5.12 shows the results of the inlet temperature of a server under PTEC and the reactive control approach with $T_U = 33°C$. Control cycle length is set to $m = 6$ (3 minutes). It can be seen that PTEC consistently maintains the temperature below T_U. The reactive control with $T^U = 33°$, however, exceeds the MAT constraint T_U frequently. By lowering the T^U to 32°C, the reactive control can maintain the temperature below T_U. Note that as all approaches adopt the DFSC, their fan power consumptions are similar. Figure 5.13 shows the AC power consumption of PTEC and the reactive control approach when the servers are idle. PTEC can reduce the power by up to 30%. In addition, PTEC also reduces the power by up to 20% when the servers are fully utilized.

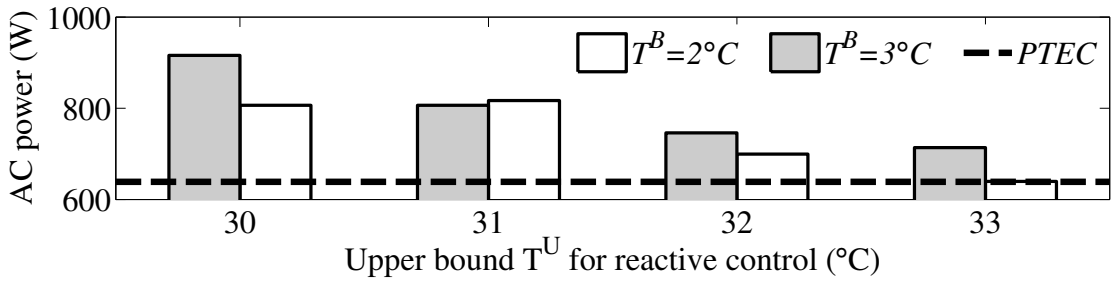


Figure 5.13 AC power consumption of reactive control baselines and PTEC when the server is idle.

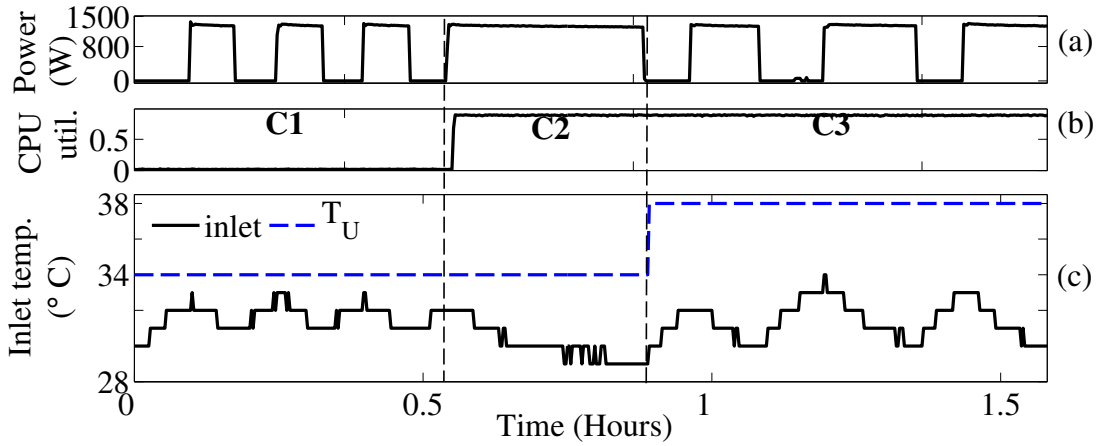


Figure 5.14 Impact of MAT. (a) AC power; (b) CPU utilization; (c) Server inlet temperature. Three periods are marked from C1 to C3.

5.6.3 Impact of Predictive Controller Settings

5.6.3.1 MAT

This section evaluates the impact of T_U on PTEC. We set $K = 18$ (9 minutes). In Period C2 of Figure 5.14, we configure all the servers to be fully utilized. With $T_U = 34$°C, the system turns on the AC to reduce the inlet temperature in response to a predicted temperature rise. At the beginning of Period C3, we intentionally reconfigure $T_U = 38°$ to allow increased inlet temperature due to the full utilization of servers. In this period, the AC is on intermittently, resulting in a higher average inlet temperature than that in Period C2. However, PTEC does not increase the inlet temperature close to T_U, because, otherwise, the increased server fan power due to higher speeds will cancel the energy saving of the AC by

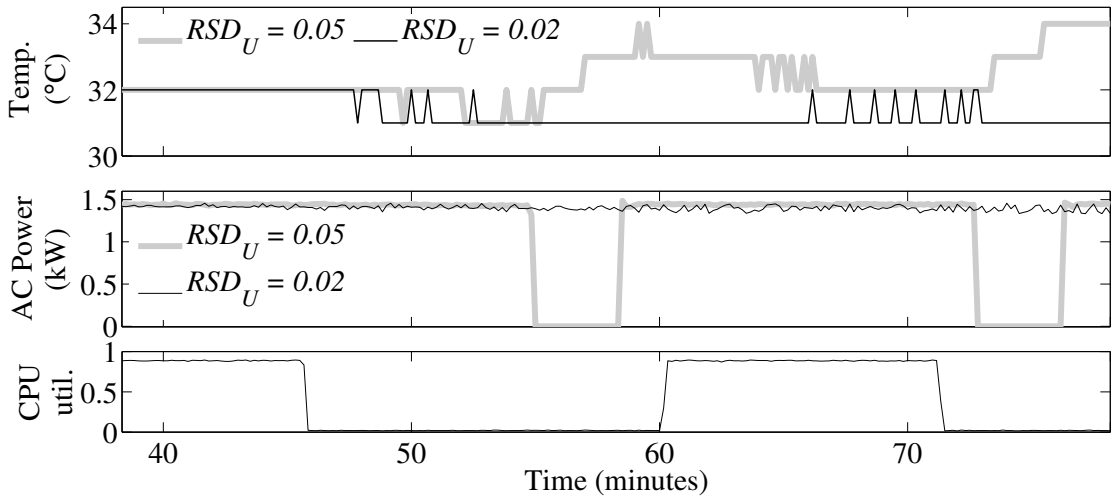


Figure 5.15 Evolution of inlet temperature under different RSD requirements.

reducing working time.

5.6.3.2 RSD_U

We now evaluate the impact of RSD_U setting. The RSD computing window size w is set to 36 (18 minutes). Other settings are: $m = 6$, $K = 12$, $T_U = 35°\text{C}$. The RSD_U is set to either 0.05 or 0.02 to evaluate its impact. Figure 5.15 shows the results of an inlet temperature under different settings of RSD_U. In the first 45 minutes, the servers are fully utilized and PTEC cannot turn off the AC without exceeding T_U. After 45 minutes under the setting $\text{RSD}_U = 0.05$, PTEC turns off the AC since the servers are idle, without violating the constraints on inlet temperatures and their RSDs. However, under the setting $\text{RSD}_U = 0.02$, the AC has to remain on to avoid temperature variations. This result suggests that an inappropriately small RSD_U may restrain the AC from changing operating status, thus eliminating the opportunity for energy saving. As suggested in [69], $\text{RSD}_U = 0.05$ is a practical setting for real data centers.

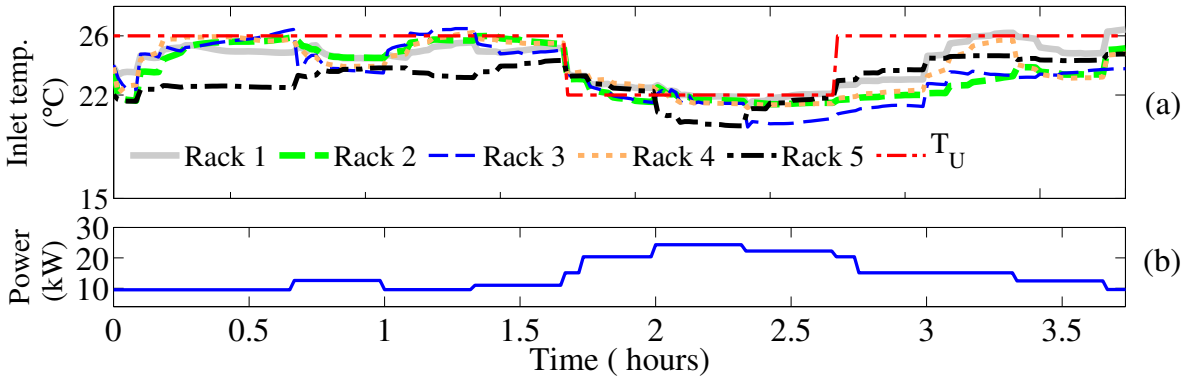


Figure 5.16 PTEC in CFD simulations. (a) Average server inlet temperatures of each server rack; (b) CRAC power.

5.6.4 Trace-Driven Computational Fluid Dynamics Simulations

In addition to the testbed evaluation, we study the performance of PTEC by Computational Fluid Dynamics (CFD) simulations driven by real workload data traces collected from the High-Performance Computing Center (HPCC) of Michigan State University. We use a commercial CFD software (ANSYS Fluent) to model a part of this data center, which hosts five racks of totally 229 servers. The racks are arranged in two rows with a cold aisle between them. Two in-row CRAC systems are installed for each row. Due to the lack of a complete CFD model of HPCC, we model each CRAC system as a cooling unit with three discrete cooling powers, i.e., 24 kW (rated power of each CRAC system in HPCC), 17 kW, and 9.6 kW. Thus, PTEC selects a cooling power instead of tuning temperature setpoint. Moreover, each CRAC system has two blower speeds. The simulations are driven by the server workload traces of these 229 servers. A time step is 5 minutes. Other settings are: $m = 4$ (20 minutes), $K = 8$ (40 minutes), and $w = 18$ (1.5 hours).

Figure 5.16(a) shows the results of server inlet temperatures under PTEC. Initially, we set $T_U = 26°C$. We can see that PTEC controls the server inlet temperatures at around 26°C. At about 1.3 hours and 2.3 hours, we set $T_U = 22°C$ and $T_U = 26°C$, respectively. All inlet temperatures are maintained at around T_U. Figure 5.16(b) shows the resulting CRAC power consumption. After we increase T_U from 22°C to 26°C at about 2.3 hours, PTEC controls

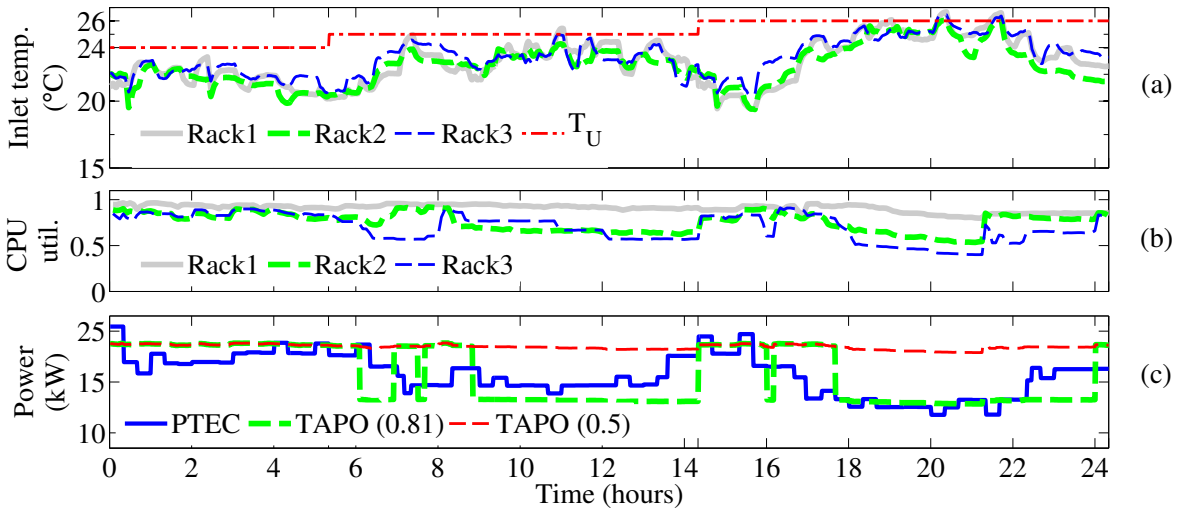
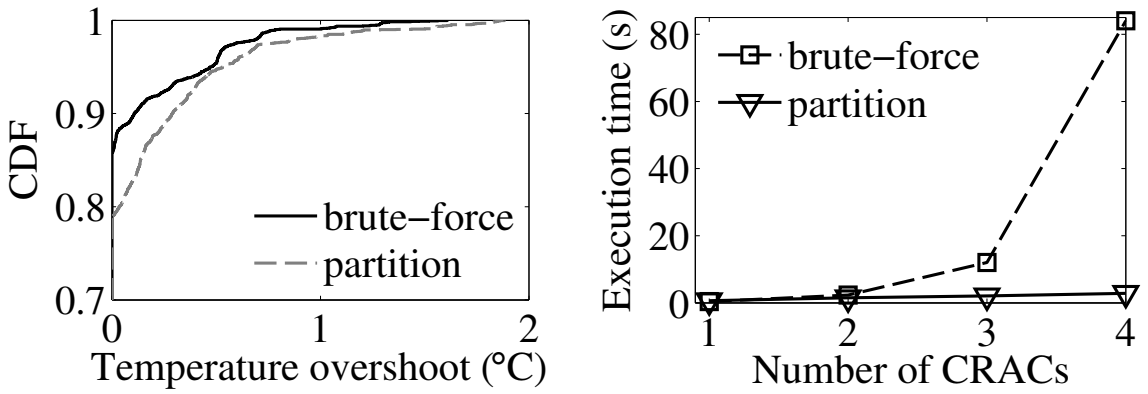


Figure 5.17 Temperature control under dynamic CPU utilizations.

the CRAC systems to gradually increase the server inlet temperatures without violating the RSD requirement, and the CRAC power consumption is reduced by up to 35%.

We now evaluate the effectiveness of PTEC under dynamic CPU utilization. The real CPU utilization traces span 96 hours. We select a portion of 24 hours that exhibit the highest dynamic levels to drive the simulation. For clear illustration, Figure 5.17 shows the results of only three racks. The average CPU utilizations of these three racks are shown in Figure 5.17(b). Initially, we set $T_U = 24°C$. Since the average utilizations of all three racks are high, PTEC keeps relatively low inlet temperatures to prevent overheating. At the 5.2 hours, we increase T_U to 25°C. Since the CPU utilization is still high, PTEC cannot further increase the inlet temperatures. After about 6 hours, the CPU utilization of Rack3 drops significantly. PTEC is then able to increase the inlet temperature without violating the new T_U. After 8 hours, PTEC reduces the inlet temperatures in response to the increased utilization of Rack3. After about 14 hours, we set $T_U = 26°C$. Initially, the inlet temperatures are maintained at a low level due to the high CPU utilization. After 16 hours, PTEC gradually increases the inlet temperatures close to T_U in response to the reduced CPU utilizations. This experiment shows that PTEC can well adapt to the dynamics of realistic data center



(a) (b)

Figure 5.18 (a) CDF of temperature overshoot; (b) Average execution time vs. the number of CRACs.

server workload.

We also compare PTEC with a baseline approach that is a variant of an existing representative control approach TAPO [27]. TAPO uses a fixed low CRAC temperature setpoint T_L if the CPU utilization is higher than a predefined threshold u. Otherwise, it uses a fixed high CRAC temperature setpoint T_H. In our simulations, we set $T_L = 22°\text{C}$ and $T_H = 26°\text{C}$. In [27], the threshold u is 0.5. Under this setting, as shown in Figure 5.17(c), TAPO always uses T_L since the CPU utilizations in the simulation never drop below 0.5. By setting $u = 0.81$, the cooling power consumption under TAPO is comparable to that under PTEC. This result shows that the setpoints of TAPO need to be manually tuned to achieve the desirable performance. As the CPU utilization is unpredictable in real data centers, TAPO may not well adapt to dynamic CPU utilization.

We finally evaluate the partition-based algorithm in Section 5.4.5. We partition HPCC to four regions, each of which contains one CRAC system. We compare our approach with a *brute-force* approach that exhaustively searches the optimal solution to Problem 1 for the whole data center. Figure 5.18(a) shows the Cumulative Distribution Function (CDF) of the inlet temperature overshoots over T_U in a 12-hour simulation. For the brute-force approach, 90% of temperatures do not exceed T_U, and more than 95% of all temperatures

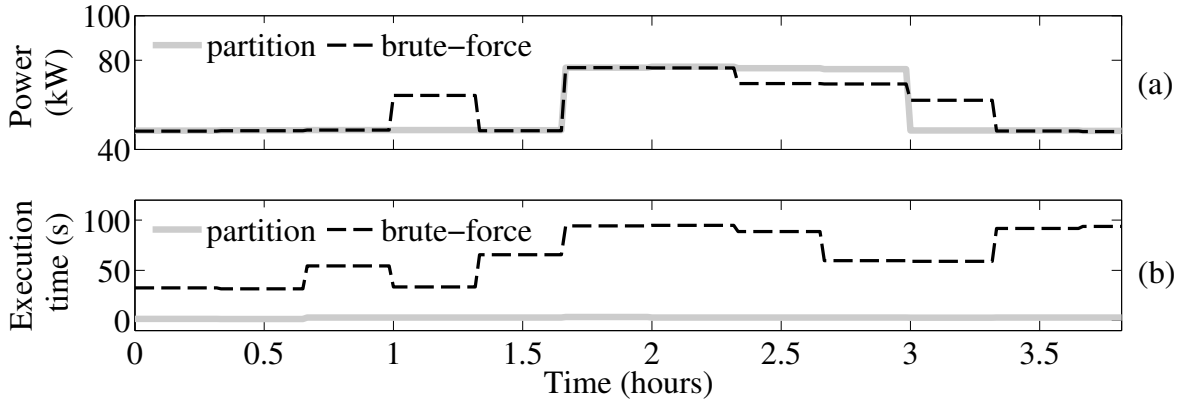


Figure 5.19 Performance comparison between the brute-force and partition-based approaches. (a) power; (b) execution time.

fall within 1°C above T_U. The performance of our approach is slightly lower than the brute-force approach. As shown in Figure 5.18(a), for a temperature overshoot of 1°C, the two approaches are comparable. In practice, we can set 1°C safety margin between the setpoint and the overheating temperature. Figure 5.19 shows that our approach achieves comparable total power consumption and reduces the execution time significantly, compared with the brute-force approach. The average execution time of our approach is only 5% of that of the brute-force approach. Moreover, Figure 5.18(b) shows that the execution time on an Intel Core i7-2600K 3.4 GHz CPU of the brute-force approach increases exponentially with the number of CRAC systems. On the contrary, the execution time of our approach increases slowly and linearly with the number of CRACs. These results show that our approach can find near-optimal solutions with satisfactory scalability. The low computational overhead enables PTEC to be implemented on portable hardware without relying on the computing infrastructure of monitored data center.

5.7 Conclusion

This chapter presents the design and evaluation of PTEC – a system for predictive thermal and energy control in data centers. PTEC leverages the server built-in sensors and

monitoring utilities, as well as a wireless sensor network to monitor the thermal and power conditions of a data center. Based on the sensor data, it predicts the server temperatures in real time, and optimizes temperature setpoints and cold air supply rates of cooling systems, as well as the speeds of server internal fans, to minimize their overall energy consumption. Moreover, PTEC enforces a set of thermal safety requirements including the upper bounds on server inlet temperatures and their variations, to prevent server overheating and reduce server hardware failure rate. Experiments on a small hardware testbed and trace-driven CFD simulations based on a production data center show that PTEC can reduce the cooling and circulation energy consumption by up to 34% and 30%, compared with an overcooling strategy and a reactive control strategy, respectively.

CHAPTER 6

CONCLUSION

This dissertation presents the holistic performance control approaches of two types of mission-critical Cyber-Physical Systems. First, it presents a fidelity-aware real-time CPU utilization control approach to jointly ensure the fidelity and timeliness performance requirements of wireless cyber-physical surveillance systems. It proposes a novel problem formulation to enforce a given upper bound on the CPU utilization while minimizing the system detection error rate. The problem formulation is based on rigorous performance models that characterize the fusion-based detection performance and the expected CPU utilization induced by processing stochastic detection results. Then, it develops Fidelity-Aware Utilization Controller (FAUC) that adaptively adjusts the data fusion threshold to bound the CPU utilization for ensuring real-time schedulability while minimizing the system detection error rate. FAUC has been implemented on a small-scale WCS testbed to evaluate the FAUC under real dynamics and conducted extensive simulations based on real acoustic data traces. The extensive experiments show that FAUC can achieve robust fidelity and timeliness control in the presence of significant physical dynamics and unreliable wireless links.

Second, we present a proactive thermal and energy control approach for data centers to improve the energy-efficiency performance while ensuring the data center reliability. It consists of a high-fidelity real-time temperature prediction approach and a predictive thermal and energy control (PTEC) system. The prediction approach integrates Computational Fluid Dynamics (CFD) modeling and real-time data-driven prediction to achieve high fidelity temperature prediction in various thermal conditions of data centers. Extensive experimental results show that this approach can accurately predict the temperatures up to 10 minutes into the future, even in the presence of highly dynamic server workloads. PTEC system leverages the server built-in sensors and monitoring utilities, as well as a network of wire-

less sensors to monitor the thermal and power conditions of a data center. Based on the sensor data, it use the proposed prediction approach to estimate the server temperatures in real-time, and optimize temperature setpoints and cold air supply rates of cooling systems, as well as the speeds of server internal fans, to minimize their overall energy consumption. To ensure the data center reliability, PTEC enforces a set of thermal safety requirements including the upper bounds on server inlet temperatures and their variations, to prevent server overheating and reduce server hardware failure rate. Extensive experiments and trace-driven CFD simulations show that PTEC can safely reduce substantial cooling and circulation energy consumption compared with traditional approaches, and can adapt to the realistic data center workload.

Overall, mission-critical Cyber-Physical Systems often have stringent performance requirements which are highly dependent on each other. Simply addressing each requirement individually may not achieve an overall satisfactory performance. This dissertation investigates the interdependencies between these performance requirements. We show that holistic and coordinated performance control are essential for the mission-critical CPSs to meet a multitude of performance requirements.

APPENDICES

APPENDIX A

MEAN AND VARIANCE UNDER TEMPORAL SAMPLING

In this appendix, we derive mean and variance of the fusion statistic Y under the temporal sampling scheme described in Sectoin 3.4.2. In the absence of target, the mean and variance of Y are given by:

$$\begin{aligned}
\mathbb{E}[Y|H_0] &= \sum_{i=1}^{N}\sum_{j=1}^{m}\mathbb{E}[u_{ij}\cdot y_{ij}|H_0] \\
&= \sum_{i=1}^{N}\sum_{j=1}^{m}\mathbb{E}[u_{ij}]\cdot\mathbb{E}[y_{ij}|H_0] \\
&= m\cdot\sum_{i=1}^{N}p_i\cdot\mu_i,
\end{aligned}$$

$$\begin{aligned}
\mathrm{Var}[Y|H_0] &= \sum_{i=1}^{N}\sum_{j=1}^{m}\mathrm{Var}[u_{ij}\cdot y_{ij}|H_0] \\
&= \sum_{i=1}^{N}\sum_{j=1}^{m}\mathbb{E}[u_{ij}^2\cdot y_{ij}^2|H_0] - \big(\mathbb{E}[u_{ij}\cdot y_{ij}|H_0]\big)^2 \\
&= \sum_{i=1}^{N}\sum_{j=1}^{m}\Big(\mathbb{E}[u_{ij}^2]\cdot\mathbb{E}[y_{ij}^2|H_0] - p_i^2\cdot\mu_i^2\Big) \\
&= m\cdot\sum_{i=1}^{N}p_i\cdot\sigma_i^2 + \mu_i^2\cdot(p_i - p_i^2).
\end{aligned}$$

Note that $\mathbb{E}[\mu_{ij}^2] = p_i$ and $\mathbb{E}[y_{ij}^2|H_0] = \sigma_i^2 + \mu_i^2$. Similarly, in the presence of target, the mean and variance of $Y|H_1$ are given by

$$\begin{aligned}
\mathbb{E}[Y|H_1] &= m \cdot \sum_{i=1}^{N} p_i \cdot (s_i + \mu_i), \\
\mathrm{Var}[Y|H_1] &= m \cdot \sum_{i=1}^{N} p_i \cdot \left(\sigma_i^2 + (s_i+\mu_i)^2\right) - p_i^2 \cdot (s_i+\mu_i)^2 \\
&= m \cdot \sum_{i=1}^{N} p_i \cdot \sigma_i^2 + (s_i + \mu_i)^2 \cdot (p_i - p_i^2).
\end{aligned}$$

Note that $\mathbb{E}[y_{ij}^2|H_1] = \sigma_i^2 + (s_i + \mu_i)^2$.

APPENDIX B

SUMMARY OF NOTATIONS IN PTEC

Table B.1 Summary of notations.

Symbol	Definition	Unit
T	temperature	°C
T_l	inlet temperature of the lth server	°C
Δt	duration of a time step	sec
t_n	beginning time instant of the nth time step	Δt
J	the total number of CRAC systems	n/a
L	the total number of servers	n/a
P_{Fl}	instantaneous fan power of the lth server	W
P_{Cj}	instantaneous power of the jth CRAC system	W
S_j	temperature setpoint of the jth CRAC system	°C
$\mathbf{S}$	$\mathbf{S} = [S_1, S_2, \ldots, S_J]$	n/a
B_j	blower speed setting of the jth CRAC system	n/a
$\mathbf{B}$	$\mathbf{B} = [B_1, B_2, \ldots, B_J]$	n/a
D_l	PWM duty cycle of the lth server	n/a
$\mathbf{D}$	$\mathbf{D} = [D_1, D_2, \ldots, D_L]$	n/a
T_{Hj}	return hot air temperature of the jth CRAC system	°C
m	control cycle length	Δt
k	prediction horizon	Δt
K	optimization horizon	Δt
$\overline{P(t_n)}$	predicted average power between t_n and $t_n + K$	W

Table B.1 (cont'd)

Symbol	Definition	Unit
T_U	user-defined maximum allowed temperature (MAT)	°C
$T_{l,k}$	true inlet temperature of the lth server at t_{n+k}	°C
$\hat{T}_{l,k}$	predicted inlet temperature of the lth server at t_{n+k}	°C
$\widetilde{T}_{l,k}$	MAT for the lth server at prediction horizon k (Eq. (5.2))	°C
RSD_l	RSD of the lth server's inlet temperature	n/a
RSD_U	maximum allowed RSD	n/a
w	size of the window for computing RSD	Δt
r_i	the ith setting of DFSC, $r_i = \langle V_i, W_i \rangle$	n/a
M	number of CPU utilization levels in DFSC	n/a
V_i	highest inlet temperature for maintaining minimal speed	°C
W_i	lowest inlet temperature for maintaining full speed	°C
$\gamma_{l,j}$	thermal correlation index (TCI)	n/a
λ	TCI threshold for associating a server and a CRAC	n/a
g	a region after partitioning	n/a
$\mathbb{G}$	set of all regions	n/a
C_g	set of CRAC systems in region g	n/a
C	set of all CRAC systems, $C = \bigcup_{\forall g \in \mathbb{G}} C_g$, $\lvert C \rvert = J$	n/a
E_g	set of servers in region g	n/a
$\widetilde{T}^g_{l,k}$	relaxed MAT for the lth ungrouped server and region g	°C
T^U	temperature upper bound of reactive control approach	°C
T^B	temperature band of reactive control approach	°C

APPENDIX C

ALGORITHM OF PARTITION-BASED PREDICTIVE CONTROLLER

C.1 Offline Stage

In the offline stage, we design a heuristic partitioning algorithm based on an Maximum-Weight Independent Set (MWIS) solver. Based on the Thermal Correlation Index (TCI), each server has one or more associated CRAC systems. We cluster the servers so that the servers in a cluster are associated with the same set of CRAC systems. The algorithm then works on a bipartite graph $G = (X, Y, Z)$ (Figure C.1(1)) that consists of the server cluster vertices X, CRAC vertices Y, and the association edges Z. The associated CRAC systems of each server cluster represent a potential CRAC group in the partition, which is referred to as *inferred CRAC group*. For example, in Figure C.1, CRAC 7, 8 and 9 form an inferred CRAC group of server cluster 8. Therefore, the partitioning problem can be solved by finding a set of server clusters whose inferred CRAC groups cover all the CRAC systems.

Now we use the example illustrated in Figure C.1 to discuss the algorithm. Let X_i denote a subset of server clusters with degree of i in the original bipartite graph. Moreover, Y_i and Z_i denote the corresponding associated CRAC systems and the association edges of server clusters in X_i, respectively. We define subgraph $G_i = (X_i, Y_i, Z_i)$. Our algorithm partitions the subgraphs $G_1, G_2, \ldots$, sequentially. It first partitions G_1 as illustrated in Figure C.1(1). The server clusters in X_1 are cluster 1, 2, and 6. Their weights are assigned as their degrees (i.e., 1), whereas the weights of their associated CRAC systems are assigned as zero. This assignment scheme is to ensure that more server clusters can be found in the results of MWIS solver. After running the MWIS solver for G_1, the resulted MWIS contains server cluster 1, 2, and 6, and thus their associated CRAC 1, 2, and 6 form individual CRAC groups, as illustrated by the dashed rectangles in Figure C.1(1).

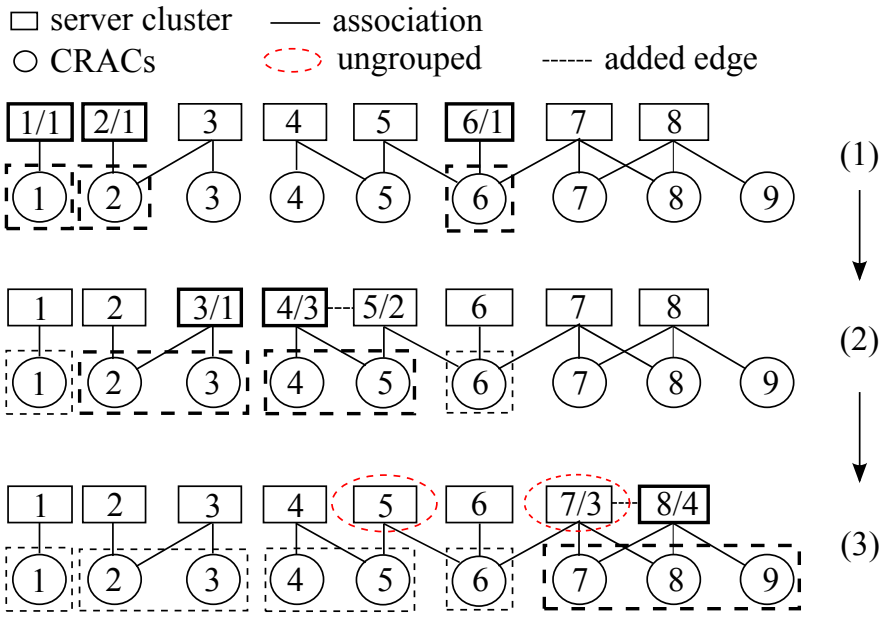


Figure C.1 Illustration of the offline stage. The server clusters are labeled by *index* or *index/weight*, whereas the CRAC systems are labeled by *index* only since their weights are constantly zero. Sub-figure (1), (2) and (3) are three example iterations that form different CRAC groups. In particular, the thick dashed rectangles show the CRAC groups formed in the current iteration while the thin dashed rectangles show the CRAC groups formed in previous iterations. In addition, the thick rectangles show the server clusters associated with the CRACs grouped in the current iteration.

Then, our algorithm partitions G_2 with server cluster 3, 4, and 5, as shown in Figure C.1(2). The dashed edge is added between cluster 4 and 5, since they are associated with a common CRAC system (i.e., CRAC 5). The added edge ensures that the shared CRAC system falls in one CRAC group only. The weights of the server clusters are assigned as the degree minus the number of CRAC systems that are already in CRAC groups. For example, the server cluster 3 has a degree of 2. However, its weight is 1 since CRAC 2 is already in a CRAC group during partitioning G_1. Then, the resulted MWIS of G_2 is server cluster 3 and 4. Therefore, CRAC 2 and 3, associated with server cluster 3, forms a CRAC group, while CRAC 4 and 5, associated with server cluster 4, form another CRAC group. Similarly, in Figure C.1(3), the MWIS of G_3 after weighting and adding edges contains server cluster 8 only. Therefore, the CRAC 7, 8 and 9 form a group. At this point, all CRACs are in one of the CRAC groups and our algorithm stops. Note that server cluster 5 and 7 are

not in any resulted MWIS and hence their servers are ungrouped servers.

In practice, our algorithm may work on G_i with large i only if several CRAC systems' association are identical, which is unlikely in a large-scale data center.

C.2 Online Stage

The pseudocode of the online stage of our partition-based predictive controller is in Algorithm 1. Note that, in Line 15, we initialize the relaxed MATs as infinity. Line 20 is the relaxation discussed in Section 5.4.5. The following proposition proves the convergence of the algorithm.

Proposition 1. *Algorithm 1 converges.*

Proof. In the ith iteration of Algorithm 1, the predicted inlet temperature of the lth ungrouped server (i.e., $\hat{T}_{l,k}$) must satisfy the MAT constraint in the previous iteration (denoted by $\widetilde{T}^g_{l,k}(i-1)$), i.e., $\hat{T}_{l,k} \leq \widetilde{T}^g_{l,k}(i-1)$. From the relaxation in Line 20, the new MAT $\widetilde{T}^g_{l,k}(i) < \hat{T}_{l,k} \leq \widetilde{T}^g_{l,k}(i-1)$. Thus, both $\hat{T}_{l,k}$ and $\widetilde{T}^g_{l,k}$ in Algorithm 1 are non-increasing. Therefore, the algorithm converges to a solution that satisfies $\hat{T}_{l,k} \leq \widetilde{T}_{l,k}$, $\forall l \in E_0$ and $\forall k \in [1, K]$. □

Algorithm 1 Online stage of the partition-based predictive controller.

Input: ① set of TCIs $\{\gamma_{l,j}|\forall j \in [1,J], \forall l \in [1,L]\}$; ② results of the offline stage: regions $\mathbb{G} = \{g_1, g_2, \ldots\}$, $\{E_{g1}, E_{g2}, \ldots\}$, $\{C_{g1}, C_{g2}, \ldots, \}$, and set of ungrouped servers E_0

Output: CRAC temperature setpoints $\mathbf{S}$ and blower speeds $\mathbf{B}$

for $\forall g \in \mathbb{G}$ **do**
  use CSA to find the sub-solution $\langle \mathbf{S}_g, \mathbf{B}_g \rangle$ for the CRAC systems in C_g and the servers in E_g with $\widetilde{T}_{l,k}$ given by Eq. (5.2) as MAT constraint
end for
iteration index $i \Leftarrow 0$
loop
  $\mathbf{S} = \bigcup_{g\in\mathbb{G}} \mathbf{S}_g$, $\mathbf{B} = \bigcup_{g\in\mathbb{G}} \mathbf{B}_g$
  for $\forall l \in E_0$, $\forall k \in [1,K]$ **do**
    predict $\hat{T}_{l,k}$ based on $\langle \mathbf{S}, \mathbf{B} \rangle$
  end for
  if thermal requirements for servers in E_0 are satisfied **then**
    return $\mathbf{S}$ and $\mathbf{B}$
  end if
  if $i = 0$ **then**
    for $\forall g \in \mathbb{G}$, $\forall l \in E_0$, $\forall k \in [1,K]$ **do**
      $\widetilde{T}^g_{l,k} = +\infty$
    end for
  end if
  for $\forall g \in \mathbb{G}$, $\forall l \in E_0$, $\forall k \in [1,K]$ **do**
    if $\hat{T}_{l,k} > \widetilde{T}_{l,k}$ **then**
      $\widetilde{T}^g_{l,k} = \hat{T}_{l,k} - \frac{\sum_{j\in C_g} \gamma_{l,j}}{\sum_{j\in C} \gamma_{l,j}} \cdot (\hat{T}_{l,k} - \widetilde{T}_{l,k})$
    end if
  end for
  for $\forall g \in \mathbb{G}$ **do**
    use CSA to find the sub-solution $\langle \mathbf{S}_g, \mathbf{B}_g \rangle$ for the CRAC systems in C_g, the servers in E_g with $\widetilde{T}_{l,k}$ as MAT constraint, and the servers in E_0 with $\widetilde{T}^g_{l,k}$ as MAT constraint
  end for
  $i \Leftarrow i + 1$
end loop

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] E. A. Lee, "Cyber physical systems: Design challenges," in *Proceedings of the 11th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC)*, 2008, pp. 363–369.

[2] R. Tan, G. Xing, X. Liu, J. Yao, and Z. Yuan, "Adaptive calibration for fusion-based wireless sensor networks," in *Proceedings of the 29th IEEE International Conference on Computer Communications (INFOCOM)*, 2010, pp. 1–9.

[3] R. Tan, G. Xing, J. Chen, W.-Z. Song, and R. Huang, "Fusion-based volcanic earthquake detection and timing in wireless sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 9, no. 17, 2013.

[4] Y. P. Fallah and R. Sengupta, "A cyber-physical systems approach to the design of vehicle safety networks," in *Proceedings of the 32nd International Conference on Distributed Computing Systems Workshops*, 2012, pp. 324–329.

[5] P. K. Varshney, *Distributed Detection and Data Fusion*. Springer, 1996.

[6] V. Bychkovskiy, S. Megerian, D. Estrin, and M. Potkonjak, "A collaborative approach to in-place sensor calibration," in *Proceedings of the 2nd ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2003, pp. 301–316.

[7] J. Feng, S. Megerian, and M. Potkonjak, "Model-based calibration for sensor networks," in *Proceedings of IEEE Sensors*, vol. 2, 2003, pp. 737–742.

[8] K. Whitehouse and D. Culler, "Calibration as parameter estimation in sensor networks," in *Proceedings of the 1st ACM International Workshop on Wireless Senor Networks and Applications (WSNA)*, 2002, pp. 59–67.

[9] G. Xing, C. Lu, Y. Zhang, Q. Huang, and R. Pless, "Minimum power configuration in wireless sensor networks," in *Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2005, pp. 390 – 410.

[10] H. Aydin, V. Devadas, and D. Zhu, "System-level Energy Management for Periodic Real-Time Tasks," in *Proceedings of the 27th IEEE Real-Time Systems Symposium (RTSS)*, 2006, pp. 313–322.

[11] H. Aydin, R. Melhem, D. Mossé, and P. Mejía-Alvarez, "Dynamic and Aggressive Scheduling Techniques for Power-Aware Real-Time Systems," in *Proceedings of the 22nd IEEE Real-Time Systems Symposium (RTSS)*, 2001, pp. 95–105.

[12] J.-J. Chen, H.-R. Hsu, K.-H. Chuang, C.-L. Yang, A.-C. Pang, and T.-W. Kuo, "Multiprocessor energy-efficient scheduling with task migration considerations," in *Proceedings of the 16th Euromicro Conference on Real-Time Systems (ECRTS)*, 2004, pp. 101–108.

[13] R. K. Sharma, C. L. Bash, c. d. patel, R. J. Friedrich, and J. S. Chase, "Balance of power: Dynamic thermal management for internet data centers," in *IEEE Internet Computing*, vol. 9, January 2005, pp. 42–49.

[14] Strix Systems, Inc., 2010, http://www.strixsystems.com/.

[15] T. He, S. Krishnamurthy, J. A. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, J. Hui, and B. Krogh, "Energy-efficient surveillance system using wireless sensor networks," in *Proceedings of the 2nd International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2004, pp. 270–283.

[16] MEMSIC Inc., "TelosB, Iris, Mica2 datasheets," 2011.

[17] J. A. Stankovic, T. He, T. Abdelzaher, M. Marley, G. Tao, S. Son, and C. Lu, "Feedback control scheduling in distributed real-time systems," in *Proceedings of the 22nd IEEE Real-Time Systems Symposium (RTSS)*, 2001, pp. 59–70.

[18] C. Lu, X. Wang, and X. Koutsoukos, "Feedback utilization control in distributed real-time systems with end-to-end tasks," *IEEE Transaction on Parallel and Distributed Systems*, vol. 16, no. 6, pp. 550–561, 2005.

[19] J. Yao, X. Liu, M. Yuan, and Z. Gu, "Online adaptive utilization control for real-time embedded multiprocessor systems," in *Proceedings of the 6th IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, 2008, pp. 85–90.

[20] U.S. Environmental Protection Agency, "Report to congress on server and data center energy efficiency," 2007.

[21] Aperture Research Institute, "Data center professionals turn to high-density computing as major boom continues," April 2007.

[22] Emerson Network Power, "State of the data center 2011," http://www.emersonnetworkpower.com.

[23] http://www.google.com/about/datacenters/.

[24] C. E. Bash, C. D. Patel, and R. K. Sharma, "Dynamic thermal management of air cooled data centers," in *Proceedings of the 10th Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronics Systems (ITHERM)*, 2006, p. 452.

[25] J. Moore, J. Chasey, P. Ranganathanz, and R. Sharmaz, "Making scheduling "cool": Temperature-aware workload placement in data centers," in *Proceedings of the USENIX Annual Technical Conference (USENIX)*, 2005, p. 5.

[26] A. Banerjee, T. Mukherjee, G. Varsamopoulos, , and S. K. S. Gupta, "Cooling-aware and thermal-aware workload placement for green hpc data centers," in *Proceedings of International Green Computing Conference (IGCC)*, 2010, pp. 245–256.

[27] W. Huang, M. Allen-Ware, J. B. Carter, E. Elnozahy, H. Hamann, T. Keller, C. Lefurgy, J. Li, K. Rajamani, and J. Rubio, "TAPO: Thermal-aware power optimization techniques for servers and data centers," in *Proceedings of International Green Computing Conference (IGCC)*, 2011, pp. 1–8.

[28] G. Xing, R. Tan, B. Liu, J. Wang, X. Jia, and C.-W. Yi, "Data fusion improves the coverage of wireless sensor networks," in *Proceedings of the 15th International Conference on Mobile Computing and Networking (MobiCom)*, 2009, pp. 157–168.

[29] R. Tan, G. Xing, B. Liu, and J. Wang, "Impact of data fusion on real-time detection in sensor networks," in *Proceedings of the 30th IEEE Real-Time Systems Symposium (RTSS)*, 2009, pp. 323–332.

[30] R. Tan, G. Xing, Z. Yuan, X. Liu, and J. Yao, "System-level calibration for fusion-based wireless sensor networks," in *The 31st IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2010, pp. 215–224.

[31] T. Heath, A. P. Centeno, P. George, L. Ramos, Y. Jaluria, and R. Bianchini, "Mercury and freon: Temperature emulation and management for server systems," in *The 12th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2006, pp. 106–116.

[32] L. Ramos and R. Bianchini, "C-oracle: Predictive thermal management for data centers," in *IEEE 17th International Symposium on High Performance Computer Architecture (HPCA)*, 2008, pp. 111–122.

[33] Q. Tang, T. Mukherjee, S. K. S. Gupta, and P. Cayton, "Sensor-based fast thermal evaluation model for energy efficient high-performance datacenters," in *Proceedings of the 4th International Conference on Intelligent Sensing and Information Processing (ICISIP)*, 2006, pp. 203–208.

[34] J. Moore, J. Chasey, and P. Ranganathanz, "Weatherman: Automated, online, and predictive thermal mapping and management," in *The 3rd IEEE International Conference on Autonomic Computing (ICAC)*, 2006, pp. 155–164.

[35] J. Choi, Y. K. An, J. Srebric, Q. Wang, and J. Lee, "Modeling and managing thermal profiles of rack-mounted servers with thermostat," in *In Proceedings of the 13th International Symposium on High-Performance Computer Architecture*, 2007, pp. 205–215.

[36] L. Li, C.-J. M. Liang, J. Liu, S. Nath, A. Terzis, and C. Faloutsos, "Thermocast: A cyber-physical forecasting model for data centers," in *Proceedings of the 17th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2011, pp. 1370–1378.

[37] J. Niemann, "Best practices for designing data centers with the infrastruxure inrow RC," *Application note of American Power Conversion*, 2006.

[38] G. C. Bell, "Improving data center efficiency with rack or row cooling devices: Results of 'chill-off 2' comparative testing," *Federal Energy Management Program*, 2012.

[39] N. Rasmussen, "Cooling options for rack equipment with side-to-side airflow," 2011.

[40] M. Jonas, R. R. Gilbert, J. Ferguson, G. Varsamopoulos, , and S. Gupta, "A transient model for data center thermal prediction," in *Proceedings of the 3rd International Green Computing Conference (IGCC)*, 2012, pp. 1–10.

[41] Q. Tang, S. K. S. Gupta, and G. Varsamopoulos, "Energy-efficient thermal-aware task scheduling for homogeneous high-performance computing data centers: A cyber-physical approach," in *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 11, 2008, pp. 1458–1472.

[42] Z. Wang, C. Bash, N. Tolia, M. Marwah, X. Zhu, and P. Ranganathan, "Optimal fan speed control for thermal management of servers," in *Proceedings of ASME 2009 InterPACK Conference*, vol. 2, 2009, pp. 709–719.

[43] S. Li, T. Abdelzaher, and M. Yuan, "TAPA: temperature aware power allocation in data center with map-reduce," in *Proceedings of International Green Computing Conference and Workshops (IGCC)*, 2011, pp. 1–8.

[44] E. K. Lee, H. Viswanathan, and D. Pompili, "VMAP: Proactive thermal-aware virtual machine allocation in hpc cloud datacenters," in *Proceedings of the 19th International Conference on High Performance Computing (HiPC)*, 2012, pp. 1–10.

[45] N. Tolia, Z. Wang, P. Ranganathan, C. Bash, M. Marwah, and X. Zhu, "Unified thermal and power management in server enclosures," in *Proceedings of ASME 2009 InterPACK Conference*, vol. 2, 2009, pp. 721–730.

[46] L. Parolini, N. Tolia, B. Sinopoli, and B. H. Krogh, "A cyber-physical systems approach to energy management in data centers," in *Proceedings of the 1st International Conference on Cyber-Physical Systems (ICCPS)*, April 2010, pp. 168–177.

[47] Z. Liu, Y. Chen, C. Bash, A. Wierman, D. Gmach, Z. Wang, M. Marwah, and C. Hyser, "Renewable and cooling aware workload management for sustainable data centers," in *Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE joint International Conference on Measurement and Modeling of Computer Systems*, 2012, pp. 175–186.

[48] R. Zhou, C. Bash, Z. Wang, A. McReynolds, T. Christian, and T. Cader, "Data center cooling efficiency improvement through localized and optimized cooling resources delivery," in *ASME 2012 International Mechanical Engineering Congress and Exposition*, vol. 7, 2012, pp. 1789–1796.

[49] C. J. M. Liang, J. Liu, L. Luo, A. Terzis, and F. Zhao, "RACNet: A high-fidelity data center sensing network," in *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2009, pp. 15–28.

[50] S. Choochaisri, V. Niennattrakul, S. Jenjaturong, C. Intanagonwiwat, and C. A. Ratanamahatana, "Senvm: Server environment monitoring and controlling system for small data center using wireless sensor network," in *Proceedings of International Computer Science and Engineering Conference (ICSEC)*, 2010.

[51] X. Wang, X. Wang, G. Xing, J. Chen, C.-X. Lin, and Y. Chen, "Towards optimal sensor placement for hot server detection in data centers," in *Proceedings of the 31st International Conference on Distributed Computing Systems (ICDCS)*, 2011, pp. 899–908.

[52] C. Mansley, J. Connell, C. Isci, J. Lenchner, J. O. Kephart, S. McIntosh, and M. Schappert, "Robotic mapping and monitoring of data centers," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 5905–5910.

[53] J. Lenchner, C. Isci, J. Kephart, C. Mansley, J. Connell, and S. McIntosh, "Toward data center self-diagnosis using a mobile robot," in *Proceedings of the 8th IEEE International Conference on Autonomic Computing (ICAC)*, 2011, pp. 81–90.

[54] S. Biswas, M. Tiwari, T. Sherwood, L. Theogarajan, and F. T. Chong, "Fighting fire with fire: modeling the datacenter-scale effects of targeted superlattice thermal management," in *Proceedings of the 38th International Symposium on Computer Architecture*, 2011, pp. 331–340.

[55] X. Sheng and Y.-H. Hu, "Maximum likelihood multiple-source localization using acoustic energy measurements with wireless sensor networks," *IEEE Transaction on Signal Processing*, vol. 53(1), pp. 44–53, 2005.

[56] M. Hata, "Empirical formula for propagation loss in land mobile radio services," *IEEE Transaction on Vehicular Technology*, vol. 29(3), pp. 317–325, 1980.

[57] D. Li and Y.-H. Hu, "Energy based collaborative source localization using acoustic micro-sensor array," *Journal of EUROSIP Applied Signal Processing*, vol. 2003, no. 4, pp. 321–337, 2003.

[58] C. Wren, U. Erdem, and A. Azarbayejani, "Functional calibration for pan-tilt-zoom cameras in hybrid sensor networks," *Multimedia Systems*, vol. 12, no. 3, 2006.

[59] P. Dutta, A. Arora, and S. Bibyk, "Towards radar-enabled sensor networks," in *Proceedings of The 5th International Conference on Information Processing in Sensor Networks (IPSN)*, 2006, pp. 467–474.

[60] T. He, P. Vicaire, T. Yan, L. Luo, L. Gu, G. Zhou, R. Stoleru, Q. Cao, J. A. Stankovic, and T. Abdelzaher, "Achieving real-time target tracking using wireless sensor networks," in *Proceedings of the 12th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2006, pp. 37–48.

[61] C. L. Liu and J. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environments," *Journal of the ACM*, vol. 20, no. 1, pp. 46–61, 1973.

[62] C. Lu, X. Wang, and C. Gill, "Feedback control real-time scheduling in ORB middleware," in *Proceedings of the 9th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2003, pp. 37–48.

[63] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, 1967, pp. 281–297.

[64] K. Ogata, *Discrete-time control systems*. Prentice-Hall, 1995.

[65] ImageMagick, 2010, http://www.imagemagick.org.

[66] M. Duarte and Y.-H. Hu, "Vehicle classification in distributed sensor networks," *Journal of Parallel and Distributed Computing*, vol. 64(7), pp. 826–838, 2004.

[67] WikiMedia Foundation, "Global outrage (cooling failure and dns)," 2010, http://blog.wikimedia.org/2010/03/24/global-outage-cooling-failure-and-dns/.

[68] C. Bash and G. Forman, "Cool job alloction: Measuring the power savings of placing jobs at cooling-efficient locations in the data center," in *Proceedings of USENIX Annual Technical Conference (USENIX)*, 2007, pp. 1–6.

[69] N. El-Sayed, I. Stefanovici, G. Amvrosiadis, A. A. Hwang, and B. Schroeder, "Temperature management in data centers:why some (might) like it hot," in *Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE joint International Conference on Measurement and Modeling of Computer Systems*, 2012, pp. 163–174.

[70] J. F. Wendt, Ed., *Computational Fluid Dynamics – An Introduction*, 3rd ed. Springer, 1995.

[71] ASHRAE Technical Committee 9.9, "2011 thermal guidelines for data processing environments – expanded data center classes and usage guidance," 2011.

[72] Active Power, Inc., "Data center thermal runaway: A review of cooling challenges in high density mission critical environments," 2007.

[73] U. Singh, A. K. Singh, P. S, and A. Sivasubramaniam, "CFD-based operational thermal efficiency improvement of a production data center," in *Proceedings of the 1st USENIX Workshop on Sustainable Information Technology (SustainIT)*, 2010, p. 6.

[74] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

[75] A. Stuart, K. Ord, and S. Arnold, *Kendall's Advanced Theory of Statistics: Classical Inference and the Linear Model*, 6th ed. John Wiley & Sons, 2009.

[76] L. Van der Maaten, E. Postma, and H. Van Den Herik, "Dimensionality reduction: A comparative review," *Journal of Machine Learning Research*, vol. 10, pp. 1–41, 2009.

[77] Degree Controls, Inc., "F333 airflow sensor user guide." 2011.

[78] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler, "TinyOS: An operating system for sensor networks," *Ambient Intelligence*, pp. 115–148, 2005.

[79] Emerson Network Power, "State of the data center," 2011, http://www.emersonnetworkpower.com/.

[80] N. Ye, Ed., *The Handbook of Data Mining*. Lawrence Erlbaum Associates, Publishers, 2003.

[81] M. Stansberry and J. Kudritzki, "Uptime institute 2012 data center industry survey," 2012.

[82] M. Stansberry, "Uptime institute 2013 data center industry survey," 2013.

[83] "Fancontrol - automated software based fan speed regulation," http://linux.die.net/man/8/fancontrol.

[84] J. Chen, R. Tan, Y. Wang, G. Xing, X. Wang, X. Wang, B. Punch, and D. Colbry, "A high-fidelity temperature distribution forecasting system for data centers," in *Proceedings of the 33rd IEEE Real-Time Systems Symposium (RTSS)*, 2012, pp. 215–224.

[85] B. W. Wah, Y. Chen, and T. Wang, "Simulated annealing with asymptotic convergence for nonlinear constrained optimization," *Journal of Global Optimization*, vol. 39, no. 1, pp. 1–37, 2007.

[86] W. Brendel and S. Todorovic, "Segmentation as maximum-weight independent set," in *Advances in Neural Information Processing Systems*, 2010.

[87] P. v. Böckh and T. Wetzel, *Heat Transfer: Basics and Practice*. Springer, 2012.