Performance Analysis of Real Time Streaming Systems for Smart Buildings

Kim Stuart

A thesis

submitted in partial fulfillment of the

requirements for the degree of

Master of Science in Computer Science and Systems

University of Washington

2017

Committee:

Matthew Tolentino

Orlando Baiocchi

Wes Lloyd

Program Authorized to Offer Degree:

Institute of Technology

University of Washington

Abstract

Performance Analysis of Real Time Streaming Systems for Smart Buildings

Kim Stuart

Chair of the Supervisory Committee:

Assistant Professor Matthew Tolentino

Institute of Technology

The Internet of Things (IoT) extends traditional cyber-physical systems by linking sensor-based edge devices to additional network-accessible services and resources. In most current IoT deployments, sensor data is streamed from edge devices to servers for storage and analysis. Analytical pipelines translate this raw sensor data into actionable information in real-time. Higher sensor densities lead to increased data volumes at higher frequencies, which increases the utilization rate of the overall system. This can lead to an increase in unacceptable response latencies for IoT analytical systems.

In this paper, we compare the impact of alternative stream processing topologies for ingesting and analyzing IoT sensor data in real-time. We use real building sensor data from a commercial, LEED certified smart building on the University of Washington campus with our real-time IoT platform Namatad. We first characterize and analyze the latency impact of how

the data streams are ingested and routed to analytical pipelines that predict occupancy at different levels of granularity. We then develop a queuing-theoretic analytical performance model for each of the four IoT streaming topologies. Our results show that as IoT systems continue to scale in density, server-side topology is critical to meet real-time requirements for analytical pipelines.

Keywords: IoT, QoS, Real-Time Analytics, sensors, topology, queueing theory

Acknowledgement

My profound gratitude goes out to Dr. Matthew Tolentino and Intelligent Platforms and Architecture (IPA) Labs for having faith in my abilities and dedication to the improvement of his research by allowing me to join his team. I truly believe that it was not until I collaborated with him that I got my first sense of industry experience in the way that his research was set up. This research alone opened up so many opportunities to expand my network and have recruiters proactively network with me because of the work I have specifically done with this research. I am thankful for my committee members: Orlando Baiocchi for allowing me the opportunity to work with sensor streaming and really establish my passion for the IoT field and Dr. Wes Lloyd, for providing guidance on performance analysis and its importance when dealing with IoT systems and with perfecting organization and formatting on technical research papers.

I am thankful to Robert “Bob” Landowski, who broadened my knowledge with systems and hardware and Anindya Dey, for his profound knowledge in machine learning and research organization techniques.

The authors would like to thank the facilities department at University of Washington for instrumenting the LEED certified Cherry Parkes building with environmental sensors and streaming them into our platform. The total duration of the data that we have got so far is more than 9 months. Getting the real world data has helped us account for the noise, the missing values and the disorders which are some of the unwelcome characteristics of real world datasets but definitely need to be accounted for. We also thank the entire team at Intelligent Platforms and Architecture laboratory for providing the necessary computing infrastructure as well as continued motivation and support.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

As modern businesses and households transition into the era of Internet of Things (IoT), traditional cyberphysical systems (CPSs) are required to become smarter and work together to meet requirements and expectations in real-time. As the need for more information and insights become critical for these smart devices to perform real-time functionality and desired behaviors with as much accuracy as possible, more sensors are being added to IoT systems. This provides rapid and large volumes of data for computational hubs to process [1]. This is one way of providing deeper and more precise information to be inferred, as well as more accurate predictions that can be processed. However, increasing the size of these wireless sensors networks (WSNs) used within IoT systems presents a challenge. While these computational hubs are now being heavily congested with an influx of data arriving at higher frequencies, this introduces added latencies to these IoT systems. While higher accuracy may be achieved, this can potentially jeopardize the state of real-time [2].

One naive approach to manage these large volumes of data, would be to simply add more computational hubs or buy state of the art hubs that can handle larger volumes of data. For businesses, the ability to upgrade existing computational hubs, choose best locations to add additional sensors and hubs, or the ability to add additional hubs in general is constrained to building blueprints that must meet safety compliance and cost of buying or upgrading hardware. For homes, cost of hardware and maintenance constrains these options. However, in both cases, the ability to manage real-time results is still a requirement, despite these constraints. This naive approach would increase accuracy and minimize latency, yielding real-time precise computations,

with no regard to cost, which is always a factor that businesses and homeowners try to minimize. A practical approach would involve increasing quality of service (QoS) metrics, while using the existing infrastructure. This approach focuses more on data stream management and controlling the flow of data prior to computation to minimize latency and maximize accuracy, which presents a challenge in real-time analytics.

This research presents a three-fold approach to address these challenges. Because we want our system to minimize latency to preserve real-time, we have extended our existing Namatad system [3] to provide more flexibility to achieve this. We will continue to ingest sensor data into our system. Our work introduces a variation into the concept of a software defined network (SDN), that allows our sensor data the flexibility of being routed to multiple software topologies based on the defined granularity at which the sensor values are being aggregated by [10]. After the additional topologies have been designed, queueing theory based performance models will be built to analytically characterize and quantify performance measurements, such as latency and utilization rates, based on the arrival rates at which sensor values are arriving [7]. The aforementioned software streaming topologies will be implemented and integrated into our existing Namatad system, empirically measured, and validated against the analytical model.

1.1 Motivation

As the need for more data is required for more precise real-time predictions, being able to ingest data and make real-time predictions while minimizing latency becomes a challenge. Real-time use of IoT data has high temporal value. Real time predictions require data to be sent and serviced in real-time. Failure to service data in real time could prove to be costly and even fatal in some cases.

Consider the case of a person trapped in a burning building. We need sensor data to be transmitted to the computational hubs to provide positioning on both the firefighter and the victim in real-time. The slightest latency in real-time predictions could prevent the firefighter from finding the victim, which can be fatal and in extreme cases, result in death. Now consider industrial buildings that have an unexplained spike in energy consumption. Such spikes will affect the average energy consumption (as energy is measured in kilowatt-hour units, or kWh) of the entire building, which that company will be charged for. For smaller buildings, this may not be problematic. For larger buildings, where energy consumption is typically used more in certain areas than others, this could serve to be very problematic when most rooms consume well under the average. That unexplained spike in consumption that occurred in a particular room could throw off the average for the entire building, which the business will be held accountable for. Real-time predictions could identify the spike and neutralize it before it presents a problem using control automation systems, saving the business thousands on energy consumption.

Currently, most systems use IoT generated data that is persisted in storage and analyzed later. These results and predictions are not representative of real-time. For IoT systems that are using real-time analytics, there is little work to show how close these results are to real-time. We expect network latencies caused from distributed systems to have an impact on response time and we can do as much hardware configuration possible to minimize network latency in that aspect. However, other factors that we are able to control can negatively impact response time. By characterizing the performance in the way we set up and route our data, we can build models that compare the response time of jobs in a system, which can provide insight to how close we are to meet real-time requirements.

We have previously coordinated with the Facilities Department at the University of Washington to gain access to the data collected by building sensors in a LEED certified building that was recently renovated. Using that data, we provided analytical techniques to provide insights beyond measured data. We routed those data streams through our real-time IoT system to predict occupancy with accuracy at greater than 95% when routing the data according to the room it came from [3]. Having a system that can ingest data and make predictions, our next step is to characterize the performance of our system.

1.2 Goals

The end goal for our Namatad system is to provide complete and multifunctional IoT streaming platform capable of making real-time computations under various use cases. This research focuses on how to enable real-time analytics on diverse sets of sensor data using analytical pipelines. As individual sensor data is ingested into a streaming platform, further insight on how to set up and configure the way the data is routed to enable real-time computations for control systems with minimal latency is desired. While our system still allows for persisted storage and offline modeling, what sets our system apart from other state of the art systems is the ability of our Namatad system to immediately handle the computation of data streams directly in real-time, by sending our sensor data straight into online models for prediction. This gives our system the capability to adapt to various granularities and minimize latency by having multiple built in software streaming topologies that we can route our data to. In order to achieve this, our paper must achieve the following goals:

- Analyze the performance of a real-time IoT analytics platform
- Identify alternative streaming topologies for IoT server platforms

- Develop analytical performance models for IoT server processing topologies

1.3 Contributions

Our contributions to improving QoS via data management in the field of IoT are as follows and serve as an outline for our paper:

- Determined alternative sensor routing paths to optimize real-time analytics and minimize response time by designing multiple software topology paths that data can be routed to based on the use case.
- Built a queueing theory based analytical performance model as a tool to provide insight on how to characterize the performance measurements for real-time IoT prediction systems using diverse sensors at various utilization rates.
- Implemented a real system that used real sensor data to empirically measure validate our analytical performance model.

Chapter 2 Related Works

The main components of this research are the abilities to stream sensor data and perform real-time computations using online models while minimizing latency, define alternative sensor streaming topologies for ingested data, and the ability to build analytical performance models for IoT server processing topologies. We have studied current state of the art systems, topology control, real-time streaming with both cloud and edge computing, and queueing theory. The following provide related works to identify similar work that has been done and work that has yet to be achieved that our research aims to target.

2.1 State of the Art Systems

As the need for extracting larger amounts of data continues to drive businesses and research, IoT systems will continue to evolve. This has motivated a considerable amount of previous work involved with leveraging IoT systems influenced by sensor driven architectures [11] [12] [13] [14] [15]. Computing data streams from a relational database has been highly influenced by Michael Stonebraker. Hoppe and Grys published a survey on one of Stonebraker's more notable works, using StreamBase [4] [16]. While Stonebrakers work noted performance increases in stream computing over a traditional RDBMS, real-time computations were not yet attainable. Motivated by Stonebraker and the emergence of IoT, our Namatad system is designed to handle real-time analytics and computations by allowing computations to occur over an online model as soon as sensors ingest data to the topic queues.

2.2 Real-Time Systems with Cloud Computing

Numerous works propose leveraging cloud computing to handle large amounts of IoT driven data as a means to manage scalability, accepting the trade-offs of higher latency and introducing more security concerns [17] [18] [19] [20]. For example, in [21], Reichherzer and et al. conducted a case study in which Arduino boards and Raspberri Pi boards were deployed to transmit sensor data into a cloud based infrastructure in examine tradeoffs between security, scalability, and efficiency in smart home sensor networks. The results showed that for wireless and wired network setups, scalability and security are concerns, while real-time results are obtainable.

2.3 Real-Time Systems with Edge Computing

In many cases, where terabytes and petabytes of data needs to be processed, cloud computing is inevitable. However, for many local and private systems, edge computing has emerged as a more optimal solution [22] [23] [24]. Just as [25] did with their GeeLytics system, we shifted the focus from cloud to edge computing over a fog server. However, our system is influenced by flexible software streaming topology management techniques that handle highly distributed IoT systems and show that not only can real-time results be computed more efficiently by edge computing, but overall QoS, including the scalability, security, and latency challenges proposed in [21] can also be managed. We have shown that by choosing a suitable software streaming topology to handle the routing of data and data management, it is possible to effectively cut down on that systems utilization rates by orders of magnitude providing more scaling potential while introducing minimal latency.

2.4 Queueing Theory

There have been notable contributions to Queueing theory in the IoT field that have been formulated and validated with through experimentation using sensor based infrastructures [5] [26] [27] [28] [29]. Many of these works validated their queueing analysis with a theoretical system design. For example, [6] has shown that by using queuing theory, it is possible to optimize QoS by knowing what arrival rates can handle specific utilization rates, meeting specific time thresholds [6]. Our Namatad system extends this concept by comparing arrival rates against multiple software topologies and determines under which software constraints that we can maximize the number of data steams computed on at a given utilization rate, further improving QoS. Zhenfei [30] applied queueing theory to a theoretical Passive Optical Networks (PON) architecture driven by Wireless Sensor Networks (WSNs) using a single topology to show that decreasing the number of sensors per Optical Network Unit (ONU) will allow for an increase in bandwidth distribution amongst the end users connected to it. Analytically, this was validated by testing it against a packet arrival rating that yielded 100% throughput. With most systems, state of the art hardware is required to handle computations at 100% throughput, which is highly costly. By applying queueing theory to our M/D/c models, our system shows that by choosing the correct software topology you can decrease utilization and still maximize the number of streams running through a hub [7] [31] [32] [33]. This shows that cost effective hardware can be used to process the same number of jobs that state of the art hardware can process at 100% utilization rates, by simply restructuring the software streaming topology before computation.

Knowing that software is available to manage data streams and handle real-time computations, our research focuses on choosing the appropriate software streaming topology under various use cases. Much work has been researched on determining effective physical node

arrangements in network topologies of ad hoc CPSs [34] [35] [36] [37], as well as Topology Control algorithms for how to maintain and manage these nodes [38] [39] [40]. In many use cases, freedom of node or sensors arrangements in IoT systems are constrained to building blueprints and safety compliance and consequently, real-time analytics constrained to existing sensor deployments. Our system provides multiple software streaming topological granularities with analytical comparisons using existing sensors that improve QoS telemetries.

Chapter 3 System Architecture

Many IoT server systems are composed of one or more web servers used to ingest sensor data streams and store them in long-term storage. This stored data is then analyzed using various tools as well as visually displayed for further evaluation by analysts. However, given the temporal utility of many IoT sensors, we really need the capability to analyze these data streams in real time.

Motivated by the Big Data movement, there are many open source frameworks available for processing large data sets, including Apache projects Hadoop, Spark, and Flink. There are also projects such as Apache Kafka [41] that provide persistent message broker functionality and Apache Nifi [42] that enable data streams to be routed through various computational and storage steps. Because our goal is to dynamically model and manage real-time data streams for diverse sensor deployments, we leveraged these existing projects, that each provide a subset of the functionality needed, to create a new end-to-end streaming system for IoT we call Namatad.

In many IoT deployments, sensor data is transmitted to a back-end server environment for storage and analysis using the lightweight MQTT protocol [43] [44].

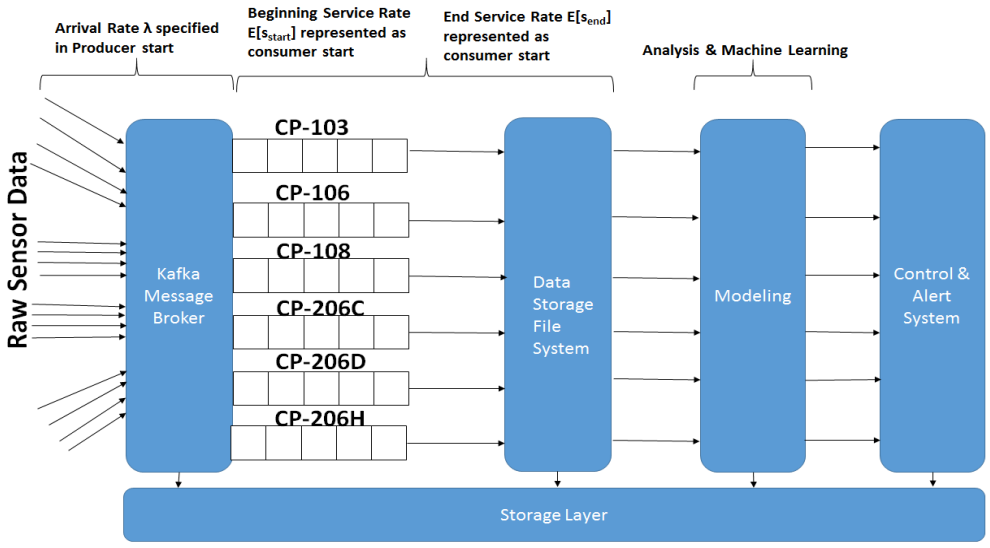

Figure 3.1 Namatad System Architecture

IoT gateways or hubs are then used to aggregate sensor data sent via MQTT and forward to Cloud-based server environments. Sensors can also be configured to send data values directly to Cloud-based servers for collection, reporting via dashboards, and historical analysis. In this work, we leverage simple, lightweight JSON-based schemas, including Apache Avro, to send sensor data between edge devices and our streaming server platform [45].

As shown in figure 3.1, we ingest all sensor data through the Kafka message broker. Kafka manages data in queues, referred to as topics, which we used to aggregate sensor data by room. The sensor devices were configured to send readings, each with a time stamp, to the Kafka room topic within which the sensor was located. For example, for the sensors in Room CP106, we routed all sensor data to the Kafka topic CP106. To enable a hierarchical view of the state of each building (with state viewable by room, floor, and the entire building), we also created additional Kafka topics to aggregate the room-level sensor data by floor. Similarly, we aggregated all floor-based sensor data into a building based topic that captured the state of all sensors for the building. While this does replicate the sensor data that is sent to Kafka, it does not change how the IoT end point devices transmit data. Moreover, it provides a simple way to couple analytical models at multiple levels of granularity in real time without having to filter very large data streams multiple times merely to gain insight into a particular floor or room. Once the sensor values are received within a Kafka topic they are persisted to storage for later retrieval.

After the sensor data is stored in the appropriate queue within Kafka, each data stream then undergoes additional filtering. For some sensor readings, this involves normalizing values, aggregation, or transformation of format for consumption by subsequent computational steps. Once a sensor reading within a data stream has been filtered, normalized, and aggregated, this data

serves as input to our machine learning model. While there are many languages that provide machine learning modeling, Python was implemented for this research [3].

Since the focus of this research is to characterize performance and minimize latency, the following sections breakdown the different components to our system architecture and provide more elaborate detail on how they impact to overall response time of the sensor values as they are added to and serviced in our Namatad system. Understanding the individual components to our system was critical in performance testing to analyze raw data that was transmitted from the sensors to observe empirically, what contributed to waiting times and aggregation times, which will be explained in the following subsections.

3.1 Sensor

Smart sensors are the driving force behind IoT systems, as they allow smart devices the ability to connect to the Internet to collect and exchange data [9]. Devices whose functionality are dependent upon and systems that model predictions based on the accuracy of sensors, rely on sensors to make sense of what is happening in the real world to perform expected behaviors and functionality. As a result, it is imperative that sensors are strategically deployed in such a way that values are accurate and are placed near hubs in a way to minimize the network latencies involved with the transmission of the data. However, in many cases, the placement of the sensors and hubs are constrained by building blueprints and safety compliance. Given the predetermined placement of these sensors, we have to determine the most effective way to transmit the data and perform computations.

The building we collected our sensor data from, Cherry Parks, is a mixed use building consisting of administrative offices, classrooms, faculty offices, and engineering laboratories. This

takes into account a multitude of external influences, such as the density and sparsity of classroom sizes relative to the number of people occupying the room at a given time, as well as the deployment topology of the environmental sensors. Each room in the Cherry Parks building is instrumented with the four environmental sensors, including, 1) room temperature(RT) sensors (Model 540-660B), 2) CO2 sensors (QPM2100), 3) HVAC supply air volume (AV, in cubic feet/minute or CFM) sensors, and 4) HVAC supply air temperature(AT) sensors. The CO2 sensors were mounted on the return air ducts, while the temperature sensors were placed near the entry door [3]. Transmitted packet data is written to a message broker, containing timestamped sensor values. The reading of a single sensor data value will follow a *[YYYY-MM-DD HH:MM:SS, sensor value]* schema. For example, the data value for CP-103-CO2 will be represented as *[2016-01-05 12:45:00, 368.67]*.

3.2 Transaction

Because we have designed our experiment to ensure the model receives all four sensor values from a room before making a prediction, we define a single transaction time as T(x), where T is the transaction at real world time x, as the aggregated tuple for all four sensors values for a room (CO2, AV, RT, AT) ready to be pulled by the model, as shown in the figure 3.2.

As figure 3.2 shows, sensor data values are pushed into Kafka, where the consumer pulls using a bijective mapping. Once all four sensor values for a room at a given real world time x, are retrieved by the consumers, they are then combined as one transaction tuple and written to Ignite, where the model pulls from and uses to make predictions. The process of forming a transaction forms during the consumer phase. The log parsing values for the timestamps of a transaction can

be seen to the right of the transaction architecture flow chart in figure 3.2, contain the following schema:

[sensor name, producer start time, producer end time, consumer start time, consumer end time, model start time, model end time]

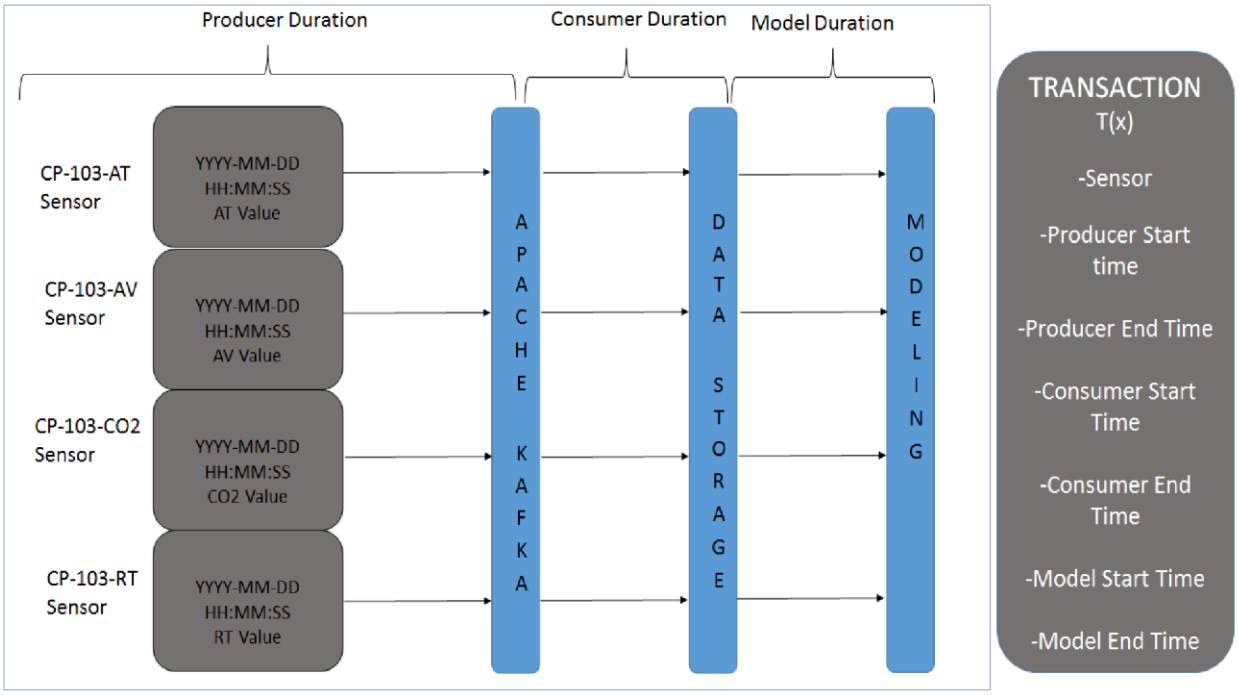

Figure 3.2 Namatad Transaction Architecture

The schema for a completed key-value transaction tuple is represented as:

[(Real World Time Stamp, Room), (AT value, AV value, Co2 value, RT value)]

and by adding sample values, it will look as follows:

[(12:45:00, CP-103), (72, 12, 401.92, 74]

The time it takes to service a transaction is dependent on three components; the aggregation time, the time it takes to write to Ignite, and the time it takes to predict. Because we are predicting occupancy at room level granularity, a transaction tuple must have all four sensor values for a

room at a given timestamp before it can be used to prediction. Because our system contains asynchronous data, aggregation is dependent on the time the sensor values arrive. The wait time is impacted by the arrival time and ultimately, the response time is a function of the order in which the sensors arrive. Figure 3.3 shows an example of how the aggregation time is effected if the sensor values arrive in order:

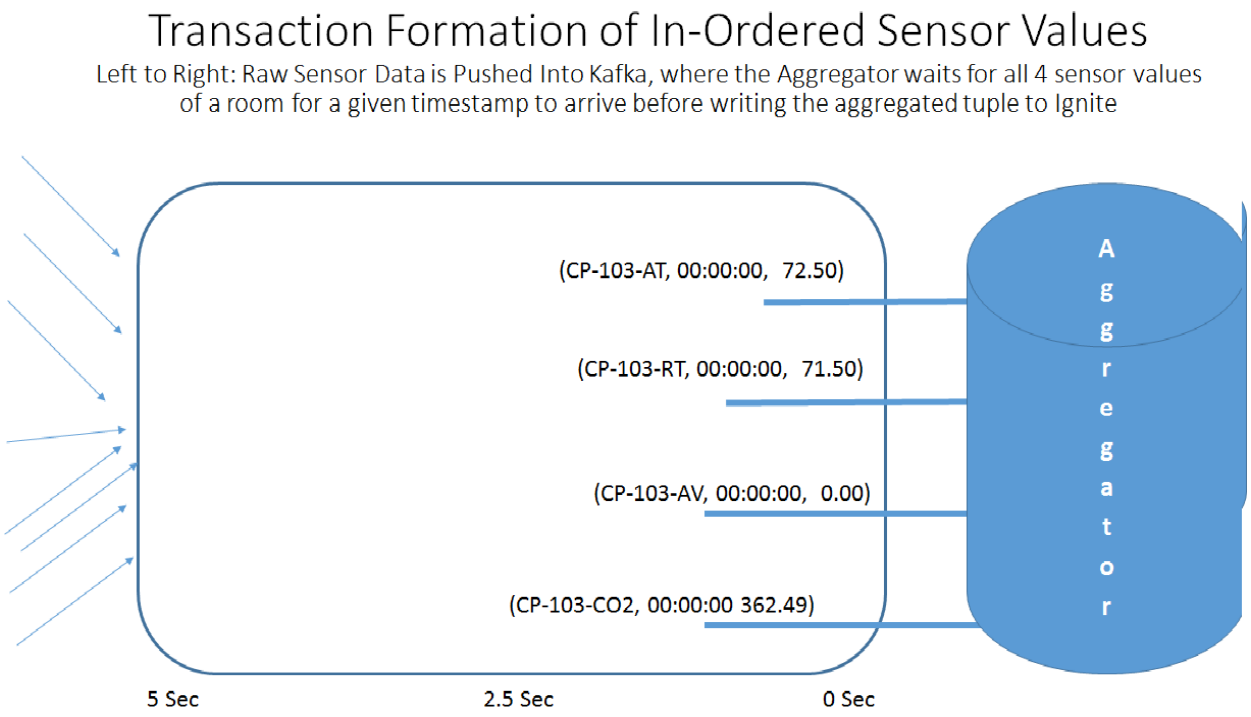

Figure 3.3 In Ordered Sensor Arrival

Figure 3.3 shows how four sensors values for a room are formed when their values arrive relatively in order. During this five second interval, all of the sensor values arrive within a time frame of one second and as a result, the aggregation can form a transaction tuple in one second. Figure 3.4 shows a case in which sensor values for a room transaction arrive out of order. The earliest sensor value arrives in 2 seconds and on average, the sensors are arriving in approximately

2.5 seconds. However, because the last sensor value arrives after 5 seconds, the entire transaction has to wait five seconds before being written to Ignite.

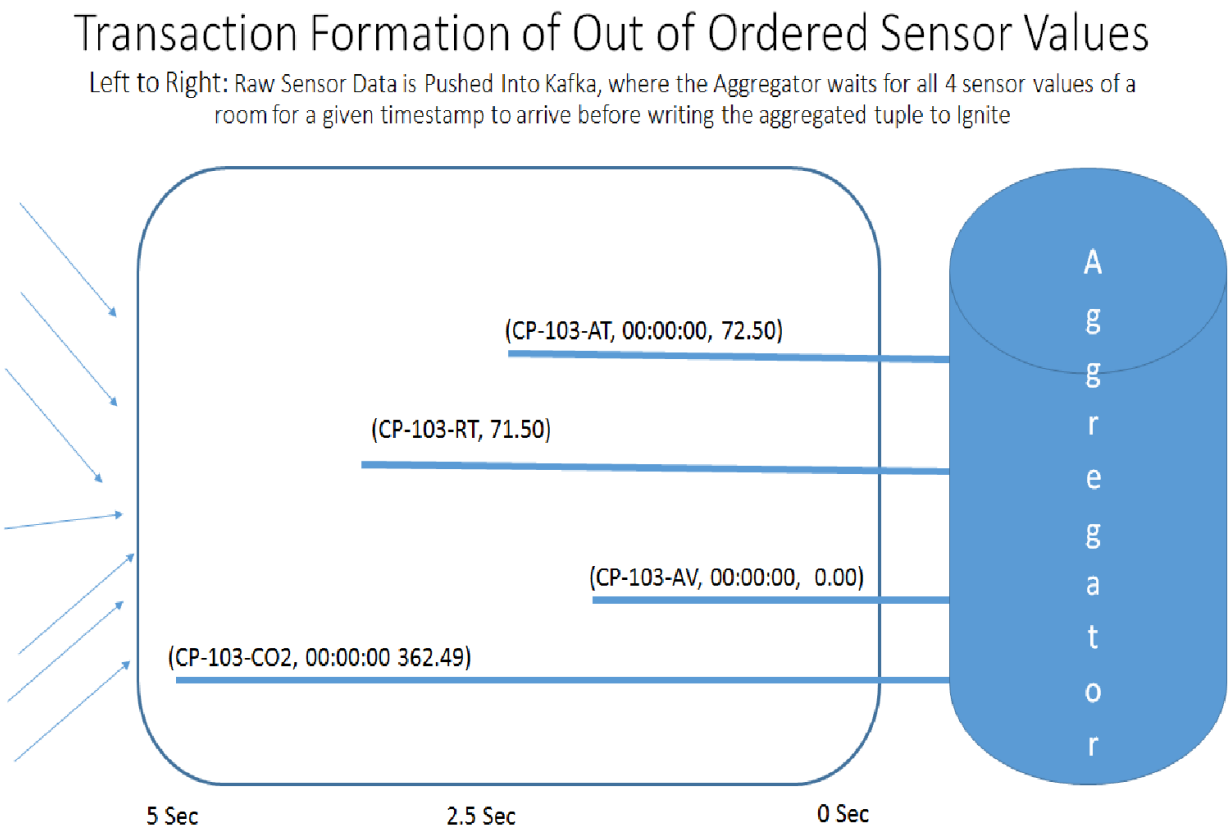

Figure 3.4 Out of Ordered Sensor Arrivals

3.3 Topology

Traditional network device environments have a control plane and a data plane on each device, leading to limited centralization. As a result, Software Defined Networks (SDNs) have been developed to offer more flexibility in providing more options to where data packets can be routed [10]. Since we can think of control planes as the learning of these different routes and the data plane as the actual forwarding to one of these routes, we can immediately see how choosing

the right route, or software streaming topology, will optimize performance and minimize latency. Our Namatad system allows for easy creation of ad-hoc topology pipelines to be created that allow data packets to be routed to the most efficient queueing system, which we are able to analytically validate. For our experiment, we have designed four different software topologies which our Cherry Parks building sensors can be routed to.

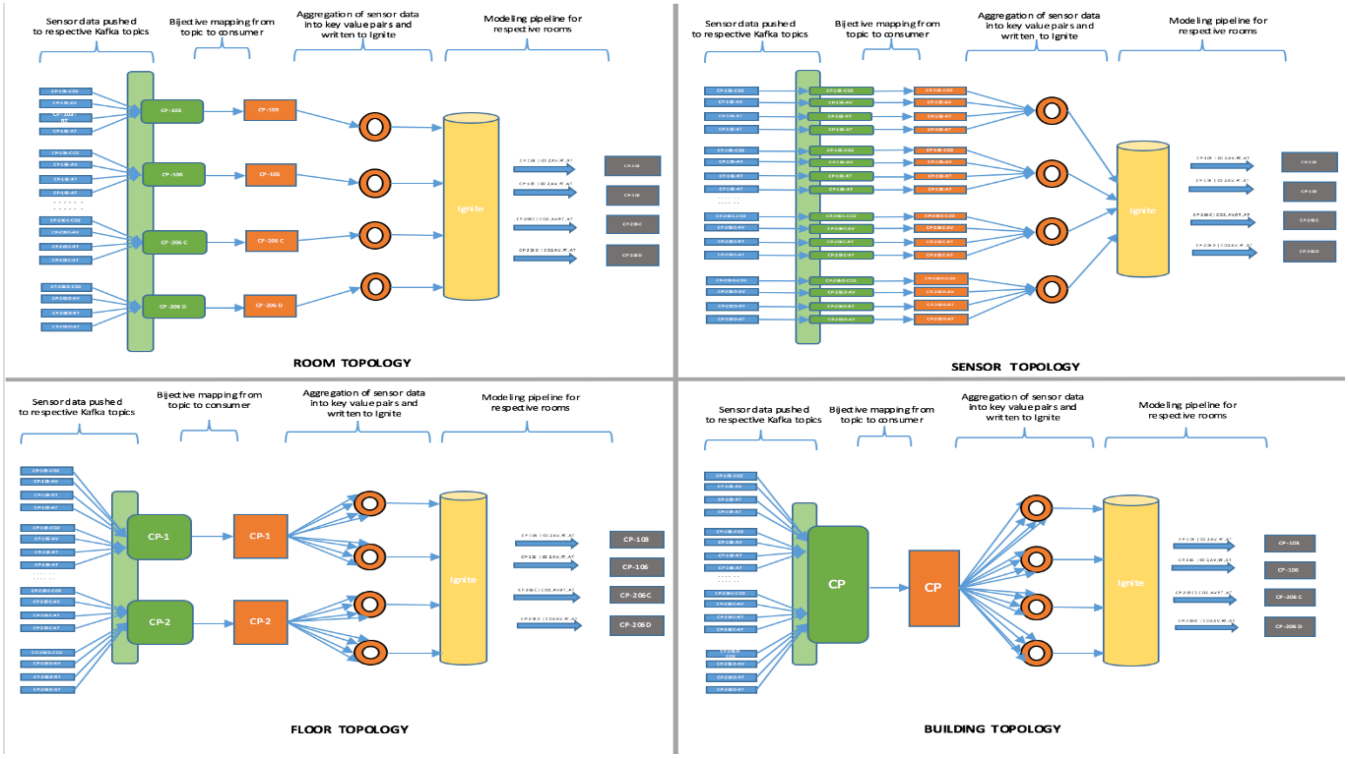

Figure 3.5 All Namatad Topologies Diagram

Figure 3.5 shows that different topologies have different number of topics. For each topology the topics are created before the experiments are run. An example of how topics are created is as follows: CP-103-CO2 topic is created for sensors topology, where all sensor values from CP-103-CO2 are routed to and a topic is created for each sensor; CP-103 for rooms topology, where all sensor values containing CP-103 go into one topic and a topic is created for each room; CP-1 for floor topology, where all first floor sensors containing CP-1 go into one topic and a topic

is created for each floor; and CP for building topology, where all sensors for the CP building go into one topic.

Because the model predicts occupancy at room level granularity, there is a specific process that each topology must undergo to properly form its respective transaction. As seen in figure 3.5, in the case of the buildings topology, individual sensor data is streamed into Kafka where it gets aggregated into one building topic. Using a bijective mapping, a single consumer pulls the values from the buildings topic, where it then segregates the values into their respective transaction tuples according to their room and timestamp and is then written to a temporary data file storage system. At that point, the model pulls the tuple from Ignite, where it makes its prediction. In the case of the floors topologies, individual sensor data is streamed into Kafka, where it gets aggregated into its respective floor topic. A consumer for each of the two floors pulls the values from its respective floor topic, where it then segregates the values into their respective transaction tuples, based on room and timestamp and is written to a data file storage system, where the model pulls the tuple for prediction. In the case of the rooms topology, individual sensor data is streamed into Kafka, where it gets aggregated into its respective room topic. As a result, there is no need for aggregation or segregation from the consumer side. A consumer for each of the six rooms pulls the values from its respective topic, where it forms transaction tuples, based solely on timestamp and is written to a temporary data file storage system, where the model pulls the tuple for prediction. In the case of the sensors topology, individual sensor data gets streamed into Kafka, where it immediately gets routed to its proper sensor topic. As a result, there is no need for the sensors topology to aggregate its data after its arrival into the system. A consumer for each of the 24 sensor pulls the values from its respective topic, where it then aggregates the values into their

respective transaction tuples, based on room and timestamp and is written to a temporary data file storage system, where the model pulls the tuple for prediction.

The number of topics for a topology determine how many threads will be computed, which we will represent as a computational server in our queueing model. The number of topics for the sensors topology is 24 (4 sensors per room X 6 rooms) and will be represented as M/D/24. The number of topics for the rooms topology is 6 (since 6 rooms in total) and will be represented as M/D/6. The number of topics for the floors topology is 2 (since 2 rooms in total) and will be represented as M/D/2. The buildings topology is 1 (since there is 1 Cherry Parks building) and will be represented as M/D/1.

For this paper, we have primarily focused on studying the effects of various topologies on the latency, or the total response time of a job represented as a transaction, of prediction. The experiment also highlights how the quality of prediction is affected by the topology. These are discussed in detail in the forthcoming sections. The advantage of using the Namatad platform is the ability to quickly create additional computational pipelines for data streams, as well as additional models which can be easily deployed and evaluated. The deployment of these models provide the capability to improve building control systems by predicting trends unobservable by traditional control systems.

3.4 Machine Learning

There are many different instruments that can predict occupancy, such as video sensors, motion sensors, etc. that can offer more precise modeling and occupancy predictions. Since we are using existing CO2 sensors provided by the university and do not know how sensitive it is to particles, there can assume to be many external factors than can influence irregular particle

readings, such as excessive burning of fossil fuels for electricity to power buildings, bringing in cigarette smoke, gasoline from cars and motorcycles, etc. This presents a challenge in choosing a "best" algorithm with the given sensors

From our Namatad paper, we know that in machine learning there are two predominant approaches to making inferences, unsupervised and supervised techniques. Unsupervised learning techniques are useful when we are trying to discover unknown patterns in a data set or stream. While this technique could be useful for uncovering patterns in newly deployed sensors, we did not find strong correlations with the installed sensor types using K-means clustering. Supervised techniques are useful when we are trying to predict a condition definitively. For example, if we know that a room is occupied by 15 people between the hours of 10 a.m. to 11 a.m. we can then evaluate the predictions for a given model against this ground truth. This can be challenging as we must ensure we have accurate ground truth established. In this paper, we describe our approach using supervised learning to predict occupancy [3].

Since the focus of this paper is on systems and since choosing a "best" algorithm for predicting occupancy using the existing sensors presents a challenge, the multi-class Random Forrest algorithm was chosen as the classifier. The Random Forest algorithm is a very common and powerful machine learning ensemble method that has been successfully used across many domains for classification. Random Forest is a tree-based machine learning algorithm that uses bagging to build an ensemble of decision trees to make predictions. We are using multidimensional data sets in which the Random Forest algorithm is good at accounting for variable dependencies, being capable of averaging out biases from multiple features. We are using supervised learning, where we have obtained class sheets that give the number of students in the

class, which we have tested against our prediction results to allow us to predict occupancy as high as 95% and as low as 71%.

Chapter 4 Emulation Experimental Results Extended

To better understand how this research is designed to characterize the performance impact topology has on response time, it is important to extend the results in our initial paper. While our initial work showed that we were successfully able to design an end to end system capable of streaming data and making predictions using an online model, the focus at that stage was not to characterize the performance impacts due to topology. For our initial work, our Namatad System only ran on a single rooms topology. It was from the success of our initial work that we wanted to see how speed and accuracy would work under different topologies to determine if the rooms topology was the most efficient to use. If we are able to analytically characterize the impact that topology has on response time for our system, then we will be able to analytically validate all IoT systems by tailoring the topology to the use case.

4.1 Sensor Readings

The initial step for this research was to identify how we could leverage machine learning techniques with our streaming platform to predict occupancy using only the existing building sensors in the Cherry Parks building. Upon collaborating with the Facilities Management Department, we were able to obtain access to the data for this work. We found that the sensors were read in five minute intervals with data collected and stored. While the temperature sensor provided direct input to the HVAC systems within the building for environmental control, the other sensors did not directly feed into other control systems except when dealing with maintenance issues.

We initially used sensors readings from a period of three weeks during the month of May in 2016. We also collected occupancy of each of the rooms we use in this study via physical

observation. Figure 4.1 shows the plot of sensor readings, values, and changes over a one-week period for one of the rooms (CP-103), which is used as a classroom as well as a general meeting room. The x-axis shows hours during the week and the y-axis shows the sensor values for normalized to the maximum value observed for the week. We observe the values change over time throughout the day depending on use. However, there is a significant difference in variance between the readings. As expected, temperature

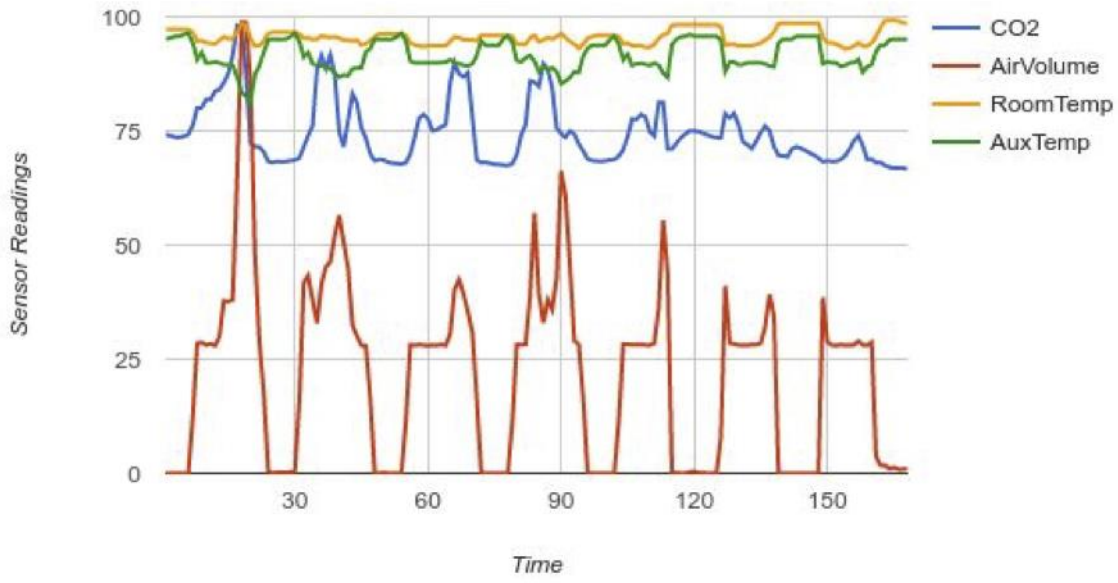

Figure 4.1 CP-103 Sensor Value Readings Over One-Week Period

remains nearly constant as it constitutes the set point the HVAC systems are set up to track and maintain. Consequently, despite the intuitive correlation between occupants and temperature, in isolation, temperature is a poor indicator of occupancy. We observe that incoming, HVAC-supplied air temperature is caused by ambient room temperature and there is significant variance in the supply air volume. However, it is unclear whether the changes in air volume are directly correlated to occupancy. For example, if room temperature increases and the HVAC system activates to reduce room temperature to the set point, the increase in air volume might be attributed

to an increase in occupancy. However, if the temperature decreases in the room and the air volume increases to raise the temperature, the increase in air volume could simply be due to building heat loss during a seasonal cold period. Consequently, in isolation air volume could be poor indicator of occupancy. CO2 does exhibit variance that can be attributed to occupancy. As occupancy increases, the CO2 levels increase. In many cases, the CO2 level increases more rapidly than temperature. Although CO2 levels are not actively controlled by the HVAC system, when the temperature increases (or decreases) sufficiently the HVAC system is activated. This has a mitigating effect on CO2 as new air is delivered to the room reducing the concentration of CO2. Despite this limitation, because we are limiting our occupancy predictions to only use these four existing sensors, we chose to leverage CO2 as a primary indicator of occupancy. Still we must take into account the mitigating impact of the volume of fresh air supplied when the HVAC system activates. In addition to the sensor readings collected, we also collected occupancy data by visiting the physical rooms used in our evaluation. Figure 4.2 shows the observed occupancy over the same one-week time period for the same room, CP103. Comparing the figure 2 with figure 3, we observe a correlation between occupancy and the air volume during the initial half of the week, but a poor correlation in the latter half. This aligns with our previous intuition about

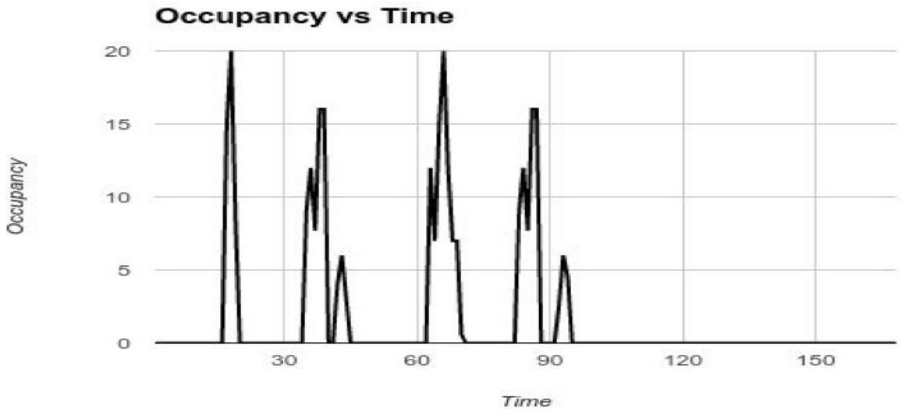

Figure 4.2 CP-103 Observed Occupancy over One-Week Period

air volume as a primary feature to predict occupancy. However, we see a closer correlation between occupancy and CO2. During the first half of the week there is a correlation between CO2 and occupancy. While there is some variance in CO2 during the second half of the week during which we observed the room was unused, it is minimal relative to the changes in readings from the other sensors. These observations motivated the development of a machine learning model to predict occupancy using the sensor values.

4.2 Occupancy Prediction

A key requirement for supervised learning is the establishment of ground truth. In our case, the ground truth consists of the occupancy levels of the monitored rooms. To establish ground truth, we observed the occupancy of the rooms over the period of several weeks throughout the day. Because our observations were of classrooms and laboratories on campus during the academic quarter, the occupancies were fairly consistent. These observations constituted the ground truth on which we based our classes labels to train our machine learning models. We defined five classes of occupancy based on the binning of people into groups. The first class (class-0) represented the case of zero people in a room. Class-1 represented the room was occupied by one to five people, and class 2 represented the occupancy level of six to ten people. A motivation for this transformation was the variance in occupancy we detected when establishing the ground truth occupancy. While we are not predicting an exact occupancy count using this approach, we are able to estimate how occupied a room is at a given time. In the case of an emergency scenario, this information could be invaluable in reducing victim search time. Given the groupings defined by the labels, the goal of our machine learning model is to classify the sensor data into different occupancy classes. We previously observed a correlation between occupancy and CO2, however, we also noted the other sensors impacted the prediction as well. To characterize the impact of each

sensor reading on prediction accuracy, we trained several version of models using three weeks of the sensor readings that had been previously collected by UW facilities. We trained one model using only CO2 sensor readings to predict occupancy. We also built several additional models that considered several combinations of sensor values. These models included:

- CO2 and HVAC air supply volume (Air Volume)
- CO2 and HVAC air supply temperature (Aux Temp)
- combination of all sensors - temperature (Room Temp), CO2, HVAC air supply volume (Air Volume), and HVAC air supply Temperature (Aux Temp).

Table 4.1 Covariance against all Pairing Combinations of the 4 environmental Sensors

Attribute 1	Attribute 2	Covariance value
Time	Occupancy	152.74
Time	CO2	2003.2
Time	Air Volume	7104.39
Time	Aux Temp	-11.24
Time	Room Temp	43.61
Occupancy	CO2	561.62
Occupancy	Air Volume	2507.09
Occupancy	Aux Temp	-46.89
Occupancy	Room Temp	-19.81
CO2	Air Volume	14659.22
CO2	Aux Temp	636.16
CO2	Room Temp	801.47
Air Volume	Aux Temp	-483.5
Air Volume	Room Temp	461.14
Aux Temp	Room Temp	137.54

As shown in Table 4.1, we were able to calculate the covariance values for the given combinations of attributes we modeled with the data we obtained from facilities. The results validate our initial intuition that as a standalone sensor, the CO2 sensor would be the biggest influence in predicting occupancy. There is a correlation between occupancy an CO2, in that as occupancy increases then CO2 levels also increase. While air volume does show the best standalone covariance when measured against time, the uncertainties of what causes an increase in HVAC air supply cannot be correlated to occupancy. However, when paired together, they show the highest covariance, indicating that that the change in one attribute are associated with changes in the other. The table also shows that room temp and auxiliary temperature are the least efficient standalone sensors to predict occupancy.

We then evaluated the prediction accuracy of the model using 10-fold cross validation. We chose to evaluate the models we built to predict occupancy using sensors from three unique rooms. The data sets for each room differed in terms of sparsity. For some rooms we had a limited set of data points for each of the sensors. This was due to a building management power policy that shut down certain sensors during periods of each day, such as at night. For other rooms, we had sensor readings from all sensors every five seconds. This reflects many heterogeneous IoT deployments where sensor densities vary over time or by location. Because we wanted to ensure our approach works consistently across alternative deployment scenarios, we selected three rooms with sensor readings with different sparsity levels. Room CP-103 included sparse sensor values, room CP-108 included dense sensor readings, and room CP-106 revealed a moderate number of readings. Figure 4.3 compares the prediction accuracy of our initial models constructed to predict occupancy using a set of

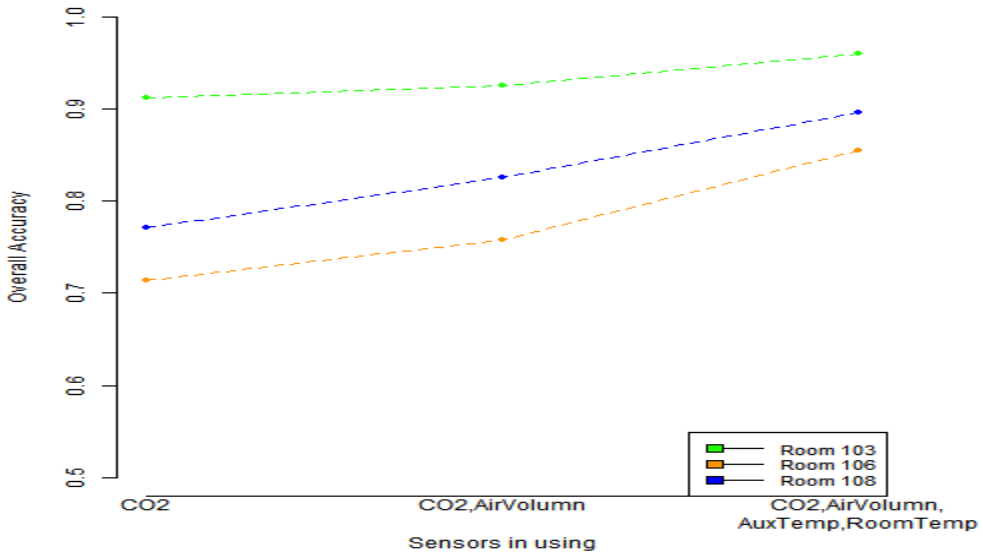

Figure 4.3 Occupancy Predictions Against CO2, (CO2, Air Volume), and (CO2, Air Volume, Aux Temp, and Room Temp)

sensor readings for three different rooms within the Cherry Parkes building. Starting with CO2 in isolation, we achieve nearly 91% prediction accuracy for occupancy for room CP103. However, recall that in many cases if we take HVAC supply air volume into account we hypothesized we could potentially increase our accuracy. This is precisely what we observe when we combine these two sensor readings into our machine learning model. The second point in figure 4 shows that our prediction accuracy increases if we incorporate HVAC air volume into our model. Finally, even though temperature is a poor indicator, we incorporate all of the sensor readings into the model, further increasing accuracy up to 95.9%. We also evaluated the combined use of CO2 combined with the HVAC incoming supply air temperature (Aux Temp) within our model. Figure 4.4 shows the prediction accuracy of the models constructed based on the different combinations of sensor readings, but focusing on the impact of CO2 combined with HVAC incoming supply air temp (Aux Temp). Similar to figure 4, we see an increase in prediction accuracy for room CP-103 of about 2% from 91% with CO2 only to 93% taking into account CO2 and Aux Temp in our model.

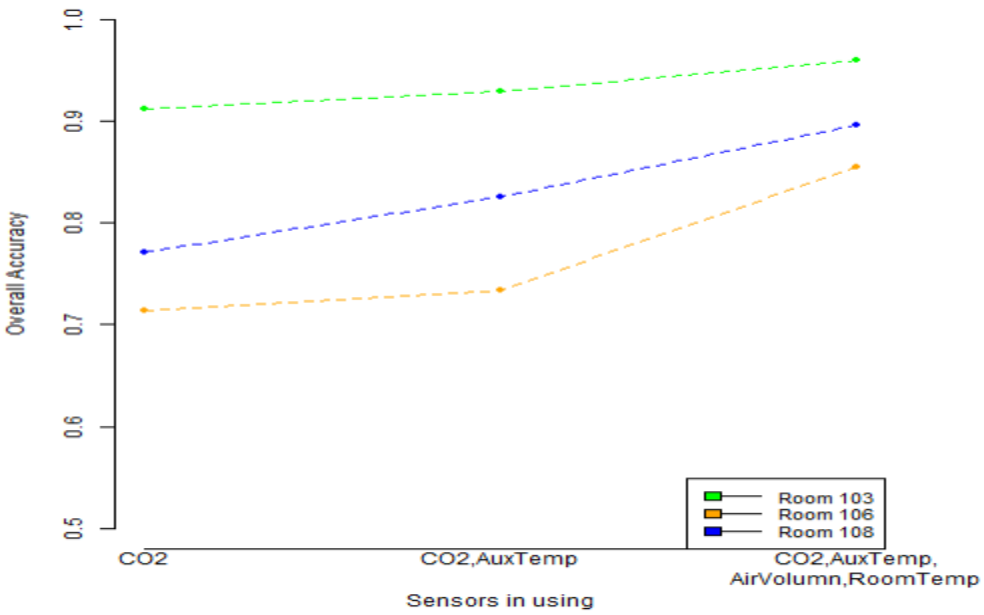

Figure 4.4 Occupancy Predictions Against CO2, (CO2, Aux Temp), and (CO2, Aux Temp, Air Volume, and Room Temp)

As noted previously. the prediction accuracy does vary with the sparsity of the data provided by the sensors. We are still quantifying the impact of sparsity on our prediction accuracy. However, in all cases, with only a single sensor we were able to achieve greater than 71% and up to 91% accuracy using only CO2 to predict occupancy. We have since incorporated these models into our Namatad streaming IoT system and are currently evaluating rooms on all floors within the building.

4.3 Initial Performance Model

Being successfully able to design an end to end system capable of streaming in sensor data, processing the data, and making predictions on the data via machine learning, we now wanted to focus on improving speed. Our next step was to identify alternative streaming topologies for IoT server platforms to see if we could gain an increase in the performance of the response time. We then created three additional topologies in which we could predict occupancy against at room level granularity in buildings, floors, and sensors. Initially, we created additional topics for the

respective topologies and immediately began predictions as soon as sensor values arrived. First testing against accuracy, we obtained the following results shown in figure 4.5 after predicting occupancy under different topologies:

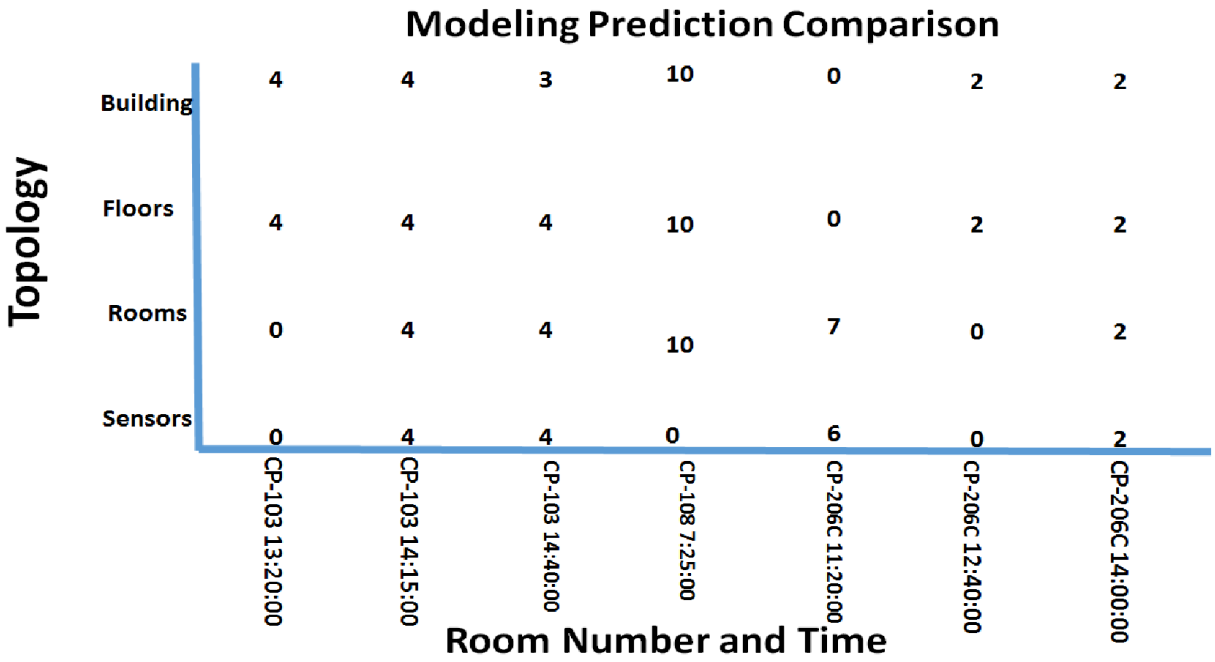

Figure 4.5 Prediction Comparison of 4 Topologies using Buckets

We were able to notice inconsistencies between the predictions across topology. While we don't expect exact results due to run to run variances, we expect nearly similar results. After confirming with our raw log files, we noticed that at certain times of the day, predictions weren't inaccurate, they were missing, in the case of the 0 prediction buckets.

While facing a major performance loss in accuracy, we considered two approaches to fixing the issue of the missing values. We introduced an aggregator component into our system, that waits and stores the four sensor values for a room, known as a transaction. Once the four values for a transaction have arrived, the aggregator sends this transaction to Ignite, where the model pulls from. While this slightly increases the overall response time of the job, this ensures

that there are no missing values and that the model is making accurate predictions. The second approach, which we will design for future works, is to add imputation to the system instead of the aggregator because the challenge is choosing the best values to substitute for a missing transaction values prior to prediction.

Having found a way to control accuracy, we then reran our system, introducing timestamps to capture the start and end times for the producer (Kafka push), consumer (preprocessing), and modeling (machine learning predictions) phases of our experiment, so that we could measure and compare the performance across the four topologies.

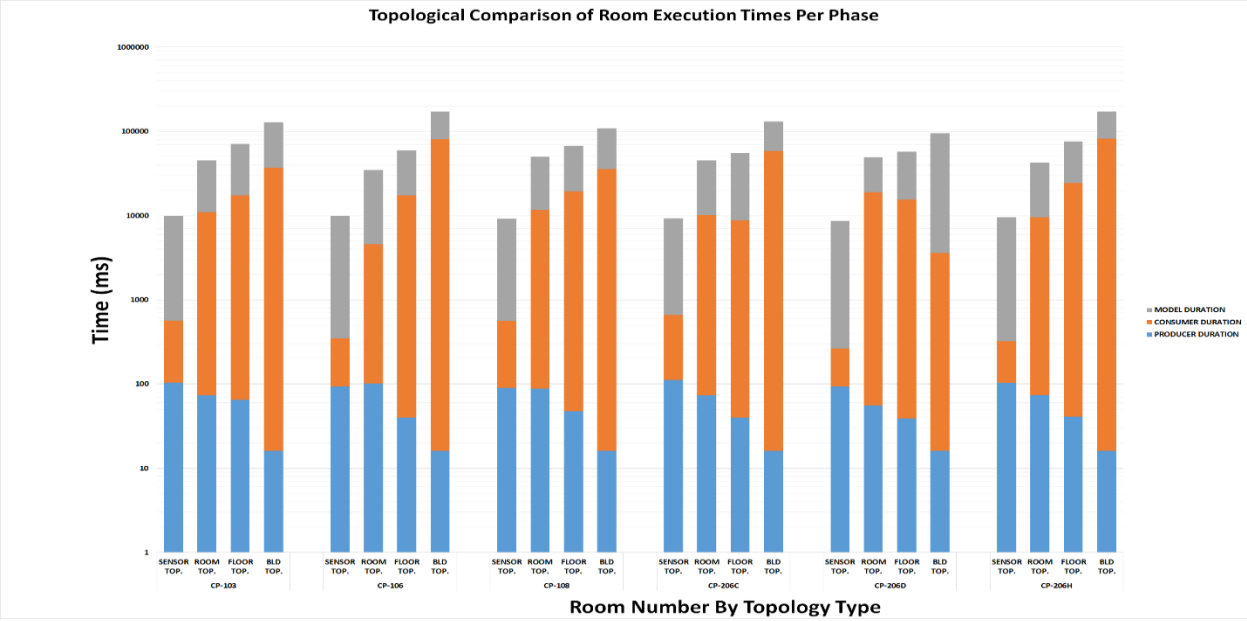

Figure 4.6 Topological Comparison of Execution Times Per Producer, Consumer, and Model Phase

Figure 4.6 shows how each of the four topologies compare during each of the three phases. We can see from the graph that during the producer phases, it takes the least amount of time to route the sensor data to the buildings topology. As you go from broad to more detailed topologies (buildings to floors, floors to rooms, rooms to sensors) the time it takes to send the data into the

respective Kafka topics increases. In the case of the buildings topology, there is only one topic that is created, so all sensor data is immediately routed into one topic queue. This is represented in the figure as the buildings topology has the shortest producer phase. However, the consumer service phase takes an exponentially longer time to process and results in the overall longest response time. In the case of floors, data now has to be carefully routed into one of two dedicated topics, based on the floor that the sensor value is associated with (first floor or second floor). In the case of the rooms topology, the sensor data now has to be carefully routed into one of six topic queues (one topic for each room), which takes even more time to route. While the producer Kafka routing phase takes longer than the buildings topology, the consumer service phase and total response time is much less than the buildings topology. Finally, in the case of the sensors topology, there sensor data has to be routed carefully into one of 24 topic queues. We can see that the more detailed granularity we choose, the longer it takes to route the sensor value correctly to its proper topic.

In the case of the consumer phase, we notice that the service time decreases as you go from broad to more detailed granularities. Because we are making predictions at room level granularity, the time to service the transaction is dependent on the time it takes to aggregate the sensor values into a proper transaction. In the case of the buildings topology, it takes longer to aggregate sensor values into transactions because it has to sort through sensor values from six different rooms to find the correct sensor values for one room for a given real world time. In the case of floors, it has to sort through sensor values from three different rooms to find the correct sensor values for one room for a given real world time. In the case of the rooms and sensors topologies, everything in the queue is already sorted by the correct room, so the sensor values only need to be sorted by the real world time value, with the sensors topology being arranged in a more precise manner because

once it finds one correct sensor value, it knows it doesn't need to look for anymore values of that specific sensor type. This is not the case for the sensors topology.

Once the aggregation is completed, the modeling pulls and predicts every run with a four sensor transaction tuple, so the time to predictions is always constant (except for run to run variances). When looking at figure 4.6, the Y-Axis is on a logarithmic scale, so the length of the modeling bar is proportionate to its time value on the logarithmic scale.

From the initial performance analysis, we were successfully able to determine that we are able to sacrifice longer producer time in the time it takes sensor values to be routed to their respective topic because the service time and overall response time will significantly decrease. Our initial results show that from what we are able to control through our experiment, the sensors topology has the shortest service time and as a result, the shortest overall response time.

Our initial performance model has proven that we can characterize and compare the performance and the impact that topology has on response time for the four topologies in the Cherry Parkes Building. However, this particular building uses 60 sensors. This model is not an accurate representation of some of the much larger buildings on campus, that have over 200 sensors. The initial performance model does not account for an increase in scalability and work load intensity. This was our motivation for moving from an empirical performance model to a general purpose analytical model, via queueing theory, where we can account for different utilization rates that are influences by different arrival rates.

Chapter 5 Analytical Model via Queueing Theory

One of the critical challenges IoT systems face is effectively managing the way in which sensor values arriving are routed to their respective queues. Congestion occurs when the rate at which jobs are arriving exceeds the number of jobs that can be serviced in that time. There are congestion control protocols in place that focus on queue length and use a simple adjustment technique to keep the length of the nodes as low as possible. This is a three phase protocol that consist of a congestion detection phase, a congestion notification phase, and a rate adjustment phase. Queueing models are primarily used for the congestion detection phase [5].

The characteristics for an M/D/1 queue, with a Markovian arrival rate and a deterministic service rate using one computational server has been completely characterized. However, there are few literatures that focus on extensive calculations of multi-server M/D/c queues [31] [32]. Tijms has shown work to validate that the addition of computational servers into a M/D/c queue is a functions of the utilization rate in an M/C/1 queue over the number of computational servers, in which we have used for the analytical modeling of our Namatad system [31].

Because our Namatad systems streams sensor data into Kafka topics, represented as queues, we can analytically quantify the waiting times, service times and response times of the transactions based on the rates at which their respective sensor values arrive in the system. We can analytically see how the system works under different utilization rates, which gives us the ability to make many different inferences. If a system is underperforming, values could be missing because they haven't arrived before the model can pull them and predictions could become inaccurate. The model could also wait for all of the values to arrive, yielding accurate predictions if the sensor values are arriving out of order. However, because of additional wait time, the

predictions would no longer be met under real time constraints. Likewise, if the system is over performing and jobs are arriving much faster than they can be serviced, sensor values could be in the system. However, if the queue depth is large and sensor values have to be searched for at the end of the queue, the added wait time could also jeopardize real time results or the model could attempt to pull values much faster than they are arriving, yielding inaccurate results based off of missing values. Our Namatad system is currently comprised of four individual M/D/c queueing systems, one for each topology.

We define the arrival rate, λ, as the number of jobs that enter the system per unit time; the service time, T[s], as the time it takes to service a single job; and the number of computational servers, c, as the number of computational processers that concurrently service jobs at a given time. For example, in the case of the rooms topology, represented as an M/D/6 queue, one job in each of the 6 Kafka topic queues can be serviced in parallel. The following equations are used in characterizing different performance components of an M/D/c queue that we will use to characterize our Namatad system:

$$\textbf{Arrival Rate} = \lambda \tag{5.1}$$

where the arrival rate, λ, defines the number of jobs that enter the system per unit time,

$$\textbf{Service Time, Constant} = T[s] \tag{5.2}$$

where the service time, T[s], defines the time it takes to serve a single job,

$$\textbf{Number of Computational Servers} = c \tag{5.3}$$

where the number of computation servers, c, defines the numbers of queues that can service jobs at one time,

$$\textbf{Utilization Rate, } \rho = \frac{\boldsymbol{\lambda T[s]}}{\boldsymbol{c}} \qquad \textbf{5.4}$$

where the utilization rate, ρ, is a function of the rate at which jobs arrive in the system, the rate at which jobs can be served, and the number of computational servers servicing jobs at one time, and defined as the fraction of time the server is busy

$$\textbf{Mean Waiting Time, T[w]} = \frac{\boldsymbol{\rho T[s]}}{\boldsymbol{2(1-\rho)}} \qquad \textbf{5.5}$$

where the mean waiting time, T[w], is a function of the utilization rate and the service time, which calculates the average time that a job has to wait in the queue before it can be serviced.

$$\textbf{Mean Response Time, T[r]} = \boldsymbol{T[s]} + \frac{\boldsymbol{\rho T[s]}}{\boldsymbol{2(1-\rho)}} \qquad \textbf{5.6}$$

Or

T[s] + T[w]

where the mean response time, T[r], is a function of the waiting time and the service time, which is defined as the total time a job is spent in the system, including both the time it has to wait in the queue and the time it takes to be serviced.

Chapter 6 Simulation Experimental Methodology

To provide more scalability and inferences on latency versus prediction quality, we have extended the research from our Namatad paper to incorporate more rooms from the Cherry Parkes building. This lets us continue our ongoing research for which we had initially predicted occupancy and compared across CP-103, CP-106 and CP-108. To provide multiple levels of granularity in terms of sensors, rooms, floors, and buildings level software streaming topology design, we added another floor that includes various room types, such as class rooms, office rooms, engineering labs etc., to account for the sparsity of data and sensor placement as well.

For consistency across runs, we have replayed the same data provided from facilities for each run. This eliminated variance due to workload performance because for each run, the same amount of data is being processed as well as the same type of data. This ensures that the only simulated inconsistencies will result from run to run variances.

For benchmarking purposes, we fixed the CPU frequency on the server to maximum which is 300000Hz on the server machine. This is done to keep the system performance consistent over each experimental run since different topologies have different loads on the CPU and as we have DVFS (dynamic voltage frequency scaling) on the server machine, which could have influenced the run time performances if we did not fix our CPU frequency. Since all components of our experiment are running on a single server, the network latency has been fixed by default, which influences a fixed service time across multiple experimental runs, aside from run to run variances.

To account for the initial backpressure startup cost that occurs if you force data to immediately begin queueing before the consumer beings to pull and service data, we began our experiment by running the consumer first. This avoids an early queueing system, which would

result in a major performance loss. The consumer will initially run and wait for data to come to process immediately, eliminating the chances of backpressure.

We are extending our existing Namatad system architecture shown earlier in figure 3.1. We are still using Apache Kafka as the message broker platform which our data is ingested into. Using a bijective mapping, the consumer pulls data from the respective Kafka topic as per the topology. The data is then held in aggregation while waiting for all four sensor values for a room at a given timestamp to arrive. For the sensors topology, this involves the aggregation sensor values contained in 4 different topics to be combined into one transactions tuple. For the floors and buildings topologies, this involves segregation of sensor values from various rooms to be separated into their respective room transaction tuple. For the rooms topology, neither aggregation nor segregation takes place, but values still need to be written according to the correct timestamp. Once all values for a transaction tuple are present, they are written as key-value pair tuples to a temporary data storage file system. We have chosen Apache Ignite as our data file storage system because of its in memory platform and its easy integration with Apache Kafka and other open source platforms that we will be integrating in the future. This handles distributed computing on large scale data sets in real-time [8]. The model has been designed to pull transaction tuples from Ignite and then to train and make room level occupancy predictions. By placing time stamps along various points in our experiment, we can capture specific values needed to validate our analytical model and can represent the aforementioned analytical telemetries.

Figure 6.1 shows an analytical to empirical decomposition of the equations used in queueing theory where we can see what the service time and waiting time translates to in respect to the empirical timestamps that we measure and at what phases throughout the run the necessary times can be found.

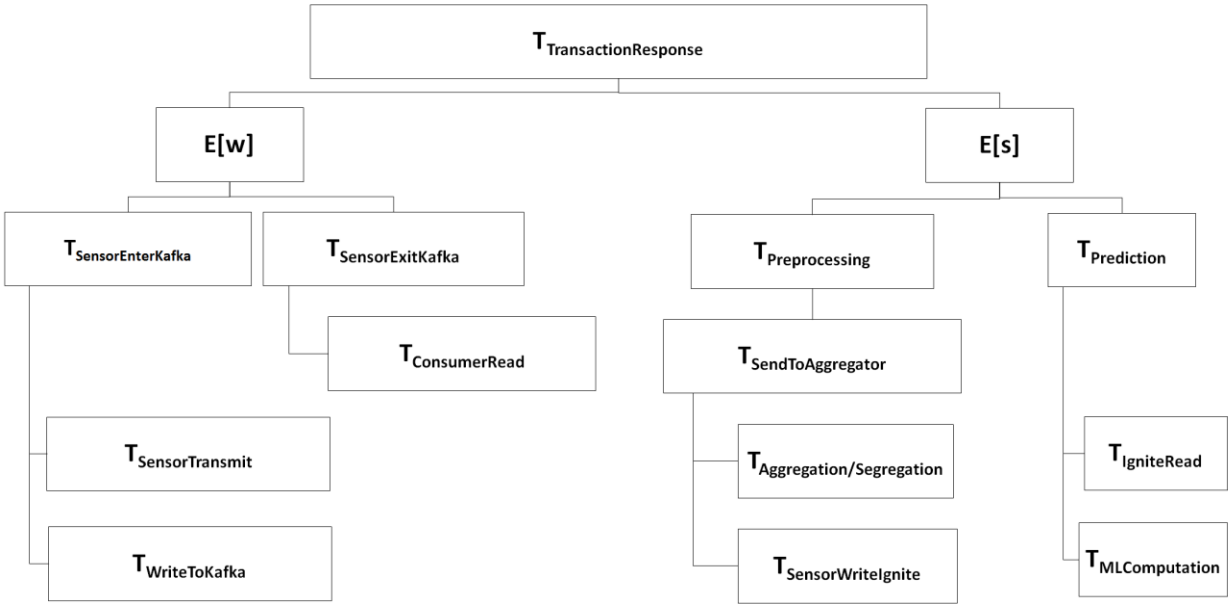

Figure 6.1 Analytical to Empirical Formulae Decomposition Breakdown

From figure 6.1, we can infer the following from our Namatad system:

$$\mathbf{T_{ResponseTime} = T_{Wait} + T_{TransactionService}} \quad \mathbf{6.1}$$

where the empirical response time, $T_{ResponseTime}$, is the summation of the time the sensor values wait in the queue during the aggregation period and the time it takes to pull those sensor values from the queue and write that transaction tuple into Ignite, and the time it takes the model to pull that tuple to perform a prediction,

$$\mathbf{T_{Wait} = T_{LastSensorPulledFromKafka} - T_{FirstSensorPushedtoKafka}} \quad \mathbf{6.2}$$

where the empirical wait time, $T_{Wait,}$ is the total time length that all four sensor values have to wait in the queue before a transaction tuple is completely written to Ignite,

$$\mathbf{T_{SensorEnterKafka} = T_{SensorTransmit} + T_{KafkaWrite}} \quad \mathbf{6.3}$$

where the empirical time it takes for a sensor to enter Kafka, $T_{SensorEnterKafka}$, is the time it takes for an environmental sensor to transmit data to Kafka and the time it takes for Kafka to route that sensor value to its respective topic

$$\mathbf{T_{SensorExitKafka} = T_{ConsumerRead}} \quad \mathbf{6.4}$$

where the time it takes for a sensor value to exit Kafka, $T_{SensorExitKafka,}$ is the time it takes for the consumer to pull the sensor value from the Kafka topic,

$$\mathbf{T_{TransactionService} = T_{SensorPreprocessing} + T_{IgniteWrite} + T_{TransactionPrediction}} \quad \mathbf{6.5}$$

where the time it takes to service a transaction, $T_{TransactionService}$, is the time it takes for sensor values to be aggregated to their respective transaction tuple, the time it takes to write that tuple to Ignite, and the time it takes for the model to pull that transaction and perform a prediction on it

$$\mathbf{T_{SensorProcessing} = T_{ConsumerRead} + T_{Aggregation/Segregation}} \quad \mathbf{6.6}$$

where the time it takes to process a sensor value, $T_{SensorProcessing}$, is the time it takes to read a value from Kafka and aggregate that value into its proper transaction tuple

$$\mathbf{T_{Prediction} = T_{IgniteRead} + T_{MLComputation}} \quad \mathbf{6.7}$$

where the time it takes to make a prediction on a transaction, $T_{Prediction}$, is the time it takes for the model to pull the transaction tuple from Ignite and make a prediction on the transaction

Recall that the response time for a transaction, according to queueing theory, is its total time spent in the system, including its queue residency time and time to service. For our experimentation, a single log entry follows the given schema:

[real world time, sensor ID, sensor reading, entry timestamp into topic, exit timestamp from topic]. Adding sample values, it looks as follows:

[12:45:00, CP-103-AT,72.5, 1491693393, 1491693406].

For gathering the values needed to analytically calculate specific values as per queueing theory and for generating the plots and remember that in order for a transaction to be completed, all four

sensor values for a room at a specific timestamp must arrive, we infer pertinent information from the entry and exit times as follows:

Eq. 6.2 translates into,

$$\mathbf{T_{Wait}} =$$

Exit Timestamp from Topic MAX (AT, AV, CO2, RT) – Entry Timestamp to Topic MIN (AT, AV, CO2, RT) **6.8**

where log generated wait time, T_{wait}, is the difference in the time that it takes the last sensor value for a room at a specific timestamp and to be written to Ignite and the time it takes the first sensor value to arrive,

Eq. 7.5 translates into,

$$\mathbf{T_{TransactionService}} =$$

Exit Timestamp from Topic MAX (AT, AV, CO2, RT) - Exit Timestamp from Topic MIN (AT, AV, CO2, RT) **6.9**

where the log generated transaction service time, $T_{TransactionService}$, is the difference in the time it takes for the last sensor value to be written to Ignite and the first sensor value to be written to Ignite,

Eq. 6.1 translates into,

$$\mathbf{T_{Response} = Eq.\ 6.8 + Eq.\ 6.9} \quad \mathbf{6.10}$$

To gain insight on how the system works under various utilization rates, we need a way to vary the arrival rate since utilization rate is a function of arrival rate and service rate and the service rate is deterministic. There are two approaches to varying the arrival rate. One approach is to explicitly set the arrival rate manually. However, this is not an accurate representation of how a real world IoT system functions. For computations to be serviced in real time, sensors will transmit

data as soon as values are received. This means that there may be time periods where data arrives at a high rates and times where data arrives less frequently over periods of time. However, the system doesn't know at what frequencies the data is arriving, it just knows to ingest it into the system. We have decided that a more accurate representation of varying the arrival rates is to add delays to the time data is pushed to Kafka. By doing this, the system will produce an arrival rate based on the delay and the system utilization. we are still allowing the system to push data as data arrives, but can simulate periods where data arrives more frequently and periods where data arrives less frequently over periods of time. We have added the following delays, measured in seconds: 0, 0.05, 0.10, 0.15, 0.20, and 1, where 0 seconds generates arrival rates as fast as the server can push and 1 second generating arrival rates at very slow frequencies.

Since the arrival rate is controlled by the system utilization, it was not possible to take the average of five runs because the arrival rates were not consistent. However, running 5 simulations for each delay gave us five different utilizations rates that we were able to plot and see the trend. Essentially, instead of this experiment generating 5 averaged points, this experiment generated 30 different points per topology. This provided more points to plot within the min and max arrival rate that was generated that we were then able to plot and compare against the analytical plots.

Chapter 7 Results

A fundamental concept in queueing theory is Little's Law, which provides a way to determine the long term average numbers of jobs in a system. When dealing with real-time computations, the goal is to limit the additional time that data arriving has to wait before being serviced. In a completely ideal scenario, jobs would be serviced as soon as they arrive and no queuing would happen, thus ensuring absolute real-time results. This would infer that the response time of jobs in a system would not be effected by any waiting time. With the data provided by facilities, we have determined that there are 6912 sensor values that result in 1729 transactions over a 24-hour period. This breaks down into 4 sensor values for a room with 6 rooms per 5-minute interval at 288-5 minute intervals, starting from 00:00:00 to 23:55:00. If all sensor values arrive in order, they would appear in increasing order, starting at time 00:00:00, with all four sensor values for a room arriving simultaneously.

We ran a total of five experiments. Considering similar behaviors over run to run variances, the following results represent the raw data obtained from our experiment over a single run. The tables 7.1-7.4 show for all topologies, the complete set of points for all five runs, which we will plot and show in section 7.2. For each set of tables, you will see the individual measurements for the average wait time, average service time, average response time, and arrival rate in the first five rows. The sixth row shows the average wait time, average service time, average response time, and average arrival rate of the five runs. The seventh row shows the standard deviations for the wait times, service times, response times, and arrival rates over the five runs. Although we did not use these directly for our results, it gives us insight on how the delays effected the system utilization, which we will consider for alternative implementations for future work.

Table 7.1 All Results for 5 Runs for Buildings Topology

Building Delay 000				
	Average Wait Time	Average Service Time	Average Response Time	Arrival Rate
	4.327453702	4.319496525	8.646950227	224.25
	3.390949	0.778663	4.169612	128.5
	3.342471	2.7814	6.123871	146
	4.235608	3.376921	7.612529	296.5
	1.890897	1.429034	3.319931	209.75
Averag	3.43747574	2.537102905	5.974578645	201
StDev	0.875462428	1.285443901	2.00757556	60.03103364

Buildings Delay 015				
	Average Response Time	Average Service Time	Average Response Time	Arrival Rate
	8.648351	4.27544	12.92379	6.5
	5.614317	5.539126	11.15344	12.5
	1.747245	1.675498	3.422743	30
	4.337118	4.137049	8.474167	6.25
	2.941609	2.802928	5.744537	3.75
Averag	4.657728	3.6860082	8.3437354	11.8
StDev	2.381549955	1.327169555	3.460067947	9.543846185

Fixed Buildings Average
7.159157023

Buildings Delay 005				
	Average Wait Time	Average Service Time	Average Response Time	Arrival Rate
	4.913218	4.79224	9.705457171	13.75
	0.102946	0.036088	0.139034	25.5
	1.571458	1.502228	3.073686	51.25
	1.404479	1.334479	2.738958	55.5
	4.784653	4.476036	9.260689	11.25
Averag	2.5553508	2.4282142	4.983564834	31.45
StDev	1.94093618	1.873935271	3.81432653	18.58588174

Buildings Delay 020				
	Average Wait Time	Average Service Rate	Average Response Time	Arrival Rate
	5.354491	5.279236	10.63373	6.25
	4.687344	4.477176	9.16452	50
	1.451823	1.378084	2.829907	25.5
	4.944427	4.731852	9.676279	9.25
	4.344867	4.103854	8.448721	21
Averag	4.1565904	3.9940404	8.1506314	22.4
StDev	1.392017709	1.362703241	2.753917751	15.5375352

Buildings Delay 010				
	Average Wait Time	Average Service Time	Average Response Time	Arrival Rate
	4.861597222	4.787269	9.648866	18.75
	4.421933	4.135249	8.557182	16.5
	5.967703	5.625897	11.5936	16.5
	6.079317	5.755434	11.83475	22.25
	3.361331	3.162691	6.524022	5.25
Averag	4.938376244	4.693308	9.631684	15.85
StDev	1.012021712	0.965492703	1.975602544	5.702192561

Buildings Delay 100				
	Average Wait Time	Average Service Time	Average Response Time	Arrival Rate
	7.122043	7.0439	14.16594	3.75
	4.151599	4.086959	8.238558	6
	6.721701	6.593137	13.31484	2
	4.243096	4.176383	8.419479	6
	7.143646	7.072593	14.21624	3
Averag	5.876417	5.7945944	11.6710114	4.15
StDev	1.379470415	1.368673955	2.748062194	1.609347694

Table 7.2 All Results for 5 Runs for Rooms Topology

Floors Delay 000				
	Average Wait Time	Average Service Time	Average Response Rate	Arrival Rate
	3.583544	2.783387576	6.366932034	346.25
	3.542917	2.745943	6.28886	346
	3.93160301	3.309693271	7.241296281	246.25
	2.513351	1.208223	3.721574	145.5
	4.209497	0.916227	5.125723	214.5
Average	3.556182402	2.192694769	5.748877063	259.7
StDev	0.575531504	0.948814195	1.21648406	77.72280875

Floors Delay 015				
	Average Wait Time	Average Service Time	Average Response Time	Arrival Rate
	4.983669	4.877639	9.861308	12.5
	4.350359	4.140891	8.49125	4.75
	4.958802082	4.785787035	9.744589117	8.75
	5.600694	5.427153	11.02785	11.25
	4.493681	4.416562	8.910243	18.75
Average	4.877441016	4.729606407	9.607048023	11.2
StDev	0.439564805	0.437315399	0.875532571	4.610856753

Fixed Floors Average
4.329480991

Floors Delay 005				
	Average Wait Time	Average Service Time	Average Response Time	Arrival Rate
	1.416718755	1.26890625	2.685625	50.25
	3.424693	3.35213	6.776823	45.75
	4.842175923	4.66606481	9.508240733	17.75
	3.323611	3.090347	6.413958	9
	4.919051	4.845833	9.764884	40.25
Average	3.585249936	3.444656212	7.029906147	32.6
StDev	1.277156933	1.290203036	2.566650271	16.25084613

Floors Delay 020				
	Average Wait Time	Average Service Time	Average Response Time	Arrival Rate
	5.390521	5.296267	10.68679	11
	3.174994	3.095961	6.270955	2.25
	1.927048611	1.821643519	3.74869213	2.25
	5.569363	5.395058	10.96442	13.25
	5.851661	5.688235	11.5399	10
Average	4.382717522	4.259432904	8.642151426	7.75
StDev	1.553737184	1.529877687	3.083470168	4.612483062

Floors Delay 010				
	Average Wait Time	Average Service Time	Average Response Time	Arrival Rate
	5.906968	5.809583	11.71655	14.25
	3.242066	3.047425	6.289491	5
	3.353368056	3.199710646	6.553078703	7
	4.389219	4.116128	8.505347	19.25
	4.173154	3.929034	8.102188	9.75
Average	4.212955011	4.020376129	8.233330941	11.05
StDev	0.957480895	0.983757975	1.940400346	5.141497836

Floors Delay 100				
	Average Wait Time	Average Service Time	Average Response Time	Arrival Rate
	7.787506	7.705459	15.49297	1.75
	6.38713	6.276458	12.66359	1
	6.531406249	6.433026618	12.96443287	2
	7.707737	7.585619	15.29336	2.5
	8.75169	8.650035	17.40172	2.5
Average	7.43309385	7.330119524	14.76321457	1.95
StDev	0.877133233	0.879012498	1.75609493	0.556776436

Table 7.3 All Results for 5 Runs for Rooms Topology

	Rooms Delay 000					Rooms Delay 015				Fixed Rooms Average
	Average Wait Time	Average Service Time	Average Response Time	Arrival Rate		Average Wait Time	Average Service Time	Average Response Time	Arrival Rate	3.945884763
	3.469403936	3.015705997	6.485109933	291		5.736574074	5.623501161	11.36007523	16.25	
	3.357991897	1.437575256	4.795567153	156.75		2.913125	2.763101852	5.676226852	3.5	
	2.54651	1.267668	3.814178	116.25		3.796719	3.647135	7.443854	5	
	3.68099	1.709792	5.390781	631.25		4.277656	4.091128	8.368785	6.25	
	3.346811	1.470874	4.817685	118		2.090932	1.971059	4.061991	1.25	
Average	3.280341367	1.780323051	5.060664217	262.65	Average	3.763001215	3.619185003	7.382186416	6.45	
StDev	0.386062242	0.633592238	0.874192877	195.0595166	StDev	1.239282156	1.240160662	2.479305334	5.1754227	

	Rooms Delay 005					Rooms Delay 020			
	Average Wait Time	Average Service Time	Average Response Time	Arrival Rate		Average Wait Time	Average Service Time	Average Response Time	Arrival Rate
	4.65429398	4.537274309	9.191568289	13.25		5.736574074	5.623501161	11.36007523	16.25
	5.208761571	4.944901625	10.1536632	41.5		4.765208	4.573119	9.338328	5
	1.527124	1.447078	2.974201	53.75		2.055417	1.940231	3.995648	4
	3.593293	3.33989	6.933183	9.75		4.918565	4.778044	9.696609	9
	3.223472	3.090127	6.3136	7		4.256725	4.008999	8.265723	17.75
Average	3.64138891	3.471854187	7.113243098	25.05	Average	4.346497815	4.184778832	8.531276646	10.4
StDev	1.275124904	1.230271319	2.504680028	18.93898097	StDev	1.240322664	1.236251358	2.476040338	5.6625966

	Rooms Delay 010					Rooms Delay 100			
	Average Wait Time	Average Service Time	Average Response Time	Arrival Rate		Average Wait Time	Average Service Time	Average Response Time	Arrival Rate
	5.736574074	5.623501161	11.36007523	16.25		4.745921594	4.664540668	9.410462261	5.5
	3.374340278	3.212997686	6.587337964	5		7.231094	7.122037	14.35313	2.25
	5.799068	5.715064	11.51413	22		5.537124	5.43941	10.97653	5.25
	3.387216	3.184294	6.57151	8.5		8.278027	8.156169	16.4342	2.25
	4.21526042	3.974583	8.189844	10.5		6.095828	6.003241	12.09907	1
Average	4.502491754	4.342087969	8.844579439	12.45	Average	6.377598919	6.277079534	12.65467845	3.25
StDev	1.07732416	1.120488696	2.197493186	6.009159675	StDev	1.248058058	1.234657015	2.482713937	1.7958285

Table 7.4 All Results for 5 Runs for Sensors Topology

Sensor Delay 000				
	Average Service Time	Average Wait Time	Average Response Time	Arrival Rate
	3.424578	3.433397	6.857975	323.25
	4.999965	4.987344	9.987309	275.5
	3.609091	3.4286	7.037691	333
	2.569606	2.117263	4.686869	120.75
	3.415914	3.243958	6.659873	241.75
Average	3.6038308	3.4421124	7.0459434	258.85
StDev	0.785656872	0.914602683	1.697133713	76.53342407

Sensor Delay 015				
	Average Wait Time	Average Service Time	Average Response Time	Arrival Rate
	4.736453	4.630712	9.367164	13.75
	1.534566	1.437187	2.971753	23.25
	5.825828	5.734091	11.55992	15
	3.47537	3.37261	6.84798	2.5
	1.573681	1.484375	3.058056	30
Average	3.4291796	3.331795	6.7609746	16.9
StDev	1.702201903	1.700727929	3.402924524	9.304031384

Fixed Sensors Average
3.975600267

Sensor Delay 005				
	Average Wait Time	Average Service Time	Average Response Time	Arrival Rate
	3.065972	2.960868	6.02684	4
	3.030041	2.914525	5.944566	19.25
	4.183576	4.091696	8.275272	8.75
	4.712245	4.594965	9.307211	14.5
	5.322726	5.232309	10.55503	14
Average	4.062912	3.9588726	8.0217838	12.1
StDev	0.903773902	0.908908241	1.812652764	5.240706059

Sensors Delay 020				
	Average Wait Time	Average Service Time	Average Response Time	Arrival Rate
	5.340185	5.441076	10.78126	14
	1.295145	1.198241	2.493385	26
	5.081481	4.979711	10.06119	6.25
	1.411476	1.316181	2.727656	24
	3.989612	3.89478	7.884392	20.75
Average	3.4235798	3.3659978	6.7895766	18.2
StDev	1.750497114	1.793909078	3.543785337	7.230836743

Sensor Delay 010				
	Average Wait Time	Average Service Time	Average Response Time	Arrival Rate
	3.816331	3.916053	7.732384	11.25
	2.413819	2.323993	4.737812	35.5
	3.063519	2.957801	6.021319	12
	3.046493	2.957297	6.003791	3.25
	5.537813	5.414861	10.95267	13.25
Average	3.575595	3.514001	7.0895952	15.05
StDev	1.077046659	1.077989923	2.153489883	10.81133664

Sensors Delay 100				
	Average Service Time	Average Wait Time	Average Response Time	Arrival Rate
	6.788137	6.863929	13.65207	2.25
	6.921973	6.843212	13.76519	1.25
	6.558397	6.475191	13.03359	2.25
	5.951846	5.869022	11.82087	1.5
	5.229907	5.15276	10.38267	3.75
Average	6.290052	6.2408228	12.530878	2.2
StDev	0.625733582	0.652407699	1.276888871	0.871779789

7.1 Per Transaction Latency

The follow “Per Transaction Latency” graphs show how for each topology, the overall response time changes as more jobs are entering the system. For each figure, the top graph shows the response time change over the duration of all 1729 transactions. The bottom left graph show the response times for the first 60 transactions and the bottom right graph shows the same for the last 60 transactions. This allows a close up of how the response times begin versus how they end. It is important to notice the response time behavior in the beginning and the end. The first 60 transactions show the response times before queueing happens. As shown in the figures below, there are clearly defined high and low point trends for best case response times and worst case response times. Because in the best case, sensor values for a transaction arrive in order, the low points in the case of the first 60 transactions reflect the response times of sensor values arriving in order, without queueing. Respectively, the high points reflect sensor values arriving with a high degree of out of orderedness, without queueing. For the last 60 transactions, the best case and worst case response times have increased, with many points in between. This still reflects best and worst case response times. However, this shows an amortized time that the transactions converge to when queueing is introduced. The low points represent the best case response time when sensor values arrive in order, when there is queueing. The high points represent the worst case response time when the sensor values arrive with the highest degree of out of orderedness, when there is queueing. The points in between represent the response time in respect to the degree of out of orderedness in which the sensor values arrive in.

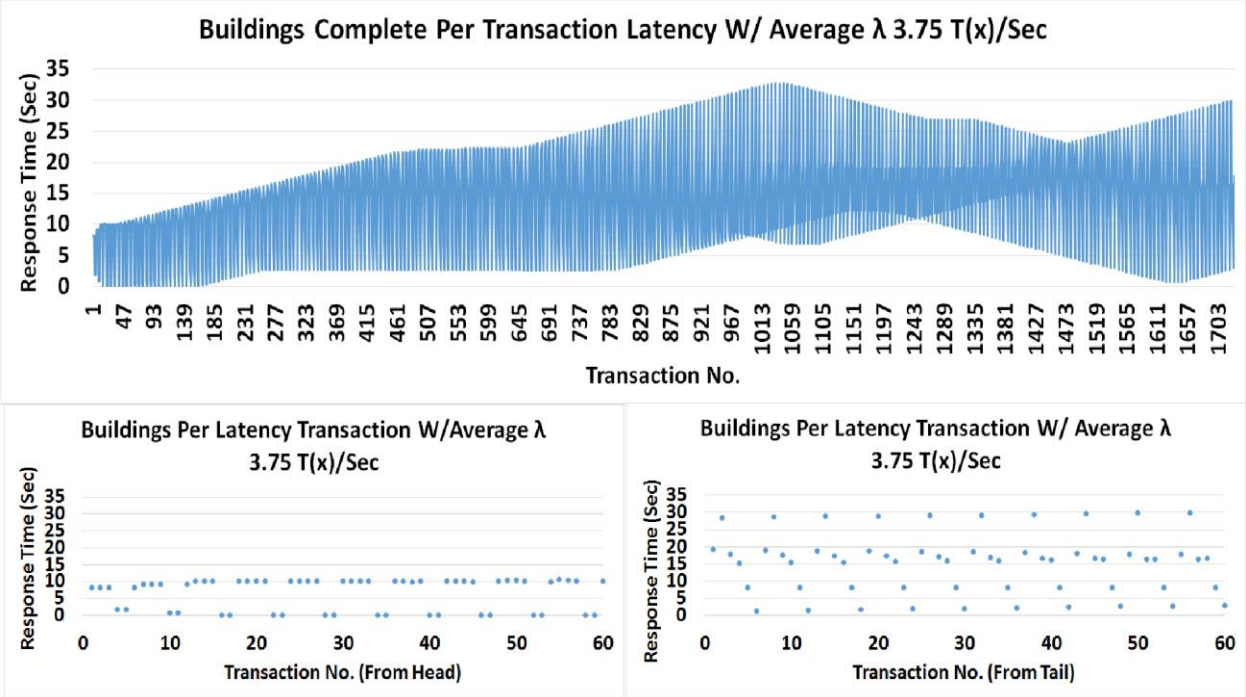

Figure 7.1 Per Transaction Latency Plots for the Buildings Topology W/ Average λ 3.75 T(x)/Sec

Figure 7.1 shows the graph from the per transaction latency for the buildings topology with an average arrival rate of 3.75 transactions per second show the highest response time occurring in comparison to the other four topologies at roughly 1035 transactions. We also notice that the buildings topology has the highest range in the low and high response times, with a difference of 26 seconds. The multiple inflection points, denoting the different variances in response times, show the different degrees of out of orderedness that the sensor values are arriving at during different points of the run.

The initial 60 transactions show consistent best and worst case times, before queueing has begun. You can see that best case times can completely aggregate a transaction tuple in less than a second if all sensor values arrive in order. Worst case, if sensor values arrive in the worst degree of out of orderedness, this can increase the response time over 500 times the amount of the best case. We can see that before queueing begins, there is either a best case time or a worst case time.

When we observe the final 60 transactions, when queueing has taken place, we notice that the degree of out of orderedness varies as queueing is taking place. We can still see that in the best case, response times can take less than a second, assuming all values are in order. However, with queueing, at worst case we can see the response times increase by as much as 1500 times the best case. We also notice that most of the transactions are in the middle, indicating an average degree of out of orderedness that influences an amortized degree of out of orderedness and response time that over time, transactions converge to.

Table 7.5 shows raw sensor data from the buildings topology logs after the experiment was ran. This table contains the sensor values that account for the first 4 transactions that arrived. If you focus on the "Time" column, you can clearly see how according to real world time, the sensor values are mostly all arriving in order, with at most an out of orderedness variance in real world time by 5 minutes and an even amount of values arriving according to room and sensor type.

Table 7.5 Raw Sensor Values corresponding to first 4 T(x)s for the Buildings Topology W/ Average λ 3.75 T(x)/Sec

Buildings Raw Sensor Data Corresponding to first 4 T(x)s W/ Average λ 3.75 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'00:00:00'	'CP'	'1'	'103'	'CO2'	'362.49']	1494276826	1494276828
['9/27/2016'	'00:00:00'	'CP'	'1'	'106'	'AV'	'0.00']	1494276826	1494276828
['9/27/2016'	'00:00:00'	'CP'	'1'	'108'	'AV'	'16.00']	1494276826	1494276828
['9/27/2016'	'00:00:00'	'CP'	'1'	'106'	'AT'	'71.50']	1494276826	1494276828
['9/27/2016'	'00:00:00'	'CP'	'1'	'108'	'RT'	'73.25']	1494276826	1494276828
['9/27/2016'	'00:00:00'	'CP'	'2'	'206C'	'AV'	'12.00']	1494276826	1494276828
['9/27/2016'	'00:00:00'	'CP'	'2'	'206C'	'RT'	'74.00']	1494276826	1494276828
['9/27/2016'	'00:00:00'	'CP'	'2'	'206D'	'AV'	'0.00']	1494276826	1494276828
['9/27/2016'	'00:00:00'	'CP'	'2'	'206C'	'CO2'	'401.92']	1494276826	1494276828
['9/27/2016'	'00:00:00'	'CP'	'2'	'206D'	'AT'	'72.00']	1494276826	1494276828
['9/27/2016'	'00:00:00'	'CP'	'2'	'206H'	'CO2'	'396.90']	1494276826	1494276828
['9/27/2016'	'00:00:00'	'CP'	'2'	'206D'	'CO2'	'399.77']	1494276826	1494276828
['9/27/2016'	'00:00:00'	'CP'	'2'	'206D'	'RT'	'74.00']	1494276826	1494276828
['9/27/2016'	'00:00:00'	'CP'	'2'	'206C'	'AT'	'72.00']	1494276826	1494276828
['9/27/2016'	'00:05:00'	'CP'	'1'	'106'	'AV'	'0.00']	1494276827	1494276828
['9/27/2016'	'00:05:00'	'CP'	'1'	'103'	'CO2'	'362.49']	1494276827	1494276828

So in the early stages of sensor values arriving, aggregation is minimally impacted by out of orderedness, which reflects the plots for the first 60 transactions by the low response times for both the best case and worst case times response times.

Table 7.6 accounts for the sensor values that arrived during transactions 1350 to 1353, which showed response times after the highest peak in response times were achieved for the buildings topology with an arrival rate of 3.75 transactions per second. In the "Time" column, the real world times that the values were pushed to Kafka are out of order. Sensor values are arriving with an out of orderedness variance of 50 minutes, with an even amount of values arriving according to room and sensor type. Towards the final stages of sensor values arriving, out of orderedness is mostly influenced by real world time.

Table 7.6 Raw Sensor Values corresponding to T(x)s 1350-1353 for the Buildings Topology W/ Average λ 3.75 T(x)/Sec

Buildings Raw Sensor Data Corresponding T(x)s 1350 through 1353 W/ Average λ 3.75 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'18:25:00'	'CP'	'1'	'108'	'AT'	'70.00']	1494277071	1494277071
['9/27/2016'	'18:45:00'	'CP'	'2'	'206C'	'RT'	'73.50']	1494277071	1494277071
['9/27/2016'	'19:00:00'	'CP'	'2'	'206H'	'CO2'	'394.03']	1494277071	1494277071
['9/27/2016'	'18:55:00'	'CP'	'2'	'206D'	'AT'	'71.50']	1494277071	1494277071
['9/27/2016'	'19:00:00'	'CP'	'1'	'106'	'AV'	'432.00']	1494277071	1494277071
['9/27/2016'	'18:25:00'	'CP'	'1'	'106'	'RT'	'72.25']	1494277071	1494277071
['9/27/2016'	'18:15:00'	'CP'	'2'	'206H'	'AV'	'304.00']	1494277071	1494277071
['9/27/2016'	'18:40:00'	'CP'	'2'	'206D'	'RT'	'74.00']	1494277071	1494277071
['9/27/2016'	'19:05:00'	'CP'	'2'	'206C'	'CO2'	'375.39']	1494277071	1494277071
['9/27/2016'	'18:35:00'	'CP'	'2'	'206H'	'AT'	'71.50']	1494277071	1494277071
['9/27/2016'	'18:35:00'	'CP'	'1'	'108'	'CO2'	'393.32']	1494277071	1494277071
['9/27/2016'	'19:05:00'	'CP'	'1'	'103'	'CO2'	'377.55']	1494277071	1494277071
['9/27/2016'	'18:35:00'	'CP'	'2'	'206H'	'RT'	'74.00']	1494277071	1494277071
['9/27/2016'	'19:00:00'	'CP'	'2'	'206D'	'CO2'	'387.58']	1494277071	1494277071
['9/27/2016'	'19:00:00'	'CP'	'1'	'108'	'RT'	'71.75']	1494277071	1494277071
['9/27/2016'	'18:20:00'	'CP'	'1'	'103'	'AT'	'67.50']	1494277071	1494277071

Figure 7.2 shows the graph from the per transaction latency for the buildings topology with an average arrival rate of 6.25 transactions per second shows considerably high response times occurring throughout the entire duration of the 1729 transactions. We notice peak response times

achieved at roughly 715 transactions. The difference in low and high response times range was 14.84 seconds.

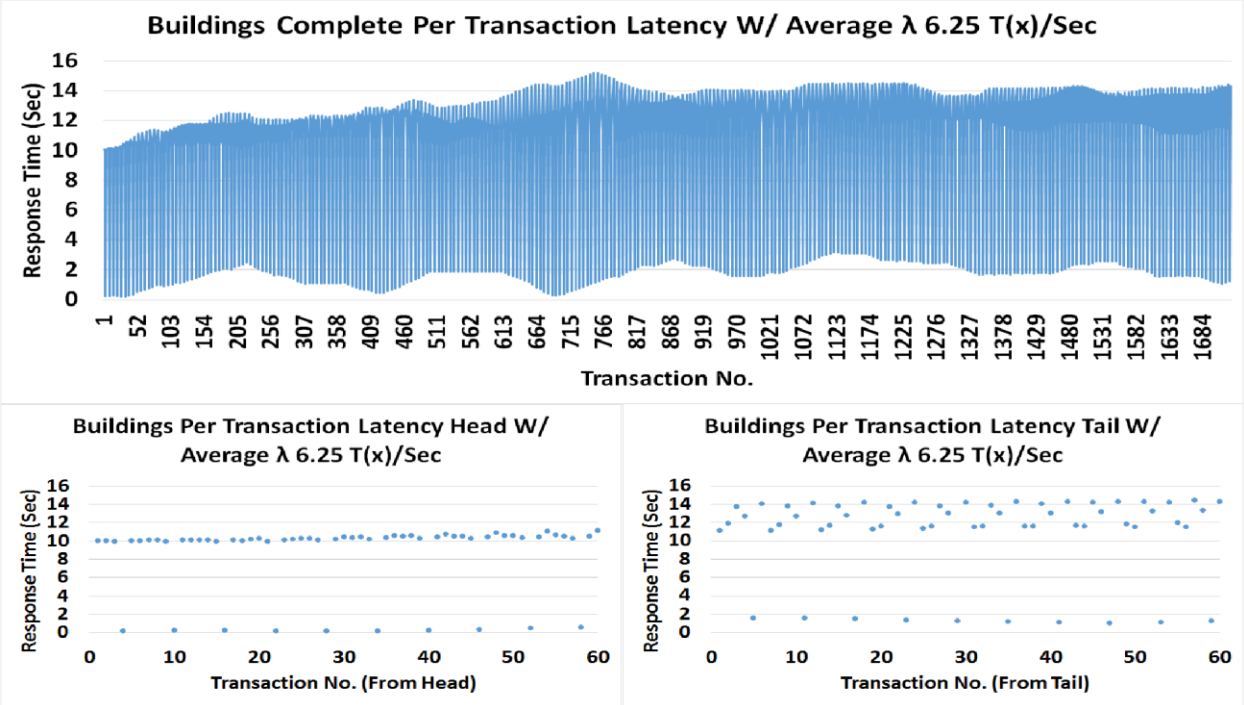

Figure 7.2 Per Transaction Latency Plots for the Buildings Topology W/ Average λ 6.25 T(x)/Sec

From the both initial 60 transactions and the final 60, we see that the response times either hit a high or a low, without many response times in between. That suggests that for this run, values were severely out of order and the degree of out of orderedness were consistently higher than they were low.

Table 7.7 shows raw sensor data from the buildings logs topology with an arrival rate of 6.25 transactions per second after the experiment was ran. This table contains the sensor values that account for the first 4 transactions that arrived. The first 24 sensor values should all have a timestamp of 00:00:00. From the "Time" column, it can be observed that values immediately arrive out of order. We see out of orderedness variances in real world time by 10 minutes right

from the beginning. In the "Room" column, not a single CP-206C value is accounted for and in the "Sensor" column, not a single CO2 value is accounted for.

Table 7.7 Raw Sensor Values corresponding to first 4 T(x)s for the Buildings Topology W/ Average λ 6.25 T(x)/Sec

Buildings Raw Sensor Data Corresponding to first 4 T(x)s W/ Average λ 6.25 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'00:00:00'	'CP'	'1'	'103'	'AT'	'72.50']	1494428148	1494428148
['9/27/2016'	'00:00:00'	'CP'	'1'	'106'	'AV'	'0.00']	1494428148	1494428148
['9/27/2016'	'00:00:00'	'CP'	'2'	'206D'	'AT'	'72.00']	1494428148	1494428148
['9/27/2016'	'00:00:00'	'CP'	'2'	'206H'	'RT'	'74.00']	1494428148	1494428148
['9/27/2016'	'00:00:00'	'CP'	'1'	'108'	'RT'	'73.25']	1494428148	1494428148
['9/27/2016'	'00:00:00'	'CP'	'2'	'206H'	'AT'	'72.00']	1494428148	1494428148
['9/27/2016'	'00:05:00'	'CP'	'1'	'103'	'AT'	'72.50']	1494428148	1494428148
['9/27/2016'	'00:05:00'	'CP'	'1'	'106'	'AV'	'0.00']	1494428148	1494428148
['9/27/2016'	'00:05:00'	'CP'	'2'	'206D'	'AT'	'72.00']	1494428148	1494428148
['9/27/2016'	'00:05:00'	'CP'	'2'	'206H'	'RT'	'74.00']	1494428148	1494428148
['9/27/2016'	'00:05:00'	'CP'	'1'	'108'	'RT'	'73.25']	1494428148	1494428148
['9/27/2016'	'00:05:00'	'CP'	'2'	'206H'	'AT'	'72.00']	1494428148	1494428148
['9/27/2016'	'00:10:00'	'CP'	'1'	'103'	'AT'	'72.50']	1494428148	1494428149
['9/27/2016'	'00:10:00'	'CP'	'1'	'106'	'AV'	'0.00']	1494428148	1494428149
['9/27/2016'	'00:10:00'	'CP'	'2'	'206D'	'AT'	'72.00']	1494428148	1494428149
['9/27/2016'	'00:10:00'	'CP'	'2'	'206H'	'RT'	'74.00']	1494428148	1494428149

This indicates that the aggregator has to experience waiting from the start of the first transactions, lasting throughout the duration of all 1729 transactions indicated by no noticeable decrease in response time. This shows how in the case of this run, out of orderedness is severely impacted by real world time, room, and sensor.

Table 7.8 accounts for the sensor values that arrived during transaction 715 to 718, which was when highest response times were achieved for the buildings topology with an average arrival rate of 6.25 transactions per second. In the "Time" column, the real world time that the values were pushed to Kafka are highly out of order. Sensor values are arriving with an out of orderedness variance of 2 hours, with an even amount of values arriving by room and sensor type. Towards the final stages of sensor values arriving, out of orderedness is highly influenced by real world time.

Table 7.8 Raw Sensor Values corresponding to T(x)s 715-718 for the Buildings Topology W/ Average λ 6.25 T(x)/Sec

Buildings Raw Sensor Data Corresponding T(x)s 715 through 718 W/ Average λ 6.25 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'09:25:00'	'CP'	'2'	'206D'	'AV'	'812.00']	1494428187	1494428187
['9/27/2016'	'11:00:00'	'CP'	'2'	'206D'	'AT'	'71.50']	1494428187	1494428187
['9/27/2016'	'09:15:00'	'CP'	'1'	'106'	'AT'	'67.50']	1494428187	1494428187
['9/27/2016'	'09:30:00'	'CP'	'1'	'103'	'AV'	'496.00']	1494428187	1494428187
['9/27/2016'	'09:45:00'	'CP'	'1'	'103'	'RT'	'72.25']	1494428187	1494428187
['9/27/2016'	'09:20:00'	'CP'	'2'	'206H'	'AV'	'300.00']	1494428187	1494428187
['9/27/2016'	'10:45:00'	'CP'	'2'	'206H'	'AT'	'71.50']	1494428187	1494428187
['9/27/2016'	'09:25:00'	'CP'	'2'	'206H'	'CO2'	'401.92']	1494428187	1494428187
['9/27/2016'	'09:40:00'	'CP'	'1'	'106'	'RT'	'71.75']	1494428187	1494428187
['9/27/2016'	'11:15:00'	'CP'	'2'	'206H'	'RT'	'74.00']	1494428187	1494428187
['9/27/2016'	'09:40:00'	'CP'	'2'	'206C'	'AV'	'468.00']	1494428187	1494428187
['9/27/2016'	'09:40:00'	'CP'	'1'	'106'	'CO2'	'373.96']	1494428187	1494428187
['9/27/2016'	'09:40:00'	'CP'	'2'	'206C'	'AT'	'71.00']	1494428187	1494428187
['9/27/2016'	'09:40:00'	'CP'	'1'	'108'	'AT'	'71.00']	1494428187	1494428187
['9/27/2016'	'09:35:00'	'CP'	'2'	'206D'	'CO2'	'403.36']	1494428187	1494428187
['9/27/2016'	'09:40:00'	'CP'	'2'	'206C'	'CO2'	'392.60']	1494428187	1494428187

In the "Sensor" column, we do see one missing AV value and one extra AT value. This indicates the sensor granularity of aggregation is also influencing out of orderedness, but at a very minimal impact.

Figure 7.3 shows the graph from the per transaction latency for the buildings topology with an average arrival rate of 6.5 transactions per second shows initially lower response times with a rapid increase occurring throughout the entire duration of the 1729 transactions. From the plots for the first 60 transactions, we can see that the best case response times are consistent. However, the worst case response times being to increase as early as the first 60 transactions, indicating that the degree of out of orderedness immediately begins to worsen. In the plots for the final 60 transactions, we can see that the worst case response time stays relatively the same. However, after the high peak in response time at the 1129 transaction mark, the degree of out of orderedness begins to amortize and the transactions begin to converge to a specific response time, where we

can see in the complete plots that towards the final transactions, the high response times begin to plateau and remain consistent.

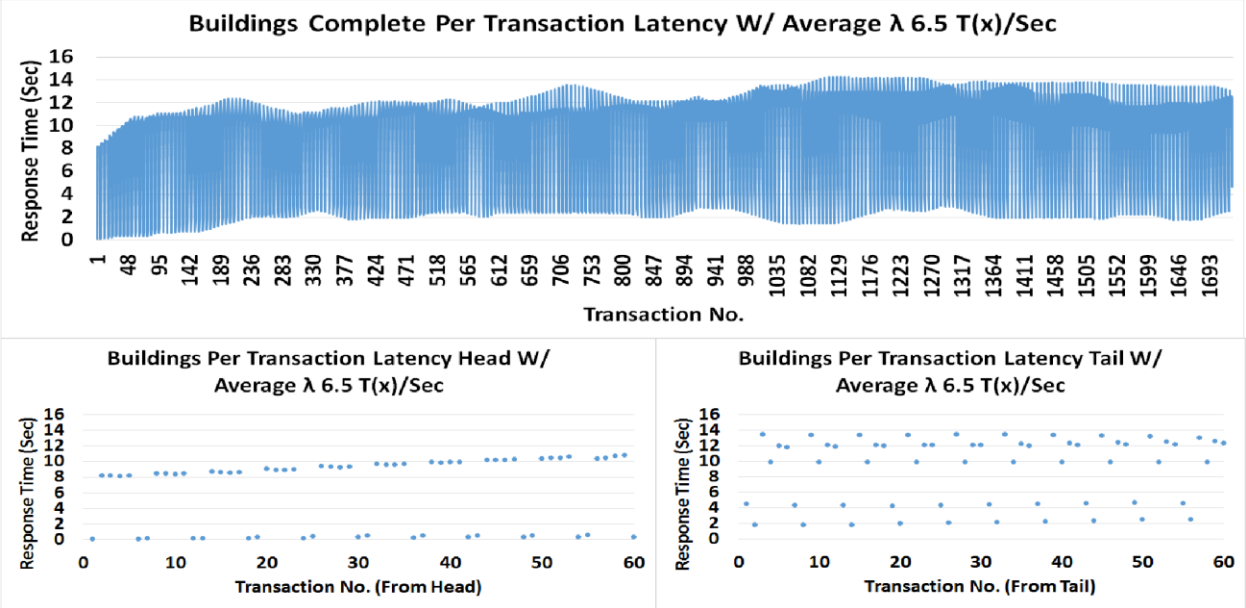

Figure 7.3 Per Transaction Latency Plots for the Buildings Topology W/ Average λ 6.5 T(x)/Sec

From the logs provided in Table 7.9, we can immediately begin to see out of orderedness initiating before the real world time values for the first set of transactions at time 00:00:00 even arrive. If you focus on the "Time" column, you can clearly see how according to real world time, the sensor values are mostly all arriving out of order, with at most an out of orderedness variance in real world time by 15 minutes. If you focus on the "Room" column, it can be seen that not a single CP-103 value has arrived. If you focus on the "Sensor" column, it can be seen that not a single RT value has arrived. This indicates that the aggregator has to experience waiting from the start of the first transactions, lasting throughout the duration of all 1729 transactions indicated by no noticeable decrease in response time. So in the early stages of sensor values arriving, aggregation is impacted by out of orderedness, being influenced by room, sensor, and by real world time.

Table 7.9 Raw Sensor Values corresponding to first 4 T(x)s for the Buildings Topology W/ Average λ 6.5 T(x)/Sec

Buildings Raw Sensor Data Corresponding to first 4 T(x)s W/ Average λ 6.5 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'00:00:00'	'CP'	'1'	'106'	'AT'	'71.50']	1494427937	1494427939
['9/27/2016'	'00:00:00'	'CP'	'1'	'108'	'CO2'	'389.73']	1494427937	1494427939
['9/27/2016'	'00:00:00'	'CP'	'2'	'206C'	'AV'	'12.00']	1494427937	1494427939
['9/27/2016'	'00:00:00'	'CP'	'2'	'206D'	'AT'	'72.00']	1494427937	1494427939
['9/27/2016'	'00:00:00'	'CP'	'2'	'206C'	'CO2'	'401.92']	1494427937	1494427939
['9/27/2016'	'00:05:00'	'CP'	'1'	'106'	'AT'	'71.50']	1494427937	1494427939
['9/27/2016'	'00:05:00'	'CP'	'1'	'108'	'CO2'	'391.17']	1494427937	1494427939
['9/27/2016'	'00:05:00'	'CP'	'2'	'206C'	'AV'	'8.00']	1494427937	1494427939
['9/27/2016'	'00:05:00'	'CP'	'2'	'206D'	'AT'	'72.00']	1494427937	1494427939
['9/27/2016'	'00:05:00'	'CP'	'2'	'206C'	'CO2'	'395.47']	1494427937	1494427939
['9/27/2016'	'00:10:00'	'CP'	'1'	'106'	'AT'	'71.50']	1494427937	1494427939
['9/27/2016'	'00:10:00'	'CP'	'1'	'108'	'CO2'	'391.17']	1494427937	1494427939
['9/27/2016'	'00:10:00'	'CP'	'2'	'206C'	'AV'	'12.00']	1494427937	1494427939
['9/27/2016'	'00:10:00'	'CP'	'2'	'206D'	'AT'	'72.00']	1494427937	1494427939
['9/27/2016'	'00:10:00'	'CP'	'2'	'206C'	'CO2'	'388.30']	1494427937	1494427939
['9/27/2016'	'00:15:00'	'CP'	'1'	'106'	'AT'	'71.50']	1494427937	1494427939

Table 7.10 accounts for the sensor values that arrived during transactions 1129 to 1132, which was when highest response times were achieved for the buildings topology with an average arrival rate of 6.5 transactions per second.

Table 7.10 Raw Sensor Values corresponding to T(x)s 1129-1132 for the Buildings Topology W/ Average λ 6.5 T(x)/Sec

Buildings Raw Sensor Data Corresponding T(x)s 1129 through 1132 W/ Average λ 6.5 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'15:20:00'	'CP'	'1'	'106'	'RT'	'71.25']	1494427987	1494427987
['9/27/2016'	'17:10:00'	'CP'	'2'	'206C'	'CO2'	'377.55']	1494427987	1494427987
['9/27/2016'	'15:30:00'	'CP'	'2'	'206H'	'RT'	'74.00']	1494427987	1494427987
['9/27/2016'	'14:50:00'	'CP'	'2'	'206H'	'AT'	'71.50']	1494427987	1494427987
['9/27/2016'	'17:15:00'	'CP'	'2'	'206D'	'AT'	'71.50']	1494427987	1494427987
['9/27/2016'	'16:55:00'	'CP'	'2'	'206C'	'AV'	'456.00']	1494427987	1494427987
['9/27/2016'	'15:25:00'	'CP'	'1'	'103'	'AV'	'456.00']	1494427987	1494427987
['9/27/2016'	'15:45:00'	'CP'	'1'	'106'	'AV'	'420.00']	1494427987	1494427987
['9/27/2016'	'15:05:00'	'CP'	'1'	'108'	'AT'	'70.50']	1494427987	1494427987
['9/27/2016'	'15:00:00'	'CP'	'2'	'206H'	'CO2'	'403.36']	1494427987	1494427987
['9/27/2016'	'15:15:00'	'CP'	'1'	'103'	'RT'	'72.00']	1494427987	1494427987
['9/27/2016'	'15:10:00'	'CP'	'2'	'206D'	'AV'	'392.00']	1494427987	1494427987
['9/27/2016'	'15:35:00'	'CP'	'2'	'206H'	'AV'	'308.00']	1494427987	1494427987
['9/27/2016'	'17:10:00'	'CP'	'1'	'106'	'AT'	'70.00']	1494427987	1494427987
['9/27/2016'	'15:00:00'	'CP'	'1'	'108'	'RT'	'72.25']	1494427987	1494427987
['9/27/2016'	'15:20:00'	'CP'	'1'	'103'	'AT'	'68.50']	1494427987	1494427987

In the “Time” column, the real world time that the values were pushed to Kafka are highly out of

order. Sensor values are arriving with an out of orderedness variance of 2 hours, with an even amount of sensor values arriving by room. However, AT and AV sensor values are dominating in comparison to RT and CO2 sensor values. Towards the final stages of sensor values arriving, out of orderedness is highly influenced by both sensor type and real world time.

Figure 7.4 shows the graph from the per transaction latency for the buildings topology with an average arrival rate of 13.75 transactions per second shows initially lower response times with a rapid increase occurring throughout the entire duration of the 1729 transactions.

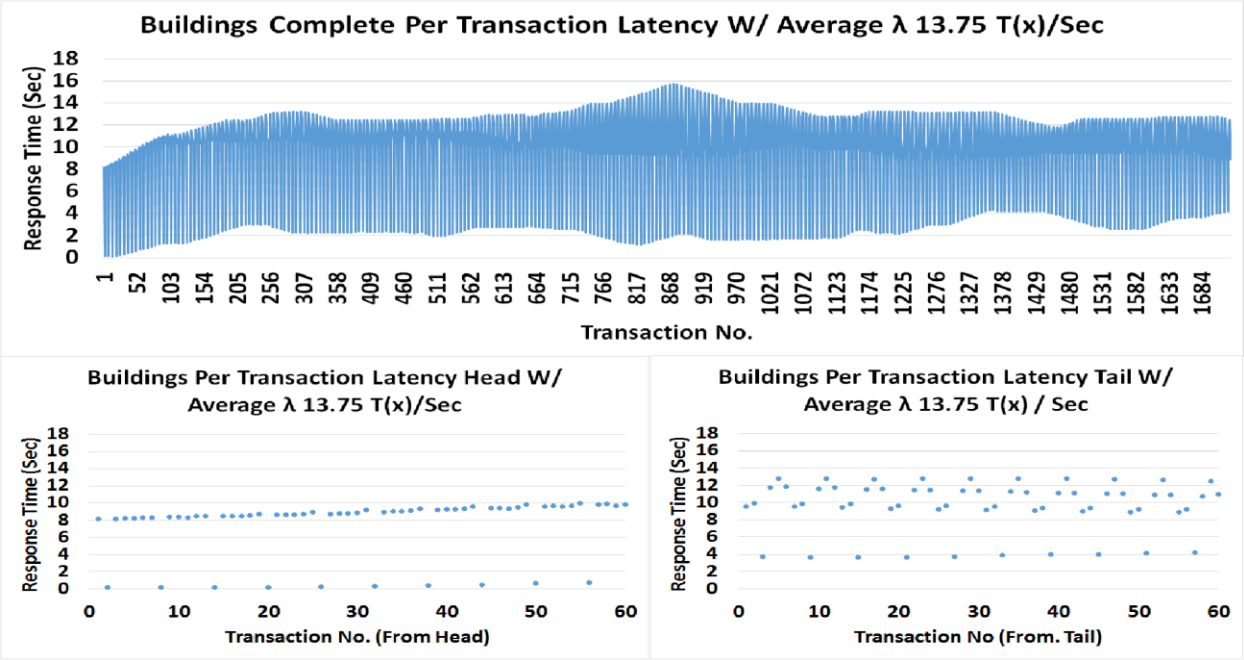

Figure 7.4 Per Transaction Latency Plots for the Buildings Topology W/ Average λ 13.75 T(x)/Sec

We notice peak response times achieved at roughly 868 transactions. The difference in low and high response times range was 14 seconds.

Table 7.11 shows raw sensor data from the buildings logs topology with an arrival rate of 13.75 transactions per second after the experiment was ran. This table contains the sensor values

that account for the first 4 transactions that arrived. The first 24 sensor values should all have a real world time of 00:00:00.

Table 7.11 Raw Sensor Values corresponding to first 4 T(x)s for the Buildings Topology W/ Average λ 13.75 T(x)/Sec

Buildings Raw Sensor Data Corresponding to first 4 T(x)s W/ Average λ 13.75 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'00:00:00'	'CP'	'2'	'206H'	'CO2'	'396.90']	1494427427	1494427429
['9/27/2016'	'00:00:00'	'CP'	'1'	'103'	'AT'	'72.50']	1494427427	1494427429
['9/27/2016'	'00:00:00'	'CP'	'1'	'108'	'RT'	'73.25']	1494427427	1494427429
['9/27/2016'	'00:00:00'	'CP'	'2'	'206C'	'AT'	'72.00']	1494427427	1494427429
['9/27/2016'	'00:05:00'	'CP'	'2'	'206H'	'CO2'	'397.62']	1494427427	1494427429
['9/27/2016'	'00:00:00'	'CP'	'2'	'206D'	'RT'	'74.00']	1494427427	1494427429
['9/27/2016'	'00:00:00'	'CP'	'2'	'206D'	'AV'	'0.00']	1494427427	1494427429
['9/27/2016'	'00:05:00'	'CP'	'1'	'103'	'AT'	'72.50']	1494427427	1494427429
['9/27/2016'	'00:05:00'	'CP'	'1'	'108'	'RT'	'73.25']	1494427427	1494427429
['9/27/2016'	'00:05:00'	'CP'	'2'	'206C'	'AT'	'72.00']	1494427427	1494427429
['9/27/2016'	'00:05:00'	'CP'	'2'	'206D'	'RT'	'74.00']	1494427427	1494427429
['9/27/2016'	'00:05:00'	'CP'	'2'	'206D'	'AV'	'0.00']	1494427427	1494427429
['9/27/2016'	'00:10:00'	'CP'	'2'	'206H'	'CO2'	'396.90']	1494427427	1494427429
['9/27/2016'	'00:10:00'	'CP'	'1'	'103'	'AT'	'72.50']	1494427427	1494427429
['9/27/2016'	'00:10:00'	'CP'	'1'	'108'	'RT'	'73.25']	1494427427	1494427429
['9/27/2016'	'00:10:00'	'CP'	'2'	'206C'	'AT'	'72.00']	1494427427	1494427429

From the log we can immediately see out of orderedness initiating before the real world time values for the first set in transactions at time 00:00:00 even arrive. If you focus on the "Time" column, you can clearly see how according to real world time, the sensor values are mostly all arriving out of order, with at most an out of orderedness variance in real world time by 10 minutes. If you focus on the "Room" column, it can be seen that not a single CP-106 value has arrived. If you focus on the "Sensor" column, it can be seen that there are 2 missing AV values. This indicates that the aggregator has to experience initial waiting from the start of the first transactions, with an immediate increase throughout the duration of all 1729 transactions indicated by several increasingly large peaks in response time. So in the early stages of sensor values arriving, aggregation is impacted by out of orderedness, being influenced by room, sensor type, and by real world time.

Table 7.12 accounts for the sensor values that arrived during transaction 715 to 718, which was when higher response times were achieved for the buildings topology with an average arrival rate of 13.75 transactions per second.

Table 7.12 Raw Sensor Values corresponding to T(x)s 715-718 for the Buildings Topology W/ Average λ 13.75 T(x)/Sec

Buildings Raw Sensor Data Corresponding T(x)s 715 through 718 W/ Average λ 13.75 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'11:50:00'	'CP'	'2'	'206D'	'RT'	'74.00']	1494427448	1494427448
['9/27/2016'	'12:40:00'	'CP'	'1'	'108'	'RT'	'72.25']	1494427448	1494427448
['9/27/2016'	'09:30:00'	'CP'	'2'	'206C'	'AV'	'456.00']	1494427448	1494427448
['9/27/2016'	'09:45:00'	'CP'	'2'	'206C'	'RT'	'73.75']	1494427448	1494427448
['9/27/2016'	'08:40:00'	'CP'	'2'	'206C'	'CO2'	'389.73']	1494427448	1494427448
['9/27/2016'	'08:50:00'	'CP'	'2'	'206H'	'AV'	'296.00']	1494427448	1494427448
['9/27/2016'	'08:55:00'	'CP'	'1'	'108'	'AT'	'71.00']	1494427448	1494427448
['9/27/2016'	'08:55:00'	'CP'	'1'	'103'	'CO2'	'380.41']	1494427448	1494427448
['9/27/2016'	'09:05:00'	'CP'	'2'	'206H'	'RT'	'74.00']	1494427448	1494427448
['9/27/2016'	'10:00:00'	'CP'	'2'	'206D'	'AT'	'71.00']	1494427448	1494427448
['9/27/2016'	'09:20:00'	'CP'	'1'	'106'	'CO2'	'371.81']	1494427448	1494427448
['9/27/2016'	'09:15:00'	'CP'	'2'	'206D'	'CO2'	'404.07']	1494427448	1494427448
['9/27/2016'	'08:40:00'	'CP'	'1'	'106'	'AT'	'68.50']	1494427448	1494427448
['9/27/2016'	'09:10:00'	'CP'	'1'	'106'	'RT'	'72.00']	1494427448	1494427448
['9/27/2016'	'09:55:00'	'CP'	'1'	'108'	'AV'	'440.00']	1494427448	1494427448
['9/27/2016'	'08:55:00'	'CP'	'1'	'103'	'AV'	'492.00']	1494427448	1494427448

In the "Time" column, the real world time that the values were pushed to Kafka are highly out of order. Sensor values are arriving with an out of orderedness variance of 4 hours, with a minimal amount of missing values from the room and sensor granularity. Towards the final stages of sensor values arriving, out of orderedness is highly influenced by real world time.

Figure 7.5 shows the graph from the per transaction latency for the buildings topology with an average arrival rate of 18.75 transactions per second shows initially lower response times with a rapid increase occurring throughout the entire duration of the 1729 transactions.

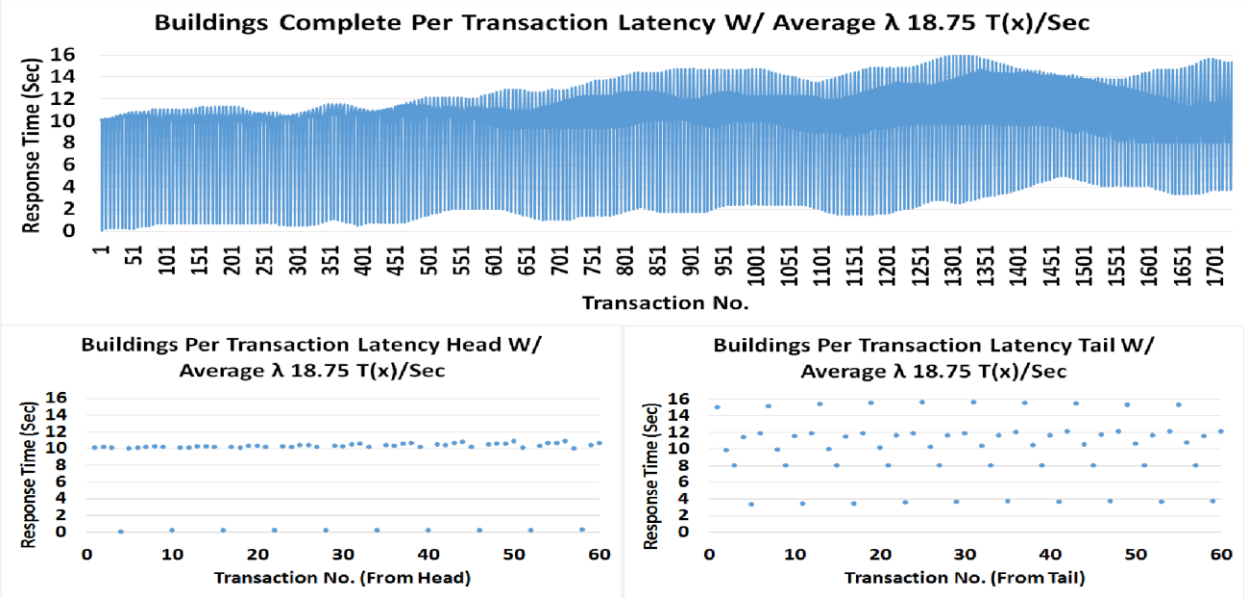

Figure 7.5 Per Transaction Latency Plots for the Buildings Topology W/ Average λ 18.75 T(x)/Sec

We notice peak response times achieved at roughly 1301 transactions. The difference in low and high response times range was 13.41 seconds.

Table 7.13 shows raw sensor data from the buildings logs topology with an arrival rate of 18.75 transactions per second after the experiment was ran.

Table 7.13 Raw Sensor Values corresponding to first 4 T(x)s for the Buildings Topology W/ Average λ 18.75 T(x)/Sec

Buildings Raw Sensor Data Corresponding to first 4 T(x)s W/ Average λ 18.75 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'00:00:00'	'CP'	'1'	'103'	'AV'	'0.00']	1494427749	1494427749
['9/27/2016'	'00:00:00'	'CP'	'1'	'106'	'AT'	'71.50']	1494427749	1494427749
['9/27/2016'	'00:00:00'	'CP'	'2'	'206C'	'RT'	'74.00']	1494427749	1494427749
['9/27/2016'	'00:00:00'	'CP'	'2'	'206H'	'RT'	'74.00']	1494427749	1494427749
['9/27/2016'	'00:00:00'	'CP'	'2'	'206C'	'AT'	'72.00']	1494427749	1494427749
['9/27/2016'	'00:00:00'	'CP'	'2'	'206C'	'CO2'	'401.92']	1494427749	1494427749
['9/27/2016'	'00:00:00'	'CP'	'1'	'108'	'CO2'	'389.73']	1494427749	1494427749
['9/27/2016'	'00:00:00'	'CP'	'2'	'206C'	'AV'	'12.00']	1494427749	1494427749
['9/27/2016'	'00:00:00'	'CP'	'2'	'206D'	'AT'	'72.00']	1494427749	1494427749
['9/27/2016'	'00:00:00'	'CP'	'1'	'108'	'RT'	'73.25']	1494427749	1494427749
['9/27/2016'	'00:00:00'	'CP'	'1'	'108'	'AT'	'72.00']	1494427749	1494427749
['9/27/2016'	'00:05:00'	'CP'	'1'	'103'	'AV'	'0.00']	1494427749	1494427749
['9/27/2016'	'00:05:00'	'CP'	'1'	'106'	'AT'	'71.50']	1494427749	1494427749
['9/27/2016'	'00:05:00'	'CP'	'2'	'206C'	'CO2'	'395.47']	1494427749	1494427749
['9/27/2016'	'00:05:00'	'CP'	'2'	'206H'	'RT'	'74.00']	1494427749	1494427749
['9/27/2016'	'00:05:00'	'CP'	'2'	'206C'	'AT'	'72.00']	1494427749	1494427749

This table contains the sensor values that account for the first 4 transactions that arrived. The first 24 sensor values should all have a real world time of 00:00:00.

From Table 7.13, we can see a minimal amount out of orderedness initiating before the real world time values for the first set in transactions even arrive. If you focus on the "Time" column, you can clearly see how according to real world time, the sensor values are mostly all arriving out of order, with at most an out of orderedness variance in real world time by 5 minutes. If you focus on the "Room" column, it can be seen that only two CP-106 values have arrived, being dominated by CP-206C values. This indicates that the aggregator has to experience initial waiting from the start of the first transactions, with an immediate increase throughout the duration of all 1729 transactions indicated by several increasingly large peaks in response time. So in the early stages of sensor values arriving, aggregation is impacted by out of orderedness, being influenced mostly by room, although slightly by sensor and real world time.

Table 7.14 accounts for the sensor values that arrived during transaction 1301 to 1304, which was when highest response times

Table 7.14 Raw Sensor Values corresponding to T(x)s 1301-1304 for the Buildings Topology W/ Average λ 18.75 T(x)/Sec

Buildings Raw Sensor Data Corresponding T(x)s 1301 through 1304 W/ Average λ 18.75 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'17:25:00'	'CP'	'2'	'206D'	'CO2'	'390.45']	1494427794	1494427794
['9/27/2016'	'19:25:00'	'CP'	'2'	'206C'	'AT'	'71.00']	1494427794	1494427794
['9/27/2016'	'18:55:00'	'CP'	'1'	'106'	'AT'	'69.50']	1494427794	1494427794
['9/27/2016'	'17:00:00'	'CP'	'1'	'108'	'AV'	'432.00']	1494427794	1494427794
['9/27/2016'	'17:05:00'	'CP'	'2'	'206D'	'RT'	'74.00']	1494427794	1494427794
['9/27/2016'	'19:15:00'	'CP'	'2'	'206C'	'CO2'	'372.53']	1494427794	1494427794
['9/27/2016'	'19:20:00'	'CP'	'1'	'103'	'AV'	'324.00']	1494427794	1494427794
['9/27/2016'	'18:35:00'	'CP'	'1'	'108'	'CO2'	'393.32']	1494427794	1494427794
['9/27/2016'	'17:00:00'	'CP'	'2'	'206H'	'CO2'	'396.90']	1494427794	1494427794
['9/27/2016'	'17:05:00'	'CP'	'2'	'206D'	'AV'	'392.00']	1494427794	1494427794
['9/27/2016'	'18:55:00'	'CP'	'2'	'206C'	'RT'	'73.00']	1494427794	1494427794
['9/27/2016'	'20:05:00'	'CP'	'1'	'108'	'RT'	'71.75']	1494427794	1494427794
['9/27/2016'	'17:20:00'	'CP'	'1'	'103'	'CO2'	'376.83']	1494427794	1494427794
['9/27/2016'	'19:20:00'	'CP'	'2'	'206D'	'AT'	'71.00']	1494427794	1494427794
['9/27/2016'	'18:45:00'	'CP'	'2'	'206C'	'AV'	'460.00']	1494427794	1494427794
['9/27/2016'	'19:45:00'	'CP'	'2'	'206H'	'RT'	'74.00']	1494427794	1494427794

were achieved for the buildings topology with an average arrival rate of 18.75 transactions per second.

In the “Time” column, the real world time that the values were pushed to Kafka are highly out of order. Sensor values are arriving with an out of orderedness variance of 3 hours and 5 minute, with an even amount of values arriving by room and sensor type. Towards the final stages of sensor values arriving, out of orderedness is highly influenced by real world.

Figure 7.6 shows the graph of the per transaction latency for the buildings topology with an average arrival rate of 224.25 transactions per second shows initially lower response times compared to the other arrival rates with more consistent response times throughout the entire duration of the 1729 transactions.

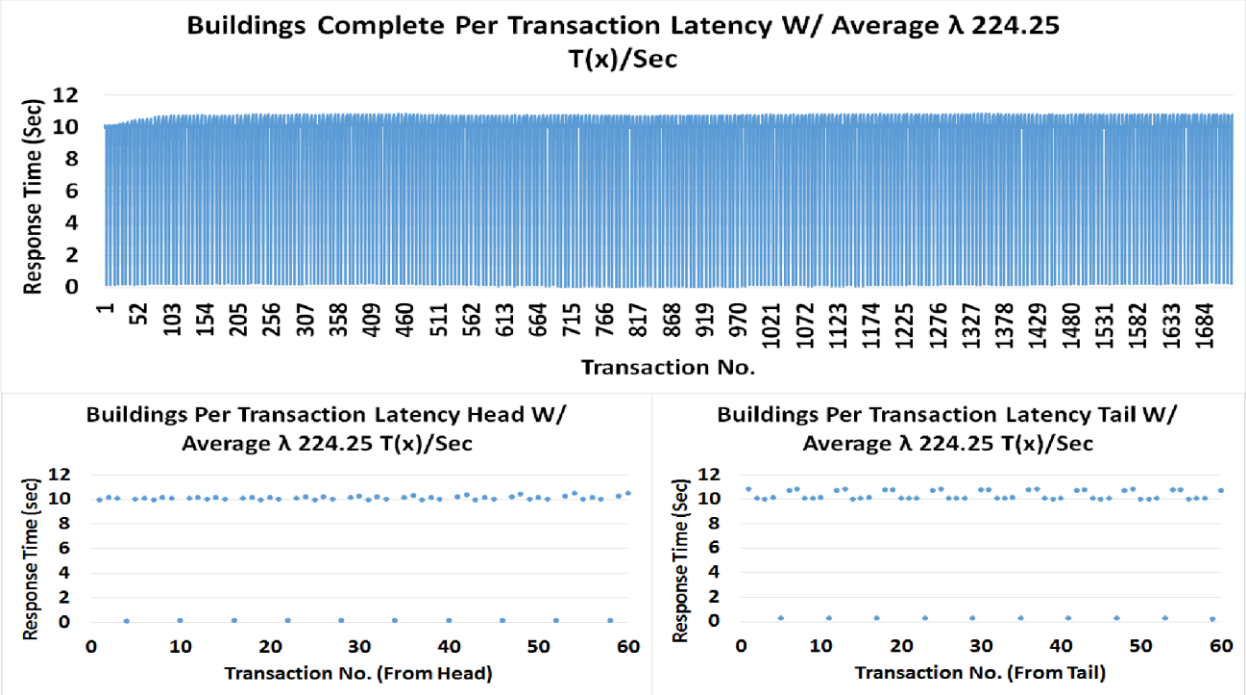

Figure 7.6 Per Transaction Latency Plots for the Buildings Topology W/ Average λ 224.25 T(x)/Sec

We can see that response times are much more evenly distributed throughout the entire duration of the 1729 transactions, with either low or high response times from beginning to end. We do not observe the spikes in response times as we saw in the experimental runs of lower arrival rates. Although queueing is taking place, as indicated by the high response times, the aggregator has a

higher chance that the sensor values it needs are already in queue because the arrival rate sends more sensor values per unit interval. In the case of the lower arrival rates, the variances in response times show cases where the aggregator has to wait even longer because the values haven't even arrived in the system yet. The difference in low and high response times range was 10.65 seconds.

Table 7.15 shows raw sensor data from the buildings logs topology with an arrival rate of 224.25 transactions per second after the experiment was ran. This table contains the sensor values that account for the first 4 transactions that arrived. The first 24 sensor values should all have a real world time of 00:00:00. From table 7.15 we can see immediate out of orderedness initiating before the real world time values for the first set in transactions even arrive. If you focus on the "Time" column, you can clearly see how according to real world time, the sensor values are mostly all arriving out of order, with at most an out of orderedness variance in real world time by 25 minutes.

Table 7.15 Raw Sensor Values corresponding to first 4 T(x)s for the Buildings Topology W/ Average λ 224.25 T(x)/Sec

Buildings Raw Sensor Data Corresponding to first 4 T(x)s W/ Average λ 224.25 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kakfa Push Time	Kafka Pull Time
['9/27/2016'	'00:00:00'	'CP'	'1'	'106'	'CO2'	'370.38']	1494425980	1494425980
['9/27/2016'	'00:00:00'	'CP'	'1'	'103'	'CO2'	'362.49']	1494425980	1494425980
['9/27/2016'	'00:00:00'	'CP'	'1'	'108'	'AV'	'16.00']	1494425980	1494425980
['9/27/2016'	'00:00:00'	'CP'	'1'	'103'	'AV'	'0.00']	1494425980	1494425980
['9/27/2016'	'00:05:00'	'CP'	'1'	'106'	'CO2'	'370.38']	1494425980	1494425980
['9/27/2016'	'00:10:00'	'CP'	'1'	'106'	'CO2'	'371.81']	1494425980	1494425980
['9/27/2016'	'00:05:00'	'CP'	'1'	'103'	'AV'	'0.00']	1494425980	1494425980
['9/27/2016'	'00:05:00'	'CP'	'1'	'103'	'CO2'	'362.49']	1494425980	1494425980
['9/27/2016'	'00:05:00'	'CP'	'1'	'108'	'AV'	'32.00']	1494425980	1494425980
['9/27/2016'	'00:10:00'	'CP'	'1'	'103'	'CO2'	'362.49']	1494425980	1494425980
['9/27/2016'	'00:15:00'	'CP'	'1'	'106'	'CO2'	'371.09']	1494425980	1494425980
['9/27/2016'	'00:20:00'	'CP'	'1'	'106'	'CO2'	'370.38']	1494425980	1494425980
['9/27/2016'	'00:10:00'	'CP'	'1'	'108'	'AV'	'24.00']	1494425980	1494425980
['9/27/2016'	'00:15:00'	'CP'	'1'	'103'	'CO2'	'362.49']	1494425980	1494425980
['9/27/2016'	'00:20:00'	'CP'	'1'	'103'	'CO2'	'363.21']	1494425980	1494425980
['9/27/2016'	'00:25:00'	'CP'	'1'	'106'	'CO2'	'370.38']	1494425980	1494425980

If you focus on the "Room" column, it can be seen that no values from the second floor have arrived. The "Sensor" column shows that all AT and RT values are missing. This indicates that

the aggregator has to experience near worst case waiting from the start of the first transactions, throughout the duration of all 1729 transactions indicated by the consistently uniform and high response times. So in the early stages of sensor values arriving, aggregation is significant impacted by out of orderedness, being influenced by real world time, room, and sensor. However, because the response times are either consistently high or low, this shows that the values are in the queue, but need to be retrieved and sent for aggregation.

Table 7.16 accounts for the sensor values that arrived during transaction 1276 to 1279, which was when highest response times were achieved for the buildings topology with an average arrival rate of 224.25 transactions per second.

Table 7.16 Raw Sensor Values corresponding to T(x)s 1276-1279 for the Buildings Topology W/ Average λ 224.25 T(x)/Sec

Buildings Raw Sensor Data Corresponding T(x)s 1276 through 1279 W/ Average λ 224.25 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kakfa Push Time	Kafka Pull Time
['9/27/2016'	'13:15:00'	'CP'	'2'	'206H'	'RT'	'74.00']	1494425987	1494425987
['9/27/2016'	'13:20:00'	'CP'	'2'	'206H'	'RT'	'74.00']	1494425987	1494425987
['9/27/2016'	'14:20:00'	'CP'	'2'	'206D'	'AV'	'392.00']	1494425987	1494425987
['9/27/2016'	'14:35:00'	'CP'	'2'	'206H'	'CO2'	'402.64']	1494425987	1494425987
['9/27/2016'	'15:05:00'	'CP'	'1'	'106'	'RT'	'71.25']	1494425987	1494425987
['9/27/2016'	'15:10:00'	'CP'	'1'	'106'	'RT'	'71.25']	1494425987	1494425987
['9/27/2016'	'15:15:00'	'CP'	'2'	'206D'	'AT'	'71.50']	1494425987	1494425987
['9/27/2016'	'15:20:00'	'CP'	'2'	'206D'	'AT'	'71.50']	1494425987	1494425987
['9/27/2016'	'15:40:00'	'CP'	'1'	'106'	'AT'	'70.00']	1494425987	1494425987
['9/27/2016'	'15:45:00'	'CP'	'1'	'103'	'AT'	'68.50']	1494425987	1494425987
['9/27/2016'	'16:00:00'	'CP'	'1'	'108'	'RT'	'72.25']	1494425987	1494425987
['9/27/2016'	'16:10:00'	'CP'	'2'	'206C'	'AT'	'72.00']	1494425987	1494425987
['9/27/2016'	'16:10:00'	'CP'	'2'	'206C'	'AV'	'460.00']	1494425987	1494425987
['9/27/2016'	'16:15:00'	'CP'	'2'	'206C'	'CO2'	'419.85']	1494425987	1494425987
['9/27/2016'	'16:20:00'	'CP'	'2'	'206C'	'CO2'	'411.24']	1494425987	1494425987
['9/27/2016'	'17:00:00'	'CP'	'1'	'103'	'RT'	'72.25']	1494425987	1494425987

In the “Time” column, the real world time that the values were pushed to Kafka are highly out of order. Sensor values are arriving with an out of orderedness variance of 3 hours and 45 minutes. Towards the final stages of sensor values arriving, room and sensor level out of orderednss began

to correct itself. However, this topology was still highly influenced by real world time induced out of orderedness. We still notice lower response times overall compared to the other arrival rates. At higher utilization rates, where the arrival rates are higher at 224.25 transactions per second in comparison to the other arrival rates, although there are higher influences of out of orderedness, values are arriving at much higher per second intervals that increase the chances of finding the value it needs. With 224.25 transactions per second at an average response time of 8.65 seconds per job, nearly 1900 sensor values are arriving in the time it takes to complete a transaction tuple, which is significantly greater than the other arrival rates.

The following are the graphs and tables from the per transaction latencies for the floors topology. While similar in response times and behavior in comparison to the buildings topology, the following graphs show that by increasing the number of queues from M/D/1 to M/D/2, highest response times are achieved towards the end of the queue.

Similar to the buildings topology, it is observed that at its highest arrival rate influenced utilization rate, 214.5 transactions per second, we can see the response times decrease and the worst case times are much more uniform in comparison to the lower arrival rates. Again, at higher arrival rates, more values may arrive out of order. Although out of orderedness in the case of the floors topology overall is significantly influenced by real world time, room, and sensor there is a significantly larger amount of values arriving per second during the time it takes on average to service jobs, which increases the chances of having all necessary values to complete a transaction to be in the queue. At 214.5 transactions per second at an average response time of 5.13 to complete each transaction tuple, nearly 1100 values are arriving during the time it takes to service a job, which is significantly higher than in the cases of the lower arrival rates for the floors topology. Likewise, we also observe that at lower utilization rates influenced by arrival rate, lower

arrival rates increase the likelihood of transactions values to become missing from a given arrival rate batch, forcing queueing and forcing the aggregator to have to wait for later batches to find the missing values.

The following figures and tables are the complete set of points for the arrival rates generated for a single floors run.

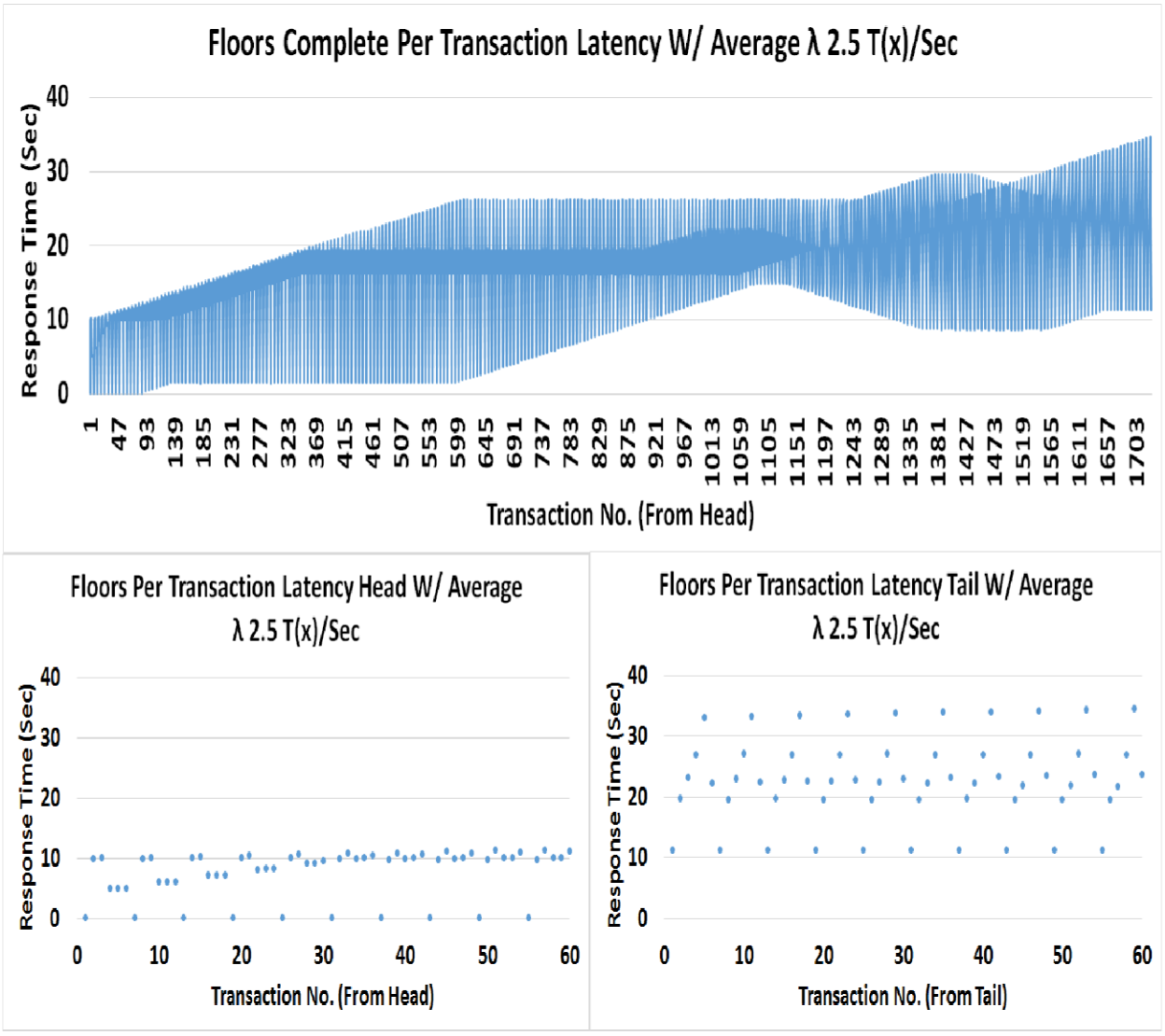

Figure 7.7 Per Transaction Latency Plots for the Floors Topology W/ Average λ 2.5 T(x)/Sec

Table 7.17 Raw Sensor Values corresponding to first 4 T(x)s for the Floors Topology W/ Average λ 2.5 T(x)/Sec

Floors Raw Sensor Data Corresponding to first 4 T(x)s W/ Average λ 2.5 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'00:00:00'	'CP'	'1'	'106'	'AT'	'71.50']	1495213359	1495213359
['9/27/2016'	'00:00:00'	'CP'	'1'	'108'	'AT'	'72.00']	1495213359	1495213359
['9/27/2016'	'00:00:00'	'CP'	'2'	'206D'	'AT'	'72.00']	1495213359	1495213364
['9/27/2016'	'00:00:00'	'CP'	'2'	'206D'	'AV'	'0.00']	1495213359	1495213364
['9/27/2016'	'00:00:00'	'CP'	'2'	'206H'	'AT'	'72.00']	1495213359	1495213364
['9/27/2016'	'00:00:00'	'CP'	'1'	'108'	'AV'	'16.00']	1495213359	1495213359
['9/27/2016'	'00:00:00'	'CP'	'1'	'106'	'RT'	'72.50']	1495213359	1495213359
['9/27/2016'	'00:00:00'	'CP'	'2'	'206C'	'AT'	'72.00']	1495213359	1495213364
['9/27/2016'	'00:00:00'	'CP'	'2'	'206C'	'AV'	'12.00']	1495213359	1495213364
['9/27/2016'	'00:00:00'	'CP'	'1'	'106'	'CO2'	'370.38']	1495213359	1495213359
['9/27/2016'	'00:05:00'	'CP'	'1'	'106'	'AT'	'71.50']	1495213360	1495213360
['9/27/2016'	'00:05:00'	'CP'	'1'	'108'	'AT'	'72.00']	1495213360	1495213360
['9/27/2016'	'00:05:00'	'CP'	'2'	'206D'	'AT'	'72.00']	1495213360	1495213364
['9/27/2016'	'00:05:00'	'CP'	'1'	'108'	'AV'	'32.00']	1495213360	1495213360
['9/27/2016'	'00:05:00'	'CP'	'2'	'206D'	'AV'	'0.00']	1495213360	1495213364
['9/27/2016'	'00:05:00'	'CP'	'2'	'206H'	'AT'	'72.00']	1495213360	1495213364

Table 7.18 Raw Sensor Values corresponding to T(x)s 1621-1624 for the Floors Topology W/ Average λ 2.5 T(x)/Sec

Floors Raw Sensor Data Corresponding T(x)s 1621 through 1624 W/ Average λ 2.5 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'22:50:00'	'CP'	'1'	'108'	'AT'	'71.50']	1495213655	1495213655
['9/27/2016'	'22:25:00'	'CP'	'1'	'108'	'RT'	'71.75']	1495213655	1495213655
['9/27/2016'	'22:30:00'	'CP'	'2'	'206H'	'RT'	'74.00']	1495213655	1495213655
['9/27/2016'	'23:05:00'	'CP'	'2'	'206C'	'AT'	'72.00']	1495213655	1495213655
['9/27/2016'	'22:00:00'	'CP'	'1'	'106'	'AV'	'400.00']	1495213655	1495213655
['9/27/2016'	'22:25:00'	'CP'	'2'	'206H'	'CO2'	'394.75']	1495213655	1495213655
['9/27/2016'	'22:55:00'	'CP'	'1'	'108'	'AV'	'0.00']	1495213655	1495213655
['9/27/2016'	'22:10:00'	'CP'	'2'	'206D'	'RT'	'74.00']	1495213655	1495213656
['9/27/2016'	'22:20:00'	'CP'	'2'	'206C'	'AV'	'468.00']	1495213655	1495213656
['9/27/2016'	'22:25:00'	'CP'	'2'	'206C'	'CO2'	'376.11']	1495213655	1495213656
['9/27/2016'	'21:50:00'	'CP'	'1'	'108'	'CO2'	'391.88']	1495213655	1495213656
['9/27/2016'	'21:55:00'	'CP'	'1'	'103'	'AT'	'71.50']	1495213656	1495213656
['9/27/2016'	'22:45:00'	'CP'	'2'	'206D'	'AT'	'72.00']	1495213656	1495213656
['9/27/2016'	'23:00:00'	'CP'	'2'	'206D'	'AV'	'0.00']	1495213656	1495213656
['9/27/2016'	'22:50:00'	'CP'	'2'	'206D'	'CO2'	'385.43']	1495213656	1495213656
['9/27/2016'	'22:25:00'	'CP'	'1'	'103'	'AV'	'336.00']	1495213656	1495213656

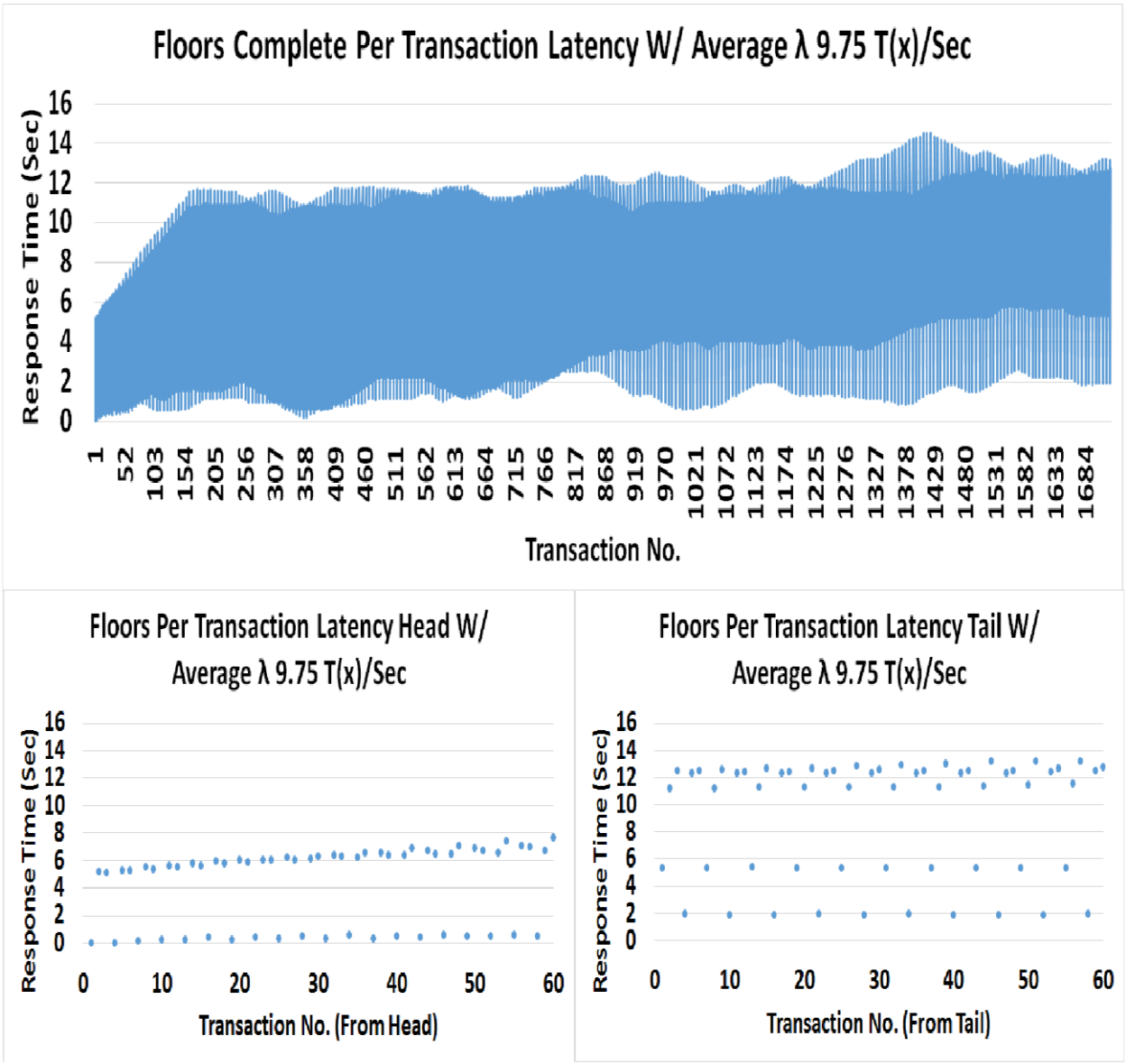

Figure 7.8 Per Transaction Latency Plots for the Floors Topology W/ Average λ 9.75 T(x)/Sec

Table 7.19 Raw Sensor Values corresponding to first 4 T(x)s for the Floors Topology W/ Average λ 9.75 T(x)/Sec

Floors Raw Sensor Data Corresponding to first 4 T(x)s W/ Average λ 9.75 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'00:00:00'	'CP'	'2'	'206D'	'RT'	'74.00']	1495184399	1495184405
['9/27/2016'	'00:00:00'	'CP'	'2'	'206H'	'RT'	'74.00']	1495184399	1495184405
['9/27/2016'	'00:00:00'	'CP'	'1'	'106'	'CO2'	'370.38']	1495184399	1495184405
['9/27/2016'	'00:00:00'	'CP'	'1'	'106'	'AV'	'0.00']	1495184399	1495184405
['9/27/2016'	'00:00:00'	'CP'	'2'	'206H'	'CO2'	'396.90']	1495184399	1495184405
['9/27/2016'	'00:00:00'	'CP'	'1'	'108'	'AV'	'16.00']	1495184400	1495184405
['9/27/2016'	'00:05:00'	'CP'	'2'	'206D'	'RT'	'74.00']	1495184400	1495184405
['9/27/2016'	'00:05:00'	'CP'	'2'	'206H'	'RT'	'74.00']	1495184400	1495184405
['9/27/2016'	'00:05:00'	'CP'	'1'	'106'	'AV'	'0.00']	1495184400	1495184405
['9/27/2016'	'00:05:00'	'CP'	'1'	'106'	'CO2'	'370.38']	1495184400	1495184405
['9/27/2016'	'00:05:00'	'CP'	'2'	'206H'	'CO2'	'397.62']	1495184400	1495184405
['9/27/2016'	'00:05:00'	'CP'	'1'	'108'	'AV'	'32.00']	1495184400	1495184405
['9/27/2016'	'00:10:00'	'CP'	'2'	'206D'	'RT'	'74.00']	1495184400	1495184405
['9/27/2016'	'00:10:00'	'CP'	'2'	'206H'	'RT'	'74.00']	1495184400	1495184405
['9/27/2016'	'00:10:00'	'CP'	'1'	'106'	'CO2'	'371.81']	1495184400	1495184405
['9/27/2016'	'00:10:00'	'CP'	'1'	'106'	'AV'	'0.00']	1495184400	1495184405

Table 7.20 Raw Sensor Values corresponding to T(x)s 1429-1432 for the Floors Topology W/ Average λ 9.75 T(x)/Sec

Floors Raw Sensor Data Corresponding T(x)s 1429 through 1432 W/ Average λ 9.75 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'19:40:00'	'CP'	'2'	'206D'	'AT'	'71.50']	1495184450	1495184450
['9/27/2016'	'21:45:00'	'CP'	'2'	'206H'	'CO2'	'403.36']	1495184450	1495184450
['9/27/2016'	'19:10:00'	'CP'	'2'	'206C'	'RT'	'73.00']	1495184450	1495184450
['9/27/2016'	'19:10:00'	'CP'	'2'	'206C'	'AV'	'460.00']	1495184450	1495184450
['9/27/2016'	'20:25:00'	'CP'	'1'	'103'	'RT'	'71.75']	1495184450	1495184450
['9/27/2016'	'21:50:00'	'CP'	'2'	'206D'	'RT'	'74.00']	1495184450	1495184450
['9/27/2016'	'19:05:00'	'CP'	'2'	'206D'	'AV'	'384.00']	1495184450	1495184450
['9/27/2016'	'19:35:00'	'CP'	'2'	'206H'	'AT'	'71.00']	1495184450	1495184450
['9/27/2016'	'19:15:00'	'CP'	'2'	'206H'	'AV'	'300.00']	1495184450	1495184450
['9/27/2016'	'21:35:00'	'CP'	'1'	'106'	'CO2'	'368.94']	1495184450	1495184450
['9/27/2016'	'19:25:00'	'CP'	'1'	'103'	'CO2'	'378.26']	1495184450	1495184450
['9/27/2016'	'19:05:00'	'CP'	'2'	'206C'	'AT'	'71.00']	1495184450	1495184450
['9/27/2016'	'19:25:00'	'CP'	'1'	'106'	'AT'	'69.50']	1495184450	1495184450
['9/27/2016'	'18:50:00'	'CP'	'1'	'108'	'RT'	'71.75']	1495184450	1495184450
['9/27/2016'	'19:00:00'	'CP'	'2'	'206D'	'CO2'	'387.58']	1495184450	1495184450
['9/27/2016'	'19:20:00'	'CP'	'1'	'103'	'AV'	'324.00']	1495184450	1495184450

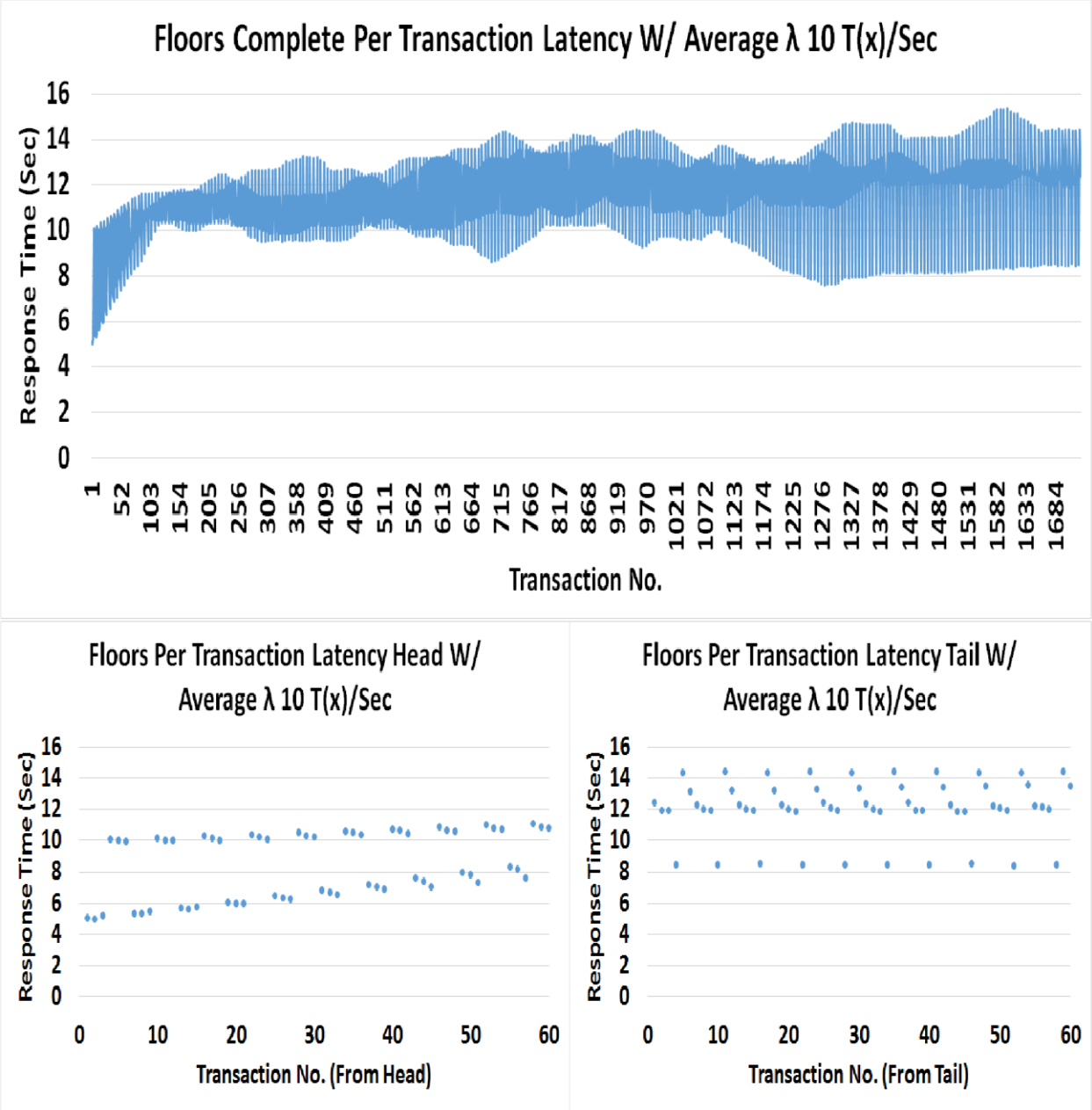

Figure 7.9 Per Transaction Latency Plots for the Floors Topology W/ Average λ 10 T(x)/Sec

Table 7.21 Raw Sensor Values corresponding to first 4 T(x)s for the Floors Topology W/ Average λ 10 T(x)/Sec

Floors Raw Sensor Data Corresponding to first 4 T(x)s W/ Average λ 10 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'00:00:00'	'CP'	'2'	'206C'	'AV'	'12.00']	1495211123	1495211123
['9/27/2016'	'00:00:00'	'CP'	'1'	'108'	'RT'	'73.25']	1495211123	1495211128
['9/27/2016'	'00:00:00'	'CP'	'1'	'103'	'AV'	'0.00']	1495211123	1495211128
['9/27/2016'	'00:00:00'	'CP'	'1'	'108'	'AV'	'16.00']	1495211123	1495211128
['9/27/2016'	'00:00:00'	'CP'	'2'	'206C'	'AT'	'72.00']	1495211123	1495211123
['9/27/2016'	'00:00:00'	'CP'	'1'	'108'	'CO2'	'389.73']	1495211123	1495211128
['9/27/2016'	'00:00:00'	'CP'	'1'	'106'	'CO2'	'370.38']	1495211123	1495211128
['9/27/2016'	'00:00:00'	'CP'	'2'	'206D'	'RT'	'74.00']	1495211123	1495211123
['9/27/2016'	'00:00:00'	'CP'	'2'	'206C'	'CO2'	'401.92']	1495211123	1495211124
['9/27/2016'	'00:00:00'	'CP'	'2'	'206H'	'AV'	'20.00']	1495211124	1495211124
['9/27/2016'	'00:05:00'	'CP'	'2'	'206C'	'AV'	'8.00']	1495211124	1495211124
['9/27/2016'	'00:05:00'	'CP'	'1'	'108'	'RT'	'73.25']	1495211124	1495211128
['9/27/2016'	'00:05:00'	'CP'	'1'	'103'	'AV'	'0.00']	1495211124	1495211128
['9/27/2016'	'00:05:00'	'CP'	'1'	'108'	'AV'	'32.00']	1495211124	1495211128
['9/27/2016'	'00:05:00'	'CP'	'2'	'206C'	'AT'	'72.00']	1495211124	1495211124
['9/27/2016'	'00:05:00'	'CP'	'1'	'108'	'CO2'	'391.17']	1495211124	1495211128

Table 7.22 Raw Sensor Values corresponding to T(x)s 1582-1585 for the Floors Topology W/ Average λ 10 T(x)/Sec

Floors Raw Sensor Data Corresponding T(x)s 1582 through 1585 W/ Average λ 10 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'21:10:00'	'CP'	'1'	'106'	'AV'	'400.00']	1495211204	1495211204
['9/27/2016'	'21:30:00'	'CP'	'1'	'103'	'RT'	'71.75']	1495211204	1495211204
['9/27/2016'	'23:10:00'	'CP'	'2'	'206C'	'AT'	'72.00']	1495211204	1495211204
['9/27/2016'	'21:10:00'	'CP'	'2'	'206H'	'RT'	'74.00']	1495211204	1495211204
['9/27/2016'	'21:15:00'	'CP'	'1'	'106'	'AT'	'69.50']	1495211204	1495211204
['9/27/2016'	'22:45:00'	'CP'	'1'	'108'	'RT'	'72.50']	1495211204	1495211204
['9/27/2016'	'23:00:00'	'CP'	'1'	'108'	'CO2'	'389.73']	1495211204	1495211204
['9/27/2016'	'22:50:00'	'CP'	'1'	'103'	'AV'	'0.00']	1495211204	1495211204
['9/27/2016'	'22:50:00'	'CP'	'2'	'206C'	'CO2'	'381.13']	1495211204	1495211204
['9/27/2016'	'22:55:00'	'CP'	'1'	'106'	'CO2'	'363.21']	1495211204	1495211204
['9/27/2016'	'21:10:00'	'CP'	'2'	'206C'	'RT'	'73.00']	1495211204	1495211204
['9/27/2016'	'22:45:00'	'CP'	'1'	'108'	'AV'	'0.00']	1495211204	1495211204
['9/27/2016'	'22:45:00'	'CP'	'2'	'206D'	'RT'	'74.00']	1495211204	1495211204
['9/27/2016'	'21:05:00'	'CP'	'2'	'206D'	'AV'	'428.00']	1495211204	1495211204
['9/27/2016'	'22:40:00'	'CP'	'2'	'206H'	'AV'	'0.00']	1495211204	1495211204
['9/27/2016'	'21:15:00'	'CP'	'1'	'106'	'RT'	'71.75']	1495211204	1495211204

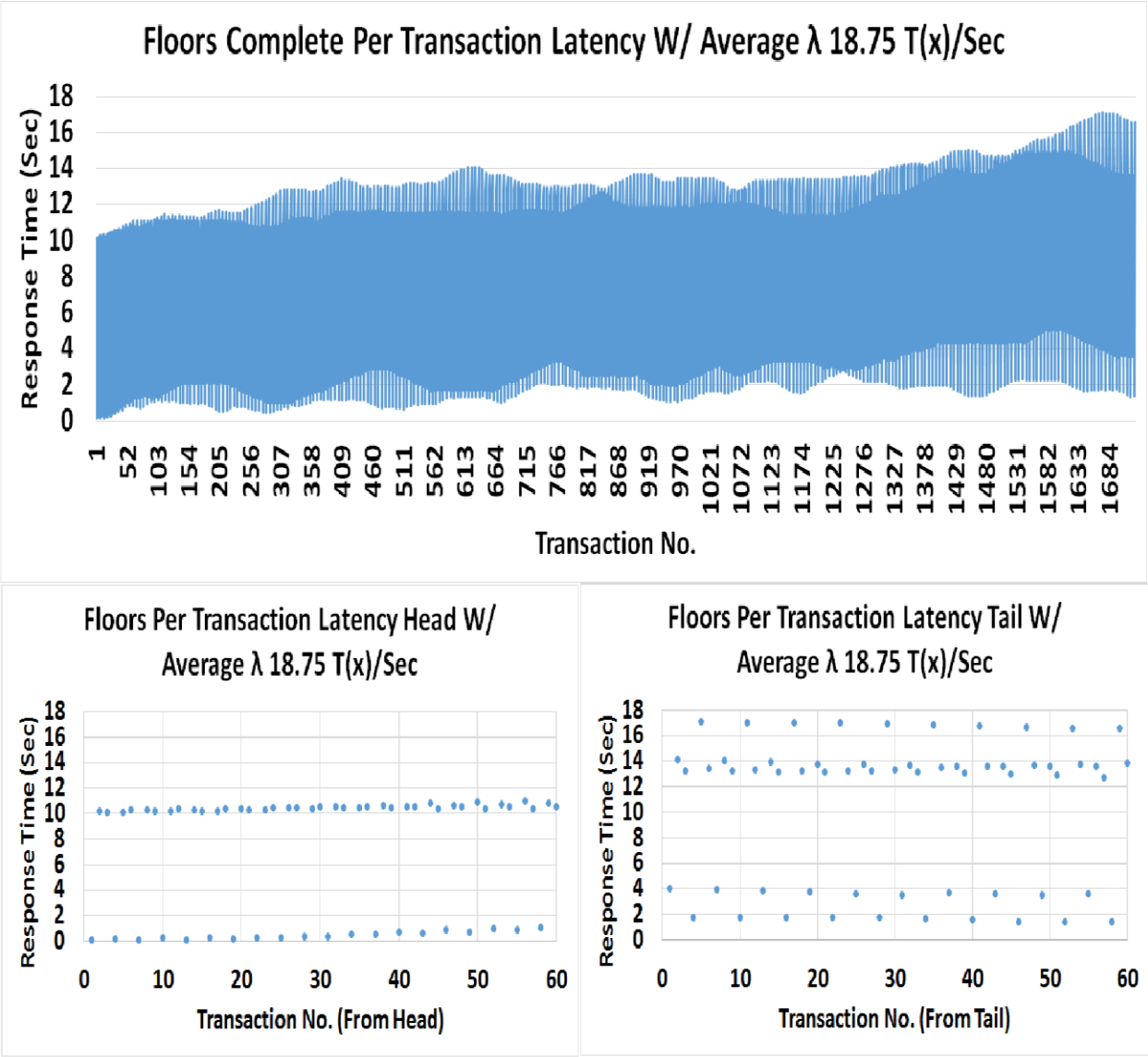

Figure 7.10 Per Transaction Latency Plots for the Floors Topology W/ Average λ 18.75 T(x)/Sec

Table 7.23 Raw Sensor Values corresponding to first 4 T(x)s for the Floors Topology W/ Average λ 18.75 T(x)/Sec

Floors Raw Sensor Data Corresponding to first 4 T(x)s W/ Average λ 18.75 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'00:00:00'	'CP'	'1'	'108'	'CO2'	'389.73']	1495185743	1495185743
['9/27/2016'	'00:00:00'	'CP'	'2'	'206C'	'AV'	'12.00']	1495185743	1495185743
['9/27/2016'	'00:00:00'	'CP'	'2'	'206H'	'AV'	'20.00']	1495185743	1495185743
['9/27/2016'	'00:00:00'	'CP'	'2'	'206D'	'RT'	'74.00']	1495185743	1495185743
['9/27/2016'	'00:00:00'	'CP'	'1'	'103'	'CO2'	'362.49']	1495185743	1495185743
['9/27/2016'	'00:00:00'	'CP'	'1'	'103'	'AT'	'72.50']	1495185743	1495185743
['9/27/2016'	'00:00:00'	'CP'	'2'	'206C'	'RT'	'74.00']	1495185743	1495185743
['9/27/2016'	'00:00:00'	'CP'	'1'	'103'	'RT'	'71.75']	1495185743	1495185743
['9/27/2016'	'00:00:00'	'CP'	'2'	'206C'	'AT'	'72.00']	1495185743	1495185743
['9/27/2016'	'00:00:00'	'CP'	'1'	'106'	'AV'	'0.00']	1495185743	1495185743
['9/27/2016'	'00:00:00'	'CP'	'1'	'106'	'CO2'	'370.38']	1495185743	1495185743
['9/27/2016'	'00:00:00'	'CP'	'1'	'108'	'AT'	'72.00']	1495185743	1495185743
['9/27/2016'	'00:00:00'	'CP'	'1'	'103'	'AV'	'0.00']	1495185743	1495185743
['9/27/2016'	'00:00:00'	'CP'	'2'	'206D'	'CO2'	'399.77']	1495185743	1495185743
['9/27/2016'	'00:00:00'	'CP'	'2'	'206C'	'CO2'	'401.92']	1495185743	1495185743
['9/27/2016'	'00:05:00'	'CP'	'1'	'108'	'CO2'	'391.17']	1495185743	1495185743

Table 7.24 Raw Sensor Values corresponding to T(x)s 1633-1636 for the Floors Topology W/ Average λ 18.75 T(x)/Sec

Floors Raw Sensor Data Corresponding T(x)s 1633 through 1636 W/ Average λ 18.75 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'21:45:00'	'CP'	'2'	'206H'	'RT'	'74.00']	1495185811	1495185811
['9/27/2016'	'21:10:00'	'CP'	'2'	'206D'	'AT'	'72.00']	1495185811	1495185811
['9/27/2016'	'21:20:00'	'CP'	'1'	'108'	'RT'	'71.75']	1495185811	1495185811
['9/27/2016'	'23:50:00'	'CP'	'2'	'206D'	'RT'	'74.00']	1495185811	1495185811
['9/27/2016'	'23:00:00'	'CP'	'2'	'206C'	'CO2'	'380.41']	1495185811	1495185811
['9/27/2016'	'23:50:00'	'CP'	'2'	'206H'	'AV'	'0.00']	1495185811	1495185811
['9/27/2016'	'23:25:00'	'CP'	'1'	'108'	'AT'	'71.50']	1495185811	1495185811
['9/27/2016'	'23:00:00'	'CP'	'1'	'108'	'CO2'	'389.73']	1495185811	1495185811
['9/27/2016'	'23:50:00'	'CP'	'1'	'103'	'AT'	'72.50']	1495185811	1495185811
['9/27/2016'	'21:05:00'	'CP'	'1'	'106'	'AT'	'69.50']	1495185811	1495185811
['9/27/2016'	'21:30:00'	'CP'	'2'	'206H'	'CO2'	'402.64']	1495185811	1495185811
['9/27/2016'	'21:45:00'	'CP'	'2'	'206D'	'AV'	'472.00']	1495185811	1495185811
['9/27/2016'	'21:30:00'	'CP'	'1'	'106'	'RT'	'71.75']	1495185811	1495185811
['9/27/2016'	'22:00:00'	'CP'	'2'	'206H'	'AT'	'72.00']	1495185811	1495185811
['9/27/2016'	'23:05:00'	'CP'	'1'	'106'	'AV'	'0.00']	1495185811	1495185811
['9/27/2016'	'23:15:00'	'CP'	'2'	'206C'	'RT'	'73.25']	1495185811	1495185811

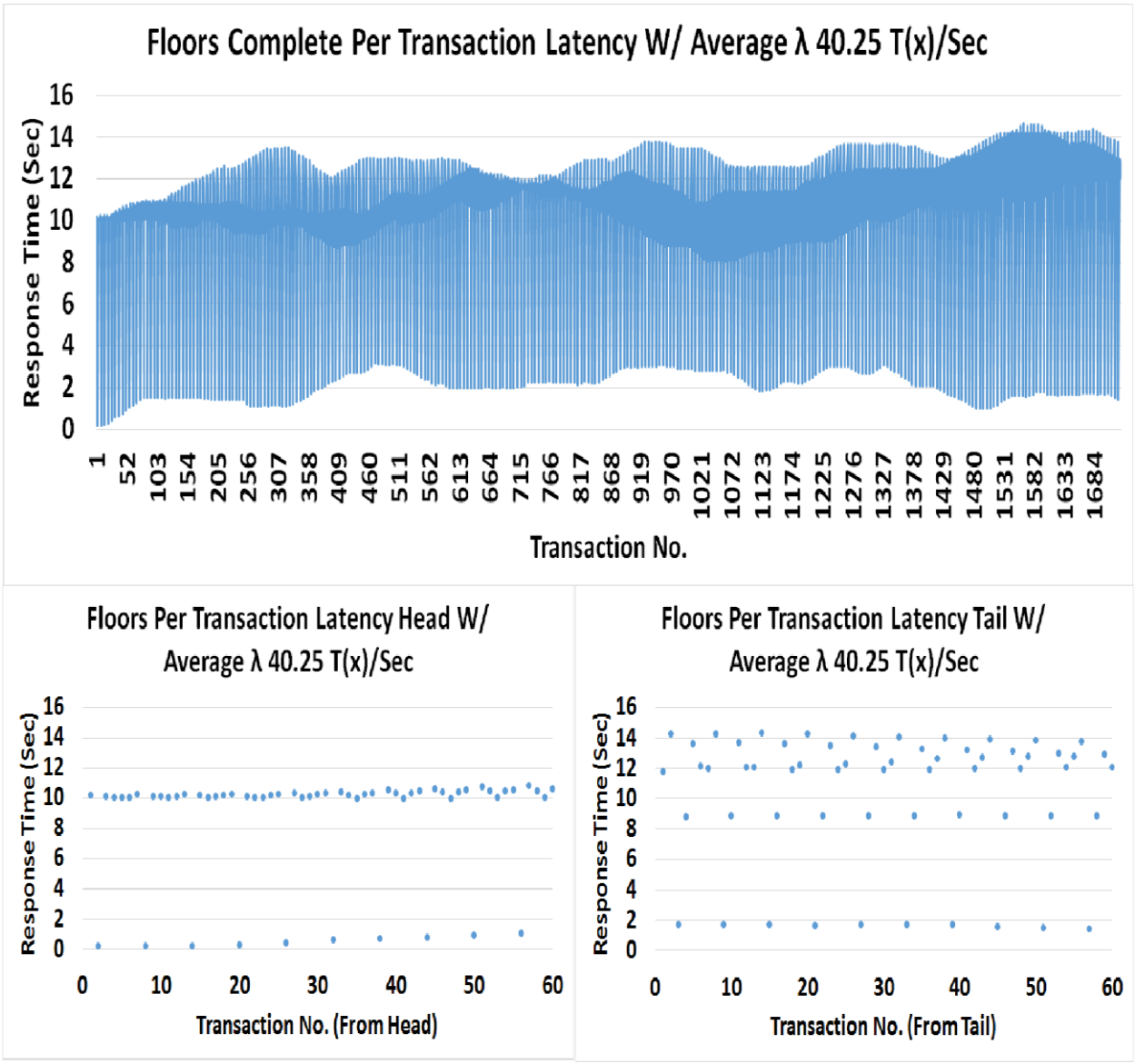

Figure 7.11 Per Transaction Latency Plots for the Floors Topology W/ Average λ 40.25 T(x)/Sec

Table 7.25 Raw Sensor Values corresponding to first 4 T(x)s for the Floors Topology W/ Average λ 40.25 T(x)/Sec

Floors Raw Sensor Data Corresponding to first 4 T(x)s W/ Average λ 40.25 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'00:00:00'	'CP'	'1'	'106'	'AV'	'0.00']	1495183423	1495183423
['9/27/2016'	'00:00:00'	'CP'	'1'	'103'	'RT'	'71.75']	1495183423	1495183423
['9/27/2016'	'00:00:00'	'CP'	'1'	'106'	'AT'	'71.50']	1495183423	1495183423
['9/27/2016'	'00:00:00'	'CP'	'1'	'103'	'AV'	'0.00']	1495183423	1495183423
['9/27/2016'	'00:00:00'	'CP'	'2'	'206H'	'CO2'	'396.90']	1495183423	1495183423
['9/27/2016'	'00:00:00'	'CP'	'2'	'206C'	'AV'	'12.00']	1495183423	1495183423
['9/27/2016'	'00:00:00'	'CP'	'2'	'206C'	'CO2'	'401.92']	1495183423	1495183423
['9/27/2016'	'00:00:00'	'CP'	'1'	'108'	'AV'	'16.00']	1495183423	1495183423
['9/27/2016'	'00:00:00'	'CP'	'2'	'206C'	'RT'	'74.00']	1495183423	1495183423
['9/27/2016'	'00:00:00'	'CP'	'2'	'206D'	'RT'	'74.00']	1495183423	1495183423
['9/27/2016'	'00:05:00'	'CP'	'1'	'103'	'RT'	'71.75']	1495183423	1495183423
['9/27/2016'	'00:00:00'	'CP'	'1'	'106'	'RT'	'72.50']	1495183423	1495183423
['9/27/2016'	'00:00:00'	'CP'	'1'	'108'	'CO2'	'389.73']	1495183423	1495183423
['9/27/2016'	'00:00:00'	'CP'	'1'	'108'	'RT'	'73.25']	1495183423	1495183423
['9/27/2016'	'00:00:00'	'CP'	'2'	'206D'	'AT'	'72.00']	1495183423	1495183423
['9/27/2016'	'00:05:00'	'CP'	'1'	'106'	'AV'	'0.00']	1495183423	1495183423

Table 7.26 Raw Sensor Values corresponding to T(x)s 1582-1585 for the Floors Topology W/ Average λ 40.25 T(x)/Sec

Floors Raw Sensor Data Corresponding T(x)s 1582 through 1585 W/ Average λ 40.25 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'19:25:00'	'CP'	'2'	'206C'	'AT'	'71.00']	1495183462	1495183463
['9/27/2016'	'22:00:00'	'CP'	'1'	'106'	'CO2'	'369.66']	1495183462	1495183463
['9/27/2016'	'22:10:00'	'CP'	'1'	'106'	'AV'	'400.00']	1495183462	1495183463
['9/27/2016'	'20:20:00'	'CP'	'1'	'103'	'CO2'	'378.26']	1495183462	1495183463
['9/27/2016'	'20:40:00'	'CP'	'2'	'206H'	'AT'	'72.00']	1495183462	1495183463
['9/27/2016'	'22:40:00'	'CP'	'1'	'108'	'AV'	'0.00']	1495183463	1495183463
['9/27/2016'	'23:15:00'	'CP'	'2'	'206C'	'CO2'	'384.71']	1495183463	1495183463
['9/27/2016'	'22:20:00'	'CP'	'1'	'108'	'CO2'	'391.88']	1495183463	1495183463
['9/27/2016'	'23:45:00'	'CP'	'2'	'206H'	'RT'	'74.00']	1495183463	1495183463
['9/27/2016'	'20:05:00'	'CP'	'2'	'206D'	'AV'	'384.00']	1495183463	1495183463
['9/27/2016'	'22:20:00'	'CP'	'1'	'108'	'RT'	'71.75']	1495183463	1495183463
['9/27/2016'	'21:55:00'	'CP'	'2'	'206D'	'AT'	'72.00']	1495183463	1495183463
['9/27/2016'	'22:50:00'	'CP'	'2'	'206C'	'RT'	'73.25']	1495183463	1495183463
['9/27/2016'	'19:00:00'	'CP'	'1'	'103'	'AT'	'68.50']	1495183463	1495183463
['9/27/2016'	'20:45:00'	'CP'	'2'	'206H'	'AT'	'72.00']	1495183463	1495183463
['9/27/2016'	'22:15:00'	'CP'	'1'	'106'	'AT'	'69.00']	1495183463	1495183463

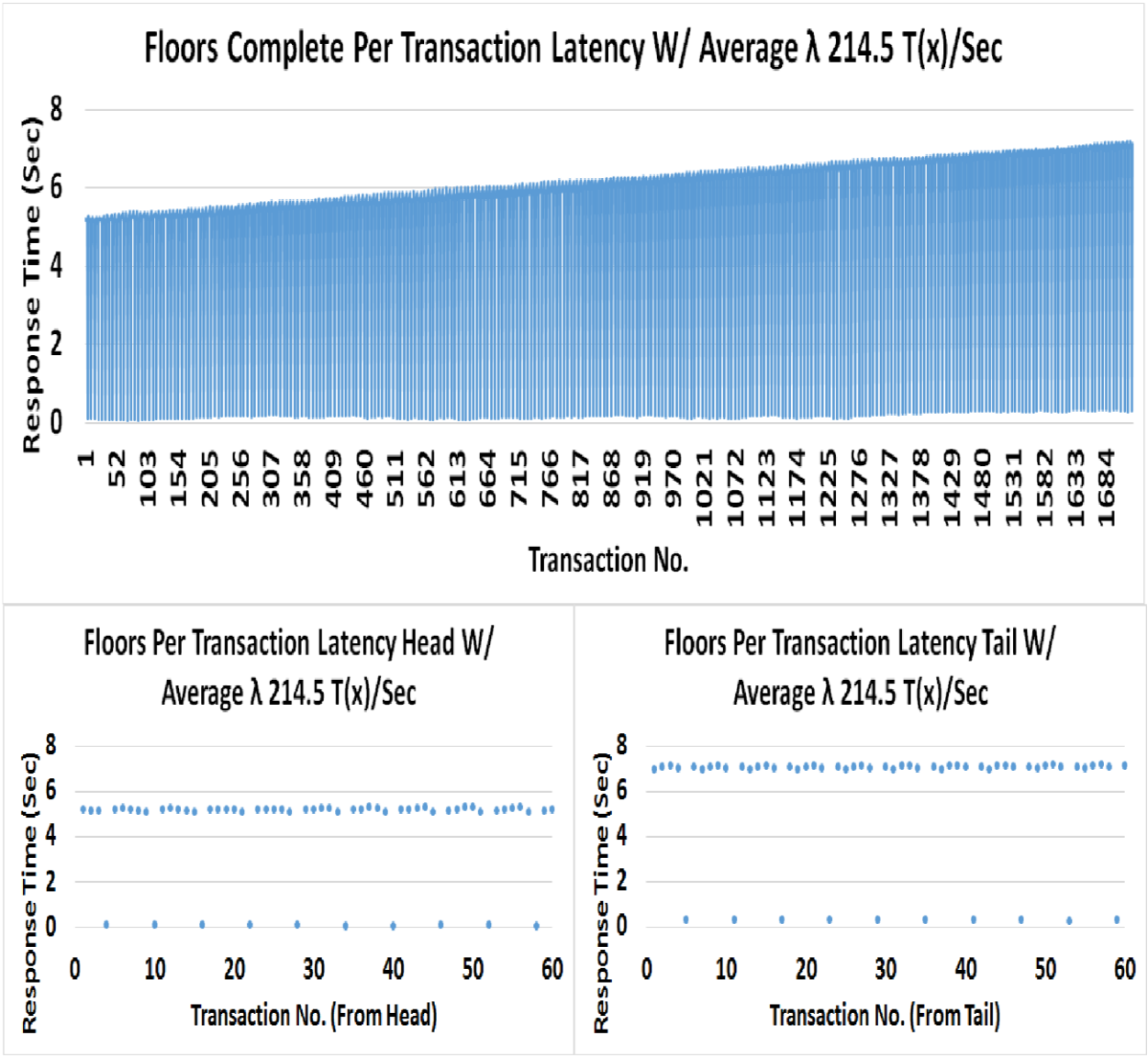

Figure 7.12 Per Transaction Latency Plots for the Floors Topology W/ Average λ 214.5 T(x)/Sec

Table 7.27 Raw Sensor Values corresponding to first 4 T(x)s for the Floors Topology W/ Average λ 214.5 T(x)/Sec

Floors Raw Sensor Data Corresponding to first 4 T(x)s W/ Average λ 214.5 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'00:00:00'	'CP'	'1'	'103'	'AV'	'0.00']	1495282875	1495282880
['9/27/2016'	'00:05:00'	'CP'	'1'	'103'	'AV'	'0.00']	1495282875	1495282880
['9/27/2016'	'00:10:00'	'CP'	'1'	'103'	'AV'	'0.00']	1495282875	1495282880
['9/27/2016'	'00:15:00'	'CP'	'1'	'103'	'AV'	'0.00']	1495282875	1495282880
['9/27/2016'	'00:00:00'	'CP'	'2'	'206H'	'RT'	'74.00']	1495282875	1495282880
['9/27/2016'	'00:20:00'	'CP'	'1'	'103'	'AV'	'0.00']	1495282875	1495282880
['9/27/2016'	'00:00:00'	'CP'	'1'	'106'	'CO2'	'370.38']	1495282875	1495282880
['9/27/2016'	'00:05:00'	'CP'	'2'	'206H'	'RT'	'74.00']	1495282875	1495282880
['9/27/2016'	'00:25:00'	'CP'	'1'	'103'	'AV'	'0.00']	1495282875	1495282880
['9/27/2016'	'00:30:00'	'CP'	'1'	'103'	'AV'	'0.00']	1495282875	1495282880
['9/27/2016'	'00:05:00'	'CP'	'1'	'106'	'CO2'	'370.38']	1495282875	1495282880
['9/27/2016'	'00:00:00'	'CP'	'2'	'206D'	'AV'	'0.00']	1495282875	1495282880
['9/27/2016'	'00:10:00'	'CP'	'2'	'206H'	'RT'	'74.00']	1495282875	1495282880
['9/27/2016'	'00:35:00'	'CP'	'1'	'103'	'AV'	'0.00']	1495282875	1495282880
['9/27/2016'	'00:00:00'	'CP'	'1'	'108'	'CO2'	'389.73']	1495282875	1495282880
['9/27/2016'	'00:05:00'	'CP'	'2'	'206D'	'AV'	'0.00']	1495282875	1495282880

Table 7.28 Raw Sensor Values corresponding to T(x)s 1633-1636 for the Floors Topology W/ Average λ 214.5 T(x)/Sec

Floors Raw Sensor Data Corresponding T(x)s 1633 through 1636 W/ Average λ 214.5 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'21:10:00'	'CP'	'1'	'103'	'CO2'	'378.26']	1495282882	1495282882
['9/27/2016'	'21:15:00'	'CP'	'2'	'206C'	'CO2'	'384.00']	1495282882	1495282882
['9/27/2016'	'21:25:00'	'CP'	'1'	'106'	'AT'	'69.50']	1495282882	1495282882
['9/27/2016'	'21:40:00'	'CP'	'1'	'108'	'AT'	'70.00']	1495282882	1495282882
['9/27/2016'	'21:45:00'	'CP'	'2'	'206H'	'AV'	'296.00']	1495282882	1495282882
['9/27/2016'	'21:50:00'	'CP'	'2'	'206H'	'AV'	'300.00']	1495282882	1495282882
['9/27/2016'	'21:55:00'	'CP'	'1'	'108'	'RT'	'71.75']	1495282882	1495282882
['9/27/2016'	'22:25:00'	'CP'	'2'	'206C'	'AT'	'72.00']	1495282882	1495282882
['9/27/2016'	'22:25:00'	'CP'	'2'	'206C'	'AV'	'456.00']	1495282882	1495282882
['9/27/2016'	'22:30:00'	'CP'	'2'	'206C'	'AT'	'72.00']	1495282882	1495282882
['9/27/2016'	'22:30:00'	'CP'	'2'	'206C'	'AV'	'456.00']	1495282882	1495282882
['9/27/2016'	'22:30:00'	'CP'	'2'	'206H'	'AT'	'72.00']	1495282882	1495282882
['9/27/2016'	'22:40:00'	'CP'	'1'	'103'	'AT'	'72.00']	1495282882	1495282882
['9/27/2016'	'22:40:00'	'CP'	'1'	'103'	'RT'	'71.75']	1495282882	1495282882
['9/27/2016'	'22:40:00'	'CP'	'2'	'206H'	'CO2'	'397.62']	1495282882	1495282882
['9/27/2016'	'22:45:00'	'CP'	'1'	'103'	'AT'	'72.00']	1495282882	1495282882

The graph from the per transaction latency for the rooms topology shows a significant decrease in the low and high response times in the M/D/6 rooms topology in comparison to both the M/D/2 floors and M/D/1 buildings topology. In the worst case scenario under the lowest utilization rates influenced by the lowest arrival rates, we see greater than a 2 times decrease in response time from 36 seconds to 15 seconds. This lets us know that the degree of out of orderedness is lower in the case of the rooms topology.

Figure 7.13 shows the graph from the per transaction latency for the rooms topology with an average arrival rate of 1.25 transactions per second shows a considerable decrease in response time in comparison to the buildings and floors topology at lower arrival rates. If we look at the per transaction latencies from the head, as shown underneath the complete transaction latency, we can see in the initial stages, almost all values are arriving in order. When response times are at its highest, more transactions are arriving out of order, but with a very low degree of out of orderedness, as many of the tail end 60 transactions still appear on the lower end of the response time range.

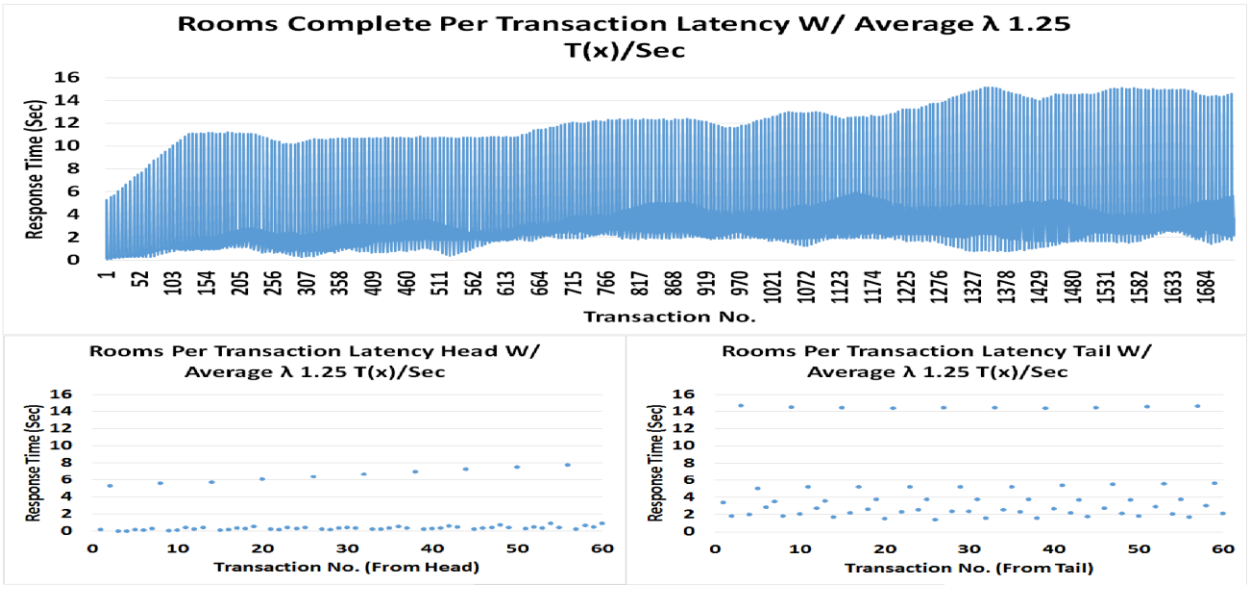

Figure 7.13 Per Transaction Latency Plots for the Rooms Topology W/ Average λ 1.25 T(x)/Sec

However, we still see a significant amount of variance in between response times per transaction.

Table 7.29 shows raw sensor data from the rooms topology logs after the experiment was ran. The first table contains the sensor values that account for the first 4 transactions that arrived. If you focus on the "Time" column, you can clearly see how according to real world time, all real world time values occurred in successive 5 minute periods. In addition, all sensors values for CP-106 as well as all AT values arrived in order.

Table 7.29 Raw Sensor Values corresponding to first 4 T(x)s for the Rooms Topology W/ Average λ 1.25 T(x)/Sec

Rooms Raw Sensor Data Corresponding to first 4 T(x)s W/ Average λ 1.25 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'00:00:00'	'CP'	'1'	'106'	'AT'	'71.50']	1495324480	1495324485
['9/27/2016'	'00:05:00'	'CP'	'1'	'106'	'AT'	'71.50']	1495324480	1495324485
['9/27/2016'	'00:10:00'	'CP'	'1'	'106'	'AT'	'71.50']	1495324480	1495324485
['9/27/2016'	'00:15:00'	'CP'	'1'	'106'	'AT'	'71.50']	1495324481	1495324485
['9/27/2016'	'00:20:00'	'CP'	'1'	'106'	'AT'	'72.00']	1495324481	1495324485
['9/27/2016'	'00:25:00'	'CP'	'1'	'106'	'AT'	'72.00']	1495324481	1495324485
['9/27/2016'	'00:30:00'	'CP'	'1'	'106'	'AT'	'72.00']	1495324481	1495324485
['9/27/2016'	'00:35:00'	'CP'	'1'	'106'	'AT'	'72.00']	1495324481	1495324485
['9/27/2016'	'00:40:00'	'CP'	'1'	'106'	'AT'	'72.00']	1495324482	1495324485
['9/27/2016'	'00:45:00'	'CP'	'1'	'106'	'AT'	'72.00']	1495324482	1495324485
['9/27/2016'	'00:50:00'	'CP'	'1'	'106'	'AT'	'72.00']	1495324482	1495324485
['9/27/2016'	'00:55:00'	'CP'	'1'	'106'	'AT'	'72.00']	1495324482	1495324485
['9/27/2016'	'01:00:00'	'CP'	'1'	'106'	'AT'	'72.00']	1495324483	1495324485
['9/27/2016'	'01:05:00'	'CP'	'1'	'106'	'AT'	'72.00']	1495324483	1495324485
['9/27/2016'	'01:10:00'	'CP'	'1'	'106'	'AT'	'72.00']	1495324483	1495324485
['9/27/2016'	'01:15:00'	'CP'	'1'	'106'	'AT'	'72.00']	1495324483	1495324485

So in the early stages of sensor values arriving, aggregation is impacted by out of orderedness, being influenced minimally by room and sensor. The logs further show that the sequence takes care of AT first, followed by AV, CO2, and RT respectively. So at room level granularity, the values are arriving in order. So the only influence that is affecting the out of orderedness in this case is the real world time, with a minimally negative impact.

Table 7.30 accounts for the sensor values that arrived during transaction 1327 to 1330, which was when highest response times were achieved for the rooms topology.

Table 7.30 Raw Sensor Values corresponding to T(x)s 1328-1330 for the Rooms Topology W/ Average λ 1.25 T(x)/Sec

Rooms Raw Sensor Data Corresponding T(x)s 1327 through 1330 W/ Average λ 1.25 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'18:25:00'	'CP'	'1'	'103'	'AT'	'67.50']	1495324539	1495324539
['9/27/2016'	'18:25:00'	'CP'	'2'	'206D'	'CO2'	'390.45']	1495324539	1495324539
['9/27/2016'	'18:25:00'	'CP'	'2'	'206H'	'CO2'	'397.62']	1495324539	1495324539
['9/27/2016'	'18:15:00'	'CP'	'1'	'108'	'CO2'	'395.47']	1495324539	1495324540
['9/27/2016'	'18:10:00'	'CP'	'1'	'108'	'RT'	'72.25']	1495324539	1495324540
['9/27/2016'	'18:25:00'	'CP'	'2'	'206D'	'AT'	'71.50']	1495324539	1495324540
['9/27/2016'	'18:45:00'	'CP'	'2'	'206C'	'RT'	'73.50']	1495324539	1495324540
['9/27/2016'	'18:30:00'	'CP'	'2'	'206C'	'CO2'	'383.28']	1495324539	1495324540
['9/27/2016'	'18:20:00'	'CP'	'1'	'103'	'RT'	'71.75']	1495324539	1495324540
['9/27/2016'	'18:20:00'	'CP'	'1'	'108'	'AV'	'432.00']	1495324539	1495324540
['9/27/2016'	'18:30:00'	'CP'	'1'	'106'	'RT'	'72.25']	1495324539	1495324540
['9/27/2016'	'18:15:00'	'CP'	'1'	'106'	'CO2'	'370.38']	1495324539	1495324540
['9/27/2016'	'18:10:00'	'CP'	'1'	'106'	'AV'	'468.00']	1495324539	1495324540
['9/27/2016'	'18:05:00'	'CP'	'1'	'103'	'CO2'	'378.26']	1495324540	1495324540
['9/27/2016'	'18:25:00'	'CP'	'2'	'206H'	'RT'	'74.00']	1495324540	1495324540
['9/27/2016'	'18:10:00'	'CP'	'2'	'206D'	'AV'	'392.00']	1495324540	1495324540

In the “Time” column, the real world time that the values were pushed to Kafka are highly out of order. However, in comparison to buildings and floors that experienced up to 4 hours of real world time variances, sensor values for the rooms topology with an average arrival rate of 1.25 transactions per second are arriving with an out of orderedness variance of 25 minutes. Towards the final stages of sensor values arriving, out of orderedness is minimally influenced by both room and sensor.

The following are the figures and tables for the rest of the arrival rates in the rooms topology.

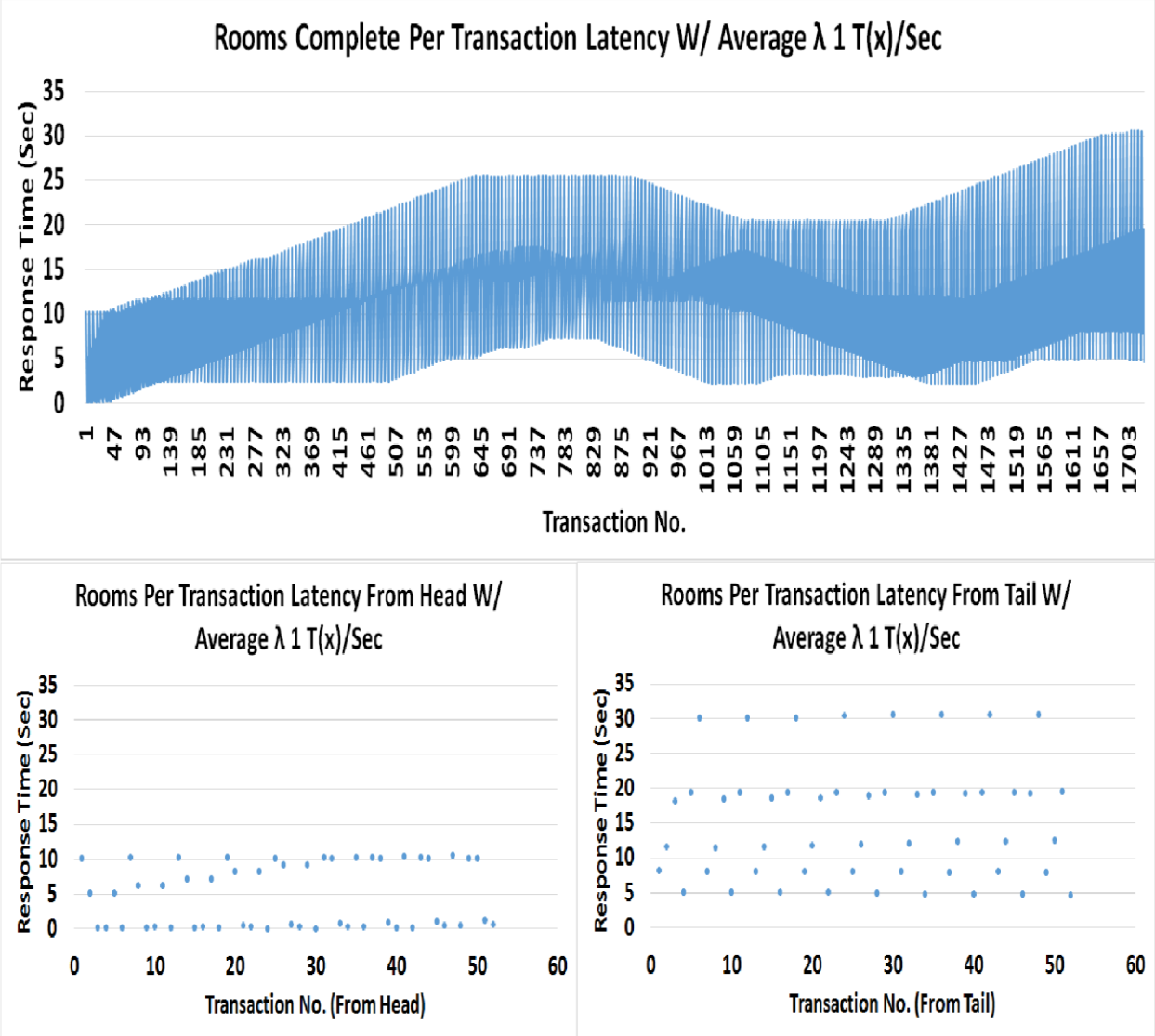

Figure 7.14 Per Transaction Latency Plots for the Rooms Topology W/ Average λ 1 T(x)/Sec

Table 7.31 Raw Sensor Values corresponding to first 4 T(x)s for the Rooms Topology W/ Average λ 1 T(x)/Sec

Rooms Raw Sensor Data Corresponding to first 4 T(x)s W/ Average λ 1 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'00:00:00'	'CP'	'1'	'103'	'CO2'	'362.49']	1495332883	1495332883
['9/27/2016'	'00:00:00'	'CP'	'1'	'106'	'RT'	'72.50']	1495332883	1495332888
['9/27/2016'	'00:00:00'	'CP'	'1'	'103'	'AV'	'0.00']	1495332883	1495332883
['9/27/2016'	'00:00:00'	'CP'	'2'	'206D'	'AV'	'0.00']	1495332883	1495332888
['9/27/2016'	'00:05:00'	'CP'	'1'	'103'	'CO2'	'362.49']	1495332884	1495332884
['9/27/2016'	'00:05:00'	'CP'	'1'	'103'	'AV'	'0.00']	1495332884	1495332884
['9/27/2016'	'00:05:00'	'CP'	'1'	'106'	'RT'	'72.50']	1495332884	1495332888
['9/27/2016'	'00:05:00'	'CP'	'2'	'206D'	'AV'	'0.00']	1495332884	1495332888
['9/27/2016'	'00:10:00'	'CP'	'1'	'103'	'CO2'	'362.49']	1495332885	1495332885
['9/27/2016'	'00:10:00'	'CP'	'1'	'103'	'AV'	'0.00']	1495332885	1495332885
['9/27/2016'	'00:10:00'	'CP'	'1'	'106'	'RT'	'72.50']	1495332885	1495332888
['9/27/2016'	'00:10:00'	'CP'	'2'	'206D'	'AV'	'0.00']	1495332885	1495332888
['9/27/2016'	'00:15:00'	'CP'	'1'	'103'	'CO2'	'362.49']	1495332886	1495332886
['9/27/2016'	'00:15:00'	'CP'	'1'	'103'	'AV'	'0.00']	1495332886	1495332886
['9/27/2016'	'00:15:00'	'CP'	'1'	'106'	'RT'	'72.50']	1495332886	1495332888
['9/27/2016'	'00:15:00'	'CP'	'2'	'206D'	'AV'	'0.00']	1495332886	1495332888

Table 7.32 Raw Sensor Values corresponding to T(x)s 1646-1649 for the Rooms Topology W/ Average λ 1 T(x)/Sec

Rooms Raw Sensor Data Corresponding T(x)s 1646 through 1649 W/ Average λ 1 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'22:40:00'	'CP'	'1'	'108'	'AV'	'0.00']	1495333182	1495333182
['9/27/2016'	'22:55:00'	'CP'	'1'	'106'	'AV'	'4.00']	1495333183	1495333183
['9/27/2016'	'22:30:00'	'CP'	'1'	'103'	'RT'	'71.75']	1495333183	1495333183
['9/27/2016'	'22:35:00'	'CP'	'1'	'108'	'RT'	'71.75']	1495333183	1495333183
['9/27/2016'	'22:55:00'	'CP'	'2'	'206H'	'CO2'	'399.77']	1495333183	1495333183
['9/27/2016'	'22:30:00'	'CP'	'1'	'106'	'CO2'	'366.07']	1495333183	1495333183
['9/27/2016'	'22:55:00'	'CP'	'2'	'206H'	'AT'	'72.00']	1495333183	1495333183
['9/27/2016'	'23:40:00'	'CP'	'1'	'106'	'RT'	'72.25']	1495333183	1495333183
['9/27/2016'	'23:10:00'	'CP'	'1'	'106'	'AT'	'71.00']	1495333183	1495333183
['9/27/2016'	'22:55:00'	'CP'	'2'	'206C'	'CO2'	'382.56']	1495333183	1495333183
['9/27/2016'	'22:45:00'	'CP'	'2'	'206H'	'AV'	'0.00']	1495333183	1495333183
['9/27/2016'	'23:00:00'	'CP'	'2'	'206D'	'AT'	'72.00']	1495333183	1495333183
['9/27/2016'	'22:45:00'	'CP'	'2'	'206H'	'RT'	'74.00']	1495333183	1495333183
['9/27/2016'	'23:15:00'	'CP'	'1'	'103'	'CO2'	'363.21']	1495333183	1495333183
['9/27/2016'	'23:10:00'	'CP'	'1'	'103'	'AV'	'0.00']	1495333183	1495333183
['9/27/2016'	'22:35:00'	'CP'	'1'	'108'	'CO2'	'391.88']	1495333183	1495333183

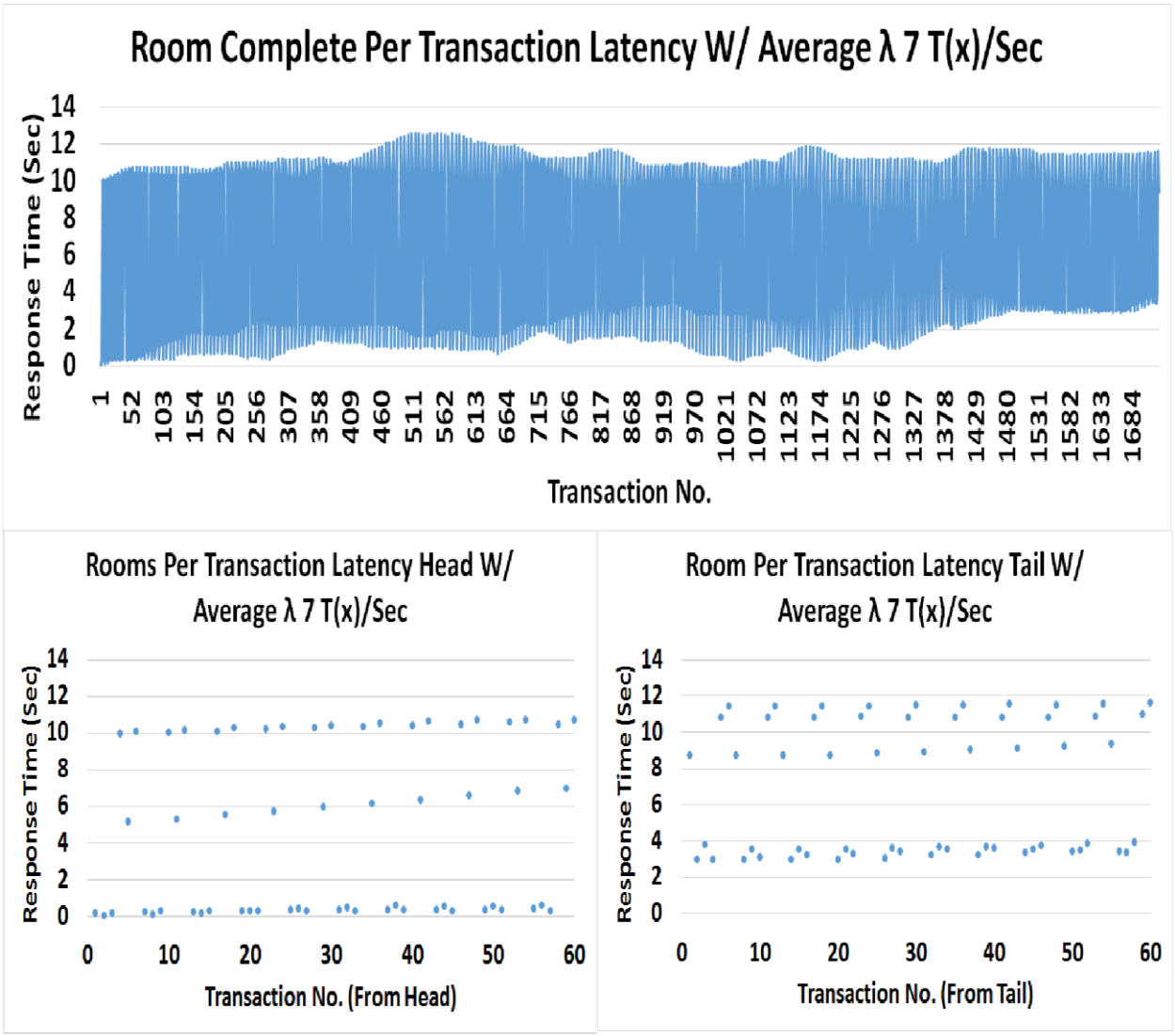

Figure 7.15 Per Transaction Latency Plots for the Rooms Topology W/ Average λ 7 T(x)/Sec

Table 7.33 Raw Sensor Values corresponding to first 4 T(x)s for the Rooms Topology W/ Average λ 7 T(x)/Sec

Rooms Raw Sensor Data Corresponding to first 4 T(x)s W/ Average λ 7 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'00:00:00'	'CP'	'2'	'206H'	'AV'	'20.00']	1495298806	1495298806
['9/27/2016'	'00:00:00'	'CP'	'2'	'206D'	'AV'	'0.00']	1495298806	1495298811
['9/27/2016'	'00:00:00'	'CP'	'2'	'206C'	'RT'	'74.00']	1495298806	1495298806
['9/27/2016'	'00:05:00'	'CP'	'2'	'206H'	'AV'	'16.00']	1495298806	1495298806
['9/27/2016'	'00:05:00'	'CP'	'2'	'206D'	'AV'	'0.00']	1495298806	1495298811
['9/27/2016'	'00:05:00'	'CP'	'2'	'206C'	'RT'	'74.00']	1495298806	1495298806
['9/27/2016'	'00:10:00'	'CP'	'2'	'206H'	'AV'	'12.00']	1495298806	1495298806
['9/27/2016'	'00:10:00'	'CP'	'2'	'206D'	'AV'	'0.00']	1495298806	1495298811
['9/27/2016'	'00:10:00'	'CP'	'2'	'206C'	'RT'	'74.00']	1495298806	1495298806
['9/27/2016'	'00:15:00'	'CP'	'2'	'206H'	'AV'	'24.00']	1495298806	1495298806
['9/27/2016'	'00:15:00'	'CP'	'2'	'206D'	'AV'	'0.00']	1495298806	1495298811
['9/27/2016'	'00:15:00'	'CP'	'2'	'206C'	'RT'	'74.00']	1495298806	1495298806
['9/27/2016'	'00:20:00'	'CP'	'2'	'206H'	'AV'	'8.00']	1495298806	1495298806
['9/27/2016'	'00:20:00'	'CP'	'2'	'206D'	'AV'	'0.00']	1495298806	1495298811
['9/27/2016'	'00:20:00'	'CP'	'2'	'206C'	'RT'	'74.00']	1495298806	1495298806
['9/27/2016'	'00:25:00'	'CP'	'2'	'206H'	'AV'	'12.00']	1495298806	1495298806

Table 7.34 Raw Sensor Values corresponding to T(x)s 562-565 for the Rooms Topology W/ Average λ 7 T(x)/Sec

Rooms Raw Sensor Data Corresponding T(x)s 562 through 565 W/ Average λ 7 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'07:55:00'	'CP'	'2'	'206H'	'CO2'	'410.52']	1495298824	1495298824
['9/27/2016'	'10:10:00'	'CP'	'2'	'206C'	'RT'	'72.50']	1495298824	1495298824
['9/27/2016'	'07:05:00'	'CP'	'1'	'108'	'CO2'	'403.36']	1495298824	1495298824
['9/27/2016'	'07:15:00'	'CP'	'1'	'103'	'CO2'	'376.83']	1495298824	1495298824
['9/27/2016'	'07:30:00'	'CP'	'1'	'106'	'AV'	'612.00']	1495298824	1495298824
['9/27/2016'	'07:35:00'	'CP'	'2'	'206D'	'AT'	'71.50']	1495298824	1495298824
['9/27/2016'	'07:25:00'	'CP'	'1'	'108'	'RT'	'72.75']	1495298824	1495298824
['9/27/2016'	'07:25:00'	'CP'	'2'	'206C'	'AT'	'71.00']	1495298824	1495298824
['9/27/2016'	'10:20:00'	'CP'	'2'	'206D'	'AV'	'576.00']	1495298824	1495298824
['9/27/2016'	'10:20:00'	'CP'	'2'	'206H'	'AV'	'300.00']	1495298824	1495298824
['9/27/2016'	'07:05:00'	'CP'	'2'	'206D'	'CO2'	'414.83']	1495298824	1495298824
['9/27/2016'	'07:30:00'	'CP'	'1'	'103'	'AT'	'72.00']	1495298824	1495298824
['9/27/2016'	'08:00:00'	'CP'	'2'	'206H'	'CO2'	'411.96']	1495298824	1495298824
['9/27/2016'	'07:35:00'	'CP'	'1'	'106'	'AT'	'69.50']	1495298824	1495298824
['9/27/2016'	'07:35:00'	'CP'	'1'	'108'	'AT'	'71.00']	1495298824	1495298824
['9/27/2016'	'07:45:00'	'CP'	'1'	'106'	'RT'	'72.50']	1495298824	1495298824

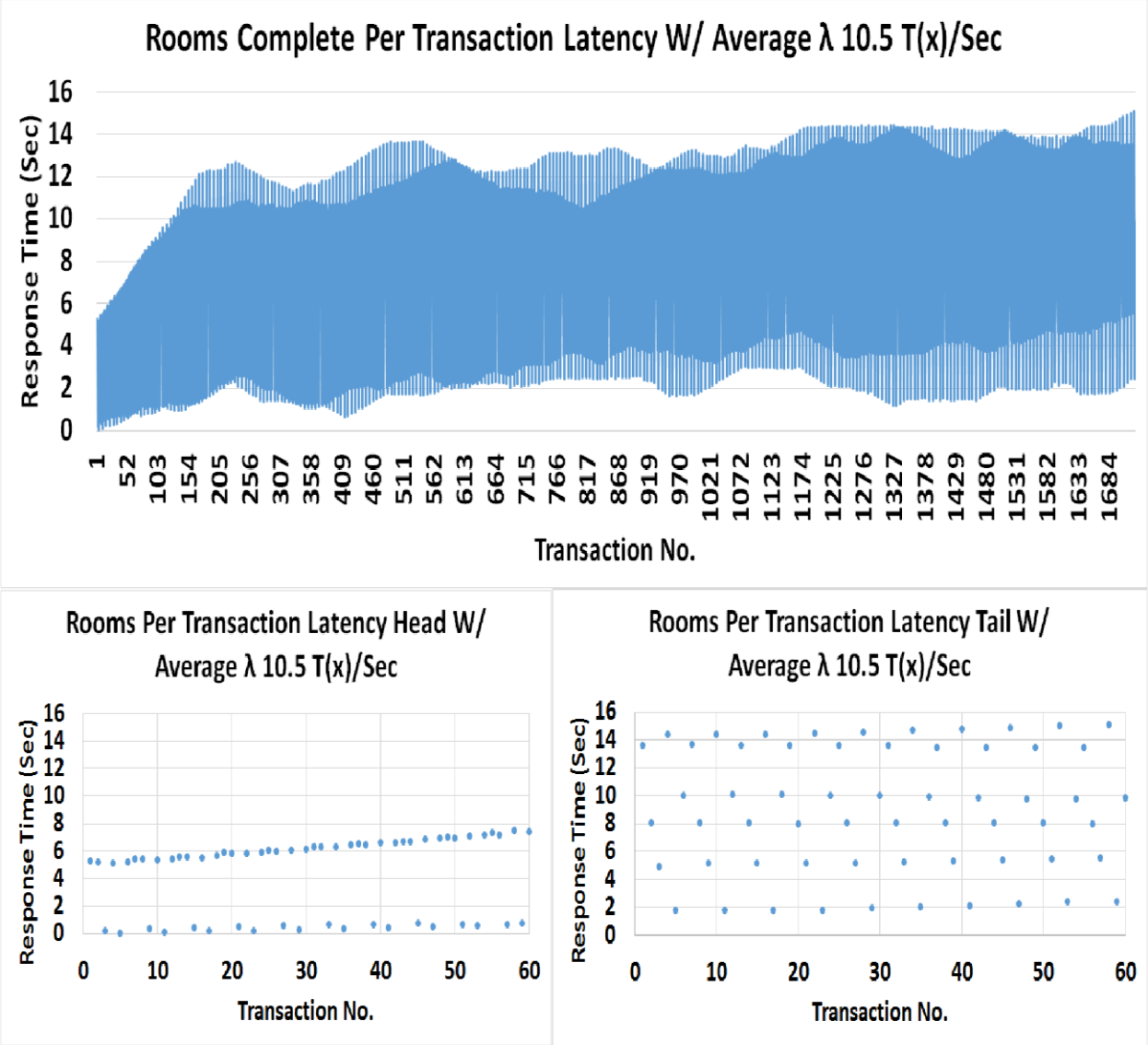

Figure 7.16 Per Transaction Latency Plots for the Rooms Topology W/ Average λ 10.5 T(x)/Sec

Table 7.35 Raw Sensor Values corresponding to first 4 T(x)s for the Rooms Topology W/ Average λ 10.5 T(x)/Sec

Rooms Raw Sensor Data Corresponding to first 4 T(x)s W/ Average λ 10.5 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'00:00:00'	'CP'	'1'	'103'	'AT'	'72.50']	1495301073	1495301078
['9/27/2016'	'00:00:00'	'CP'	'1'	'106'	'AT'	'71.50']	1495301073	1495301078
['9/27/2016'	'00:00:00'	'CP'	'1'	'106'	'RT'	'72.50']	1495301073	1495301078
['9/27/2016'	'00:00:00'	'CP'	'2'	'206C'	'CO2'	'401.92']	1495301073	1495301078
['9/27/2016'	'00:00:00'	'CP'	'2'	'206C'	'AT'	'72.00']	1495301073	1495301078
['9/27/2016'	'00:00:00'	'CP'	'2'	'206C'	'RT'	'74.00']	1495301073	1495301078
['9/27/2016'	'00:00:00'	'CP'	'2'	'206H'	'AV'	'20.00']	1495301073	1495301078
['9/27/2016'	'00:05:00'	'CP'	'1'	'103'	'AT'	'72.50']	1495301073	1495301078
['9/27/2016'	'00:05:00'	'CP'	'1'	'106'	'AT'	'71.50']	1495301073	1495301078
['9/27/2016'	'00:05:00'	'CP'	'1'	'106'	'RT'	'72.50']	1495301073	1495301078
['9/27/2016'	'00:05:00'	'CP'	'2'	'206C'	'AT'	'72.00']	1495301073	1495301078
['9/27/2016'	'00:05:00'	'CP'	'2'	'206C'	'CO2'	'395.47']	1495301073	1495301078
['9/27/2016'	'00:05:00'	'CP'	'2'	'206C'	'RT'	'74.00']	1495301073	1495301078
['9/27/2016'	'00:05:00'	'CP'	'2'	'206H'	'AV'	'16.00']	1495301073	1495301078
['9/27/2016'	'00:10:00'	'CP'	'1'	'103'	'AT'	'72.50']	1495301073	1495301078
['9/27/2016'	'00:10:00'	'CP'	'1'	'106'	'AT'	'71.50']	1495301073	1495301078

Table 7.36 Raw Sensor Values corresponding to T(x)s 1225-1228 for the Rooms Topology W/ Average λ 10.5 T(x)/Sec

Rooms Raw Sensor Data Corresponding T(x)s 1225 through 1228 W/ Average λ 10.5 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'15:55:00'	'CP'	'2'	'206H'	'AT'	'71.50']	1495301116	1495301116
['9/27/2016'	'16:15:00'	'CP'	'2'	'206D'	'RT'	'74.00']	1495301116	1495301116
['9/27/2016'	'15:50:00'	'CP'	'2'	'206D'	'AT'	'71.50']	1495301116	1495301116
['9/27/2016'	'16:20:00'	'CP'	'1'	'108'	'RT'	'72.25']	1495301116	1495301116
['9/27/2016'	'16:40:00'	'CP'	'1'	'103'	'RT'	'72.25']	1495301116	1495301116
['9/27/2016'	'19:10:00'	'CP'	'1'	'103'	'AT'	'68.50']	1495301116	1495301116
['9/27/2016'	'18:25:00'	'CP'	'1'	'106'	'AT'	'69.50']	1495301116	1495301116
['9/27/2016'	'18:05:00'	'CP'	'2'	'206H'	'AV'	'300.00']	1495301116	1495301116
['9/27/2016'	'16:45:00'	'CP'	'1'	'106'	'AV'	'448.00']	1495301116	1495301116
['9/27/2016'	'15:50:00'	'CP'	'2'	'206C'	'AV'	'464.00']	1495301116	1495301116
['9/27/2016'	'16:25:00'	'CP'	'1'	'103'	'CO2'	'401.20']	1495301116	1495301116
['9/27/2016'	'16:45:00'	'CP'	'1'	'108'	'AV'	'440.00']	1495301116	1495301116
['9/27/2016'	'16:25:00'	'CP'	'2'	'206D'	'CO2'	'395.47']	1495301116	1495301116
['9/27/2016'	'16:35:00'	'CP'	'1'	'103'	'AV'	'576.00']	1495301116	1495301116
['9/27/2016'	'16:05:00'	'CP'	'2'	'206H'	'RT'	'74.00']	1495301116	1495301116
['9/27/2016'	'16:35:00'	'CP'	'1'	'108'	'CO2'	'394.03']	1495301116	1495301116

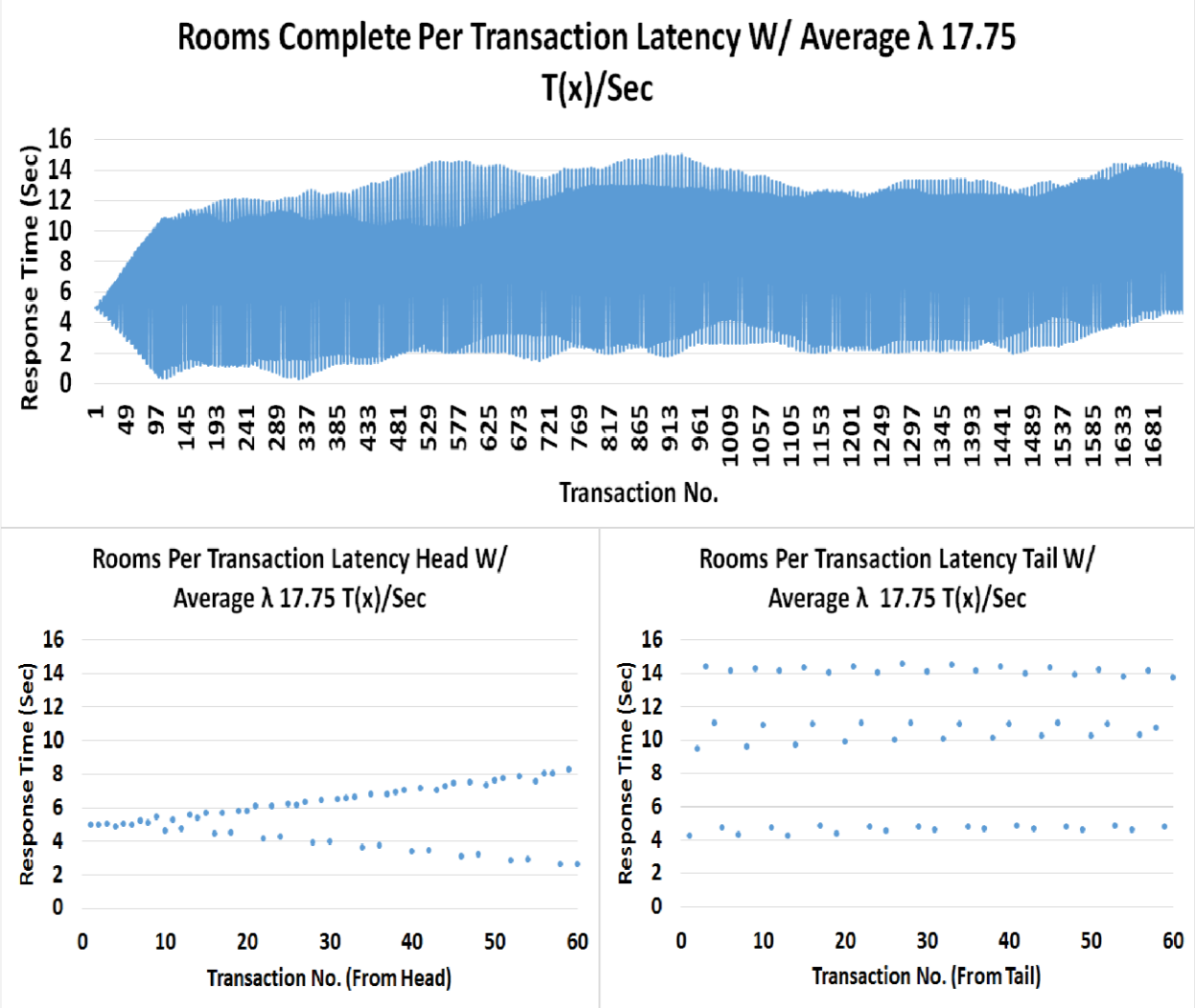

Figure 7.17 Per Transaction Latency Plots for the Rooms Topology W/ Average λ 17.75 T(x)/Sec

Table 7.37 Raw Sensor Values corresponding to first 4 T(x)s for the Rooms Topology W/ Average λ 17.75 T(x)/Sec

Rooms Raw Sensor Data Corresponding to first 4 T(x)s W/ Average λ 17.75 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'00:00:00'	'CP'	'1'	'108'	'AV'	'16.00']	1495329867	1495329872
['9/27/2016'	'00:00:00'	'CP'	'1'	'103'	'AT'	'72.50']	1495329867	1495329872
['9/27/2016'	'00:00:00'	'CP'	'1'	'103'	'AV'	'0.00']	1495329867	1495329872
['9/27/2016'	'00:00:00'	'CP'	'1'	'103'	'RT'	'71.75']	1495329867	1495329872
['9/27/2016'	'00:00:00'	'CP'	'1'	'106'	'AT'	'71.50']	1495329867	1495329872
['9/27/2016'	'00:00:00'	'CP'	'1'	'108'	'AT'	'72.00']	1495329867	1495329872
['9/27/2016'	'00:00:00'	'CP'	'2'	'206H'	'AV'	'20.00']	1495329867	1495329872
['9/27/2016'	'00:00:00'	'CP'	'1'	'106'	'RT'	'72.50']	1495329867	1495329872
['9/27/2016'	'00:00:00'	'CP'	'2'	'206D'	'RT'	'74.00']	1495329867	1495329872
['9/27/2016'	'00:00:00'	'CP'	'2'	'206H'	'CO2'	'396.90']	1495329867	1495329872
['9/27/2016'	'00:00:00'	'CP'	'2'	'206H'	'RT'	'74.00']	1495329867	1495329872
['9/27/2016'	'00:00:00'	'CP'	'2'	'206C'	'AT'	'72.00']	1495329867	1495329872
['9/27/2016'	'00:00:00'	'CP'	'2'	'206C'	'AV'	'12.00']	1495329867	1495329872
['9/27/2016'	'00:00:00'	'CP'	'2'	'206C'	'CO2'	'401.92']	1495329867	1495329872
['9/27/2016'	'00:00:00'	'CP'	'2'	'206C'	'RT'	'74.00']	1495329867	1495329872
['9/27/2016'	'00:00:00'	'CP'	'2'	'206D'	'AT'	'72.00']	1495329867	1495329872

Table 7.38 Raw Sensor Values corresponding to T(x)s 913-916 for the Rooms Topology W/ Average λ 17.75 T(x)/Sec

Rooms Raw Sensor Data Corresponding T(x)s 913 through 916 W/ Average λ 17.75 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'12:55:00'	'CP'	'2'	'206C'	'RT'	'71.75']	1495329913	1495329913
['9/27/2016'	'11:25:00'	'CP'	'1'	'106'	'CO2'	'378.98']	1495329913	1495329913
['9/27/2016'	'13:05:00'	'CP'	'2'	'206D'	'AT'	'71.50']	1495329913	1495329913
['9/27/2016'	'12:45:00'	'CP'	'2'	'206H'	'RT'	'74.00']	1495329913	1495329913
['9/27/2016'	'11:30:00'	'CP'	'2'	'206D'	'AV'	'384.00']	1495329913	1495329913
['9/27/2016'	'11:45:00'	'CP'	'1'	'108'	'RT'	'72.25']	1495329913	1495329913
['9/27/2016'	'12:00:00'	'CP'	'2'	'206D'	'CO2'	'405.51']	1495329913	1495329913
['9/27/2016'	'13:05:00'	'CP'	'2'	'206C'	'AT'	'71.50']	1495329913	1495329913
['9/27/2016'	'13:10:00'	'CP'	'2'	'206H'	'CO2'	'406.94']	1495329913	1495329913
['9/27/2016'	'13:05:00'	'CP'	'1'	'103'	'AT'	'69.00']	1495329913	1495329913
['9/27/2016'	'11:40:00'	'CP'	'1'	'108'	'CO2'	'398.34']	1495329913	1495329913
['9/27/2016'	'13:05:00'	'CP'	'1'	'108'	'AT'	'71.00']	1495329913	1495329913
['9/27/2016'	'12:50:00'	'CP'	'2'	'206H'	'AV'	'300.00']	1495329913	1495329913
['9/27/2016'	'13:05:00'	'CP'	'1'	'106'	'RT'	'70.25']	1495329913	1495329913
['9/27/2016'	'13:00:00'	'CP'	'1'	'103'	'AV'	'328.00']	1495329913	1495329913
['9/27/2016'	'13:10:00'	'CP'	'1'	'108'	'AV'	'436.00']	1495329913	1495329913

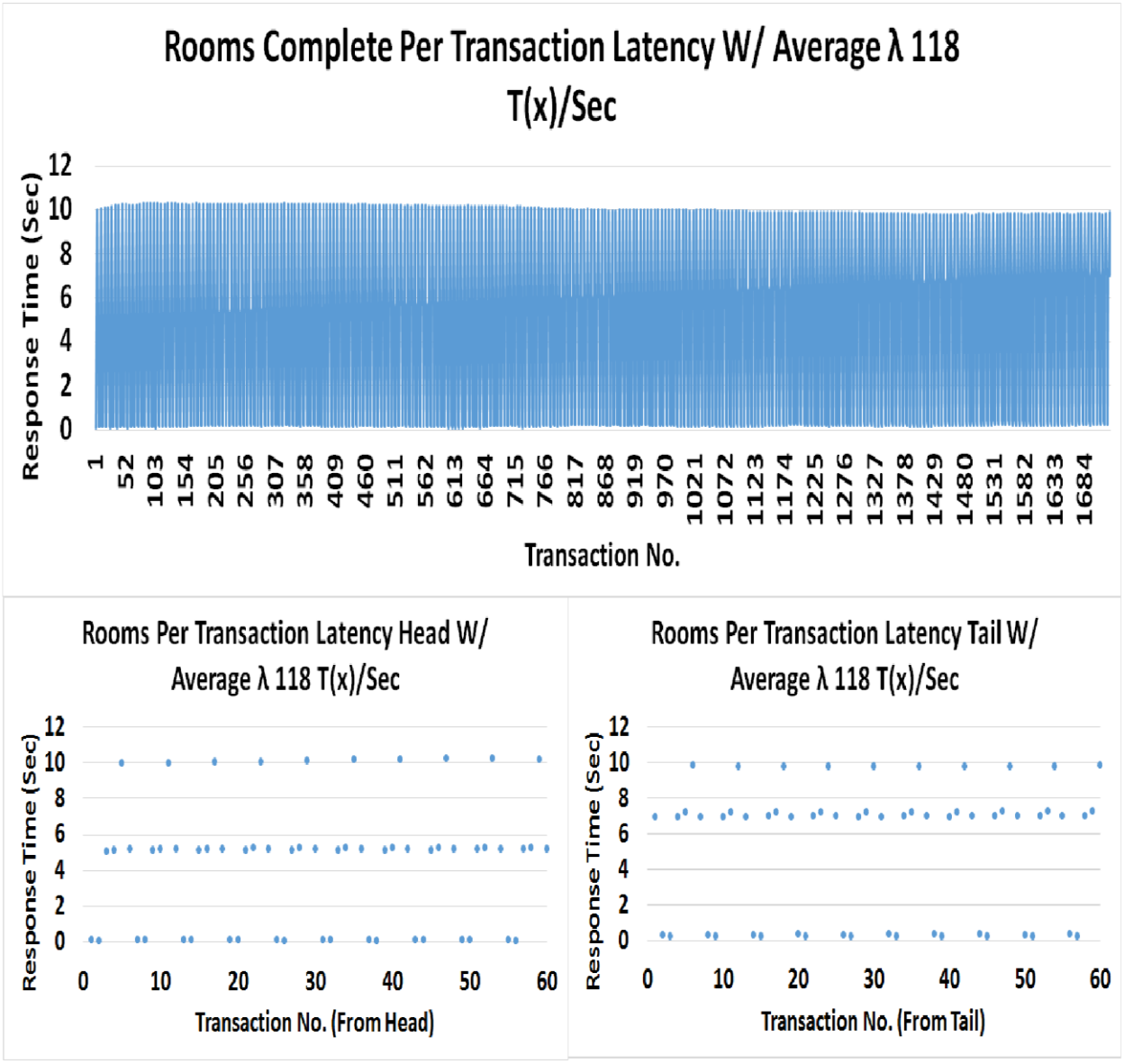

Figure 7.18 Per Transaction Latency Plots for the Rooms Topology W/ Average λ 118 T(x)/Sec

Table 7.39 Raw Sensor Values corresponding to first 4 T(x)s for the Rooms Topology W/ Average λ 118 T(x)/Sec

Rooms Raw Sensor Data Corresponding to first 4 T(x)s W/ Average λ 118 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'00:00:00'	'CP'	'2'	'206H'	'AT'	'72.00']	1495296901	1495296906
['9/27/2016'	'00:05:00'	'CP'	'2'	'206H'	'AT'	'72.00']	1495296901	1495296906
['9/27/2016'	'00:00:00'	'CP'	'1'	'108'	'AT'	'72.00']	1495296901	1495296906
['9/27/2016'	'00:10:00'	'CP'	'2'	'206H'	'AT'	'72.00']	1495296901	1495296906
['9/27/2016'	'00:00:00'	'CP'	'2'	'206C'	'CO2'	'401.92']	1495296901	1495296906
['9/27/2016'	'00:00:00'	'CP'	'2'	'206D'	'AV'	'0.00']	1495296901	1495296901
['9/27/2016'	'00:05:00'	'CP'	'1'	'108'	'AT'	'72.00']	1495296901	1495296906
['9/27/2016'	'00:15:00'	'CP'	'2'	'206H'	'AT'	'72.50']	1495296901	1495296906
['9/27/2016'	'00:05:00'	'CP'	'2'	'206C'	'CO2'	'395.47']	1495296901	1495296906
['9/27/2016'	'00:05:00'	'CP'	'2'	'206D'	'AV'	'0.00']	1495296901	1495296901
['9/27/2016'	'00:10:00'	'CP'	'1'	'108'	'AT'	'72.00']	1495296901	1495296906
['9/27/2016'	'00:15:00'	'CP'	'1'	'108'	'AT'	'72.00']	1495296901	1495296906
['9/27/2016'	'00:20:00'	'CP'	'1'	'108'	'AT'	'72.00']	1495296901	1495296906
['9/27/2016'	'00:20:00'	'CP'	'2'	'206H'	'AT'	'72.50']	1495296901	1495296906
['9/27/2016'	'00:10:00'	'CP'	'2'	'206C'	'CO2'	'388.30']	1495296901	1495296906
['9/27/2016'	'00:10:00'	'CP'	'2'	'206D'	'AV'	'0.00']	1495296901	1495296901

Table 7.40 Raw Sensor Values corresponding to T(x)s 1684-1687 for the Rooms Topology W/ Average λ 118 T(x)/Sec

Rooms Raw Sensor Data Corresponding T(x)s 1684 through 1687 W/ Average λ 118 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'23:25:00'	'CP'	'2'	'206C'	'AV'	'16.00']	1495296908	1495296908
['9/27/2016'	'23:25:00'	'CP'	'2'	'206H'	'CO2'	'395.47']	1495296908	1495296908
['9/27/2016'	'23:25:00'	'CP'	'2'	'206H'	'RT'	'74.00']	1495296908	1495296908
['9/27/2016'	'23:30:00'	'CP'	'2'	'206C'	'AV'	'16.00']	1495296908	1495296908
['9/27/2016'	'23:30:00'	'CP'	'2'	'206H'	'RT'	'74.00']	1495296908	1495296908
['9/27/2016'	'23:35:00'	'CP'	'1'	'106'	'CO2'	'363.92']	1495296908	1495296908
['9/27/2016'	'23:35:00'	'CP'	'1'	'108'	'RT'	'72.75']	1495296908	1495296908
['9/27/2016'	'23:35:00'	'CP'	'2'	'206C'	'AT'	'72.00']	1495296908	1495296908
['9/27/2016'	'23:40:00'	'CP'	'1'	'106'	'CO2'	'363.21']	1495296908	1495296908
['9/27/2016'	'23:40:00'	'CP'	'1'	'108'	'RT'	'72.75']	1495296908	1495296908
['9/27/2016'	'21:30:00'	'CP'	'2'	'206C'	'RT'	'73.00']	1495296908	1495296908
['9/27/2016'	'21:35:00'	'CP'	'2'	'206D'	'CO2'	'390.45']	1495296908	1495296908
['9/27/2016'	'21:40:00'	'CP'	'2'	'206D'	'CO2'	'389.02']	1495296908	1495296908
['9/27/2016'	'22:40:00'	'CP'	'1'	'103'	'AT'	'72.00']	1495296908	1495296908
['9/27/2016'	'22:45:00'	'CP'	'1'	'103'	'AT'	'72.00']	1495296908	1495296908
['9/27/2016'	'22:45:00'	'CP'	'2'	'206D'	'AT'	'72.00']	1495296908	1495296908

The graph from the per transaction latency for the sensors topology in Figure 7.4 shows the least amount of variance in response times. As we add the most number of computational servers to this M/D/24 model, the variance in response time drops from 26 seconds in the case of buildings topology to 11 seconds. This indicates that the sensors topology has the most sensor values arriving in order.

Figure 7.19 shows the graph from the per transaction latency for the sensors topology with an average arrival rate of 241.75 transactions per second shows the best results out of all topologies.

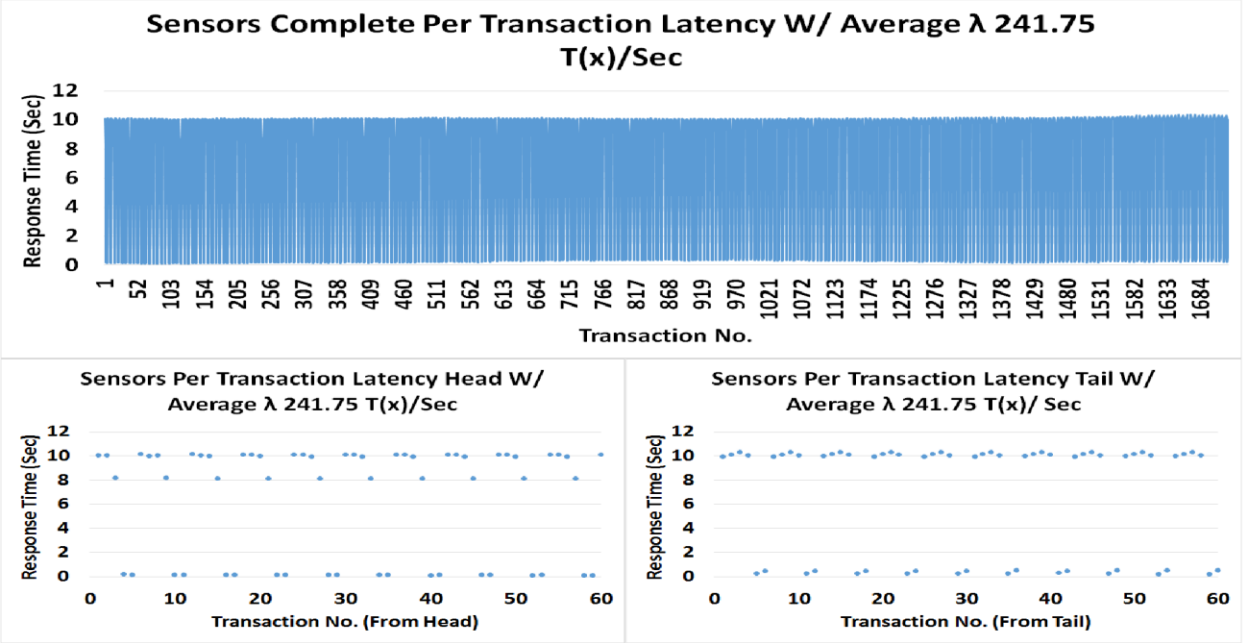

Figure 7.19 Per Transaction Latency Plots for the Sensors Topology W/ Average λ 241.75 T(x)/Sec

In the per transaction latency plots, we can see that in the worst case, the response time is approximately 10 seconds to service jobs. In the worst case, the sensors topology at higher utilization rates perform 3.5 times better than buildings and floors and 1.6 times better than the rooms topology.

In the case of the buildings and floors topologies, sensor values were routed in such a way that its out of orderedness was influenced by real world time, room, and sensor type. We were able to minimize the influence of room in the case of the rooms topology, where data was routed and pre-aggregated according to room before it was sent for aggregation. At its finest granularity, the sensors topology pre-aggregates by room and sensor type, only considering real world time to influence its out of orderedness. Since the aggregator knows that a transaction it needs 4 sensor values from the same room and real world time, it knows that once it receives one sensor value or a specific sensor type, it no longer needs to consider any values or that sensor type to complete the rest of its transaction. This is not the case for the rooms topology, where even if, for example, it finds a sensor value for CP-103-AT at real world time 00:00:00, it still runs the likelihood of running across more CP-103-AT sensors of different sensor types when it should only consider the remaining AV, CO2, and RT sensors. This is considered in the sensors topology.

From table 7.41, we can observe an immediate out of orderedness initiating before the real world time values for the first set in transactions even arrive.

Table 7.41 Raw Sensor Values corresponding to first 4 T(x)s for the Sensors Topology W/ Average λ 241.75 T(x)/Sec

Sensors Raw Sensor Data Corresponding to first 4 T(x)s W/ Average λ 241.75 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'00:00:00'	'CP'	'1'	'108'	'CO2'	'389.73']	1495336165	1495336167
['9/27/2016'	'00:00:00'	'CP'	'1'	'108'	'AT'	'72.00']	1495336165	1495336167
['9/27/2016'	'00:00:00'	'CP'	'2'	'206H'	'AV'	'20.00']	1495336165	1495336167
['9/27/2016'	'00:05:00'	'CP'	'2'	'206H'	'AV'	'16.00']	1495336165	1495336167
['9/27/2016'	'00:00:00'	'CP'	'1'	'103'	'AV'	'0.00']	1495336165	1495336165
['9/27/2016'	'00:00:00'	'CP'	'1'	'106'	'AT'	'71.50']	1495336165	1495336165
['9/27/2016'	'00:00:00'	'CP'	'2'	'206H'	'CO2'	'396.90']	1495336165	1495336165
['9/27/2016'	'00:05:00'	'CP'	'1'	'108'	'AT'	'72.00']	1495336165	1495336167
['9/27/2016'	'00:05:00'	'CP'	'1'	'108'	'CO2'	'391.17']	1495336165	1495336167
['9/27/2016'	'00:10:00'	'CP'	'2'	'206H'	'AV'	'12.00']	1495336165	1495336167
['9/27/2016'	'00:15:00'	'CP'	'2'	'206H'	'AV'	'24.00']	1495336165	1495336167
['9/27/2016'	'00:05:00'	'CP'	'1'	'103'	'AV'	'0.00']	1495336165	1495336165
['9/27/2016'	'00:10:00'	'CP'	'1'	'103'	'AV'	'0.00']	1495336165	1495336165
['9/27/2016'	'00:10:00'	'CP'	'1'	'108'	'AT'	'72.00']	1495336165	1495336167
['9/27/2016'	'00:15:00'	'CP'	'1'	'103'	'AV'	'0.00']	1495336165	1495336165
['9/27/2016'	'00:20:00'	'CP'	'2'	'206H'	'AV'	'8.00']	1495336165	1495336167

If you focus on the “Time” column, you can clearly see how according to real world time, the

sensor values are mostly all arriving out of order, with at most an out of orderedness variance in real world time by 25 minutes. While according to the log, there is out of orderedness that occurs on the real world time, room, and sensor level, what is not depicted in this log is that these are the values of 24 consumer threads running in parallel. So in the early stages of sensor values arriving, aggregation is significant impacted by out of orderedness, being influenced by real world time, room, and sensor type.

Table 7.42 Raw Sensor Values corresponding to T(x)s 1684-1687 for the Sensors Topology W/ Average λ 241.75 T(x)/Sec

Sensors Raw Sensor Data Corresponding T(x)s 1684 through 1687 W/ Average λ 241.75 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'21:20:00'	'CP'	'2'	'206D'	'CO2'	'390.45']	1495336173	1495336173
['9/27/2016'	'21:55:00'	'CP'	'1'	'106'	'CO2'	'369.66']	1495336173	1495336173
['9/27/2016'	'22:00:00'	'CP'	'1'	'106'	'CO2'	'369.66']	1495336173	1495336173
['9/27/2016'	'22:30:00'	'CP'	'2'	'206C'	'AT'	'72.00']	1495336173	1495336173
['9/27/2016'	'22:35:00'	'CP'	'1'	'103'	'AT'	'72.50']	1495336173	1495336173
['9/27/2016'	'22:40:00'	'CP'	'1'	'103'	'AT'	'72.00']	1495336173	1495336173
['9/27/2016'	'23:10:00'	'CP'	'1'	'108'	'AV'	'16.00']	1495336173	1495336173
['9/27/2016'	'23:15:00'	'CP'	'2'	'206C'	'RT'	'73.25']	1495336173	1495336173
['9/27/2016'	'23:20:00'	'CP'	'2'	'206C'	'CO2'	'382.56']	1495336173	1495336173
['9/27/2016'	'23:25:00'	'CP'	'2'	'206C'	'CO2'	'381.13']	1495336173	1495336173
['9/27/2016'	'23:45:00'	'CP'	'1'	'108'	'RT'	'72.75']	1495336173	1495336173
['9/27/2016'	'23:50:00'	'CP'	'1'	'103'	'CO2'	'360.34']	1495336173	1495336173
['9/27/2016'	'23:50:00'	'CP'	'1'	'108'	'RT'	'72.75']	1495336173	1495336173
['9/27/2016'	'23:50:00'	'CP'	'2'	'206D'	'AV'	'0.00']	1495336173	1495336173
['9/27/2016'	'23:50:00'	'CP'	'2'	'206H'	'AT'	'72.00']	1495336173	1495336173
['9/27/2016'	'23:55:00'	'CP'	'2'	'206D'	'AV'	'0.00']	1495336173	1495336173

Table 7.42 shows that over time, the out of orderedness according to room and sensor is at a much lower risk, with most of those values arriving in order. Again, the biggest influence on out of orderedness in this topology is the real world time at which they arrive.

The following are the figures and tables for the remaining arrival rates for the sensors topology.

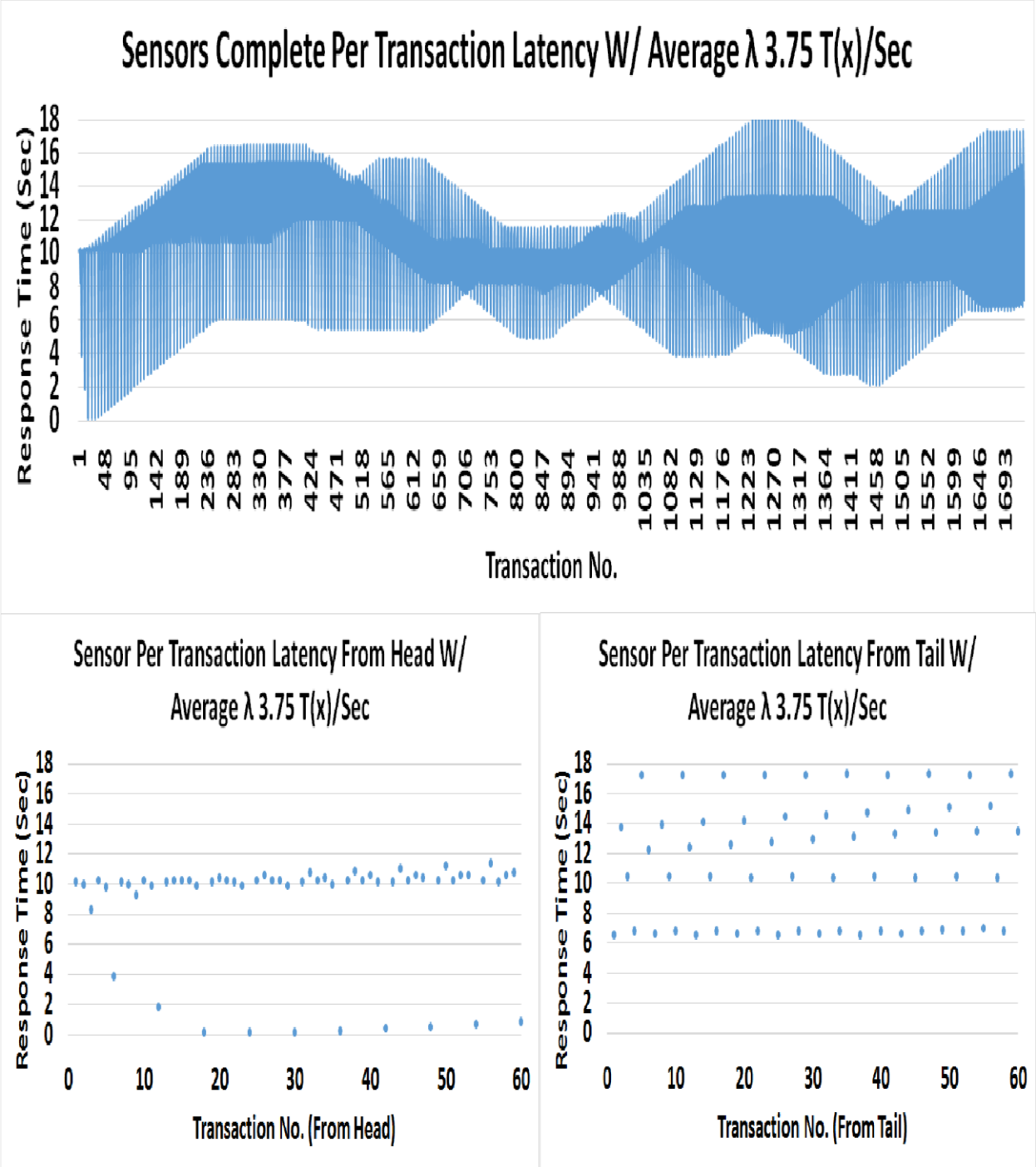

Figure 7.20 Per Transaction Latency Plots for the Sensors Topology W/ Average λ 3.75 T(x)/Sec

Table 7.43 Raw Sensor Values corresponding to first 4 T(x)s for the Sensors Topology W/ Average λ 3.75 T(x)/Sec

Sensors Raw Sensor Data Corresponding to first 4 T(x)s W/ Average λ 3.75 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'00:00:00'	'CP'	'1'	'108'	'CO2'	'389.73']	1495381705	1495381707
['9/27/2016'	'00:00:00'	'CP'	'2'	'206C'	'AV'	'12.00']	1495381705	1495381705
['9/27/2016'	'00:00:00'	'CP'	'1'	'103'	'AV'	'0.00']	1495381705	1495381707
['9/27/2016'	'00:00:00'	'CP'	'1'	'106'	'AT'	'71.50']	1495381705	1495381707
['9/27/2016'	'00:00:00'	'CP'	'1'	'108'	'RT'	'73.25']	1495381705	1495381707
['9/27/2016'	'00:00:00'	'CP'	'1'	'103'	'CO2'	'362.49']	1495381705	1495381705
['9/27/2016'	'00:00:00'	'CP'	'1'	'106'	'AV'	'0.00']	1495381705	1495381705
['9/27/2016'	'00:00:00'	'CP'	'2'	'206D'	'CO2'	'399.77']	1495381705	1495381705
['9/27/2016'	'00:00:00'	'CP'	'2'	'206H'	'AT'	'72.00']	1495381705	1495381705
['9/27/2016'	'00:00:00'	'CP'	'2'	'206D'	'AT'	'72.00']	1495381705	1495381705
['9/27/2016'	'00:00:00'	'CP'	'2'	'206D'	'RT'	'74.00']	1495381705	1495381707
['9/27/2016'	'00:00:00'	'CP'	'2'	'206H'	'AV'	'20.00']	1495381705	1495381705
['9/27/2016'	'00:00:00'	'CP'	'2'	'206H'	'RT'	'74.00']	1495381705	1495381705
['9/27/2016'	'00:00:00'	'CP'	'2'	'206H'	'CO2'	'396.90']	1495381705	1495381707
['9/27/2016'	'00:05:00'	'CP'	'2'	'206C'	'AV'	'8.00']	1495381706	1495381706
['9/27/2016'	'00:05:00'	'CP'	'1'	'108'	'CO2'	'391.17']	1495381706	1495381707

Table 7.44 Raw Sensor Values corresponding to T(x)s 1726-1729 for the Sensors Topology W/ Average λ 3.75 T(x)/Sec

Sensors Raw Sensor Data Corresponding T(x)s 1726 through 1729 W/ Average λ 3.75 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'23:30:00'	'CP'	'2'	'206C'	'AT'	'72.00']	1495382017	1495382017
['9/27/2016'	'23:45:00'	'CP'	'2'	'206C'	'RT'	'73.25']	1495382017	1495382017
['9/27/2016'	'23:50:00'	'CP'	'2'	'206D'	'AV'	'0.00']	1495382018	1495382018
['9/27/2016'	'23:55:00'	'CP'	'2'	'206D'	'AT'	'72.00']	1495382018	1495382018
['9/27/2016'	'23:55:00'	'CP'	'1'	'108'	'AT'	'71.50']	1495382018	1495382018
['9/27/2016'	'23:45:00'	'CP'	'1'	'103'	'RT'	'72.25']	1495382018	1495382018
['9/27/2016'	'23:35:00'	'CP'	'2'	'206C'	'AT'	'72.00']	1495382018	1495382018
['9/27/2016'	'23:50:00'	'CP'	'2'	'206C'	'RT'	'73.25']	1495382018	1495382019
['9/27/2016'	'23:55:00'	'CP'	'2'	'206D'	'AV'	'0.00']	1495382019	1495382019
['9/27/2016'	'23:40:00'	'CP'	'2'	'206C'	'AT'	'72.00']	1495382019	1495382019
['9/27/2016'	'23:50:00'	'CP'	'1'	'103'	'RT'	'72.25']	1495382019	1495382020
['9/27/2016'	'23:55:00'	'CP'	'2'	'206C'	'RT'	'73.25']	1495382020	1495382020
['9/27/2016'	'23:45:00'	'CP'	'2'	'206C'	'AT'	'72.00']	1495382020	1495382020
['9/27/2016'	'23:55:00'	'CP'	'1'	'103'	'RT'	'72.25']	1495382021	1495382021
['9/27/2016'	'23:50:00'	'CP'	'2'	'206C'	'AT'	'72.00']	1495382021	1495382021
['9/27/2016'	'23:55:00'	'CP'	'2'	'206C'	'AT'	'72.00']	1495382022	1495382023

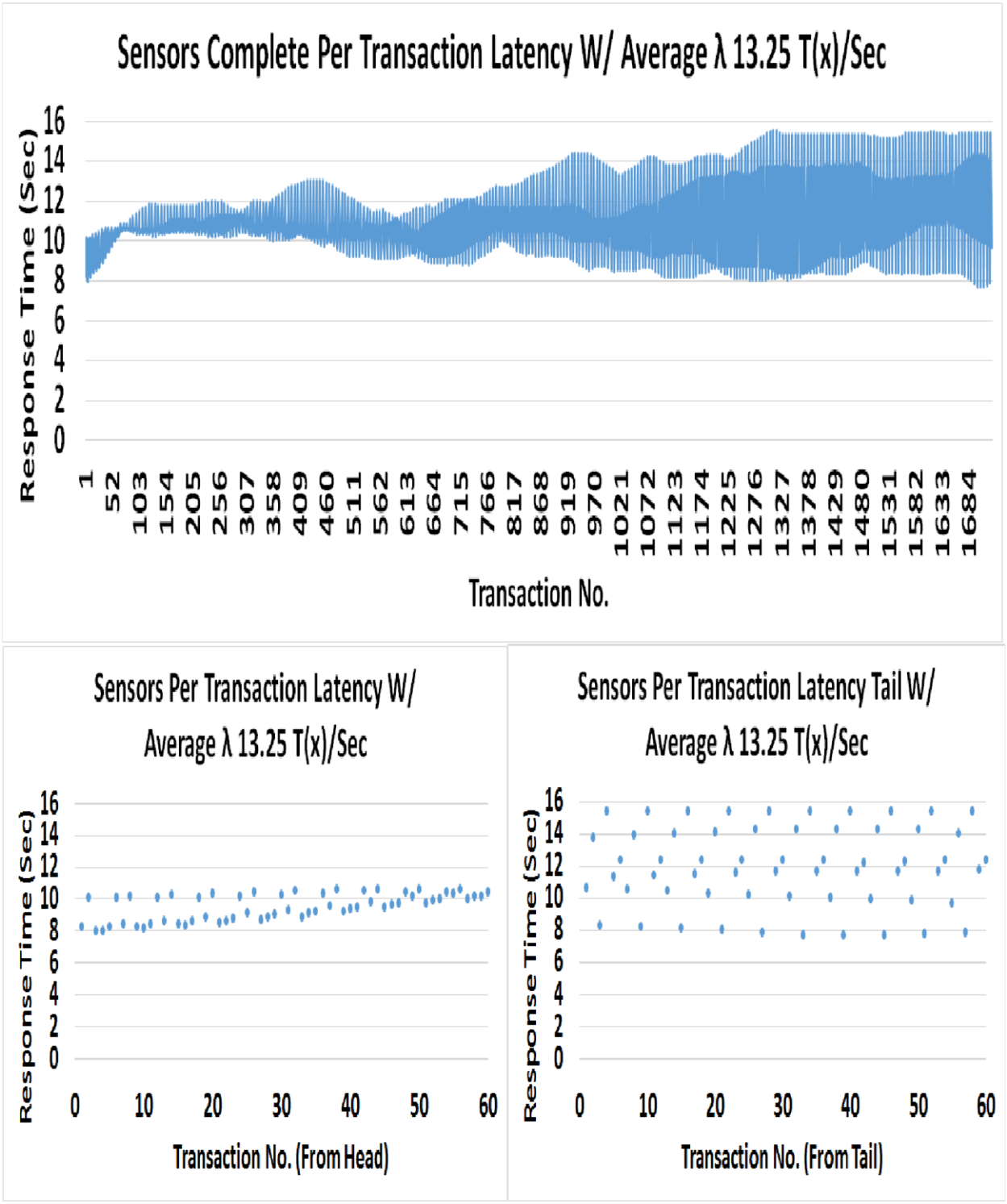

Figure 7.21 Per Transaction Latency Plots for the Sensors Topology W/ Average λ 13.25 T(x)/Sec

Table 7.45 Raw Sensor Values corresponding to first 4 T(x)s for the Sensors Topology W/ Average λ 13.25 T(x)/Sec

Sensors Raw Sensor Data Corresponding to first 4 T(x)s W/ Average λ 13.25 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'00:00:00'	'CP'	'1'	'103'	'RT'	'71.75']	1495343093	1495343095
['9/27/2016'	'00:00:00'	'CP'	'2'	'206D'	'AT'	'72.00']	1495343093	1495343096
['9/27/2016'	'00:00:00'	'CP'	'1'	'108'	'RT'	'73.25']	1495343093	1495343096
['9/27/2016'	'00:00:00'	'CP'	'1'	'108'	'AV'	'16.00']	1495343093	1495343095
['9/27/2016'	'00:00:00'	'CP'	'1'	'106'	'AV'	'0.00']	1495343093	1495343094
['9/27/2016'	'00:00:00'	'CP'	'2'	'206C'	'AT'	'72.00']	1495343094	1495343096
['9/27/2016'	'00:00:00'	'CP'	'2'	'206D'	'RT'	'74.00']	1495343094	1495343095
['9/27/2016'	'00:00:00'	'CP'	'2'	'206H'	'RT'	'74.00']	1495343094	1495343094
['9/27/2016'	'00:05:00'	'CP'	'2'	'206D'	'AT'	'72.00']	1495343094	1495343096
['9/27/2016'	'00:05:00'	'CP'	'1'	'103'	'RT'	'71.75']	1495343094	1495343095
['9/27/2016'	'00:05:00'	'CP'	'1'	'108'	'RT'	'73.25']	1495343094	1495343096
['9/27/2016'	'00:05:00'	'CP'	'1'	'108'	'AV'	'32.00']	1495343094	1495343095
['9/27/2016'	'00:05:00'	'CP'	'1'	'106'	'AV'	'0.00']	1495343094	1495343094
['9/27/2016'	'00:05:00'	'CP'	'2'	'206C'	'AT'	'72.00']	1495343094	1495343096
['9/27/2016'	'00:05:00'	'CP'	'2'	'206D'	'RT'	'74.00']	1495343094	1495343095
['9/27/2016'	'00:05:00'	'CP'	'2'	'206H'	'RT'	'74.00']	1495343094	1495343094

Table 7.46 Raw Sensor Values corresponding to T(x)s 1276-1279 for the Sensors Topology W/ Average λ 13.25 T(x)/Sec

Sensors Raw Sensor Data Corresponding T(x)s 1276 through 1279 W/ Average λ 13.25 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'18:25:00'	'CP'	'1'	'106'	'AV'	'468.00']	1495343138	1495343138
['9/27/2016'	'17:05:00'	'CP'	'2'	'206H'	'CO2'	'401.20']	1495343138	1495343138
['9/27/2016'	'17:05:00'	'CP'	'1'	'108'	'AT'	'70.00']	1495343138	1495343138
['9/27/2016'	'17:50:00'	'CP'	'1'	'106'	'CO2'	'368.22']	1495343138	1495343138
['9/27/2016'	'19:10:00'	'CP'	'2'	'206D'	'RT'	'74.00']	1495343138	1495343138
['9/27/2016'	'16:25:00'	'CP'	'1'	'103'	'CO2'	'401.20']	1495343138	1495343138
['9/27/2016'	'16:45:00'	'CP'	'1'	'106'	'RT'	'71.75']	1495343138	1495343138
['9/27/2016'	'16:45:00'	'CP'	'2'	'206C'	'CO2'	'375.39']	1495343138	1495343138
['9/27/2016'	'17:00:00'	'CP'	'2'	'206H'	'AV'	'296.00']	1495343138	1495343138
['9/27/2016'	'17:25:00'	'CP'	'1'	'103'	'AT'	'67.50']	1495343138	1495343138
['9/27/2016'	'18:40:00'	'CP'	'2'	'206H'	'RT'	'74.00']	1495343138	1495343138
['9/27/2016'	'17:10:00'	'CP'	'1'	'108'	'CO2'	'394.03']	1495343138	1495343138
['9/27/2016'	'19:00:00'	'CP'	'2'	'206C'	'AT'	'71.50']	1495343138	1495343138
['9/27/2016'	'19:00:00'	'CP'	'2'	'206D'	'AT'	'71.50']	1495343138	1495343138
['9/27/2016'	'16:35:00'	'CP'	'1'	'103'	'AV'	'576.00']	1495343138	1495343138
['9/27/2016'	'16:50:00'	'CP'	'2'	'206C'	'AV'	'456.00']	1495343138	1495343138

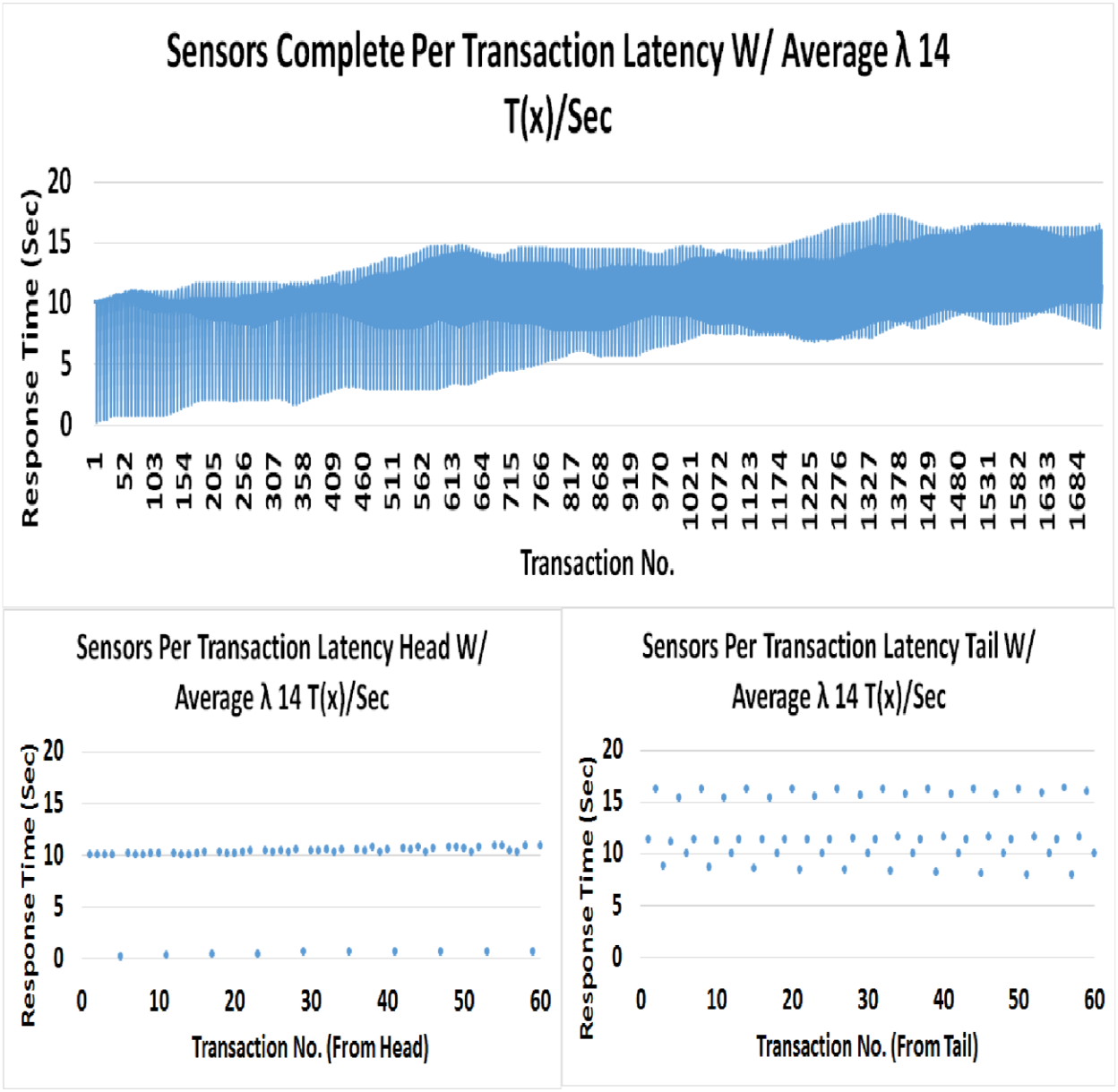

Figure 7.22 Per Transaction Latency Plots for the Sensors Topology W/ Average λ 14 T(x)/Sec

Table 7.47 Raw Sensor Values corresponding to first 4 T(x)s for the Sensors Topology W/ Average λ 14 T(x)/Sec

Sensors Raw Sensor Data Corresponding to first 4 T(x)s W/ Average λ 14 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'00:00:00'	'CP'	'2'	'206H'	'RT'	'74.00']	1495339930	1495339930
['9/27/2016'	'00:00:00'	'CP'	'1'	'106'	'AT'	'71.50']	1495339930	1495339930
['9/27/2016'	'00:00:00'	'CP'	'1'	'103'	'CO2'	'362.49']	1495339930	1495339930
['9/27/2016'	'00:00:00'	'CP'	'1'	'108'	'AV'	'16.00']	1495339930	1495339930
['9/27/2016'	'00:00:00'	'CP'	'2'	'206C'	'RT'	'74.00']	1495339930	1495339930
['9/27/2016'	'00:00:00'	'CP'	'2'	'206H'	'CO2'	'396.90']	1495339930	1495339930
['9/27/2016'	'00:05:00'	'CP'	'2'	'206H'	'RT'	'74.00']	1495339930	1495339930
['9/27/2016'	'00:05:00'	'CP'	'1'	'106'	'AT'	'71.50']	1495339930	1495339930
['9/27/2016'	'00:05:00'	'CP'	'1'	'108'	'AV'	'32.00']	1495339930	1495339930
['9/27/2016'	'00:05:00'	'CP'	'2'	'206C'	'RT'	'74.00']	1495339930	1495339930
['9/27/2016'	'00:05:00'	'CP'	'1'	'103'	'CO2'	'362.49']	1495339930	1495339930
['9/27/2016'	'00:05:00'	'CP'	'2'	'206H'	'CO2'	'397.62']	1495339930	1495339930
['9/27/2016'	'00:10:00'	'CP'	'2'	'206C'	'RT'	'74.00']	1495339930	1495339930
['9/27/2016'	'00:10:00'	'CP'	'2'	'206H'	'RT'	'74.00']	1495339930	1495339930
['9/27/2016'	'00:10:00'	'CP'	'1'	'108'	'AV'	'24.00']	1495339930	1495339930
['9/27/2016'	'00:10:00'	'CP'	'1'	'103'	'CO2'	'362.49']	1495339930	1495339930

Table 7.48 Raw Sensor Values corresponding to T(x)s 1378-1381 for the Sensors Topology W/ Average λ 14 T(x)/Sec

Sensors Raw Sensor Data Corresponding T(x)s 1378 through 1381 W/ Average λ 14 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'19:35:00'	'CP'	'2'	'206C'	'CO2'	'376.11']	1495339966	1495339966
['9/27/2016'	'17:50:00'	'CP'	'2'	'206C'	'AV'	'460.00']	1495339966	1495339966
['9/27/2016'	'20:25:00'	'CP'	'2'	'206H'	'CO2'	'403.36']	1495339966	1495339966
['9/27/2016'	'18:10:00'	'CP'	'1'	'106'	'CO2'	'368.94']	1495339966	1495339966
['9/27/2016'	'19:05:00'	'CP'	'2'	'206D'	'AT'	'71.00']	1495339966	1495339966
['9/27/2016'	'21:55:00'	'CP'	'1'	'103'	'CO2'	'374.68']	1495339966	1495339966
['9/27/2016'	'17:50:00'	'CP'	'2'	'206H'	'AV'	'292.00']	1495339966	1495339966
['9/27/2016'	'17:55:00'	'CP'	'1'	'103'	'AV'	'520.00']	1495339966	1495339966
['9/27/2016'	'18:05:00'	'CP'	'2'	'206D'	'CO2'	'389.73']	1495339966	1495339966
['9/27/2016'	'18:50:00'	'CP'	'2'	'206H'	'AT'	'71.50']	1495339966	1495339966
['9/27/2016'	'17:55:00'	'CP'	'1'	'108'	'CO2'	'396.90']	1495339966	1495339966
['9/27/2016'	'17:45:00'	'CP'	'1'	'103'	'AT'	'68.50']	1495339966	1495339966
['9/27/2016'	'18:30:00'	'CP'	'1'	'106'	'RT'	'72.25']	1495339966	1495339966
['9/27/2016'	'18:30:00'	'CP'	'1'	'108'	'RT'	'71.75']	1495339966	1495339966
['9/27/2016'	'20:35:00'	'CP'	'1'	'106'	'AT'	'69.50']	1495339966	1495339966
['9/27/2016'	'17:20:00'	'CP'	'2'	'206D'	'RT'	'74.00']	1495339966	1495339966

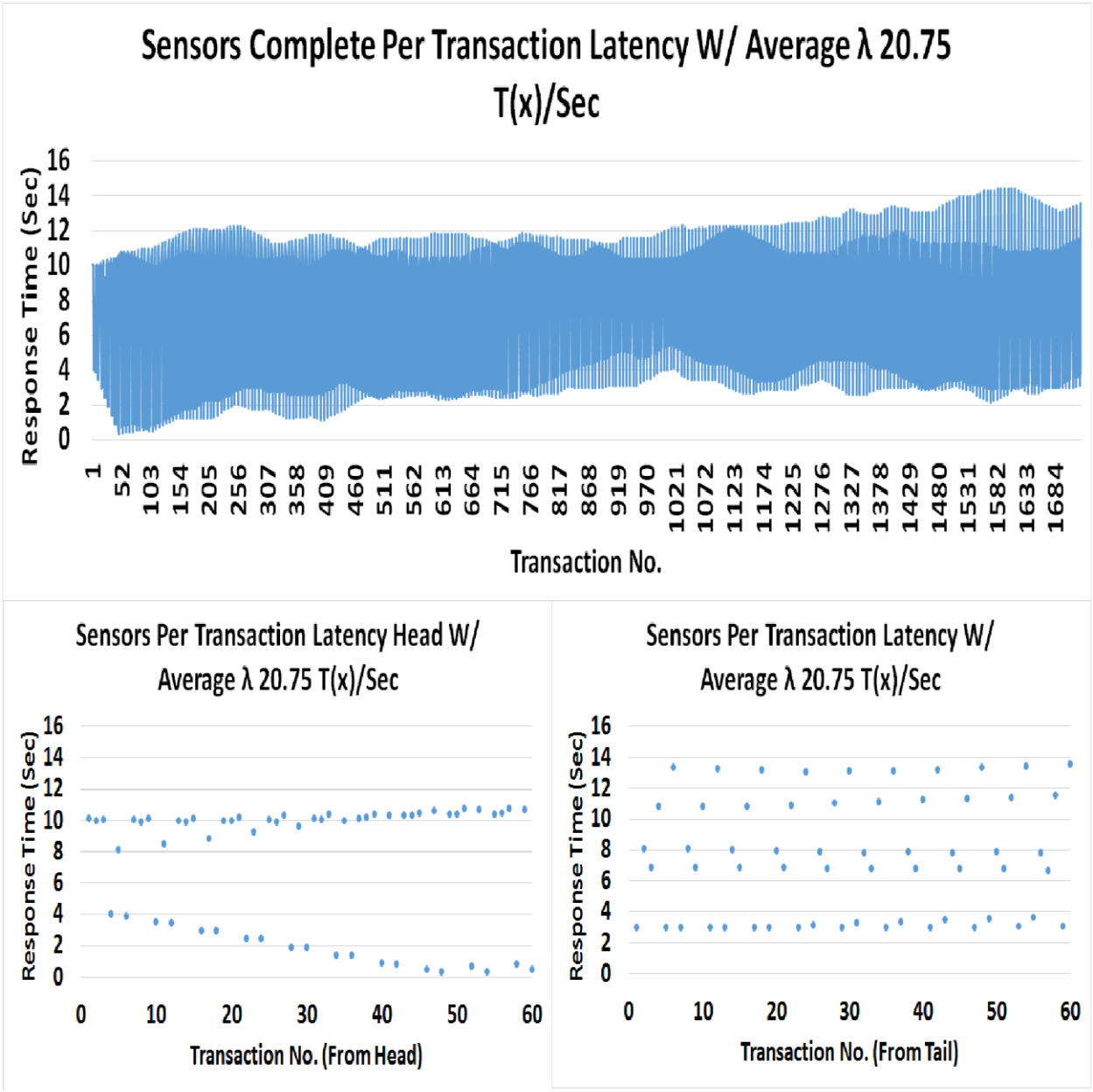

Figure 7.23 Per Transaction Latency Plots for the Sensors Topology W/ Average λ 20.75 T(x)/Sec

Table 7.49 Raw Sensor Values corresponding to first 4 T(x)s for the Sensors Topology W/ Average λ 20.75 T(x)/Sec

Sensors Raw Sensor Data Corresponding to first 4 T(x)s W/ Average λ 20.75 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'00:00:00'	'CP'	'1'	'103'	'CO2'	'362.49']	1495350615	1495350615
['9/27/2016'	'00:00:00'	'CP'	'2'	'206C'	'CO2'	'401.92']	1495350615	1495350615
['9/27/2016'	'00:00:00'	'CP'	'1'	'106'	'CO2'	'370.38']	1495350615	1495350615
['9/27/2016'	'00:00:00'	'CP'	'2'	'206C'	'AV'	'12.00']	1495350615	1495350615
['9/27/2016'	'00:00:00'	'CP'	'1'	'103'	'AT'	'72.50']	1495350615	1495350617
['9/27/2016'	'00:00:00'	'CP'	'1'	'103'	'RT'	'71.75']	1495350615	1495350617
['9/27/2016'	'00:00:00'	'CP'	'1'	'106'	'AV'	'0.00']	1495350615	1495350615
['9/27/2016'	'00:00:00'	'CP'	'1'	'108'	'AT'	'72.00']	1495350615	1495350617
['9/27/2016'	'00:00:00'	'CP'	'1'	'108'	'AV'	'16.00']	1495350615	1495350615
['9/27/2016'	'00:00:00'	'CP'	'1'	'106'	'AT'	'71.50']	1495350615	1495350615
['9/27/2016'	'00:00:00'	'CP'	'1'	'108'	'RT'	'73.25']	1495350615	1495350617
['9/27/2016'	'00:00:00'	'CP'	'2'	'206C'	'AT'	'72.00']	1495350615	1495350617
['9/27/2016'	'00:00:00'	'CP'	'2'	'206D'	'CO2'	'399.77']	1495350615	1495350617
['9/27/2016'	'00:00:00'	'CP'	'2'	'206H'	'RT'	'74.00']	1495350615	1495350615
['9/27/2016'	'00:00:00'	'CP'	'2'	'206D'	'AT'	'72.00']	1495350615	1495350617
['9/27/2016'	'00:00:00'	'CP'	'2'	'206D'	'AV'	'0.00']	1495350615	1495350617

Table 7.50 Raw Sensor Values corresponding to T(x)s 1582-1585 for the Sensors Topology W/ Average λ 20.75 T(x)/Sec

Sensors Raw Sensor Data Corresponding T(x)s 1582 through 1585 W/ Average λ 20.75 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'22:20:00'	'CP'	'2'	'206H'	'RT'	'74.00']	1495350693	1495350693
['9/27/2016'	'21:15:00'	'CP'	'1'	'106'	'RT'	'71.75']	1495350693	1495350693
['9/27/2016'	'22:05:00'	'CP'	'1'	'103'	'RT'	'71.75']	1495350693	1495350693
['9/27/2016'	'22:00:00'	'CP'	'2'	'206C'	'AV'	'464.00']	1495350693	1495350693
['9/27/2016'	'22:15:00'	'CP'	'2'	'206H'	'CO2'	'397.62']	1495350693	1495350693
['9/27/2016'	'22:20:00'	'CP'	'1'	'106'	'AT'	'69.00']	1495350693	1495350693
['9/27/2016'	'22:20:00'	'CP'	'1'	'106'	'AV'	'400.00']	1495350693	1495350693
['9/27/2016'	'22:30:00'	'CP'	'2'	'206D'	'AT'	'72.00']	1495350693	1495350693
['9/27/2016'	'22:25:00'	'CP'	'2'	'206C'	'AT'	'72.00']	1495350693	1495350693
['9/27/2016'	'22:00:00'	'CP'	'2'	'206H'	'AV'	'292.00']	1495350693	1495350693
['9/27/2016'	'22:15:00'	'CP'	'1'	'108'	'AT'	'70.00']	1495350693	1495350693
['9/27/2016'	'22:00:00'	'CP'	'1'	'103'	'CO2'	'374.68']	1495350693	1495350693
['9/27/2016'	'22:00:00'	'CP'	'1'	'106'	'CO2'	'369.66']	1495350693	1495350693
['9/27/2016'	'20:45:00'	'CP'	'1'	'108'	'CO2'	'394.75']	1495350693	1495350693
['9/27/2016'	'22:10:00'	'CP'	'2'	'206D'	'AV'	'452.00']	1495350693	1495350693
['9/27/2016'	'21:45:00'	'CP'	'1'	'108'	'AV'	'440.00']	1495350693	1495350693

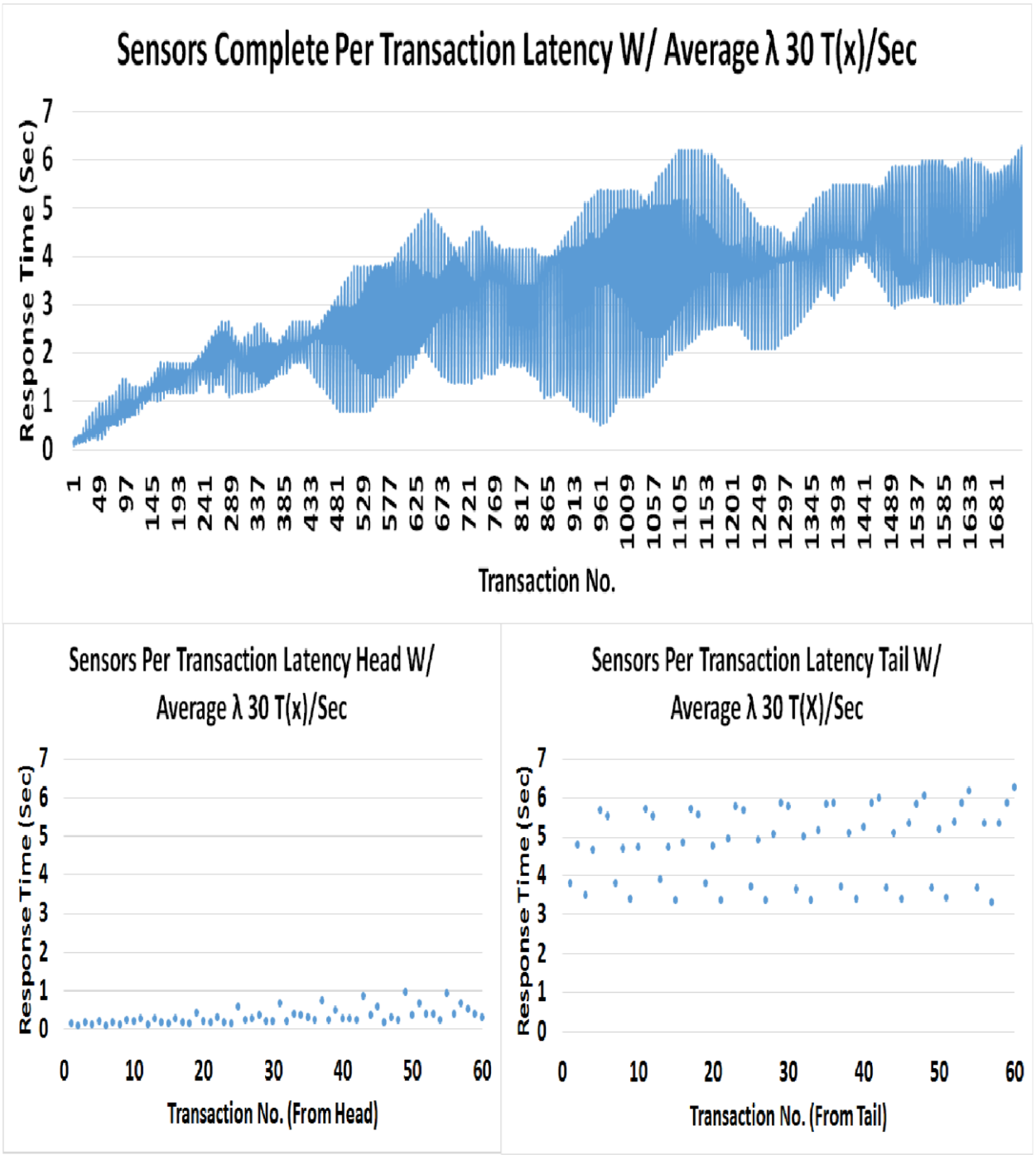

Figure 7.24 Per Transaction Latency Plots for the Sensors Topology W/ Average λ 30 T(x)/Sec

Table 7.51 Raw Sensor Values corresponding to first 4 T(x)s for the Sensors Topology W/ Average λ 30 T(x)/Sec

Sensors Raw Sensor Data Corresponding to first 4 T(x)s W/ Average λ 30 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'00:00:00'	'CP'	'1'	'103'	'AV'	'0.00']	1495346767	1495346768
['9/27/2016'	'00:00:00'	'CP'	'2'	'206D'	'AV'	'0.00']	1495346767	1495346768
['9/27/2016'	'00:00:00'	'CP'	'1'	'103'	'AT'	'72.50']	1495346767	1495346768
['9/27/2016'	'00:00:00'	'CP'	'1'	'108'	'CO2'	'389.73']	1495346768	1495346768
['9/27/2016'	'00:00:00'	'CP'	'2'	'206C'	'AV'	'12.00']	1495346768	1495346768
['9/27/2016'	'00:00:00'	'CP'	'1'	'103'	'CO2'	'362.49']	1495346768	1495346768
['9/27/2016'	'00:00:00'	'CP'	'1'	'106'	'RT'	'72.50']	1495346768	1495346768
['9/27/2016'	'00:00:00'	'CP'	'1'	'108'	'RT'	'73.25']	1495346768	1495346768
['9/27/2016'	'00:00:00'	'CP'	'2'	'206C'	'AT'	'72.00']	1495346768	1495346768
['9/27/2016'	'00:00:00'	'CP'	'2'	'206H'	'AV'	'20.00']	1495346768	1495346768
['9/27/2016'	'00:00:00'	'CP'	'1'	'108'	'AT'	'72.00']	1495346768	1495346768
['9/27/2016'	'00:00:00'	'CP'	'2'	'206C'	'RT'	'74.00']	1495346768	1495346768
['9/27/2016'	'00:00:00'	'CP'	'1'	'106'	'CO2'	'370.38']	1495346768	1495346768
['9/27/2016'	'00:00:00'	'CP'	'2'	'206H'	'AT'	'72.00']	1495346768	1495346768
['9/27/2016'	'00:00:00'	'CP'	'1'	'103'	'RT'	'71.75']	1495346768	1495346768
['9/27/2016'	'00:00:00'	'CP'	'1'	'106'	'AT'	'71.50']	1495346768	1495346768

Table 7.52 Raw Sensor Values corresponding to T(x)s 1105-1108 for the Sensors Topology W/ Average λ 30 T(x)/Sec

Sensors Raw Sensor Data Corresponding T(x)s 1105 through 1108 W/ Average λ 30 T(x)s/Sec								
Date	Time	Building	Floor	Room	Sensor	Value	Kafka Push Time	Kafka Pull Time
['9/27/2016'	'15:05:00'	'CP'	'2'	'206D'	'RT'	'74.00']	1495346812	1495346813
['9/27/2016'	'15:40:00'	'CP'	'1'	'106'	'CO2'	'369.66']	1495346812	1495346812
['9/27/2016'	'15:40:00'	'CP'	'1'	'106'	'RT'	'71.25']	1495346812	1495346812
['9/27/2016'	'15:00:00'	'CP'	'2'	'206C'	'AV'	'460.00']	1495346812	1495346813
['9/27/2016'	'15:25:00'	'CP'	'1'	'103'	'CO2'	'466.45']	1495346812	1495346812
['9/27/2016'	'15:25:00'	'CP'	'2'	'206H'	'AT'	'71.50']	1495346812	1495346813
['9/27/2016'	'15:55:00'	'CP'	'2'	'206D'	'AV'	'392.00']	1495346812	1495346813
['9/27/2016'	'15:00:00'	'CP'	'1'	'103'	'RT'	'72.00']	1495346812	1495346813
['9/27/2016'	'15:40:00'	'CP'	'1'	'108'	'RT'	'72.25']	1495346812	1495346813
['9/27/2016'	'15:30:00'	'CP'	'2'	'206D'	'AT'	'71.50']	1495346812	1495346813
['9/27/2016'	'15:00:00'	'CP'	'1'	'106'	'AT'	'69.50']	1495346812	1495346813
['9/27/2016'	'15:35:00'	'CP'	'2'	'206H'	'CO2'	'401.20']	1495346812	1495346813
['9/27/2016'	'15:30:00'	'CP'	'2'	'206C'	'AT'	'71.50']	1495346813	1495346813
['9/27/2016'	'15:00:00'	'CP'	'2'	'206D'	'CO2'	'401.20']	1495346813	1495346813
['9/27/2016'	'14:55:00'	'CP'	'1'	'108'	'AV'	'436.00']	1495346813	1495346813
['9/27/2016'	'15:20:00'	'CP'	'1'	'103'	'AV'	'440.00']	1495346813	1495346813

7.2 Analytical Vs Empirical

After empirically finding the response times for each of the topologies at the given arrival rates, we were able compare them against the analytical model. We expected the empirical and analytical models to behave with similar trends. However, after comparing the plots, discovered that there is an extra term of aggregation time that the analytical model does not account for in queueing theory. As we recall, the response time of a job is the summation of the service time of a job and the wait time. After running our experiment and from equations 6.8, 6.9, and 6.10, we were able to extract the time the sensor values were pushed to Kafka and pulled from Kafka and the time it took to aggregate the sensor values into transactions. Since we are modeling our transactions at room level granularity for all topologies, the only time invariant in the service time is the time for aggregation. After the sensor values have been aggregated into a transaction tuple, the time it takes to write the tuple to Ignite and to model the transaction are constant. The following tables show analytical calculations and empirical measurements for each of the four topologies.

Table 7.53 Analytical Model for Buildings Topology

Analytical Model for Buildings Topology						
Type	M/D/1	M/D/1	M/D/1	M/D/1	M/D/1	M/D/1
N	1	1	1	1	1	1
λ	4	11	15	22	31	201
μ	259.3697316	259.3697316	259.3697316	259.3697316	259.3697316	259.3697316
ρ	1.54%	4.24%	5.78%	8.48%	11.95%	77.50%
E[s]	0.0038555	0.0038555	0.0038555	0.0038555	0.0038555	0.0038555
ρ_num	0.9	10.05	20.1	30	40.2	50.1
E[w]	0.000030195	0.000085378	0.000118330	0.000178669	0.000261682	0.006638334
E[r]	0.003885695	0.003940878	0.003973830	0.004034169	0.004117182	0.010493834

Table 7.54 Empirical Results for Buildings Topology

Empirical Model for Buildings Topology						
Type	M/D/1	M/D/1	M/D/1	M/D/1	M/D/1	M/D/1
N	1	1	1	1	1	1
λ	4	11	15	22	31	201
μ	172.5747637	271.2960627	125	250.3730559	411.825283	394.1503508
ρ	2.32%	4.05%	7.04%	8.79%	7.53%	51.00%
E[s]	0.00579459	0.00368601	0.004693308	0.00399404	0.002428214	0.002537103
ρ_num	2.317836	4.054611	7.039962	8.786888	7.52746402	50.99576839
E[w]	0.000068748	0.000077885	0.000177715	0.000192380	0.000098831	0.001320106
E[re]	0.011671000	0.008343740	0.009631684	0.008150631	0.004983565	0.005974579

Table 7.55 Analytical Model for the Floors Topology

Floors Topology Analytical						
Type	M/D/2	M/D/2	M/D/2	M/D/2	M/D/2	M/D/2
N	2	2	2	2	2	2
λ	1	7	11	11	32	259
μ	230.9735535	230.9735535	230.9735535	230.9735535	230.9735535	230.9735535
ρ	0.22%	1.52%	2.38%	2.38%	6.93%	56.07%
E[s]	0.0043295	0.0043295	0.0043295	0.0043295	0.0043295	0.0043295
ρ_num	0.45	5.03	10.05	15	20.1	25.05
E[w]	0.000004696	0.000033308	0.000052805	0.000052805	0.000161117	0.002762642
E[r]	0.004334196	0.004362808	0.004382305	0.004382305	0.004490617	0.007092142

Table 7.56 Empirical Results for Floors Topology

	Floors Topology Empirical					
Type	M/D/2	M/D/2	M/D/2	M/D/2	M/D/2	M/D/2
N	2	2	2	2	2	2
λ	1	7	11	11	32	259
μ	136.4234146	234.7730373	248.7329463	211.4339237	290.3055752	456.0598284
ρ	0.37%	1.49%	2.21%	2.60%	5.51%	28.40%
E[s]	0.00733012	0.004259433	0.004020376	0.00472961	0.003444646	0.002192695
ρ_num	0.366506	1.490801516	2.211206871	2.6012855	5.511433939	28.39539726
E[w]	0.000013482	0.000032230	0.000045455	0.000063158	0.000100462	0.000434766
E[re]	0.014763200	0.008642151	0.008233331	0.009607050	0.007029906	0.005748877

Table 7.57 Analytical Model for the Rooms Topology

Rooms Topology Analytical						
Type	M/D/6	M/D/6	M/D/6	M/D/6	M/D/6	M/D/6
N	6	6	6	6	6	6
λ	3	6	10	12	25	262
μ	253.4276084	253.4276084	253.4276084	253.4276084	253.4276084	253.4276084
ρ	0.20%	0.39%	0.66%	0.79%	1.64%	17.23%
E[s]	0.0039459	0.0039459	0.0039459	0.0039459	0.0039459	0.0039459
ρ_num	0.15	1.68	3.35	5	6.7	8.35
E[w]	0.000003900	0.000007816	0.000013061	0.000015694	0.000032980	0.000410716
E[r]	0.003949800	0.003953716	0.003958961	0.003961594	0.003978880	0.004356616

Table 7.58 Empirical Results for Rooms Topology

	Rooms Topology Empirical					
Type	M/D/6	M/D/6	M/D/6	M/D/6	M/D/6	M/D/6
N	6	6	6	6	6	6
λ	3	6	10	12	25	262
μ	159.3097427	276.3049191	238.9612546	230.3039476	288.0305295	561.695811
ρ	0.31%	0.36%	0.70%	0.87%	1.45%	7.77%
E[s]	0.00627708	0.00361919	0.004184779	0.004342088	0.003471854	0.001780323
ρ_num	0.313854	0.361919	0.697463139	0.868417594	1.446605911	7.774077323
E[w]	0.000009881	0.000006573	0.000014696	0.000019019	0.000025481	0.000075035
E[re]	0.012654700	0.007382190	0.008531277	0.008844579	0.007113243	0.005060664

Table 7.59 Analytical Model for Sensors Topology

Sensors Topology Analytical						
Type	M/D/24	M/D/24	M/D/24	M/D/24	M/D/24	M/D/24
N	24	24	24	24	24	24
λ	2	12	15	16	18	258
μ	251.5343596	251.5343596	251.5343596	251.5343596	251.5343596	251.5343596
ρ	0.03%	0.20%	0.25%	0.27%	0.30%	4.27%
E[s]	0.0039756	0.0039756	0.0039756	0.0039756	0.0039756	0.0039756
ρ_num	0.04	0.42	0.84	1.25	1.68	2.09
E[w]	0.000000659	0.000003959	0.000004951	0.000005282	0.000005945	0.000088747
E[r]	0.003976259	0.003979559	0.003980551	0.003980882	0.003981545	0.004064347

Table 7.60 Empirical Results for Sensors Topology

	Sensors Topology Empirical					
Type	M/D/24	M/D/24	M/D/24	M/D/24	M/D/24	M/D/24
N	24	24	24	24	24	24
λ	2	12	15	16	18	258
μ	160.2353537	252.5971662	284.5759008	300.1380635	297.0887266	290.5192753
ρ	0.05%	0.20%	0.22%	0.22%	0.25%	3.70%
E[s]	0.00624082	0.003958873	0.003514001	0.0033318	0.003365998	0.003442112
ρ_num	0.052006833	0.19794363	0.219625063	0.22212	0.252449835	3.70027083
E[w]	0.000001624	0.000003926	0.000003867	0.000003709	0.000004259	0.000066131
E[re]	0.012530900	0.008021784	0.007089595	0.006760970	0.006789577	0.007045943

For the empirical experiment, we were able to determine the service time, the waiting time, and the arrival rate from log parsing. The response times and the arrival rates for both the empirical and analytical model were then plotted for each topology as seen in figures 7.25 and 7.26.

From these tables, the analytical vs. empirical models shows one distinctive commonality amongst all four topologies. In the analytical model, low utilization rates due to lower arrival rates yield low response times and high utilization rates due to higher arrival rates yield longer response times. In an M/D/c queue, the ideal scenario assumes that all sensor values arrive in order and does not consider aggregation times. So analytically, it takes more time overall to service more jobs that arrive in the system under a deterministic service rate. This means that in the ideal situation, all values arrive in order so all four sensor values for a transaction are arriving in order causing no additional waiting time upon aggregation. However, because values arrive out of order, there is a level of entropy which the empirical model takes into account that the analytical model does not. Recall that for a room level transaction to be complete, all four sensor values for that

room at that given timestamp but be present. As a result, under larger arrival rates, there is a higher likelihood that all four sensor values will arrive in the same sensor value batch arriving in the system and that the aggregator has less time to wait create a transaction tuple to write to Ignite. Under lower utilization rates influenced by lower arrival rates, queueing is essentially being forced and there is a higher likelihood that the four sensor values needed to complete a transaction will not arrive in the same sensor value batch arriving in the system and that the aggregator has to wait significantly longer to create a transaction tuple to write to Ignite. This can be seen in the high response times for low arrival rates from the per transaction latency plots shown in the previous subsection.

Another observation is that when adding higher delays into the system, the arrival rates eventually converge to a certain arrival rate range. From the figure, we can see how the arrival rates seem to cluster around the 0 to 50 range.

Table 7.61 Complete Arrival Rates for all delays for Buildings Topology

Buildings Arrival Rates	
Delay	Lambda
0	201
0.05	31
0.1	22
0.15	15
0.2	11
0.25	17
0.3	18
0.33	5
0.4	8
0.5	12
1	4

The table lists the complete set of arrival rates from the 11 different delays used in the building topology. It can be seen that the difference in arrival rates occur from delay 0 to 0.05.

Delay 0.01 to 0.3 all cluster around the arrival rate range from 15 to 20. Delay 0.33 to 1 all cluster around the arrival rate of 5 to 10. This arrival rate range gaps based off of the delay used was common between all topologies. This is expected because the utilization rate irrespective of topology is determined by the computing power of the hardware. The results from the empirical model calculations show that under specific hardware constraints, the system dictates the same utilization rate for each topology and shows essentially the impact of performance across utilization rates and how they vary across topology under specific system generated arrival rates.

The results from the analytical model seen in Figure 7.25 show the increase in utilization rate, influenced by the arrival rate results in an increase in response time. In the case of the M/D/1 buildings topology, the arrival rate causes 100% utilization at roughly 100 transactions per second. Graphically this shows how limited the buildings topology is in the amount of jobs that can be serviced.

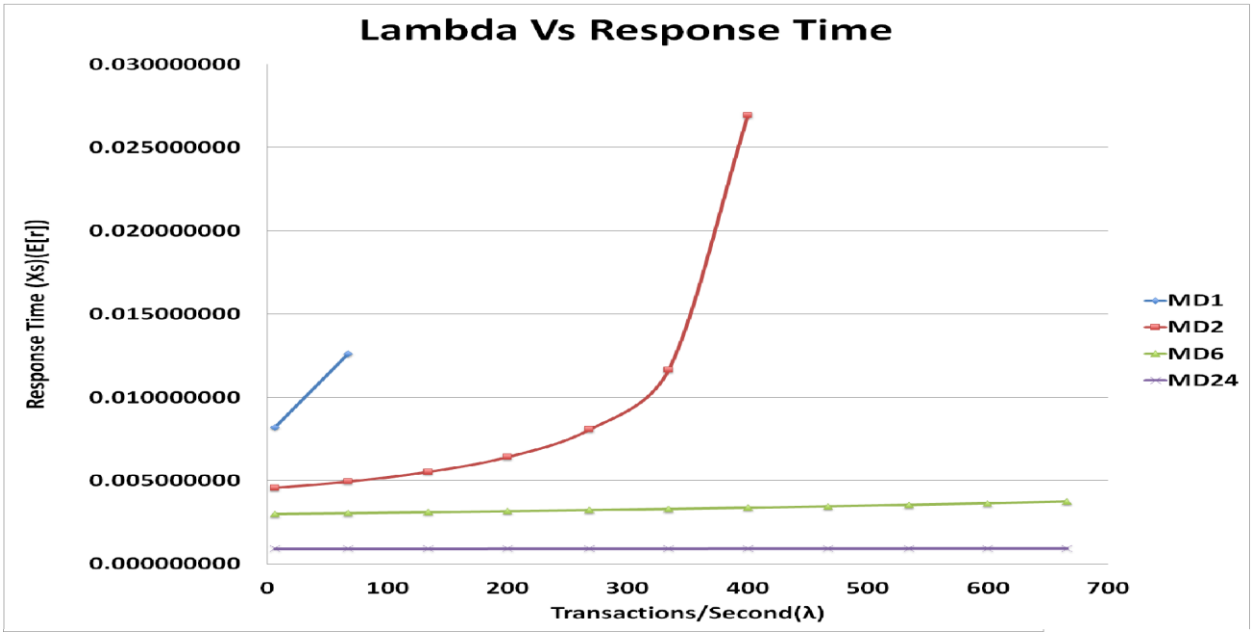

Figure 7.25 Analytical Comparison of λ Vs Response Time for 4 Topologies

In the case of the M/D/2 floors topology, more transactions are able to service before achieving 100% throughput. However, there is an exponentially high increase in response time after achieving 75% utilization at 334 transactions per second. In the case of the M/D/6 rooms topology, the arrival rate of 700 jobs per second as shown, only reaches 30% utilization rate. With the rooms topology, more jobs are able to be serviced in a much lower response time than either floors and buildings, at 0.004 seconds. Lastly, in the case of the M/D/24 sensors topology, the arrival rate of 700 jobs per second only reaches 2.5% utilization rate, capable of servicing the same amount of jobs as the rooms topology in 0.0009 seconds. Analytically, the following table shows the performance increase against the worst case scenario of buildings at 100% utilization.

Table 7.62 Analytical Performance Measure Comparison for 4 Topologies

Performance Increase for 100% utilization on builings topology (666 T(x)/Sec			
New value	Old value	((new-old)/old)*100	(new/old)*100
Floors	Building	99.7004993	0.299500699
Room	Floors	26.6257978	73.3742022
Room	Buildings	99.78024375	0.219756249
Sensors	Rooms	7.106554822	92.89344518
Sensors	Floors	31.8401757	68.1598243
Sensors	Buildings	99.79586085	0.20413915

From our experiment, we can visually see how lower utilization rates influenced by lower arrival rates negatively impact our performance. In the first section of the graph, we can see how lower arrival rates force queueing on the system. Because we are dealing with room level transactions, lower arrival rates increase the chances of the aggregator having to wait on missing sensor values to arrive before being able to complete and write a transaction tuple. The second section shows the impact of queueing. In our case, depending on the order that the real world

times arrive can either positively or negatively impact performance. We expect sensor values to arrive out of order.

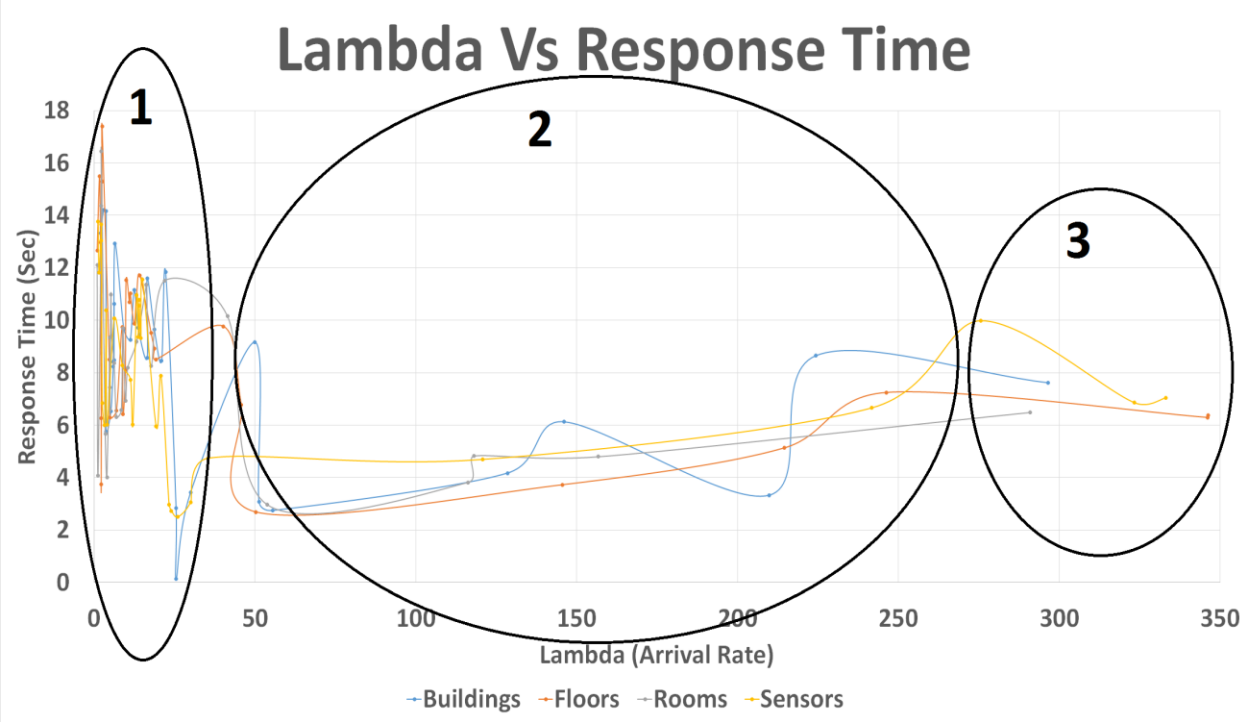

Figure 7.26 Empirical Comparison of λ Vs Response Time for 4 Topologies

However, if time stamp values are also arriving out of order, then the aggregator will still have to experience heavy wait times because it needs four sensor values for a room with the same real world time. However, if the real world time values arrive in order, this positively impacts performance because how there is a higher chance that within a given sensor value arrival rate batch, all four sensor values for a room with the same real world time will arrive. In the case of the buildings topology, we can see five inflection points during that arrival rate range. The high points indicate runs that experienced a high degree of out of orderedness and the low points indicate runs that experienced a minimal degree of out of orderedness. The third section of the graph shows how at some arrival rate, because we are aggregating at room level granularity, the response time will eventually converge, as the degree of out of orderedness begins to amortize to.

This indicates that at some point, the queue no longer will receive new sensor values and the degree of out of orderedness will no longer worsen to negatively impact the response time.

Chapter 8 Future Works

The end goal for our research is to provide our Namatad system a fully interactive interface for different end users. Our research will continue to extend our current work, as components will need to be added to ensure this end to end system is fully complete. Since we have analytically modeled an M/D/C queue, which was validated against our empirical implementation, we have identified an additional aggregation term. which we defined in our results. Our future work will involve exploring the following methods to quantify the additional aggregation term:

- Our assumption was that at no time during the experiment will any of our sensors fail. We have ensured this by adding aggregation into our system, in which the model can only pull complete transaction tuples. In some cases, the aggregator might have to wait longer because data arrives asynchronously. In a true IoT deployment, sensors may fail and restart and for future work, we will use imputation as an alternative to aggregation, to handle these missing values.
- Our experiment decomposed one M/M/n queueing architecture into 4 M/D/c pipelines to obtain measurable data for topology comparison. For our experiment, we ran the buildings, floors, rooms, and sensor topologies on individual runs. Running multiple pipelines in parallel as a system, would no longer yield a deterministic service rate. We will continue to extend our system by enabling the runs of multiple different pipelines in parallel to represent an M/M/n Model.
- Because the only term of our service time that is not constant is the aggregation time, dependent on the degree of out of orderedness, we will represent our system as a queueing network. By modeling our system as a queueing network, we can allow the write and

modeling times to remain constant, but isolate the aggregation time as its own Markovian queueing model.

- Once we have determined the most efficient way to quantify the degree of out of orderedness that effects the aggregation time, we can distribute the different components of our system (sensor transmission, Kafka platform, Ignite platform, modeling platform, etc) onto different servers. Most IoT systems are fully distributed and with a fully distributed Namatad system, we can now represent our Namatad system as a more practical M/M/n model.

- We will introduce Stochastic Petri Nets as another component to our performance model to improve the modeling by being able to characterize the out of orderedness that appears during the aggregation and segregation phases.

Chapter 9 Conclusions

In this paper, we have identified alternative streaming topologies for IoT platforms with the additions of the buildings, floors, and sensors topologies with our existing rooms topology. As we move from general to more detailed topologies, in the case of buildings to sensors, and introduce more computational servers, we can effectively decrease the response time of jobs in the system. We were successfully able to characterize the impact that topology has on response time. Our initial performance model, in the case of the Cherry Parkes Buildings, showed that the sensors topology performed the best out of the four topologies in terms of time spent during the consumer phase. Since we were able to characterize the performance in one building that consisted of 60 sensors, we then wanted to come up with a general model that could characterize all IoT driven systems under various utilization rates that were influenced by increasing the scalability of more sensors. We moved to our M/D/c analytical model to show that the sensors topology shows the highest performance increase at 99% when compared to the worst case buildings topology and validated this empirically.

There is still significant work we plan to pursue in this space as we integrate additional sensors into the buildings. We have identified that there is an additional aggregation term that plays a critical role in the time it takes to service a transaction. There is work that has been done to identify random variables that account for delays in expected service times for systems. As we continue to explore the impact of alternative streaming topologies on both the time to insight from our machine learning models as well as the ability to affect control systems, we will explore alternative methods for quantifying that extra aggregation term.

REFERENCES

[1] M. Ringwald and K. Romer, “Deployment of sensor networks: Problems and passive inspection,” in Intelligent Solutions in Embedded Systems, 2007 Fifth Workshop on. IEEE, 2007, pp. 179–192.

[2] M. Abu-Elkheir, M. Hayajneh, and N. A. Ali, “Data management for the internet of things: Design primitives and solution,” Sensors, vol. 13, no. 11, pp. 15 582–15 612, 2013.

[3] A. Dey, X. Ling, A. Syed, Y. Zheng, B. Landowski, D. Anderson, K. Stuart, and M. E. Tolentino, “Namatad: Inferring occupancy from building sensors using machine learning,” in Internet of Things (WF- IoT), 2016 IEEE 3rd World Forum on. IEEE, 2016, pp. 478–483.

[4] H. Kitagawa and Y. Watanabe, “Stream data management based on integration of a stream processing engine and databases,” in Network and Parallel Computing Workshops, 2007. NPC Workshops. IFIP International Conference on. IEEE, 2007, pp. 18–22.

[5] S. Lall, A. Alfa, and B. Maharaj, “The role of queueing theory in the design and analysis of wireless sensor networks: An insight,” in Industrial Informatics (INDIN), 2016 IEEE 14th International Conference on. IEEE, 2016, pp. 1191–1194.

[6] R. Tolosana-Calasanz, J. Diaz-Montes, O. F. Rana, and M. Parashar, "Feedback-control & queueing theory-based resource management for streaming applications," IEEE Transactions on Parallel and Distributed Systems, vol. 28, no. 4, pp. 1061–1075, 2017.

[7] P. N. D. Bukh and R. Jain, "The art of computer systems performance analysis, techniques for experimental design, measurement, simulation and modeling," 1992.

[8] (2015) Apache ignite. [Online]. Available: https://ignite.apache.org/

[9] M. Kocakulak and I. Butun, "An overview of wireless sensor networks towards internet of things," in Computing and Communication Workshop and Conference (CCWC), 2017 IEEE 7th Annual. IEEE, 2017, pp. 1–6.

[10] O. Chato and W. E. S. Yu, "An exploration of various quality of service mechanisms in an openflow and software defined networking environment in terms of latency performance," in Information Science and Security (ICISS), 2016 International Conference on. IEEE, 2016, pp. 1–7.

[11] R. Marculescu and P. Bogdan, "Cyberphysical systems: workload modeling and design optimization," IEEE Design & Test of Computers, vol. 28, no. 4, pp. 78–87, 2011.

[12] J. L. Horey, "Challenges in scheduling aggregation in cyberphysical information processing systems," in Data Mining Workshops (ICDMW), 2010 IEEE International Conference on. IEEE, 2010, pp. 148–153.

[13] N. Zompakis and K. Siozios, “A framework for reducing the modeling and simulation complexity of cyberphysical systems,” in Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS), 2015 International Conference on. IEEE, 2015, pp. 360–365.

[14] H. Suo, J. Wan, L. Huang, and C. Zou, “Issues and challenges of wireless sensor networks localization in emerging applications,” in Computer Science and Electronics Engineering (ICCSEE), 2012 International Conference on, vol. 3. IEEE, 2012, pp. 447–451.

[15] M. Gomes, R. da Rosa Righi, and C. A. da Costa, “Internet of things scalability: Analyzing the bottlenecks and proposing alternatives,” in Ultra-Modern Telecommunications and Control Systems and Workshops (ICUMT), 2014 6th International Congress on. IEEE, 2014, pp. 269–276.

[16] A. Hoppe and J. Gryz, “Stream processing in a relational database: A case study,” in Database Engineering and Applications Symposium, 2007. IDEAS 2007. 11th International. IEEE, 2007, pp. 216–224.

[17] K. Laubhan, K. Talaat, S. Riehl, M. S. Aman, A. Abdelgawad, and K. Yelamarthi, “A low-power iot framework: From sensors to the cloud,” in Electro Information Technology (EIT), 2016 IEEE International Conference on. IEEE, 2016, pp. 0648–0652.

[18] A. Singhal, R. Tomar et al., “Intelligent accident management system using iot and cloud computing,” in Next Generation Computing Technologies (NGCT), 2016 2nd International Conference on. IEEE, 2016, pp. 89–92.

[19] J. Wang and S. Duan, “An online anomaly learning and forecasting model for large-scale service of internet of things,” in Identification, Information and Knowledge in the Internet of Things (IIKI), 2014 International Conference on. IEEE, 2014, pp. 152–157.

[20] H. Y. Shwe and P. H. J. Chong, “Scalable distributed cloud data storage service for internet of things,” in Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld), 2016 Intl IEEE Conferences. IEEE, 2016, pp. 869–873.

[21] T. Reichherzer, A. Mishra, E. Kalaimannan, and N. Wilde, “A case study on the trade-offs between security, scalability, and efficiency in smart home sensor networks,” in Computational Science and Computational Intelligence (CSCI), 2016 International Conference on. IEEE, 2016, pp. 222–225.

[22] Z. Huang, K.-J. Lin, and C.-S. Shih, “Supporting edge intelligence in service-oriented smart iot applications,” in Computer and Information Technology (CIT), 2016 IEEE International Conference on. IEEE, 2016, pp. 492–499.

[23] D. Nunes, J. S. Silva, A. Figueira, H. Dias, A. Rodrigues, V. Pereira, F. Boavida, and S. Sinche, “Fotsechuman security in fog of things,” in Computer and Information Technology (CIT), 2016 IEEE International Conference on. IEEE, 2016, pp. 743–749.

[24] M. Samaniego and R. Deters, “Hosting virtual iot resources on edge- hosts with blockchain,” in Computer and Information Technology (CIT), 2016 IEEE International Conference on. IEEE, 2016, pp. 116–119.

[25] B. Cheng, A. Papageorgiou, F. Cirillo, and E. Kovacs, “Geelytics: Geo- distributed edge analytics for large scale iot systems based on dynamic topology,” in Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on. IEEE, 2015, pp. 565–570.

[26] Z. Wang, K. Yang, and D. K. Hunter, “Modelling and analysis of convergence of wireless sensor network and passive optical network using queueing theory,” in Wireless and mobile computing, networking and communications (WiMob), 2011 IEEE 7th international conference on. IEEE, 2011, pp. 37–42.

[27] D. Shen, J. Luo, and F. Dong, “Energy-efficient resource allocation model with qos assurance for ubiquitous and heterogeneous environment,” in Ubi-Media Computing and Workshops (UMEDIA), 2014 7th International Conference on. IEEE, 2014, pp. 37–42.

[28] S. S. Sarma and G. Chakraborty, “Queuing model-based optimal traffic flow in a grid network,” in Advanced Networks and Telecommuncations Systems (ANTS), 2015 IEEE International Conference on. IEEE, 2015, pp. 1–3.

[29] W. M. Kempa, “Time-dependent analysis of transmission process in a wireless sensor network with energy saving mechanism based on threshold waking up,” in Signal Processing Advances in Wireless Communications (SPAWC), 2015 IEEE 16th International Workshop on. IEEE, 2015, pp. 26–30.

[30] Z. Wang, K. Yang, and D. K. Hunter, "A dynamic bandwidth allocation algorithm for a multi-sink wireless sensor network converged with a passive optical network," in Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on. IEEE, 2012, pp. 1548–1554.

[31] H. Tijms, "New and old results for the m/d/c queue," AEU-International Journal of Electronics and Communications, vol. 60, no. 2, pp. 125–130, 2006.

[32] V. Marianov and D. Serra, "Location models for airline hubs behaving as m/d/c queues," Computers & Operations Research, vol. 30, no. 7, pp. 983–1003, 2003.

[33] G. J. Franx, "A simple solution for the m/d/c waiting time distribution," Operations Research Letters, vol. 29, no. 5, pp. 221–229, 2001.

[34] Y. Liu and W. Trappe, "Topology adaptation for robust ad hoc cyber- physical networks under puncture-style attacks," Tsinghua Science and Technology, vol. 20, no. 4, pp. 364–375, 2015.

[35] J. Huang, Q. Duan, C.-C. Xing, and H. Wang, "Topology control for building a large-scale and energy-efficient internet of things," IEEE Wireless Communications, vol. 24, no. 1, pp. 67–73, 2017.

[36] T. Yu, X. Wang, and A. Shami, "Physical topology discovery scheme for wireless sensor networks using random walk process," in Global Communications Conference (GLOBECOM), 2016 IEEE. IEEE, 2016, pp. 1–5.

[37] L. Zhang, J. Qu, and J. Fan, “Topology evolution based on the complex networks of heterogeneous wireless sensor network,” in Computational Intelligence and Design (ISCID), 2016 9th International Symposium on, vol. 2. IEEE, 2016, pp. 317–320.

[38] G. Yi, J. H. Park, and S. Choi, “Energy-efficient distributed topology control algorithm for low-power iot communication networks,” IEEE Access, 2016.

[39] A. A. Aziz, Y. A. Sekercioglu, P. Fitzpatrick, and M. Ivanovich, “A survey on distributed topology control techniques for extending the lifetime of battery powered wireless sensor networks,” IEEE communications surveys & tutorials, vol. 15, no. 1, pp. 121–144, 2013.

[40] M. Stein, T. Petry, I. Schweizer, M. Brachmann, and M. Muhlhauser, “Topology control in wireless sensor netowrks: What blocks the breakthrough?” in *Local Computer Networks* (LCN), 2016 IEEE 41st Conference on. IEEE, 2016, pp. 389-397.

[41] J. R. Jay Kreps, Neha Narkhede, “Kafka: A distributed messaging system for log processing,” June 2011.

[42] (2015) Apache nifi. [Online]. Available: http://nifi.apache.org

[43] U. Hunkeler, H. L. Truong, and A. Stanford-Clark, “Mqtt-sa publish/subscribe protocol for wireless sensor networks,” in Communication systems software and middleware and workshops, 2008. comsware 2008. 3rd international conference on. IEEE, 2008, pp. 791–798.

[44] (2015) Message queue telemetry transport. [Online]. Available: http://mqtt.org

[45] (2015) Apache avro. [Online]. Available: http://avro.apache.org