Characterizing the Impact of Topology on IoT Stream Processing

Anindya Dey, Kim Stuart, Matthew E. Tolentino
Intelligent Platforms & Architecture Lab
University of Washington, Tacoma, WA, USA
{*andy1602, ksstuart, metolent*} *@uw.edu*

***Abstract*—The Internet of Things (IoT) extends traditional cyber-physical systems by linking sensor based edge devices to network accessible services and resources. In most current IoT deployments, sensor data is streamed from edge devices to servers for storage. Analytical pipelines are then used to translate this raw sensor data into actionable information in real-time. As additional IoT devices are deployed, the volume and rate of data received on the server side can increase dramatically. This has a possibility of offsetting the response latencies beyond acceptable limits for IoT analytical systems.**

In this paper, we compare the impact of alternative server-side stream processing topologies for ingesting and analyzing IoT sensor data in real-time. We use real building sensor data with our real-time IoT platform called Namatad. We have characterized and analyzed the latency and QoS impact due to the different levels of granularity of the ingestion and routing process by which we transmit data into the analytical pipelines. Our results show that as IoT systems continue to scale in density, server-side topology management for IoT data streams is critical for latency-sensitive control and analysis applications.

***Keywords*-IoT, Server, Topology, Sensors, Machine Learning**

I. Introduction

Internet of Things (IoT) systems consist of multiple compute platforms, notably edge devices and servers. Within IoT systems, most of the focus has been on the development and integration of novel new edge devices. For example, the recent proliferation of wearable devices designed to monitor personal health has increased significantly in recent years yielding unprecedented awareness of fitness. Similarly, new *smart* buildings are integrating new sensors with building control systems to improve energy efficiency, occupant comfort, and safety [1], [2]. The raw data obtained using IoT devices provides tremendous operational insight, which is driving the deployment of additional IoT devices.

For deployed IoT devices, once sensor values are read the data generated is transmitted across the network and stored on server platforms for later analysis [3]. Once stored, this data is then analyzed, leveraging recent advances in machine learning. To date, most of these IoT analytics have been performed offline, using batch-oriented techniques. However, as IoT analytics transition to online, real-time pipelines that immediately translate raw data into actionable information, earlier approaches to manage streaming data becomes challenging. Additionally, as the number of deployed IoT devices increases, how IoT data is routed and processed must be handled judiciously to prevent overloading and ensure scalability.

The server entry point for IoT data generated by edge devices are often message queues. Effectively managing these ingress queues is key to efficiently route streaming data to analytical pipelines while maximizing scalability. One common approach is to use message brokers, such as MQTT or Apache Kafka to establish a single ingress queue. This simplifies the configuration of IoT devices because all data can be sent to the same destination message queue. However, as an IoT deployment scales from a few edge devices to thousands, this single message queue will quickly become a bottleneck. The additional latency caused by using a single queue will then have a ripple effect impacting all downstream analytical processing pipelines. In the extreme this additional latency can cause poor predictions and analysis.

In this paper, we describe and characterize the impact of alternative IoT data processing topologies in the context of real-time machine processing pipelines. Our goal is twofold. First, we identify and compare the performance impact of using different topologies that leverage different levels of queue-level parallelism for server-side processing of IoT data. Second, we identify patterns using real deployment data (e.g. smart buildings) that can be generalized for diverse IoT deployments. We leverage the complete real-time streaming platform that we developed in previous work [4] and extend this system to use alternative topologies that increase queue parallelism. We use real building sensor data collected by the facilities department at the University of Washington for a LEED certified building that was recently renovated. These data streams, sourced from multiple installed sensors are then routed through our real-time streaming system to our analytical pipelines to gain insights into room occupancy within that building. We then analyze the performance impact of several alternative topologies. More specifically, our contributions from this paper include:

- Characterization of the impact of topology on an IoT

978-1-4673-9944-9/18/$31.00

system in terms of latency to prediction as well as the impact on prediction accuracy.
- Characterization of the impact of real world parameters including degree of disorder in data arrivals, varied data set sparsity, and impact of missing values.
- Impact of new prediction models that enable occupancy forecasting.

II. System Architecture & Topologies

Most IoT systems are composed of edge devices connected to web servers which ingest sensor data streams and transfer them to persistent storage. This stored data is then analyzed and visually displayed using dashboard interfaces. Unfortunately, this approach fails to leverage the temporal utility of sensor data.

To immediately take advantage of new sensor data, data streams must be received and analyzed in real-time with minimal latency. Many IoT servers use a message broker such as MQTT or Apache Kafka to reliably receive and queue incoming data and then use analytical pipelines for processing the data. As an example, in previous work we leveraged Kafka and open source analytical engines to create a new end-to-end streaming system for IoT called Namatad and used this system to infer occupancy from a minimal set of environmental sensors using machine learning [4]. However, during several subsequent scaling experiments for high-density sensor deployments within buildings, we quickly encountered bottlenecks with our system that caused our machine learning models to produce wildly inaccurate predictions. These problems motivated us to conduct several scaling experiments including adding IoT devices to a deployment as well as increasing the rate at which these devices send data to the server.

Several scaling issues became evident during these initial experiments. First the prediction latencies of our real-time analytical pipelines increased. We initially used a single server-side queue within Kafka to capture the data from all sensors within the building. We found that increasing the number of IoT devices and rate at which each edge device sent data to the server significantly increased the depth of this single receive queue. Second, the computational requirements on the server increased. This was due to the increase in the sensor types that had to be parsed, split, correlated with other sensor values in time, and then relayed to the appropriate analytic pipeline for processing. Third, the increase in rate and volume in IoT device generated data caused the sensor values to often arrive at the server out of order. This required additional buffer space within the server given each analytic pipeline required multiple correlated sensor values to provide a prediction.

A simple approach to coping with these issues would be to increase the number of servers we used for processing real-time analytic pipelines. However, particularly for many IoT deployments, it is impractical to install a cluster of servers

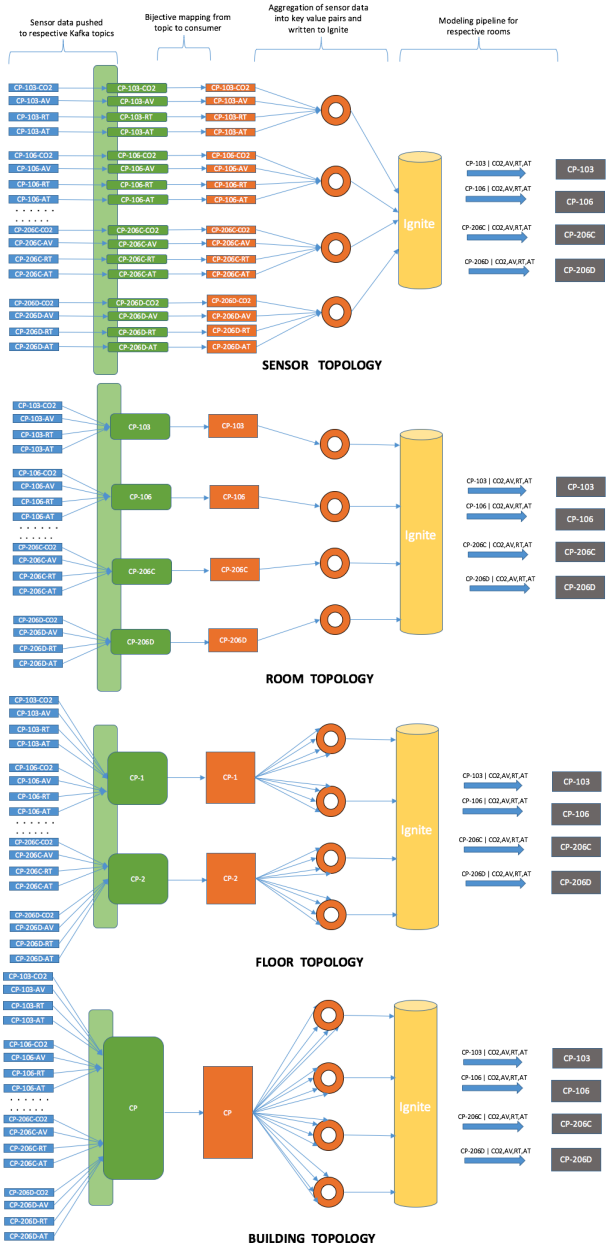

Figure 1. Topology Comparisons

just to analyze sensor data. Additionally, after profiling we found that not all cores of our single server were being utilized, meaning the server had additional unused computational capacity. Instead, how we *used* the server was problematic. This was due to how sensor values from different areas of the building were managed within the streaming platform and routed through the analytic pipelines. In other words, the topology used for receiving, filtering, and processing IoT data in real-time limited our scalability.

To address this issue, we experimented with using alternative topologies within our real-time analytic system.

Using the data from our previous work to predict room occupancy, we developed four different topologies that could be configured. Figure 1 shows the four different topologies used in this study which we refer to as sensor, room, floor, and building based on how we group the data received. Each topology has a different number of receive queues or topics. Each topic consists the destination that IoT devices send data to the server for processing. For each topology the topics are created before the experiments are run. A sample topic is named as CP-103-CO2 for sensor topology, CP-103 for room topology, CP-1 for floor topology, CP for building topology. In our experimental setup, the number of topics for the sensor topology is 24 (4 sensors per room * 6 rooms), for room topology its 6 (since 6 rooms in total), for floor topology its 2 (since 2 rooms in total), for building topology its 1(since 1 building).

We have also configured a bijective mapping between Kafka topic and Kafka consumers. This means that as we increase the number of topics, we also increase the number of computational pipelines that will be used to process the data. Consumers pull data from distinct topics. Based on the topology, the data is either segregated or aggregated into room level key-value pair tuples and written to our key-value temporary store using Apache Ignite. The tuples are at room granularity since our machine learning models have been trained to predict occupancy at room level.

III. Experimental Methodology

We experimented using sensor values from six rooms from the Cherry Parkes building. This is consistent with our previous work where we had predicted occupancy and compared across several rooms [4]. To evaluate multiple levels of granularity in terms of sensor, room, floor and building levels we needed another floor along with various types of rooms like class rooms, office rooms so that we could also account for the sparsity of data. For a single room, there are four sensors, which gives us time stamped sensor values. For example, for room CP-103, we have CO2(carbon dioxide), AV(supply air volume), AT(supply air temperature) and RT(room temperature) sensors. So 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*.

The motivation for our experiments was to evaluate the time for orchestrating the flow of data from individual sensors in a given room, aggregation time, and time for writing the resulting tuples to Ignite which feeds them to the trained machine learning models for prediction and forecasting. In this evaluation, we captured the execution times for the entire end to end run with a focus on the aggregation time given this is dependent on topology. For all runs, we fixed the CPU frequency on both the PI boards and server to 900 MHz and 3 GHz respectively. This was done to maintain experimental consistency, particularly given different topologies have different CPU loads. Additionally, given our system supports dynamic voltage frequency scaling we wanted to ensure any observed performance differences were due to topology and not system effects.

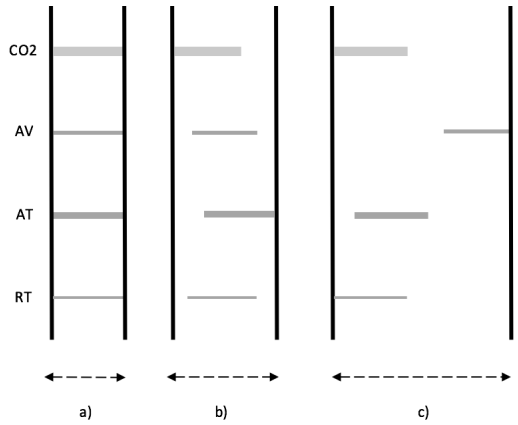

Figure 2. T(x) Duration : Each grey line represents the time it takes for the sensor value to arrive and pass through its respective Kafka topic to reach the consumer at the end. The length of the line can be assumed to be roughly consistent for a topology.
The thickness of each of the grey lines is proportional to the information gain (importance in quality of prediction) for that sensor type. In our case, CO2 has the highest information gain followed by AT, whereas RT and AV do not provide as much information. This provides a choice to trigger the model when just the features with higher information gains have arrived without waiting for all the features, as the quality of prediction will not be too bad and we save on time to insight.
T(x) duration is shown by the dotted double headed arrow. Its least in case of a) when all sensor values arrive in order, slightly more in case of b) when sensor values arrive slightly out of order and a lot more in case of c) when the arrival rates differ a lot.

We define a single transaction time as T(x), where T is the transaction and x is the corresponding time stamp. Figure 2 shows that we have defined a transaction as the total time it takes for all four sensor (CO2,AV,AT,RT) data values in a room to get pushed to their respective Kafka topics via the Kafka producer on the sensors, the time to perform any necessary processing, such as aggregating the data based on their topics and creating and storing key-value pairs in Ignite through the Kafka consumer code. When these respective scripts are executed, their starting time and end times are logged. We then analyzed the performance impact of the execution times across the producer, consumer, and modeling phases while generating twenty-four hours' worth of forecasting values. For all results shown, we conducted five runs per experiment to characterize run time variances and minimize any random errors. The plots in Figure 3 constitute the average of all the runs. Given that we quiesced the system before each run, we observed minimal run-to-run variance.

IV. Occupancy Prediction & Forecasting

Based on discussions with first response teams and facilities management team there was a need to predict current occupancy but also the occupancy in a future time interval. Hence we evaluated several forecasting techniques and used Linear Regression to predict the next ten set of values for each of the four features (CO2,AV,RT,AT). To forecast the feature values the forecasting model we use a predefined time window of the historic feature values to learn the pattern and then guess the next certain number of feature sets. We set the model to predict the next ten feature sets. The longer history the model looks at, the better is the guess at the future set. However a limitation of this approach is the inability to detect inflection points. These are important since they effect the actuation of the control systems attached to the sensors. We are currently exploring ways to tackle this limitation by using ensemble techniques to get forecasts from multiple models like Linear Regression, Logistic Regression, ARIMA, etc. Once we have the forecast feature set, we use that with the trained Random Forest occupancy model for scoring, yielding forecasted occupancy. An important difference between the Random Forest model and the Linear Regression model is that the Random Forest model was trained offline. We use it only for scoring online, whereas the Linear Regression model is learning and updating online. In future work we could use advanced modeling techniques like Deep Learning to potentially improve the quality of forecast but at the expense of longer training times.

The goal of this study was to empirically determine the impact of using the different topologies mentioned in section II above on how close to real-time can we make predictions while maintaining high quality predictions. In previous work we concluded our model needed all four features to provide the best prediction. In a real time scenario there can be situations when all the values may not arrive to the modeling phase at the same time, as shown in Figure 2 and the trained model has to either score with fewer features than trained on or must wait for all features to arrive. In the case of missing features, the quality of prediction will decrease and if the most relevant feature, that is the one with highest information gain is absent, the prediction may be meaningless. One way to tackle this is to use statistical techniques to impute some of the missing values; however, this impacts predictions. Alternatively our learning pipeline could wait for all the features to arrive and then predict the class label. The challenge with waiting is to ensure values arrive in the correct order. For example, the four features for a real world time stamp T(i) should all arrive before the next set of four features for real world time stamp T(i+1). In the worst case, predictions will be delayed which dilutes the near real-time aspect of the system and thus the utility of the prediction is reduced.

V. Results

The performance impact of topology on IoT stream processing in terms of total execution time and phase-wise execution times are shown in Figure 3. When there is minimal contention in the data flow, the time to prediction is minimized. When contention is high, this increases to time to prediction. Interestingly, data aggregation and segregation, which essentially is a data transformation step within the topologies is notably different between the topologies. The aggregation time for each transaction type is further explained in Figure 2.

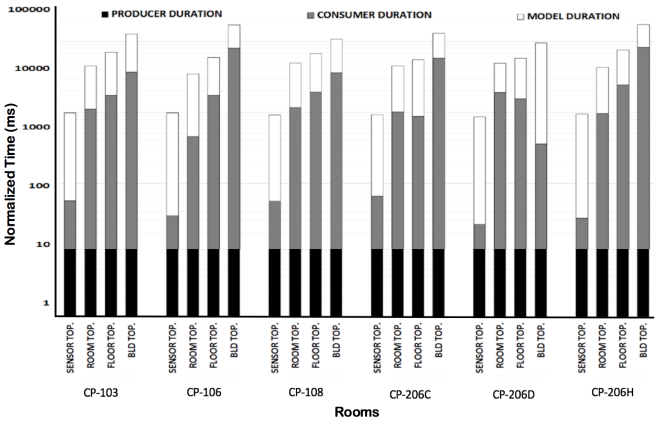

Figure 3. Phase Wise Execution Times

In Figure 3 the y-axis represents the execution time (in milliseconds) relative to the start time of the producer phase and the x-axis compares the topologies. The phase wise execution time follows a consistent trend across all the rooms. The producer phase includes transit time from the producer (raspberry PIs as hardware sensors) to the respective Kafka topics. The consumer phase starts once the consumer pulls the data from the topics and ends after it transforms the data into required tuples and writes to the key-value store in Ignite. The modeling phase runs when the machine learning pipeline picks up the tuple from ignite and scores the predictions followed by occupancy forecasting. Each of the phases is described in further detail below.

A. Producer phase

As shown in figure 3 the producer duration is consistent across topologies. This is as expected because there is no blocking or waiting time for a producer to write to a topic irrespective of topology. In the case of the sensor topology each topic is being written to by a single producer, whereas in the case of the building topology a single topic is being written to by multiple producers. Consequently, the producer phase could be longer in the latter case. Our experimental results suggest otherwise, though we suspect our experiments did not exhibit a sufficient load to showcase this impact in the producer phase.

B. Consumer phase

The consumer phase pulls data from the respective topics, aggregates and transforms them into a key value schema as explained before, and stores in Ignite. In the case of the sensor topology each consumer must pull data from the respective topics and create the key value pair; hence it has the lightest computational load. However, in the case of the room, floor and building topology more sensors write to a single topic. In these cases every consumer has to read the messages from its topic, parse and segregate them into individual sensor values and then aggregate into the appropriate key-value pair before it can be written to Ignite. So as we progress from sensor topology towards building topology, the workload of the consumer increases which is supported by the results as well. We also observed that the degree to which sensor data arrived out of order caused the consumer to wait for the right set of values in case of sensor topology. Doing multiple runs averaged out the differences in the case of a single room.

C. Modeling phase

The modeling phase is divided into two phases - current occupancy prediction and forecast occupancy prediction. The current occupancy prediction nearly overlaps with the ending of the consumer phase since it just reads the latest tuple from Ignite and scores it against the trained Random Forest room level occupancy model. The forecast occupancy model as explained in Section IV uses a certain time window of features so that it is able to forecast a specific number of future feature values. The scoring is then done using the forecast feature values. So the time for forecast depends on how quickly the forecasting model can use the required time window of features. In case of sensor topology, at the end of each consumer phase only one set of values corresponding to a T(x) is available, which then gets written to Ignite. It must then wait for the next set of values to come through the producer and consumer phase before writing to Ignite. However, in case of the other extreme i.e. the building topology, at the end of every consumer phase we have a set of six values that are written to Ignite. So it reduces the time required to build up the size of the historic feature set. In the case of room topology, we have one set of values at the end of each consumer phase, but this does not require the aggregation time the sensor topology requires. Consequently, the computational load is reduced for the room topology relative to the sensor topology. In case of floor topology, we have three sets of values which can be written to Ignite after each consumer phase completes. These analogies are supported by the results shown in above plots.

The room CP-206D shows anomalous behaviour for consumer and modeling phase which can be attributed to extremely sparse data that could be collected for that room. Experimental observation shows that sparsity accentuates degree of disorder. This is discussed below.

We further explored the challenges introduced due to out of order arrival of sensor data, which lead to increased aggregation time. The plots in Figure 3 show that although the volume and transmission rate of data being collected from each room is the same, due to varying sparsity and varying degrees of out of order of data arrivals across the different rooms, we see a difference in consumer and modeling phase execution time for the same topology across different rooms. This is because a single sensor value quickly reaching its corresponding topic is not enough since it is the transaction time that is required. Therefore even if one of the sensor values arrive out of order, the total transaction time increases as explained in Figure 2. Since the plotted results are averaged over five sets of runs (standard deviation was minimal) it shows that in the minimal contention scenario there are always values arriving out of order as compared to the other topologies, where there is more contention. Therefore we conclude that increased contention in the pipeline reduces the degree of disorder with the trade off being an increased delay in the total execution time. We are running further experiments to formulate an analytical model to estimate aggregation time due to the data transformations in our experiment.

Table I
COMPARING PREDICTIONS ACROSS TOPOLOGIES : T4 >T3 >T2 >T1 I.E. T4 OCCURS LATER IN THAN T3 AFTER THE PREDICTION PIPELINE HAS BEEN STARTED.

Topology / Timestamp	T1	T2	T3	T4
Sensor	2	2	2	2
Room	2	2	2	2
Floor	0	0	1	1
Building	0	0	1	0

Table I shows that the quality of modeling was affected by the time it took for the sensor values to reach storage within Ignite. For the building topology, which has the lowest arrival rate and has 6 values being written to ignite at a time, we see that our model predicted zero occupancy several times. This was due to the lack of values within Ignite to serve as input to the model; consequently the model assumed all values were zero. This revealed that the modeling phase must be configured to operate at a frequency independent of the consumer phase, but dependent on the topology. We are currently exploring the configuration of synchronization gearing ratios as well as leveraging statistical imputation techniques to handle missing values within our models.

VI. RELATED WORK

There has been considerable work in IoT systems that leverage analytical systems for real-time processing [5], [6] and sensor driven architectures [7], [8]. Other work has proposed leveraging cloud computing as a means to extend scalability, accepting the trade-offs of higher latency. For example, [9] conducted a case study in which Arduino boards and Raspberry Pi boards were used to transmit sensor

data into a cloud based infrastructure to examine trade-offs 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.

Edge computing has recently emerged to minimize latency for real-time systems [10]. Just as [11] did with their Geelytics system, we shifted the focus from cloud to edge computing using a fog server. However, our system provides flexible software topology management techniques rather than handling highly distributed IoT systems. We 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 [9] can be managed. We have shown that by choosing a suitable processing topology for data management, we can more effectively use existing edge servers and reduce latency.

Similar to our approach, there has been work to determine the effective physical node arrangements in network topologies of ad hoc cyber-physical systems [12], [13], as well as topology control algorithms for how to maintain and manage these nodes [14]. In many cases, node or sensors arrangements are constrained by building or geographical blueprints and real-time analytics are computed based on existing sensor deployments. Our system provides multiple ways to manage data flows through real-time analytical pipelines and enables these to be tuned to minimize latency.

VII. Conclusion

In this paper, we explored the impact of alternative real-time streaming topologies within the edge server of IoT analytical systems. We evaluated these topologies in terms of the time to insight from our machine learning models as well as the quality of predictions. Our results show that topology impacts stream processing in multiple ways and real world parameters like missing values, out of order arrivals, varying sparsity have a significant impact as we scale up the density of sensor deployments.

VIII. Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. CNS-1742899.

References

[1] B. Balaji, J. Xu, A. Nwokafor, R. Gupta, and Y. Agarwal, "Sentinel: occupancy based hvac actuation using existing wifi infrastructure within commercial buildings," in *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*. ACM, 2013, p. 17.

[2] S. Mamidi, Y.-H. Chang, and R. Maheswaran, "Improving building energy efficiency with a network of sensing, learning and prediction agents," in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, 2012, pp. 45–52.

[3] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad Hoc Networks*, vol. 10, no. 7, pp. 1497–1516, 2012.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] M. Stein, T. Petry, I. Schweizer, M. Brachmann, and M. Mühlhäuser, "Topology control in wireless sensor networks: What blocks the breakthrough?" in *Local Computer Networks (LCN), 2016 IEEE 41st Conference on*. IEEE, 2016, pp. 389–397.