A Utility-Driven Data Transmission Optimization Strategy in Large Scale Cyber-Physical Systems

Soumi Chattopadhyay
Indian Statistical Institute Kolkata
Email: soumi_r@isical.ac.in

Ansuman Banerjee
Indian Statistical Institute Kolkata
Email: ansuman@isical.ac.in

Bei Yu
The Chinese University of Hong Kong
Email: byu@cse.cuhk.edu.hk

***Abstract*—In this paper, we examine the problem of data dissemination and optimization in the context of a large scale distributed cyber-physical system (CPS), and propose a novel rule-based mechanism for effective observation collection and transmission. Our work rests on the idea that all observations on all parameters are not required at all times, and thereby, selective data transmission can reduce sensor workload significantly. Experiments show the efficacy of our proposal.**

I. Introduction

An important challenge in the large scale sensor-enabled CPS context today is handling the large data volume collected and transmitted by sensors distributed across wide areas of operation. In this paper, we address this problem from a formal perspective. We consider a simple operational model of a CPS that comprises of a set of distributed sensors communicating with a back-end controller to monitor and regulate system parameters based on some system objectives. The operations of these systems involve sensing, networking and processing of significant amount of data. The inputs are received from a set of information observers, often called the front-end, (e.g. sensors, monitors, data sources, network feeds etc). The back-end is usually a processing unit (running either on cloud servers or on mobile gateway / edge devices [1]) that accumulates all observations received over the network from different disparate sources, and monitors / evaluates a set of system objectives for deciding on necessary actuations.

Management and interpretation of data of large magnitudes in real-time is a stupendous challenge today in the CPS context. While computational scalability has received most of the recent research attention in the effort of designing better CPS, communication scalability of information flow processing has also been looked at [2], [3], A number of articles [4], [5] have focused on efficient means of regulating message observation and transmission using energy reducing optimizations, efficient query processing or classical compression techniques.

In this work, we propose a fundamentally different operational model for data dissemination and optimization. The crux of our proposal is to identify the relevance of each sensor observation as far as the objectives in the back-end are concerned. By transmitting only the necessary set of events and thereby avoiding redundancies makes the back-end computation more efficient. The motivation stems from the observation that not all data generated in the system is always important, as far as the end objectives being monitored at the back-end are concerned, and hence, need not be distributed always. This saves network resources.

To the best of our knowledge, only a few articles have looked at the problem from a similar perspective as ours, with the motivation of analyzing the system objectives and deciding on data dissemination. Most of the proposals in this direction have a topic based filtering approach [6], [7] that reduce the overall communication overhead in the network. A similar work has been done in [8], where the authors preprocess the end objectives and try to reduce the number of transmissions, however, data optimization at execution time has not been considered, and this is the main focus of our proposal here.

II. A Motivating Example

Consider a smart home scenario with 4 components: an air-conditioner (AC), a light and a wall mounted web camera, communicating with a central home automation controller. The AC can sense and communicate the temperature and humidity of the room. The light can sense and transmit whether the room is bright enough. The camera device can sense and communicate if there is any person in the room. The back-end information processing engine inside the automation controller contains a set of rules and the corresponding actuations which are basically *If-then* rules on the information observation data. If the rule evaluates to true, necessary action is triggered. Consider the following rules.

- ***Rule 1***: Turn on AC if the room is not empty and (room temperature $\geq 30^oC$ or humidity $\geq 85\%$);
- ***Rule 2***: Turn off the light if (the room is empty or the room is bright).

We define the following Boolean predicates.
p_1 : Room temperature $>= 30\ ^oC$; p_2 : Humidity $>= 85\%$;
p_3 : The room is empty; p_4 : Room is bright.
The devices evaluate the values (*true* / *false*) of the predicates, before sending the update to the back-end controller. The communication is done using the Boolean predicates, which takes fewer message bits, as opposed to transmitting the exact value of the observations. In terms of the predicates, we have

- *Rule 1*: $(p_1 + p_2).\bar{p_3}$: Turn on AC
- *Rule 2*: $p_3 + p_4$: Turn off light.

Here, . represents the AND operator and + represents the OR operator. Table I contains a randomly generated snapshot of 5 time instants. Each row in the table shows the values assigned to the predicates. Each entry in the table contains a 0 or 1 depending on whether the corresponding Boolean predicate is

978-3-9815370-8-6/17/$31.00

TABLE I. Random snapshot of 5 time instants and Transmission profile

Time Instant	p_1	p_2	p_3	p_4	p_1	p_2	p_3	p_4
1	1	1	1	1	1	1	1	1
2	1	0	1	0	X	X	X	X
3	0	0	1	1	X	X	X	X
4	0	0	1	0	X	X	X	X
5	1	1	1	1	X	X	X	X
Total	3	3	1	5	1	1	1	1

true or false. For example, if Room temperature $>= 30\ ^oC$, the predicate p_1 is 1, otherwise p_1 is 0.

In a classical transmission scheme, whenever any predicate changes from its current value, the value of the changed predicate is transmitted. Considering the snapshot in Table I, total 12 transmissions are required as shown as the total number of changes in Columns 2-5 of Table I.

We observe that not all predicates need to be observed / transmitted or processed at all times, even if they have changed. Consider Rule 1. It can be observed that if the room is empty, then the first rule can never be true. Once the information that the room is empty is received, the back-end server does not need any other observation about room temperature, humidity to evaluate the first rule. If we observe the second rule, we see that only one predicate is enough to decide whether to turn off the light. Based on the preprocessing data and the current snapshot, the back-end server communicates to the sensors the change of which predicates are needed to evaluate the rules. With our proposed optimizations, 4 transmissions are required, as shown in Columns 6-9 of Table I, where each cell entry denotes the value transmitted to the back-end (X indicates no transmission). In the following, we formalize our proposal.

III. Detailed Methodology

Our model assumes the following operational workflow. A set of sensors (i.e. the information sources or collectors) collect information about a set of parameters. Based on the values observed, they evaluate and transmit the truth values of a set of predicates. The observations may be of arbitrary data type but the predicates are Boolean. The sensors communicate with a back-end controller through a reliable network. The back-end controller has a set of predicate logic [9] rules $\hat{\mathcal{C}}$ and corresponding actions. Based on the values of the predicates, the back-end evaluates the rules and based on the evaluation status of the rules, triggers necessary actuations.

At the back-end, before the network is initialized, the back-end pre-processes and at run time communicates to the sensors, which predicates (and hence, which observations) are needed and at what instants. Based on the values received from the sensors, the back-end evaluates the rules and takes necessary steps. At the sensor end, the necessary observations at the appropriate instants (as communicated to them by the back-end) are made, and transmitted if needed. We illustrate our data dissemination optimization strategies in the following.

Optimization Strategies

The intuition behind the optimization techniques is to analyze the back-end rule repository and derive the effective importance of the predicates (in other words, the observed parameters) with respect to the actuation of the rule actions. Let $\mathcal{P}$ be the set of predicates and $\hat{\mathcal{C}}$ be the set of rules.

Definition 3.1: **Cofactor [10]:** The positive (negative) cofactor of a rule $\mathcal{C} \in \hat{\mathcal{C}}$ defined over a set of Boolean predicates $\mathcal{P} = \{p_1, p_2, \ldots p_n\}$ with respect to a predicate $p_i \in \mathcal{P}$ is obtained by substituting 1 (true) or 0 (false) in $\mathcal{C}$. ■

The positive cofactor, denoted by $\mathcal{C}_{p_i}$ is obtained as $\mathcal{C}_{p_i} = \mathcal{C}(p_1, p_2, \ldots, p_i = 1, \ldots, p_n)$. The negative cofactor $\mathcal{C}_{\bar{p_i}}$ is obtained as $\mathcal{C}(p_1, p_2, \ldots, p_i = 0, \ldots, p_n)$. For a rule $\mathcal{C}$, for a predicate p_i, if $\mathcal{C}_{pi} \oplus \mathcal{C}_{\bar{pi}} = 1$, every change of p_i is needed always. Here $\oplus$ denotes the XOR operator. The xor of the cofactors designates that every change (from 0 to 1 or 1 to 0) of p_i is *critical* to evaluate $\mathcal{C}$.

Cofactors with respect to multiple predicates can be obtained similarly. As an example, the cofactor $\mathcal{C}_{pq}$ is derived by substituting the value of $p = 1$ and $q = 1$ in $\mathcal{C}$. Co-factor analysis involving multiple predicates can help in reducing message transmissions even further.

A. Data transmission optimizations based on rule analysis:

Consider a rule $\mathcal{C}$ which is defined over a set of predicates $\mathcal{P}_C$. Also consider a set of predicates $\{p_{i_1}, p_{i_2}, \ldots, p_{i_l}\} \subset \mathcal{P}_C$, such that if we substitute the current values of $\{p_{i_1}, p_{i_2}, \ldots, p_{i_l}\}$ in $\mathcal{C}$, $\mathcal{C}$ evaluates to either 0 or 1 and no subset of $\{p_{i_1}, p_{i_2}, \ldots, p_{i_l}\}$ can lead to a constant value for $\mathcal{C}$. In this case, to evaluate $\mathcal{C}$, the back-end does not need to know the status of the predicates in $\mathcal{P} \setminus \{p_{i_1}, \ldots, p_{i_l}\}$ until one predicate from $\{p_{i_1}, p_{i_2}, \ldots, p_{i_l}\}$ changes, where $\setminus$ is the set difference operation. If this information is made available to the sensors, it may result in a significant reduction in the number of transmissions, and this is what we use in our method. If a sensor observes all these remaining set of predicates $\mathcal{P} \setminus \{p_{i_1}, \ldots, p_{i_l}\}$, it can judiciously decide when to send observations next, which in turn leads to reduction in the number of transmissions through the network. This can be done in the preprocessing step itself when the network is initialized, and kept as a look-up table at the sensor end or at run-time when the rule evaluates to 1. If these set of predicates are not co-observed, in other words, multiple sensors capture their values, the back-end can send this information to the corresponding sensors at run-time. As long as the back-end does not receive a changed value of any predicate from $\{p_{i_1}, p_{i_2}, \ldots, p_{i_l}\}$, it does not poll the other sensors, and hence, in turn, observations and transmissions are both reduced.

Example 3.1: Consider a rule $\mathcal{C}$ over the predicates p_3, p_4 as: $\mathcal{C}(p_1, p_3, p_4) = (p_1.p_3) + p_4$. $\mathcal{C}_{p_4} = 1 + p_4 = 1$, $\mathcal{C}_{\bar{p_4}} = p_1.p_3 + 0 = p_1.p_3$. So, when $p_4 = 1$, the back-end does not need the status of p_1, p_3 until p_4 becomes 0. ■

The optimizations discussed above can be done at initialization time by processing the rules and appropriately passing on the relevant information to the sensors. Additionally, at run time, the back-end may send further updates as and when necessary, to abstain the sensors from observing / transmitting information. The sensors take this information over and above the information they had at the initialization step, and this leads to further reductions in transmissions as we have explained above. We now delve on some specific run-time strategies that can lead to further reductions by selectively filtering on the predicates to be observed or transmitted, depending on the rule

semantics. Depending on the current snapshot, a rule may need one of the multiple sets of predicates on which it depends, which we formalize as the notion of the *interest set* below.

Definition 3.2: **Interest Set:** The interest set $\mathcal{I}_c$ of a rule $\mathcal{C}$ is a set of predicates which is sufficient to evaluate the truth value (true / false) of $\mathcal{C}$ based on the current snapshot. ■

Example 3.2: Consider the rule $\mathcal{C} = p_1 + p_2.p_3 + p_4$ and a snapshot $(p_1 = 1, p_2 = 1, p_3 = 1, p_4 = 0)$. To evaluate $\mathcal{C}$, any of the sets $\{p_1\}, \{p_2, p_3\}$ is sufficient to take the decision. Hence the *interest set* of $\mathcal{C}$ consists of $\{p_1\}$ and $\{p_2, p_3\}$. ■

An arbitrary choice of the interest set may end up in a large number of predicate transmissions though the optimal set is quite small. Additionally, for a rule, there may exist more than one interest set. We now define a minimal interest set.

Definition 3.3: **Minimal Interest Set:** The Minimal Interest Set (MIS) of a set of rules is the minimal set of predicates that is sufficient to evaluate all the rules to a constant value. ■

The MIS computed dynamically may not always be unique. The MIS can also be computed at initialization for a given set of rules and appropriate sensors may be notified about what to transmit at what instants. This may however, need a substantial resource requirement at the sensor end, and may not be feasible. In contrast, we propose here, how to compute the minimal interest sets based on a value snapshot, such that the back-end can communicate the necessary information to the sensors and poll them when needed.

Example 3.3: Consider the following set of rules $\mathcal{C}_1 = p_1 + p_2 + p_3, \mathcal{C}_2 = p_2 + p_4 + p_5, \mathcal{C}_3 = p_4 + p_6$ and a snapshot $p_1 = 1, p_2 = 1, p_3 = 0, p_4 = 1, p_5 = 0, p_6 = 1$. Some of the interest sets that can help us evaluate the rules are $\{p_1, p_2, p_4\}, \{p_1, p_2, p_6\}, \{p_2, p_4\}, \{p_1, p_4\}, \{p_2, p_6\}$. The minimal interest set in this case is one of the following sets $\{p_2, p_4\}, \{p_1, p_4\}, \{p_2, p_6\}$. From the example, we can see that the minimal interest set is not unique. ■

Algorithms 1 and 2 present the strategy for minimal interest set selection. The idea is to compute the interest sets of each rule and then select the minimum combination. Analyzing the structure of the rules, we form a Boolean formula that captures the requirements of each rule and use a Boolean satisfiability solver [11] to extract all satisfying solutions. Any of the solutions corresponds to an interest set and we choose the one with the minimal cardinality.

Algorithm 1 RunTimeInterestSetSelection

1: Input: Current Predicate Snapshot, Set of rules $\hat{\mathcal{C}}$
2: Output: $\mathcal{U}$ ▷ *$\mathcal{U}$ is set of predicates needed by the back-end*
3: $\mathcal{Q} = \hat{\mathcal{C}}$.
4: **for** $\mathcal{C} \in \mathcal{Q}$ **do**
5: Compute the interest set of $\mathcal{C}$.
6: **end for**
7: All predicates p, such that $\mathcal{C}_p \oplus \mathcal{C}_{\bar{p}} = 1$, for any rule $\mathcal{C}$, are included in $\mathcal{U}$;
8: ▷ $\oplus$: *XOR operator; these are the predicates for which every observation is needed*
9: If the interest set of a rule contains only one component, all elements of that set are included in $\mathcal{U}$ and the corresponding rule is removed from $\mathcal{Q}$.
10: If an interest set of a rule $\mathcal{C} \in \mathcal{Q}$, contains a component whose elements are already included in $\mathcal{U}$, then the corresponding rule is removed from $\mathcal{Q}$.
11: MinimalInterestSetSelection($\mathcal{Q}, \mathcal{U}$)

Example 3.4: In this example, we show how the algorithm MinimalInterestSetSelection works. Let, $\mathcal{C}_1 = p_1 + p_2.p_3 + p_4$; $\mathcal{C}_2 = p_2 + p_5$; $\mathcal{C}_3 = p_3 + p_6.p_7$ be three rules. Let the current snapshot be $(p_1 = 1, p_2 = 1, p_3 = 1, p_4 = 0, p_5 = 1, p_6 = 1, p_7 = 1)$. So, the interest set of $\mathcal{C}_1$ is $\{p_1\}, \{p_2, p_3\}$, the interest set of $\mathcal{C}_2$ is $\{p_2\}, \{p_5\}$ and the interest set of $\mathcal{C}_3$ is $\{p_3\}, \{p_6, p_7\}$. After step 7 of Algorithm 2, the Boolean formula obtained is: $\mathcal{F} : (x_{p_1} \vee (x_{p_2} \wedge x_{p_3})) \wedge (x_{p_2} \vee x_{p_5}) \wedge (x_{p_3} \vee (x_{p_6} \wedge x_{p_7}))$. Consider all satisfiable solutions of $\mathcal{F}$ and we choose the solution $(x_{p_1}, x_{p_2}, x_{p_3}, x_{p_4}, x_{p_5}, x_{p_6}, x_{p_7}) :: (0, 1, 1, 0, 0, 0, 0)$ and include p_2, p_3 in $\mathcal{U}$. This is the minimal set of predicates the back-end needs to know in this case. ■

Algorithm 2 MinimalInterestSetSelection

1: Input: $\mathcal{Q}, \mathcal{U}$
2: Output: $\mathcal{U}$ ▷ *Set of predicates needed by the back-end*
3: **for** $\mathcal{C} \in \mathcal{Q}$ **do**
4: Construct a Boolean formula ($\mathcal{F}$) as follows:
5: For each predicate p, a variable x_p is defined as:
$x_p = 0$, if p is being considered as an element of $\mathcal{U}$
$= 1$, otherwise
6: Combine each Boolean variable corresponding to each member of every interest set of $\mathcal{C}$ using AND to get a product term.
7: If one of the predicates in a set is already included in $\mathcal{U}$, then the value of its corresponding Boolean variable is always 1, hence we can remove the Boolean variable from the formula.
8: All product terms in one interest set are combined using OR operator to get a sum of product term.
9: All sum of product terms corresponding to $\mathcal{C}$ are combined using AND operator. i.e., $\mathcal{F} = \mathcal{F}$ AND (sum of product term corresponding to $\mathcal{C}$).
10: **end for**
11: Get all possible satisfiable solutions of the formula $\mathcal{F}$. Choose the solution whose cardinality of number of 1s is minimum. In case of tie, break it arbitrarily. The corresponding predicates in $\mathcal{U}$ are included.

Lemma 3.1: Algorithm 2 gives a minimal interest set.
Proof: The proof follows from the construction procedure. ■

B. Optimizations based on rule clustering:

The strategy above involves a satisfiability solving step, which may not scale if there are too many interest sets and we have a large formula to handle. A further optimization can be done by handling rules in clusters and creating *rule families* based on the predicates they have in common, instead of dealing with each in isolation. Rule families are mutually exclusive and collectively exhaustive. The predicate set corresponding to the rule families are pairwise disjoint. Algorithm 3 presents an approach for rule family generation. Since the set of predicates and rules are both finite, the algorithm terminates.

Algorithm 3 GenerateRuleFamily

1: Input: $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k\}$
2: Output: Rule families $\{\mathcal{C}_{F_1}, \mathcal{C}_{F_2}, \ldots, \mathcal{C}_{F_m}\}$
3: Initialize $i = 1; p = p_1$. A flag variable is set to false for each rule and predicate.
4: Initialize $\mathcal{C}_{F_1} = \phi; \mathcal{P}_{F_1} = p_1$
5: **repeat**
6: $\mathcal{C}_{F_i} = \mathcal{C}_{F_i} \cup \{\mathcal{C}_j$ such that $\mathcal{C}_{jp} \oplus \mathcal{C}_{j\bar{p}} \neq 0\}$
7: Mark the flag of p as true.
8: Revise the set $\mathcal{P}_{F_i}$. Include all predicates associated with each rule $\mathcal{C} \in \mathcal{C}_{F_i}$ and not already added in $\mathcal{P}_{F_i}$.
9: Mark flag of each rule $\mathcal{C} \in \mathcal{C}_{F_i}$ as true.
10: **if** $\mathcal{P}_{F_i}$ contains any predicate p_j with flag as false **then**
11: $p = p_j$
12: **else**
13: $i = i + 1$;
14: $p =$ A predicate whose flag is false; If no such p exists, terminate the procedure.
15: **end if**
16: **until** (All predicates or all rules are examined)

Example 3.5: Consider a set of 5 rules $\mathcal{C}_1 = p_1 + p_2.p_3 + p_4$, $\mathcal{C}_2 = p_1 + p_5$, $\mathcal{C}_3 = p_5 + p_6$, $\mathcal{C}_4 = p_7 + p_8$ and $\mathcal{C}_5 = p_7 + p_9$. The rule families are $\mathcal{CF}_1 = \mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$, $\mathcal{CF}_2 = \mathcal{C}_4, \mathcal{C}_5$. ■
Once the rule clusters are formed, we compute the minimal interest sets for each cluster using Algorithm 2 as earlier. The

main bottleneck for this procedure is still the satisfying solution generation step, which is computationally expensive.

C. Optimizations using prediction:

Our final proposal is based on value prediction. The interest set of all the rules in $\hat{\mathcal{C}}$ are computed at first. The final set $\mathcal{U}$ of the predicates that need to be captured by the back-end server, generated in each round, contains all the predicates p for which $\mathcal{C}_p \oplus \mathcal{C}_{\bar{p}} = 1$, for any rule $\mathcal{C} \in \hat{\mathcal{C}}$ and the predicates from the interest set corresponding to the rules having singleton interest sets satisfying the current snapshot at that round. Rules for which at least one interest set is considered in the final set $\mathcal{U}$ are excluded, the remaining participate in this prediction. In other words, rules which participate in this prediction algorithm have two or more interest sets, and the decision to select one of them is what this heuristic does. We use a simple predicate history outcome based prediction strategy. We wish to select interest sets which remain unchanged for a long time, and thereby, can lead to more transmission savings. For each rule, we choose an interest set consisting of predicates that change less frequently. This leads to a further optimization in our methodology. However, we do not lose out on any information or rule evaluation. If the predicted set does change at runtime, we poll the remaining sensors for the other predicates.

For each predicate, we store a history h at the back-end, where h denotes the number of changes (0 to 1 or 1 to 0 is considered as a change) that the back-end has noticed over the history of computation. A lower value of h implies the predicate changes less frequently. An interest set usually contains one or more predicates. In an interest set, predicate(s) with low history values are considered as the representative predicates for that interest set. While selecting an interest set for each rule, we select the one whose representative predicate has lowest history value. As is intuitively obvious, the final set $\mathcal{U}$ that is generated in each round using prediction is correct but may not be optimal in terms of cardinality.

IV. Experimental Result

We tested our method on synthetically generated use cases by randomly varying the rules and the number of predicates. The results are shown in Table II. We considered a total window of 15 snapshots over time. We present the effect of different optimization procedures through our experiments. Each row of the table corresponds to the results obtained on one experimental dataset. Columns 1 and Column 2 of Table II represent the number of randomly generated predicates and rules respectively. Column 3 shows the number of different clusters found in the rule set. We compare our method with the asynchronous transmission scheme with respect to the number of transmissions. We also present comparative results with [8]. Column 4 presents the number of transmissions in a usual asynchronous transmission scheme. Column 5 presents the number of transmissions with minimal interest set selection (Case 1) and finally Column 6 shows the number of transmissions using prediction (Case 2). Column 7 shows the number of transmissions without runtime optimization as described in [8] (Case 3). As can be seen from the results, the number of transmissions reduces with our optimizations (Cases 1 and 2) compared to both the usual case and [8]. In some cases, the reduction is substantial, whereas in some cases, it is negligible. These optimizations can lead to the same performance as in the asynchronous method in the worst case if the rule set does not lend itself to any of the optimizations we propose in this work. It is guaranteed theoretically and also as seen through the experiments, that our method (Case 1 and Case 2) does not transmit more than the asynchronous method. However, the prediction step being non-deterministic, may or may not, yield the desired improvement.

TABLE II. Test-case Statistics (#Tx = No. of Transmissions)

No. of predicates	No. of rules	No. of clusters	#Tx (usual case)	#Tx (Case 1)	#Tx (Case 2)	#Tx (Case 3)
14	6	1	716	404	398	663
28	28	1	1420	1111	957	1276
38	31	2	1848	1198	1099	1692
46	38	2	2247	1843	1554	2102
49	103	2	2416	2367	2208	2416
60	27	5	3013	1306	1149	2507
68	68	2	3340	2424	2079	3242
74	77	2	3651	2470	2113	3494
87	92	2	4335	3290	2876	4131
98	116	2	4802	3728	3326	4563

V. Conclusion

In this paper, we present a novel architecture for reducing data transmission for a large scale sensor enabled cyber-physical systems. Experimental results show promising improvements in the number of transmissions. With information processing and sensing systems gaining widespread popularity in recent times, we believe our work will have interesting applications.

Acknowledgment

This work is supported, in part, by a special PPEC-grant (2012-2017) to Nanotechnology Research Triangle provided by Indian Statistical Institute, Kolkata.

References

[1] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM, 2012, pp. 13–16.

[2] G. Cugola and A. Margara, "Processing flows of information: From data stream to complex event processing," *ACM Comput. Surv.*, vol. 44, no. 3, pp. 15:1–15:62, Jun. 2012.

[3] B. Babcock *et al.*, "Load shedding for aggregation queries over data streams," in *Proceedings of the 20th International Conference on Data Engineering*, ser. ICDE, 2004, p. 350.

[4] Y. Xu *et al.*, "Prediction-based strategies for energy saving in object tracking sensor networks," in *IEEE MDM*, 2004, pp. 346–357.

[5] J. Yick *et al.*, "Wireless sensor network survey," *Computer networks*, vol. 52, no. 12, pp. 2292–2330, 2008.

[6] D. Kossmann, "The state of the art in distributed query processing," *ACM Computing Surveys (CSUR)*, vol. 32, no. 4, pp. 422–469, 2000.

[7] S. Adali *et al.*, "Query caching and optimization in distributed mediator systems," in *ACM SIGMOD*, vol. 25, no. 2, 1996, pp. 137–146.

[8] S. Chattopadhyay, A. Banerjee, and N. Banerjee, "A data distribution model for large-scale context aware systems," in *MobiQuitous*, 2013.

[9] S. J. Russell *et al.*, *Artificial intelligence: a modern approach*. Prentice hall Upper Saddle River, 2003, vol. 2.

[10] G. D. Hachtel and F. Somenzi, *Logic Synthesis and Verification Algorithms*, 1st ed. Norwell, MA, USA: Kluwer Academic Publishers, 2000.

[11] Princeton University, "zChaff." [Online]. Available: https://www.princeton.edu/~chaff/zchaff.html