Maximum Achievable Throughput in a Wireless Sensor Network Using In-network Computation for Statistical Functions

Rajasekhar Sappidi, André Girard and Catherine Rosenberg

***Abstract*—Many applications require the sink to compute a function of the data collected by the sensors. Instead of sending all the data to the sink, the intermediate nodes could process the data they receive to significantly reduce the volume of traffic transmitted: this is known as in-network computation. Instead of focusing on asymptotic results for large networks as is the current practice, we are interested in explicitly computing the maximum achievable throughput of a given network when the sink is interested in the first M statistical moments of the collected data. Here, the k-th statistical moment is defined as the expectation of the k-th power of the data.**

Flow models have been routinely used in multi-hop wireless networks when there is no in-network computation and they are typically tractable for relatively large networks. However, deriving such models is not obvious when in-network computation is allowed. We develop a discrete-time model for the real-time network operation and perform two transformations to obtain a flow model that keeps the essence of in-network computation. This gives an upper bound on the maximum achievable throughput. To show its tightness, we derive a numerical lower bound by computing a solution to the discrete-time model based on the solution to the flow model. This lower bound turns out to be close to the upper bound proving that the flow model is an excellent approximation to the discrete-time model. We then provide several engineering insights on these networks.

***Index Terms*—Sensor networks, in-network computation, throughput, flow model.**

I. Introduction

Sensor networks are an emerging class of networks with many interesting applications and configuring them to obtain the best performance is of prime importance. In several applications, there is a sink that requires a function of the information gathered by all the sensors for further decision making, for instance, a fire-alarm system where the sink is monitoring the maximum temperature in a building. The standard approach known as *convergecast* or data collection in the literature, is to send all the individual measurements to the sink that then computes the required function. However, the volume of transmitted data can be significantly reduced by allowing the intermediate nodes to process the data they have received and send only an essential summary to the next node along the route to the sink. This is known as *in-network computation* in the literature. This approach could potentially lead to significant improvements in the network performance in terms of lifetime, delay and throughput.

Rajasekhar Sappidi and Catherine Rosenberg are with Electrical and Computer Engineering, University of Waterloo, Waterloo, Ontario, Canada.
André Girard is with INRS-EMT, Montréal, Québec, Canada.

The nature of the function that the sink is interested in computing plays an important role in determining the performance gains achievable utilizing in-network computation compared to convergecast. Computing the first M moments of the data is of interest in many sensor network applications in order to reconstitute the distribution of the measured phenomenon via its moment generating function. The accuracy of this process depends on M with larger M resulting in better accuracy. For instance, consider an application where a wireless sensor network is deployed to measure the level of air pollution in a city at a given time. Factors such as wind, time of the day, cloud cover, etc., could effect the measurement at any given point. Hence, we measure the pollution at multiple points in the city at a given time. The average of the ensemble of data collected by all the sensors at a given time is a very accurate indication of the level of air pollution at that time in the city and the variance of this collected data shows how "non-uniform" the pollution is at that time. Note that in this example we are interested in measuring a phenomenon (pollution) in a potentially noisy environment.

Formally, the k-th moment of a discrete random variable X, μ_k is defined as the expectation of X^k, i.e.,

$$\mu_k = E(X^k) \tag{1}$$

We can estimate μ_k as the average of the k-th powers of all the observed data, i.e.,

$$\mu_k \simeq \frac{\sum_{i=1}^{N} X_i^k}{N} \tag{2}$$

In this paper, we explicitly compute the maximum achievable throughput (defined in the next sub-section) in a wireless sensor network when the sink is interested in computing the first M moments of the data collected by the sensor nodes.

We focus on sensor networks that use conflict-free scheduling and consider the physical model of interference. We are interested in the optimal joint routing and scheduling that gives maximum throughput when in-network computation is allowed. Although random access protocols have many desirable features like robustness to node and link failures, low maintenance, etc., they are usually outperformed by optimal centralized protocols. Even if centralized protocols are not implementable in certain contexts, e.g., random deployment of sensor nodes in war zones, etc., they can serve as a benchmark, i.e., provide an upper bound on the performance and guide the design of better distributed random access protocols.

In the next sub-section, we explain the mechanism of in-network computation and how it leads to data aggregation.

A. In-Network Computation and Data Aggregation

Assume that there are n nodes in the network and that time is slotted. Also assume that every node i makes a measurement periodically after every δ time slots. Let the w-th measurement of node i be $x_i(w)$. Let $\mathbf{x}(w)$ denote the collection of the w-th measurements made by all the nodes. Assuming that all the nodes collect data at the same time, we call the collection $\mathbf{x}(w)$, the w-th *wave* of information. In applications like the fire-alarm system, the sink is interested only in some function $f(\mathbf{x}(w))$ of these measurements. We say that the sink has received the wave w if it is able to compute the function of the data in wave w. We define *throughput* as the rate at which the sink receives the waves. This throughput can be viewed as the "monitoring rate" of the phenomenon by the sensor network. However, we use throughput in this paper as it is the term traditionally used in convergecast.

Consider the instance when the sink is interested in $E[\mathbf{x}(w)]$, the average of the measurements over all nodes in a wave, i.e., $f(\mathbf{x}(w)) = \sum_i x_i(w)/n = S(w)/n$. In the convergecast mode, all the raw data $x_i(w)$ are individually sent to the sink where the average is computed. However, the sink can compute this average even if it has received only partial sums instead of all the individual raw data. Thus, when in-network computation is allowed, each time a node i is allowed to transmit (as defined by the schedule), it will aggregate all the data of the wave, w, it has in its buffer (which may include its own data $x_i(w)$) into one (partial) sum $S_i(w)$ and send it as one packet to the next node along the path to the sink. Note that these partial sums must be computed only among the measurements made at the same time, i.e., each wave of information must be processed separately. Data from one wave cannot be aggregated with that of another wave, i.e., there must not be mixing among different waves. This is an important condition that we will have to take into account later on.

Computation of more complicated functions, such as the first two moments, i.e., variance $\text{var}(\mathbf{x}(w)) = \sqrt{R(w) - S(w)/n}$ and mean, can also be delegated to the nodes in the network. In this case, the sink needs two pieces of information: $S(w)$, the sum of the $x_i(w)$ and $R(w)$, the sum of the squares $x_i^2(w)$. Thus, in-network computation creates two *types* of partial sums, i.e., a node i now computes not only the partial sum $S_i(w)$ but also the partial sum of squares $R_i(w)$ and transmits these two values and sends each value as one packet. Similarly, the sink can compute the first M moments of the data if it receives just the M sums of the first M powers of the data.

Data aggregation is the main reason for allowing in-network computation since it can significantly reduce the total volume of the data transmitted compared to convergecast. This comes at the expense of an additional complexity since processing has to be performed at intermediate nodes. It is interesting to note that aggregation must not be performed automatically at all nodes when we want to compute more than one moment. To see this, consider a case where the sink is interested in the first two moments. Assume that some node i which is allowed to transmit has only its own data $x_i(w)$ in its buffer. If node i performs in-network computation blindly, it creates two packets corresponding to sum, $S_i(w) = x_i(w)$ and sum of squares, $R_i(w) = x_i^2$ and would require two time slots to send these packets. But, it is more efficient if i just sends its data as such and delays in-network computation to further nodes along the path to the sink. This can be done easily by allowing the existence of three (instead of just two) different types of packets per wave, namely raw data, partial sum and partial sum of squares. More generally, when the sink is interested in computing the first M moments, there would be $M+1$ different types of packets per wave.

In the next sub-section, we summarize our contributions and give an overview of this paper.

B. Our Contributions

In this paper, instead of focusing on asymptotic results for large networks as is the current practice [1], we explicitly compute the maximum achievable throughput when the sink is interested in the first M moments of the collected data. Flow models are routinely used in multi-hop wireless networks when there is no in-network computation and are typically tractable for relatively large networks. However, deriving such models is not obvious when in-network computation is allowed. We develop a discrete-time model for the real-time network operation and perform two transformations to obtain a flow model that keeps the essence of in-network computation. This gives an upper bound on the maximum achievable throughput. To show its tightness, we derive a numerical lower bound by computing a solution to the discrete-time model based on the solution to the flow model. This turns out to be close to the upper bound proving that the flow model is an excellent approximation to the discrete-time model. We describe our methodology in more detail in Section III-B.

In summary, our main contributions in this paper are:

1) A novel non-intuitive modeling of the in-network computation when the sink is interested in the first M moments.
2) A rigorous derivation of a flow model and a proof that its solution is an upper bound on the maximum achievable throughput using in-network computation.
3) A method to compute a lower bound on the maximum achievable throughput.
4) Numerical evidence that the two bounds are close. This validates our flow model formulation.
5) Insights on the value of in-network computation.

Throughout the paper, we use the physical interference model and we make no restricting assumptions on the channel gains beyond the fact that they should be quasi time-invariant.

The rest of the paper is organized as follows. We start with a review of the related work in Section II. In Section III, we first present the models for the wireless sensor network and the interference. We then present our methodology to solve this problem. Next, we present the discrete-time model and derive the flow model from it. In Section IV, we validate our flow model formulation and present numerical results on some random networks and conclude in Section V.

II. Related Work

Optimizing throughput or delay in sensor networks for the two modes of data gathering, viz., convergecast [2]–[4] and in-network computation [1], [5]–[7] has received a lot of attention in recent years. In these studies, the two most commonly used models for wireless interference are the protocol model and the physical model [8]. In the protocol model, a receiver can successfully decode the message intended for it if and only if it is within the transmission range of the sender and there are no other nodes transmitting within its interference range while in the physical model, a receiver can successfully decode if and only if its SINR exceeds a minimum threshold (see Section III-A for more details). Next, we first discuss the work on throughput and delay with no in-network computation followed with the one that considers in-network computation.

(A) **Throughput with no in-network computation:** Gupta and Kumar [8] studied the capacity of a wireless network and gave asymptotic results for the achievable throughput under both the protocol and physical models of interference. Tassiulas and Ephremides [9] have proposed the *backpressure* algorithm for scheduling in wireless multi-hop networks and proved that it is throughput-optimal. Jain *et al* [3], Stuedi *et al* [4], Karnik *et al* [2] have all studied the problem of explicitly finding the maximum achievable throughput of a fixed wireless network. Lower and upper bounds on the maximum achievable throughput are obtained in [3]. A linear program formulation for explicitly computing the optimal throughput of a network under the physical interference model is given in [2] and Luo *et al* [10] used linear programming techniques like column generation to solve this problem efficiently for larger networks. In this paper, we are interested in a similar formulation when the network allows in-network computation.

(B) **Delay with no in-network computation:** Another performance measure that is considered in the literature is delay. Florens *et al* [11] considered the minimization of delay for the convergecast problem under a protocol interference model. They assume that there is a single communication channel and that time is slotted such that every slot is long enough to send just one packet of data. They give closed form expressions for delay in terms of the number of time slots for some special topologies like line, multi-line, etc. However, they have considered only a restricted set of network topologies where the distance between any two neighboring nodes is the same. Gargano and Rescigno [12] also consider the problem of minimizing delay in the convergecast problem. They assume that the nodes have directional antennas so that any two links can be active at the same time if they do not have any nodes in common. In effect, the interference model they consider uses only transceiver constraints, i.e., a node can either receive or transmit but cannot do both at the same time and if it is receiving, it can receive from only one node. They present an algorithm using collision-free graph coloring that gives the optimal solution for the overall delay. Chen *et al* [13] also studied the problem of minimizing the delay under a variation of the protocol interference model. They showed that finding the optimal schedule for a general graph is NP-hard and proposed an heuristic that performs better than the RTS/CTS scheme. In our paper, although the discrete-time model we formulate could be used to find the optimal schedule that minimizes delay, our focus is on computing the maximum achievable throughput when the network allows in-network computation.

(C) **Throughput with in-network computation:** The earliest study on in-network computation was made by Tiwari [14]. They give lower bounds on the communication complexity, defined as the number of bits that have to be sent in the worst case to compute the function. Since the performance measure in this study is communication complexity, it does not matter whether the links are wired or wireless. Also, the functions considered are for just two sources. Giridhar and Kumar [1] studied the problem of distributively computing the maximum rate at which symmetric functions of sensor measurements can be computed and communicated to the sink. They classified the symmetric functions into two subclasses, type-sensitive and type-threshold. They show that the type-sensitive functions like mean, mode, median can be computed at $O(1/n)$ rate in collocated networks and at $O(1/\log n)$ rate in random planar multihop networks while the type-threshold functions like min, max, range can be computed at $O(1/\log n)$ rate in collocated networks and at $O(1/\log\log n)$ rate in random planar multihop networks. Note however that these sub-classes do not cover all the symmetric functions. These are asymptotic results and they do not compute the maximum achievable throughput explicitly which is the problem we consider in this paper.

(D) **Delay with in-network computation:** Sudeep and Manjunath [15] studied the problem of computing MAX in an unstructured sensor network assuming the protocol model of interference. In an unstructured network, the nodes need not have unique identifiers. They show that in $O(\sqrt{n/\log n})$ slots, MAX can be made available at the sink with high probability. This is interesting as the best coordinated protocol given in [16] also takes the same $O(\sqrt{n/\log n})$ slots to compute MAX. Most-Aoyama and Shah [17] give an elegant distributed scheme based on exponential random variables to compute the class of *separable* functions such as SUM in a wireless network using gossip protocols. Ghosh *et al* [7] have considered in-network computation where the sink is interested in a function like mean, min or max, i.e., aggregation results in a single packet regardless of the number of input packets (perfect aggregation). They assume multiple channels and the protocol model of interference and give some tree-based heuristics to minimize delay and maximize throughput for the convergecast and the perfect aggregation problems. There is a lot of similar recent work on perfect aggregation [18]–[22] that consider the problem of minimizing delay. In this paper, we consider aggregation when the sink is interested in the first M moments, of which the perfect aggregation is an example.

(E) **Others:** The modeling of compressed sensing in [23] is very similar to the way we model the computation of M moments. Using compressed sensing, the sink could retrieve all the raw data unlike in our current problem where it only retrieves the M moments of the data. However, they consider minimizing energy consumption as their performance measure and do not consider finding the optimal joint routing and scheduling that maximizes throughput which we consider.

To the best of our knowledge, there are no previous results on explicitly computing the maximum achievable throughput of a given network that allows in-network computation. In the next section, we define and formulate this problem when the sink is interested in the first M moments of the data collected by the sensor nodes.

III. Problem Formulation

Consider a wireless sensor network in which the sensor nodes are measuring some quantity (e.g., temperature, pressure, etc.) and the sink is interested in computing the first M moments of these measurements. As mentioned in Section I-A, we assume that the sensor nodes are periodically making these measurements. Let the w-th measurement of node i be $x_i(w)$. Let $\mathbf{x}(w)$ denote the collection of the w-th measurement made by all the nodes. We call the collection $\mathbf{x}(w)$, the w-th *wave* of information. We say that the sink has received a wave if it is able to compute the first M moments of the data from all the nodes in that wave. Defining *throughput* as the rate at which the sink receives the waves, we are interested in explicitly computing the maximum achievable throughput when using in-network computation in a given wireless sensor network.

In this section, we present a systematic approach to solve this problem. We begin by introducing the network model including the wireless interference model.

A. Network Model

We assume that there is a static wireless sensor network with n nodes and that the channel gains are quasi time-invariant. In addition, there is a special node called the *sink* that collects and analyzes the data from all the sensor nodes. We also assume that the sink knows the positions of all the n nodes. Let them be labeled $0, 1, \dots n$ with 0 representing the sink. We assume that there is a single communication channel and every node communicates with its neighbors through a transceiver which can either transmit or receive but not do both at the same time. We assume that the time is slotted and each slot is long enough to transmit one packet of data. We assume that the data collected by the sensor nodes is real numbers each of which fits in a single packet. We also assume that all the arithmetic operations that we do on the data result in a real number which remains within the allowed range when represented in the binary format and thus still fits in a single packet of data. We do not consider power control and assume that all the nodes transmit with the same power P.

We model the wireless interference using the physical model which is based on Signal to Interference and Noise Ratio (SINR) [2], [8]. Experimental results [24] show that this models wireless interference more accurately than any other simpler model. Also, Iyer *et al.* [25] showed that the results obtained from simpler models may be qualitatively different from those obtained from the physical model. However, note that our technique could use any interference model.

A directed link from node i to node j is said to exist if $P_{i,j} \geq \beta N_0$ where $P_{i,j}$ is the total power received from node i at node j. Here N_0 is the thermal noise power and β is the minimum Signal to Noise Ratio (SNR) required for successful decoding of the message. Given a channel propagation model for $P_{i,j}$, we can compute the set of all feasible directed links $\mathcal{L}$ in the network using this condition. We make no restricting assumptions on the channel propagation model.

In a wireless network, even though all the links use the same channel, we can typically schedule a group of links to transmit at the same time without causing excessive interference to any intended receivers. We call such a subset of links an *independent set* (Iset). The interference model dictates which subset of links can be successfully activated at the same time. In the physical interference model, when two or more links are active on the same channel, every receiver considers the power from the transmitters other than its own as interference.

Under the physical interference model, a subset of feasible links, I, is an Iset only if they form a matching i.e.,

$$i \neq i' \wedge i \neq j' \wedge j \neq i' \wedge j \neq j' \quad \forall (i,j), (i',j') \in I \tag{3}$$

and all the corresponding receivers have an SINR greater than or equal to β i.e., for all the links (i,j) in the Iset I,

$$P_{i,j} \geq \beta \left[N_0 + \sum_{(i',j') \in I \setminus \{(i,j)\}} P_{i',j} \right]. \tag{4}$$

The sum in (4) is the total interference received by the destination node j of link (i,j) due to the transmissions on all other links (i',j') in I.

In the next sub-section, we discuss the methodology we adopt and the reasons behind it to solve the problem of computing the maximum achievable throughput using in-network computation in a given wireless sensor network.

B. Methodology

The network model described above naturally results in a discrete-time model formulation for computing the maximum achievable throughput for convergecast. However, the discrete-time model is typically an integer program that is not tractable and is solvable only for very small networks. For this reason, the standard approach taken in the literature is to ignore the discrete nature of the packets and assume that packets flow on links like fluids flow in pipes. This is known as the *flow model* and it does not require the notion of time slots and waves. In the following, we say that a problem formulation is a flow model of the network operation (e.g., convergecast or in-network computation) if it can be used to compute the maximum achievable throughput (exactly or approximately) and it does not include the notion of time slots or of waves. Flow models are typically derived from intuition without any reference to the underlying discrete-time model. The nonlinear multi-commodity flow model used to compute the minimum average packet delay in wired networks [26] is an example of such a model. Deriving a flow model for convergecast in wireless networks is also quite intuitive and they have worked well to explicitly compute the maximum achievable throughput [2], [3], [10]. In all these cases, solving the flow model has brought insights on the optimal network structure and operation that would have been impossible to obtain from the discrete-time model as we cannot solve it within a reasonable time.

Trying to derive a flow model straight away for in-network computation problem does not result in a tractable accurate formulation. The reasons are

1) There is no conservation of flows at the nodes since some packets effectively disappear when they are aggregated.
2) Aggregation can happen only between data belonging to the same wave. And this is not easy to ensure using flows.

Hence, we have to start by formulating the problem using the discrete-time model. There is more than one way to formulate a discrete-time model. We have tried the straightforward and intuitive way to formulate a discrete model for the case of $M = 1$. It was based on keeping track of the number of packets on a per-wave basis at each node. This led to constraints with product forms, i.e., non-linear constraints. We found that this problem is not tractable even for small networks. In one of our attempts to simplify this formulation, we tried to linearize these constraints but this led to a very poor approximation of the original problem. Also, as the number of required moments M increased, the complexity of these constraints increased as well.

We have been able to overcome these difficulties by taking a completely different view regarding the modeling of in-network computation in a sensor network. It is based on tracking of each measurement, also called a piece of information until it reaches the sink as opposed to the conventional method that focuses on tracking of packets at every node. In this problem, although packets are not conserved, information is. To understand how we model the in-network computation, consider the example of computing the *mean*, i.e., $f(\mathbf{x}(w)) = \sum_i x_i(w)/n$. Whenever a node has more than one packet of data from the same wave, it computes their sum and transmits only the sum when one of its link is activated. In our approach, although only the sum is being transmitted, we consider that all the information units making up the sum are transmitted in the same slot. We capture this in our model by allowing the node to create a *virtual information* unit per wave for every source that is involved in the sum and to transmit all the *virtual information* units in the same time slot. The model thus keeps track of all the information units until they reach the sink. Note that if the node is transmitting raw information, it can only send raw data from one source in one time slot. This novel modeling technique results in a discrete-time model which is a linear integer program.

However, this integer program is still hard to solve. Thus, we transform it into a flow model which could be solved optimally for much larger networks than that was possible for the discrete-time model. We show that this flow model provides an upper bound on the maximum achievable throughput. To validate the tightness of this upper bound, in Section IV, we derive a feasible solution to the discrete-time model based on the solution to the flow model and show that it yields a throughput that is close to the optimal throughput computed with the flow model. In summary,

1) We rigorously derive a flow model formulation.
2) We show that this flow model provides an upper bound on the maximum achievable throughput.
3) We show that this upper bound is tight by computing a near-optimal feasible solution to the discrete-time model based on the solution to the flow model. This validates our flow model and enables us to use it to study the effects of in-network computation on network performance.
4) When we apply our methodology (i.e., formulating the discrete model, followed by the derivation of the flow model) to convergecast, it resulted in the same flow model as the one given in the literature [2]. So, our method is consistent with what has been done intuitively in the past.
5) We also show that when $M = n$, where n is the number of nodes in the network, the flow model formulation for in-network computation is the same as the convergecast formulation. This further strengthens the confidence in our model as the $M = n$ case is intuitively convergecast.

Next, we present the discrete-time model formulation.

C. The Discrete Model

Assume that we want to compute the number of time slots T_m, taken to receive m waves at the sink. Recall that for every wave, there are $M + 1$ different types of data in the network, viz., the raw data (represented with $p = 0$), the partial sums (represented with $p = 1$), the partial sum of squares (represented with $p = 2$) and so on. Define the state variable $q_i^{s,w,p}(k)$ as a binary variable that is 1 if the information[1] from source s, wave w and type p is stored at node i at the end of time slot k. Hence, $q_i^{s,w,p}(k)$ tracks the information (raw or virtual) on a per wave, per source and per type basis. By convention, we assume that it is 0 when $k = 0$.

Assume that the sensors are collecting data every δ slots, i.e., $\sigma = 1/\delta$ is the input rate of the information. Let $v^w(k)$ be a binary value that is 1 if the sensor nodes are collecting data in time slot k. Thus, we have

$$v^w(k) \triangleq \begin{cases} 1 & \text{if } k = \delta w \\ 0 & \text{otherwise} \end{cases}$$

The decision variables in the model are all binary variables with the following definitions:

1) $x_{i,j}(k)$: If the link (i, j) is active in slot k, it is 1, otherwise it is 0. This is the scheduling variable.

[1]The information is raw data if $p = 0$ and virtual information if $p > 0$

2) $y_{i,j}^{s,w,p}(k)$: If the information from source s, wave w and type p is sent on link (i,j) in slot k, it is 1, else 0.
3) $u_i^{s,w}(k)$: This variable tracks if node i aggregates raw data from source s and wave w, i.e., in the model if it is 1, M virtual information units are created for this source out of the raw data.
4) $z_{i,j}^{w,p}(k)$: It is 1 if the information of type p in wave w is selected for transmission on link (i,j) in slot k, else 0.

By convention, the x, y and z variables are defined only when the link (i,j) exists.

With these definitions, the discrete-time model formulation $\mathcal{P}_d(m)$ is as follows. All constraints are over $w = 1 \dots m$, $s = 1 \dots n$ and $k = 1 \dots T_m$ wherever w, k or s appear, unless there is a summation over them. The constraints are also either over all nodes $i = 1 \dots n$ or over all links $(i,j) \in \mathcal{L}$ depending upon the constraint. Also, all the variables are non-negative.

$$\mathcal{P}_d(m): \quad \underset{\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{u}}{\text{Min}}\ T_m \tag{5}$$

subject to

$$T_m \geq \delta m \tag{6}$$

$$q_i^{s,w,p}(T_m) = 0 \tag{7}$$

$$\begin{aligned} q_i^{i,w,0}(k) &= v^w(k-1) + q_i^{i,w,0}(k-1) - u_i^{i,w}(k) \\ &\quad - \sum_j y_{i,j}^{i,w,0}(k) + \sum_j y_{j,i}^{i,w,0}(k) \end{aligned} \tag{8}$$

$$\begin{aligned} q_i^{s,w,0}(k) &= q_i^{s,w,0}(k-1) - u_i^{s,w}(k) \\ &\quad - \sum_j y_{i,j}^{s,w,0}(k) + \sum_j y_{j,i}^{s,w,0}(k) \quad \forall i \neq s \end{aligned} \tag{9}$$

$$\begin{aligned} q_i^{s,w,p}(k) &= q_i^{s,w,p}(k-1) + u_i^{s,w}(k) \\ &\quad - \sum_j y_{i,j}^{s,w,p}(k) + \sum_j y_{j,i}^{s,w,p}(k) \\ &\quad \forall p = 1 \dots M \end{aligned} \tag{10}$$

$$y_{i,j}^{s,w,p}(k) \leq q_i^{s,w,p}(k-1) \tag{11}$$

$$\sum_{w=1}^{m} \sum_{p=0}^{M} z_{i,j}^{w,p}(k) \leq x_{i,j}(k) \tag{12}$$

$$\sum_{s=1}^{n} y_{i,j}^{s,w,0}(k) \leq z_{i,j}^{w,0}(k) \tag{13}$$

$$y_{i,j}^{s,w,p}(k) \leq z_{i,j}^{w,p}(k) \quad \forall p = 1 \dots M \tag{14}$$

$$\sum_{(i,j)} x_{i,j}(k) + \sum_{(j,i)} x_{j,i}(k) \leq 1 \quad \forall i \tag{15}$$

$$\begin{aligned} Q\left(1 - x_{i,j}(k)\right) + P_{i,j} &\geq \beta \times \\ &\left[N_0 + \sum_{(i',j') \in I \neq (i,j)} P_{i',j} x_{i',j'}(k)\right] \end{aligned} \tag{16}$$

The constraints (7–10) are the information conservation equations which guarantee that all the information reaches the sink. Constraint (6) ensures that the network operates at least as long as is necessary to generate all the waves.

Constraint (7) is the final target state where all the information units have reached the sink. Constraints (8–10) keep track of the flow of information. Whenever $u_i^{s,w}(k) = 1$, the raw information from source s and wave w disappears at node i which behaves as a sink for this information, and appears at node i as M higher types of data for that source and wave. This is how the information units for the aggregated data created.

The actual constraints on the decisions to transmit are modelled by (11–14). Constraint (11) states the fact that we cannot transmit a unit of information on a link if this information is not in the buffer at the end of the previous time slot. Constraint (12) states that information of at most one type of one wave can be transmitted in one time slot by a node.

The actual effect of aggregation is represented by (13–14). Recall that some data might be transmitted as raw data without aggregation. If there is no aggregation, a packet carries the information for a single source. In other words, if we decide to transmit a commodity $(s,w,0)$, we must choose a single value for s. This is modelled by constraint (13) which restricts the flow of raw data to just one packet at each slot so that raw data is transmitted like in convergecast. When we use aggregation, on the other hand, a packet can carry more than one virtual information unit for a given commodity (s,w,p), for instance the aggregated values of the measurements of some subset of sources. This is modelled by constraint (14), which allows multiple virtual information units from different sources to be sent on the same packet if they belong to the same aggregated type and wave (for $p \geq 1$). This constraint essentially captures the effect of in-network computation. Comparing constraints (13) and (14), we can see the advantage of aggregation which permits more than one unit of information to be carried on a given packet.

Finally, constraints (15) and (16) represent the physical model of wireless interference. In (16), Q is a very large positive constant [27]. The magnitude of Q is chosen so that constraint (16) is trivially true whenever any link (i,j) is inactive, i.e., $x_{i,j}(k) = 0$. On the other hand, if the link is active, constraint (16) reduces to (4) which is the minimum SINR requirement constraint of the physical model.

Let T_m^d be the optimal solution to problem $\mathcal{P}_d(m)$. This problem is a very large integer linear program that cannot be solved optimally except for very small networks and even then, with large computation times. Thus, we are interested in obtaining a problem formulation that is based on a flow model and hopefully is more tractable. The two dimensions, time and waves are the primary sources of complexity in $\mathcal{P}_d(m)$. We transform $\mathcal{P}_d(m)$ such that both time and waves disappear resulting in a flow model formulation.

We use two types of transformations on $\mathcal{P}_d(m)$ to derive the flow model formulation. They are:

1) Averaging over time.
2) Bundling of waves.

We explain in detail how they are performed in the next two sub-sections.

D. First Transformation: Averaging Over Time

The idea is to sum the constraints over time and replace the variables with averages. Let

$$x_{i,j} \triangleq \frac{1}{T_m} \sum_{k=1}^{T_m} x_{i,j}(k) \tag{17}$$

$$y_{i,j}^{s,w,p} \triangleq \frac{1}{T_m}\sum_{k=1}^{T_m} y_{i,j}^{s,w,p}(k) \tag{18}$$

$x_{i,j}$ represents the fraction of time link (i,j) is active while $y_{i,j}^{s,w,p}$ represents the fraction of time information from source s of wave w and type p is flowing on the link (i,j). We define similar averages for the variables u and z as well. Note that as the input is just one packet for every wave at every source, we have $\sum_{k=1}^{T_m} v^w(k) = 1$.

Averaging out the constraints (8–14) using these definitions is straightforward. However, time also appears in the interference constraints (15–16). It is possible in principle to average them too but the resulting flow model is very inaccurate since the interference model is averaged out. Instead, we reformulate these constraints using the extensive formulation of the problem. First, we generate all the ISets compatible with (15–16). This is equivalent to computing the link-Iset incidence matrix

$$A_{(i,j),I} \triangleq \begin{cases} 1 & \text{if link } (i,j) \in I \\ 0 & \text{otherwise} \end{cases}$$

We then replace constraints (15–16) by introducing another set of variables $\alpha_I(k)$, a binary variable that is 1 if we use Iset I in slot k and 0 otherwise. The modified interference constraints are

$$\sum_I \alpha_I(k) \le 1 \tag{19}$$

$$x_{i,j}(k) \le \sum_I \alpha_I(k) A_{(i,j),I} \tag{20}$$

Constraint (19) ensures that only one Iset is chosen in slot k and constraint (20) allows a link to be active in slot k if and only if an Iset containing it is active in that slot. Note that all the links in any given Iset satisfy the interference constraints by construction and hence the original interference constraints (15–16) are redundant in the problem formulation.

For the purpose of averaging, we then define

$$\alpha_I \triangleq \frac{1}{T_m}\sum_{k=1}^{T_m} \alpha_I(k)$$

α_I represents the fraction of time Iset I would be active in the schedule. The discrete time dimension disappears from the constraints (7–14) and (19–20) when we sum each of them over k and divide with T_m, leading to problem $\mathcal{P}_2(m)$. Note that after this summation, all $q_i^{s,w,p}(k)$ variables cancel from the equations (8), (9) and (10), except when $k = 0$ and $k = T_m$. When $k = T_m$, the q's are 0 due to the final condition and when $k = 0$, the q's are 0 by convention. The only non-zero entity in the R.H.S of constraints (23) and (24) in $\mathcal{P}_2(m)$ is due to the summation $\sum_{k=1}^{T_m} v^w(k)$ which is equal to 1. All constraints are over all $w = 1 \ldots m$ and all $s = 1 \ldots n$ wherever w, or s appear, unless there is a summation over them. The constraints are also either over all nodes $i = 1 \ldots n$ or over all links $(i,j) \in \mathcal{L}$ depending upon the constraint. Also, all the variables are non-negative.

$$\mathcal{P}_2(m): \qquad \underset{\boldsymbol{\alpha},\mathbf{y},\mathbf{z},\mathbf{u}}{\text{Min}}\ T_m \tag{21}$$

subject to

$$T_m \ge \delta m \tag{22}$$

$$\sum_j y_{i,j}^{s,w,0} - \sum_j y_{j,i}^{s,w,0} + u_i^{s,w} = \begin{cases} \frac{1}{T_m} & \text{if } i = s \\ 0 & \text{otherwise} \end{cases} \tag{23}$$

$$\sum_j y_{i,j}^{s,w,p} - \sum_j y_{j,i}^{s,w,p} - u_i^{s,w} = 0 \quad \forall p = 1 \ldots M \tag{24}$$

$$\sum_{w=1}^{m}\sum_{p=0}^{M} z_{i,j}^{w,p} \le x_{i,j} \tag{25}$$

$$\sum_{s=1}^{n} y_{i,j}^{s,w,0} \le z_{i,j}^{w,0} \tag{26}$$

$$y_{i,j}^{s,w,p} \le z_{i,j}^{w,p} \quad \forall p = 1 \ldots M \tag{27}$$

$$x_{i,j} \le \sum_I A_{(i,j),I}\alpha_I \tag{28}$$

$$\sum_I \alpha_I \le 1 \tag{29}$$

We can interpret the left-hand side of equation (23) as the average information rate out of node i of wave w. It shows that in the time-averaged model, this rate is the same for all sources s and all waves w since the right-hand side does not depend on either of these indices. It is equal to the inverse of the total time T_m if i is the source of information s and 0 otherwise. Replacing $1/T_m$ with λ_m and changing the objective to maximize λ_m, we have a linear program. Note however that because of constraint (22) and since the number of constraints depends on m, this problem is not independent of m.

Since a feasible solution to $\mathcal{P}_2(m)$ (21–29) can be constructed from any feasible solution to $\mathcal{P}_d(m)$ (5–16), we have

$$T_m^2 \le T_m^d$$

where T_m^d is the optimal solution to $\mathcal{P}_d(m)$ and T_m^2 is the optimal solution to $\mathcal{P}_2(m)$.

Next, we transform $\mathcal{P}_2(m)$ into $\mathcal{P}_f$ by bundling the waves. This gives us the flow model for computing the maximum achievable throughput using in-network computation.

E. Second Transformation: Bundling of Waves

If T_m^d is the optimal solution to $\mathcal{P}_d(m)$ then m/T_m^d is the optimal rate at which the statistical function of each of the m waves is made available at the sink. Bundling the waves of $\mathcal{P}_2(m)$ removes the dependence on m and formulates the problem as maximizing this rate. For this, we define new variables that bundle the waves together, i.e.,

$$\sum_{w=1}^{m} y_{i,j}^{s,w,p} \triangleq y_{i,j}^{s,p}$$

where $q_i^{s,p}$ represents the total amount of information from source s, type p and all waves at node i and $y_{i,j}^{s,p}$ represents the total amount of information flow from source s, type p and all waves on link $(i,j$. We define similar variables for z and u as well. The waves lose their identity with the introduction of these variables. We sum constraints (23–27) over w and obtain problem $\mathcal{P}_3$. Again, all constraints are over all $s = 1 \dots n$ and all $k = 1 \dots T_m$ wherever k or s appear. The constraints are also either over all nodes $i = 1 \dots n$ or over all links $(i, j) \in \mathcal{L}$ depending upon the constraint. Also, all the variables are non-negative.

$$\mathcal{P}_3 : \quad \underset{\boldsymbol{\alpha},\mathbf{y},\mathbf{z},\mathbf{u}}{\text{Min}} \; T_m \tag{30}$$

subject to

$$T_m \geq \delta m \tag{31}$$

$$\sum_j y_{i,j}^{s,0} - \sum_j y_{j,i}^{s,0} + u_i^s = \begin{cases} \dfrac{m}{T_m} & \text{if } i = s \\ 0 & \text{otherwise} \end{cases} \tag{32}$$

$$\sum_j y_{i,j}^{s,p} - \sum_j y_{j,i}^{s,p} - u_i^s = 0 \quad \forall p = 1 \dots M \tag{33}$$

$$\sum_{p=0}^{M} z_{i,j}^{p} \leq x_{i,j} \tag{34}$$

$$\sum_{s=1}^{n} y_{i,j}^{s,0} \leq z_{i,j}^{0} \tag{35}$$

$$y_{i,j}^{s,p} \leq z_{i,j}^{p} \quad \forall p = 1 \dots M \tag{36}$$

$$x_{i,j} \leq \sum_I A_{(i,j),I} \alpha_I \tag{37}$$

$$\sum_I \alpha_I \leq 1 \tag{38}$$

The right-hand side of Eq. (32) can be interpreted as the number of waves flowing out of node i per unit time when it is the source of information s, i.e., it is the average rate of information flow out of s. We replace it as $m/T_m \triangleq \lambda_m$ and define the objective as the maximization of this rate. With this replacement, the constraint (31) changes to

$$\lambda_m \leq \frac{1}{\delta} \tag{39}$$

This implies that the output rate cannot exceed the input rate which is as expected. The constraints and the variables are no longer dependent on m and we can replace λ_m with just λ. As we are interested in computing the maximum achievable throughput, it is customary to remove constraint (31) (which corresponds to constraint (39)).

Thus, the final flow model formulation $\mathcal{P}_f$ is as follows.

$$\mathcal{P}_f : \quad \underset{\boldsymbol{\alpha},\mathbf{y},\mathbf{z},\mathbf{u}}{\text{Max}} \; \lambda \tag{40}$$

subject to

$$\sum_j y_{i,j}^{s,0} - \sum_j y_{j,i}^{s,0} + u_i^s = \begin{cases} \lambda & \text{if } i = s \\ 0 & \text{otherwise} \end{cases} \tag{41}$$

$$\sum_j y_{i,j}^{s,p} - \sum_j y_{j,i}^{s,p} - u_i^s = 0 \quad \forall p = 1 \dots M \tag{42}$$

$$\sum_{p=0}^{M} z_{i,j}^{p} \leq x_{i,j} \tag{43}$$

$$\sum_{s=1}^{n} y_{i,j}^{s,0} \leq z_{i,j}^{0} \tag{44}$$

$$y_{i,j}^{s,p} \leq z_{i,j}^{p} \quad \forall p = 1 \dots M \tag{45}$$

$$x_{i,j} \leq \sum_I A_{(i,j),I} \alpha_I \tag{46}$$

$$\sum_I \alpha_I \leq 1 \tag{47}$$

Let λ^f be the optimal solution to problem $\mathcal{P}_f$. Given a feasible solution to $\mathcal{P}_2(m)$ (21–29), we can construct a feasible solution to the flow problem $\mathcal{P}_f$ (40–47) so we have

$$\frac{m}{\lambda^f} \leq T_m^2 \leq T_m^d \tag{48}$$

This inequality (48) proves that the throughput computed by the flow model $\mathcal{P}_f$ is an upper bound on the maximum achievable throughput using in-network computation when the sink is interested in the first M moments. Note that we would get the same problem formulation $\mathcal{P}_f$ even if we had performed bundling of waves before averaging over time. Thus, we have derived a formulation $\mathcal{P}_f$ (40–47) which has all the desirable features. It is independent of m, has continuous and fewer variables and constraints than in $\mathcal{P}_d(m)$ and it is also a linear program.

Recall that aggregation can happen only between the data belonging to the same wave. In problem formulation $\mathcal{P}_d(m)$, constraint (12) and in $\mathcal{P}_2(m)$, constraint (25) explicitly forbid aggregation among waves. However, due to the transformations, there is no wave superscript in the flow model $\mathcal{P}_f$, and hence no equivalent constraint. Thus, there is a possibility that the upper bound on throughput computed using $\mathcal{P}_f$ is too loose, i.e., it is not very close to the maximum achievable throughput obtained with the discrete-time model. Hence, there is a need to validate the tightness of the upper bound computed by the flow model $\mathcal{P}_f$. We do this in two steps. First (in the next sub-section), we show that the throughputs computed by $\mathcal{P}_2(m)$ and $\mathcal{P}_f$ are equal, which proves that there is no inflation in the throughput computed by $\mathcal{P}_f$ due to the lack of the wave superscript and the absence of a constraint preventing aggregation among waves. Second (in Section IV), based on the solution to the flow model, we compute a near-optimal feasible solution to the discrete-time model whose throughput is close to the throughput computed by the flow model. This shows that $\mathcal{P}_f$ provides a very close upper bound to the maximum achievable throughput.

F. Equivalence of the throughputs of $\mathcal{P}_f$ and $\mathcal{P}_2(m)$

In this sub-section, we prove that the throughputs computed by $\mathcal{P}_f$ and by $\mathcal{P}_2(m)$ are equal. Define a new problem $\mathcal{P}_4(m)$ as $\mathcal{P}_2(m)$ with the following additional constraint

$$\sum_{p=0}^{M} z_{i,j}^{w,p} \leq \frac{x_{i,j}}{m} \quad \forall s \tag{49}$$

Since we added a potentially restrictive constraint, we have

$$T_m^2 \leq T_m^4 \tag{50}$$

where T_m^4 is the optimal solution of $\mathcal{P}_4(m)$. Also, we see that constraint (25) is now redundant in $\mathcal{P}_4(m)$ so that it separates into m identical sub-problems for each w, each with the same optimal solution T_m^4. The sub-problem is

$$\mathcal{P}_4^{sub}(m) : \quad \underset{\boldsymbol{\alpha},\mathbf{y},\mathbf{z},\mathbf{u}}{\text{Min}} \; T_m \tag{51}$$

subject to

$$T_m \geq \delta m \tag{52}$$

$$\sum_j y_{i,j}^{s,w,0} - \sum_j y_{j,i}^{s,w,0} + u_i^{s,w} = \begin{cases} \dfrac{1}{T_m} & \text{if } i = s \\ 0 & \text{otherwise} \end{cases} \tag{53}$$

$$\sum_j y_{i,j}^{s,w,p} - \sum_j y_{j,i}^{s,w,p} - u_i^{s,w} = 0 \quad \forall p = 1 \ldots M \tag{54}$$

$$\sum_{p=0}^{M} z_{i,j}^{w,p} \leq \frac{x_{i,j}}{m} \tag{55}$$

$$\sum_{s=1}^{n} y_{i,j}^{s,w,0} \leq z_{i,j}^{w,0} \tag{56}$$

$$y_{i,j}^{s,w,p} \leq z_{i,j}^{w,p} \quad \forall p = 1 \ldots M \tag{57}$$

$$x_{i,j} \leq \sum_I A_{(i,j),I}\alpha_I \tag{58}$$

$$\sum_I \alpha_I \leq 1 \tag{59}$$

But, this is precisely problem $\mathcal{P}_f$ (40–47) with the change of variables $y_{i,j}^{s,w,p} \rightarrow y_{i,j}^{s,p}/m$, $z_{i,j}^{w,p} \rightarrow z_{i,j}^{p}/m$ and $T_m \rightarrow m/\lambda$. Hence,

$$\frac{m}{\lambda^f} = T_m^4 \tag{60}$$

From inequalities (50) and (60), we conclude

$$T_m^2 \leq \frac{m}{\lambda^f} \tag{61}$$

But from inequality (48), we know

$$T_m^2 \geq \frac{m}{\lambda^f} \tag{62}$$

so that $T_m^2 = \frac{m}{\lambda^f} = T_m^4$. This proves that there is no inflation in the throughput computed by $\mathcal{P}_f$ due to the lack of the wave superscript and the absence of a constraint explicitly preventing aggregation among different waves.

Thus, we have rigorously derived the problem formulation $\mathcal{P}_f$ (40–47) which is the desired flow model. In the next section, we validate this model and obtain some insights using it into the effects of in-network computation on the network performance.

IV. Validation and Insights

In this section, we validate our flow model formulation $\mathcal{P}_f$, i.e., we attempt to show that it computes an excellent upper bound to the solution of the discrete-time model $\mathcal{P}_d(m)$ and give some insights on the effects of in-network computation on network performance. We first prove some propositions related to the maximum achievable throughput using in-network computation and prove that when $M = n$ the flow model formulation $\mathcal{P}_f$ is equivalent to the convergecast formulation in the literature. Next, we propose a heuristic that computes a feasible solution to the discrete-time model based on the solution to the flow model. As these solutions are very close to the flow model solutions in terms of maximum achievable throughput, we have validated our flow model formulation. Finally, we give some insights on the effects of in-network computation on maximum achievable throughput.

A. Some Results

As explained in the introduction (Section I-A), in-network computation leads to data aggregation. This reduces the total volume of data transmitted in the network and potentially increases the achievable throughput. Hence, we can conclude that using in-network computation could never perform worse than convergecast in terms of the maximum achievable throughput. This leads to the following proposition.

Proposition 4.1: If λ_M is the maximum achievable throughput with in-network computation when the sink is interested in the first M moments of the data collected by the sensor nodes and λ_C is the maximum achievable throughput using convergecast, then

$$\lambda_M \geq \lambda_C \quad \forall M \geq 1 \tag{63}$$

Again consider the problem where the sink is interested in the first M moments of the data collected by the sensor nodes in the network. If in-network computation is allowed, any node that has k different raw data packets of the same wave in its buffer decides to aggregate them if and only if $k > M$. It creates the M different partial sums using those k data and thus reduces the number of time slots it requires to transmit the information in its buffer from k to M. Because of this nature of in-network computation, it is obvious that the potential savings in the volume of the transmitted data are inversely proportional to M. Thus, we have

Proposition 4.2: If λ_M is the maximum achievable throughput with in-network computation when the sink is interested in the first M moments of the data collected by the sensor nodes then

$$\lambda_M \geq \lambda_{M+1} \quad \forall M \geq 1 \tag{64}$$

The maximum number of raw data packets a node can potentially receive of a particular wave is limited by the number of nodes n in the network because every node collects only one data packet per wave. So, when $M \geq n$, no node would be able to perform aggregation and reduce the volume of the transmitted data. Thus, the maximum throughput we can expect cannot be more than that is possible through convergecast. This proves the following proposition.

Proposition 4.3: If n is the number of nodes in the network excluding the sink, λ_M is the maximum achievable throughput with in-network computation when the sink is interested in the first M moments of the collected data and λ_C is the maximum achievable throughput using convergecast then

$$\lambda_n = \lambda_{n+1} = \ldots = \lambda_C \tag{65}$$

Based on this proposition, it would give us another level of confidence if we could show that the solution given by our flow model $\mathcal{P}_f$ when $M = n$ is equal to the solution of the flow model for convergecast given in [2]. Next, we show this.

B. The Special Case of $M = n$

Let λ_n be the optimal throughput of $\mathcal{P}_f$ when $M = n$ and let λ_C be the optimal convergecast throughput obtained by solving the following problem $\mathcal{P}_C$ [2]. Recall that there is a flow from every source s to the sink in convergecast.

$$\mathcal{P}_C: \qquad \max_{\boldsymbol{\alpha},\mathbf{y}} \lambda \tag{66}$$

$$\sum_j y^s_{i,j} - \sum_j y^s_{j,i} = \begin{cases} \lambda & \text{if } i = s \\ 0 & \text{otherwise} \end{cases} \tag{67}$$

$$\sum_{s=1}^{n} y^s_{i,j} \leq \sum_I A_{(i,j),I}\alpha_I \tag{68}$$

$$\sum_I \alpha_I \leq 1 \tag{69}$$

From Proposition 4.1, we have

$$\lambda_n \geq \lambda_C \tag{70}$$

Next, we perform a series of transformations on $\mathcal{P}_f$ when $M = n$. The idea is to get rid of the dependence on the flow type p. From constraint (42), we get the value of u^s_i

$$u^s_i = \sum_j y^{s,p}_{i,j} - \sum_j y^{s,p}_{j,i}$$

Replacing u^s_i in constraint (41) and rearranging the terms gives

$$\sum_j (y^{s,0}_{i,j} + y^{s,p}_{i,j}) - \sum_j (y^{s,0}_{j,i} + y^{s,p}_{j,i}) = \begin{cases} \lambda & \text{if } i = s \\ 0 & \text{otherwise} \end{cases}$$

We sum on both sides over all p to get

$$\sum_{p=1}^{n}\sum_j (y^{s,0}_{i,j} + y^{s,p}_{i,j}) - \sum_{p=1}^{n}\sum_j (y^{s,0}_{j,i} + y^{s,p}_{j,i}) = \begin{cases} n\lambda & \text{if } i = s \\ 0 & \text{otherwise} \end{cases}$$

Next, we add constraint (44) to constraint (45) and get

$$\sum_{s=1}^{n} (y^{s,0}_{i,j} + y^{s,p}_{i,j}) \leq z^0_{i,j} + n z^p_{i,j} \quad \forall p \quad \forall (i,j) \in \mathcal{L}$$

We sum the above inequality over all p to get

$$\sum_{p=1}^{n}\sum_{s=1}^{n} (y^{s,0}_{i,j} + y^{s,p}_{i,j}) \leq n \sum_{p=0}^{n} z^p_{i,j}$$

Now, we replace

$$\sum_{p=1}^{n} (y^{s,0}_{i,j} + y^{s,p}_{i,j}) \to n y^s_{i,j} \text{ and } \sum_{p=0}^{n} z^p_{i,j} \to z_{i,j}$$

With these changes, problem $\mathcal{P}_f$ becomes problem $\mathcal{P}_5$

$$\mathcal{P}_5: \qquad \max_{\boldsymbol{\alpha},\mathbf{y},\mathbf{z}} \lambda \tag{71}$$

$$\sum_j y^s_{i,j} - \sum_j y^s_{j,i} = \begin{cases} \lambda & \text{if } i = s \\ 0 & \text{otherwise} \end{cases} \tag{72}$$

$$z_{i,j} \leq x_{i,j} \tag{73}$$

$$\sum_{s=1}^{n} y^s_{i,j} \leq z_{i,j} \tag{74}$$

$$x_{i,j} \leq \sum_I A_{(i,j),I}\alpha_I \tag{75}$$

$$\sum_I \alpha_I \leq 1 \tag{76}$$

It is not hard to see the equivalence of this new problem with the flow model for convergecast (66–69). Thus, if λ_5 is the optimal throughput of this new problem, then we have

$$\lambda_C = \lambda_5 \tag{77}$$

Since, a feasible solution to $\mathcal{P}_5$ can be constructed from the optimal solution to $\mathcal{P}_f$ with $M = n$, we have

$$\lambda_5 \geq \lambda_n \tag{78}$$

From (70), (77) and (78), we conclude that when $M = n$, $\mathcal{P}_f$ (40–47) is equivalent to the convergecast formulation given in the literature [2].

In the next sub-section, we give an heuristic to construct a feasible solution to the discrete-time model formulation, whose achievable throughput is close to optimal.

C. Feasible Solution: Lower Bound

By design, the flow model $\mathcal{P}_f$ (40–47) gives an upper bound on the maximum achievable throughput. However, solving $\mathcal{P}_f$ is not straightforward as the number of possible Isets is exponentially large in the number of links, making it a very large linear program. So, we use the column generation technique to solve $\mathcal{P}_f$ to optimality, which involves solving a series of small linear programs [10]. Unlike the column generation technique in [10], we a priori enumerate all the possible Isets (using the efficient algorithm for enumeration proposed in [10]). In the first iteration, we solve $\mathcal{P}_f$ with only the single link Isets as the input. For every iteration after this, the most useful Isets to be added are determined using the dual values of the constraints of the current iteration. We iterate until all the constraints have non-positive dual values which ensures optimality. We have successfully employed this technique for networks with up to 30 nodes. For larger networks, a priori enumeration of Isets is not scalable and we need more sophisticated column generation techniques that generates new Isets as required after every iteration. However, past work on wireless sensor networks suggests that a clustering approach which divides a large network into many small manageable clusters is the most efficient approach for many applications [28], [29]. Thus, the approach we took in this work is useful in most practical scenarios.

An obvious question now is how close the upper bound computed by $\mathcal{P}_f$ is to the optimal solution of the discrete-time model given in Section III-C. We answer this question by computing the optimality gap between the upper bound and some lower bound. If the gap is small, we know that the upper bound is good. In this section, we show that this is the case by obtaining a feasible solution (based on the solution to the flow model) to the discrete-time model. The achievable throughput of this feasible solution is naturally a lower bound on the maximum achievable throughput.

Typically, a feasible solution to the discrete-time model $\mathcal{P}_d(m)$ is a sequence of Isets (one Iset per time slot generally with some periodic pattern). Assume that such a feasible solution to $\mathcal{P}_d(m)$ is given. To compute the achievable throughput of this feasible solution, we have designed a simulator that imitates the actual network operation. It works on two simple rules that capture the working of in-network computation when the sink is interested in the first M moments:

1) A transmitter sends the highest type packet from the oldest wave available in its queue.
2) A receiver aggregates the raw data it has in its queue for a given wave when the total number of packets (including all types and raw data) of that wave is more than M.

We activate the Isets in the sequence given in the feasible solution and update the buffers at all nodes after every time slot. From this, we get the number of time slots (Δ_m) required to empty the network of the first m waves (we use $m = 10000$). Using this, we can now compute the throughput of the given feasible solution as m/Δ_m and compare it with the solution λ of $\mathcal{P}_f$.

Next, we describe the method we use to obtain a feasible schedule to the discrete-time model that gives near-optimal throughput. As mentioned earlier, we derive the feasible solution from the optimal flow solution. Much information is lost in the transformation from $\mathcal{P}_d(m)$ to $\mathcal{P}_f$ so that converting the optimal solution of $\mathcal{P}_f$ to a feasible solution of $\mathcal{P}_d(m)$ is difficult. Especially, the routing produced by the flow model is generally too complex to be of any practical use. In order to simplify routing, we impose new constraints on $\mathcal{P}_f$ to force a single path routing. For this, we define a new set of binary variables $w_{i,j}$ that indicate whether the link (i, j) is used or not. The new constraints are

$$y_{i,j}^{s,p} \leq w_{i,j} \quad \forall (i,j) \quad \forall s \quad \forall p$$
$$\sum_j w_{i,j} \leq 1 \quad \forall i, \quad w_{i,j} \in \{0,1\}$$

The solution to this problem is a set of Isets with the fractions of time each of them should be activated to achieve the optimal throughput possible through single-path routing. We normalize these fractions to find the number of times each Iset needs to be active. We assume a random order of activation of these Isets and thus we have a sequence of Isets that is a feasible solution to the discrete model. In the simulator, we assume that this Iset pattern is repeated until all the waves reach the sink and compute the throughput of this feasible solution. However, adding the binary variables $w_{i,j}$ has transformed $\mathcal{P}_f$ from a standard LP into a mixed IP so that computing an optimal solution becomes extremely time consuming as M or the size of the network increases. In the cases where M becomes too large to solve the problem $\mathcal{P}_f$ with single path routing within a reasonable time, we fix the routing as follows:

1) We either use a min-hop tree or
2) a tree using the most active links in the optimal solution to $\mathcal{P}_f$.

We then solve $\mathcal{P}_f$ (40–47) considering only this tree as the network. We obtain a feasible solution to the discrete-time model and compute the achievable throughput using the simulator following the same approach as described before. For some instances, the min-hop tree gives better throughput than the other tree and vice versa.

For numerical results, we have randomly generated two 16-nodes networks (netA and netB) and a 30-nodes network

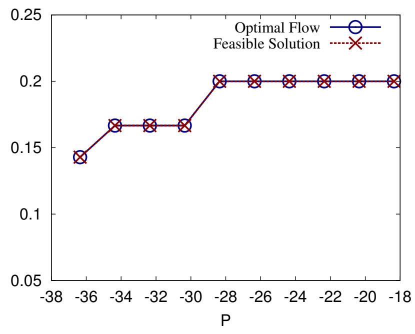


Fig. 2: λ vs P for the 16-nodes network, netA when M = 1

(netC), given in Figure (1). We used the network parameters given in Table I with the following channel propagation model.

$$P_{i,j} = \frac{G_{i,j}P}{(\frac{d_{i,j}}{d_0})^{\eta}}; \quad SNR = \frac{P_{i,j}}{N_0}$$

where $d_{i,j}$ is the physical distance between nodes i and j, $G_{i,j}$ is the channel gain on link (i, j) that accounts for channel fading and shadowing, d_0 is the near-field cross over distance, η is the path-loss exponent and N_0 is the thermal noise power in the frequency band of operation. Recall that we assume the channel gains to be quasi time-invariant. For simplicity in numerical calculations, we have assumed the same constant G on all the links. However, this should not be construed as a limitation as computations with different G on different links are not more complex.

β	6.4 dB
N_0	-100 dBm
η	3
d_0	0.1 m
G_l	1

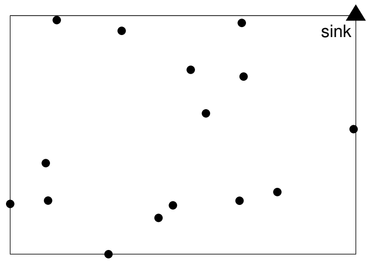
TABLE I: Network parameters used for obtaining numerical results

Network	At the lowest power	At the highest power
netA	2	15
netB	3	15
netC	6	29

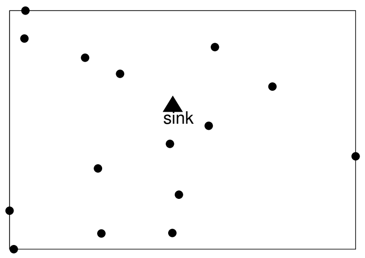
TABLE II: Average node degree in netA, netB and netC

The average node degree of the three networks at the lowest and the highest powers is given in Table II. For the two 16-nodes networks, we were able to optimally solve the single-path problem for $M < 8$ but for higher M, we have fixed the routing using the two methods described before. For the 30-nodes network, CPLEX was unable to solve the optimal single-path problem for any M as it is a very large binary program. Thus, we use the other heuristics that fix the routing using other methods to obtain the feasible solutions.

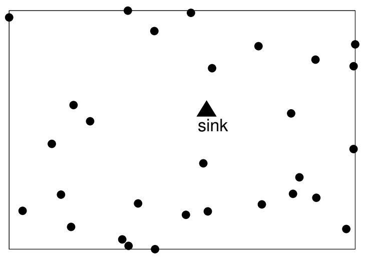
Plots of maximum achievable throughput λ versus the transmit power P (used by all the nodes) for some selected M for the three networks (netA, netB and netC) are given in Figures (2 – 9). These plots show that the upper bound



(a) First 16 nodes network, netA



(b) Second 16 nodes network, netB



(c) A 30 nodes network, netC

Fig. 1: Random Networks

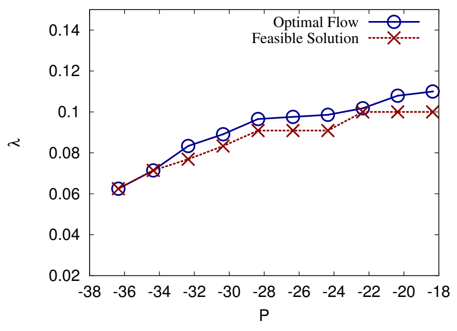


Fig. 3: λ vs P for the 16-nodes network, netA when M = 3

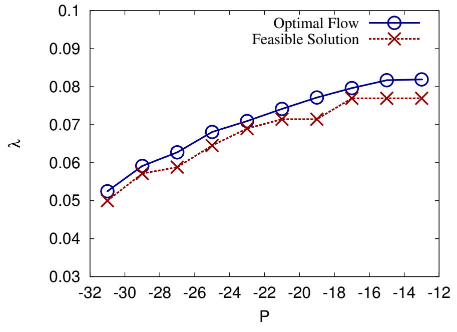


Fig. 6: λ vs P for the 16-nodes network, netB when M = 5

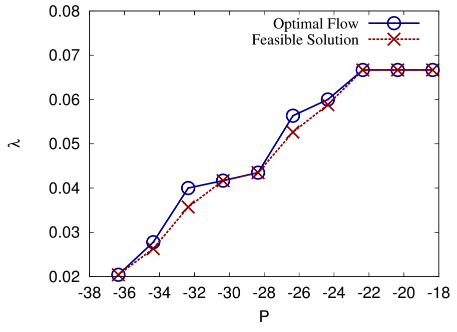


Fig. 4: λ vs P for the 16-nodes network, netA when M = 15

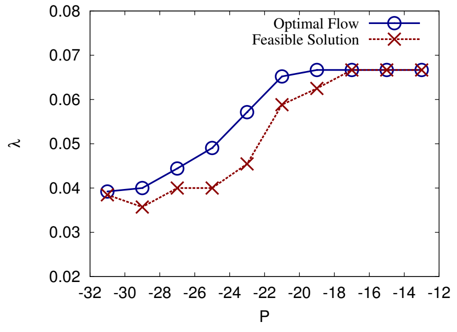


Fig. 7: λ vs P for the 16-nodes network, netB when M = 10

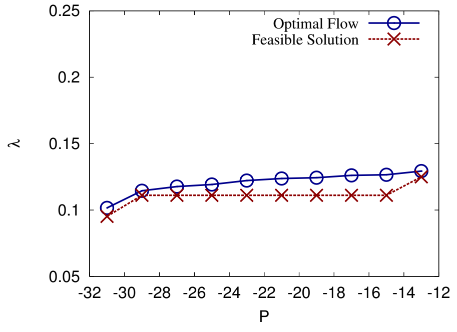


Fig. 5: λ vs P for the 16-nodes network, netB when M = 2

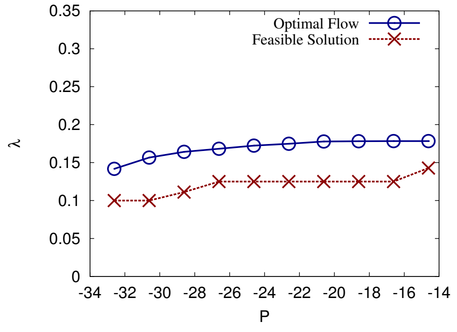


Fig. 8: λ vs P for the 30-nodes network, netC when M = 1

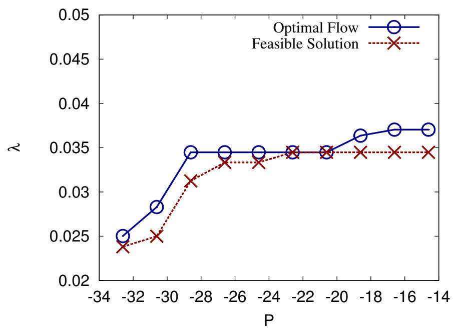


Fig. 9: λ vs P for the 30-nodes network, netC when $M = 12$

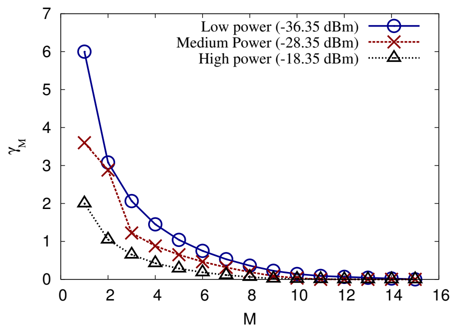


Fig. 10: γ (gain) vs M for netA at three different powers

(labelled optimal flow) is very close to the lower bound (labelled feasible solution). This validates the flow model $\mathcal{P}_f$.

D. Insights

We can now use the flow model to gain some insights on the effect of in-network computation. The first result comes from Figures (2–7). Because of the way we constructed the feasible solution, they show that we can get near-optimal throughput using a single-path routing. This is important since single path routing is easy to implement and our results confirm that this is not a bad idea. The feasible solutions in Figure (8) for the 30-nodes network when $M = 1$ are not as close to the upper bound as those for the 16-nodes network in Figures (2). The reason for this is that we were unable to optimally solve the large binary program (that imposes single path routing) for the 30-nodes network that gives the best feasible solutions for the 16-nodes network for small M.

As mentioned in the introduction, in-network computation is expected to potentially improve the throughput significantly compared to convergecast. In this paper, we have attempted to quantify this improvement and to the best of our knowledge, this is the first such attempt in the literature. For the plots in Figures (10–11), we define the gain γ_M as follows and plot it with respect to M.

$$\gamma_M = \frac{\lambda_M - \lambda_C}{\lambda_C} \tag{79}$$

We observe from this plots that for low M, in-network computation does lead to significant gains in throughput compared to convergecast at all powers. For $M = 1$, the throughput using in-network computation is several times larger than that

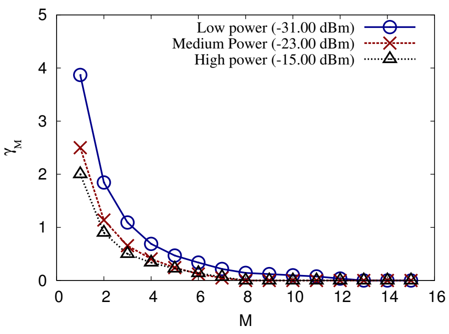


Fig. 11: γ (gain) vs M for netB at three different powers

is possible using convergecast and even for relatively large M like 5, the gains in throughput are surprisingly still significant. We also note that these gains are much larger at lower powers than at higher powers which implies that λ_M increases slower than λ_C when the transmission power increases. This is because the increased spatial reuse resulting from higher transmission power is more useful for the convergecast that transports larger volume of data compared to when in-network computation is allowed. We believe that our model can further shed light on the optimal topology, the optimal placement of gateway, etc., and provide further insights on the effects of in-network computation on network performance. We would explore this as a part of our future work.

V. Conclusion

In this paper, we have rigorously developed a framework to study the problem of in-network computation in a wireless sensor network using scheduling under a physical interference model. In particular, we have derived a flow model to compute the maximum achievable throughput when the sink is interested in the first M moments of the collected data. In addition, we have given a method to obtain near-optimal feasible solutions to the discrete-time model.

From the numerical results, we observed that it was possible to achieve close to optimal throughput using a single path routing. The gains due to in-network computation are significant even for relatively large M and they are much larger at lower powers than at higher powers.

Maximizing the lifetime of a sensor network is an important objective in sensor networks context. In-network computation has the potential to reduce the energy consumption per node and thus increase the overall lifetime. Modifying our flow model to study this problem is straightforward. Apart from this, our future work would involve developing similar flow models when the sink is interested in functions other than the first M moments and further understanding the effects of in-network computation on network performance.

References

[1] A. Giridhar and P. Kumar, "Computing and communicating functions over sensor networks," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 4, pp. 755–764, Apr. 2005.

[2] A. Karnik, A. Iyer, and C. Rosenberg, "Throughput-optimal configuration of fixed wireless networks," *IEEE/ACM Trans. Netw.*, vol. 16, no. 5, pp. 1161–1174, 2008.

[3] K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu, "Impact of interference on multi-hop wireless network performance," in *the 9th annual international conference on Mobile computing and networking*. New York, NY, USA: ACM, 2003, pp. 66–80.

[4] P. Stuedi and G. Alonso, "Computing throughput capacity for realistic wireless multihop networks." in *MSWiM*, 2006, pp. 191–198.

[5] O. Durmaz Incel and B. Krishnamachari, "Enhancing the data collection rate of tree-based aggregation in wireless sensor networks," in *5th IEEE ComSoc Conference on Sensor, Mesh and Ad Hoc Communications and Networks. San Francisco, USA*, July 2008, pp. 569–577.

[6] A. Ghosh, O. Incel, V. Kumar, and B. Krishnamachari, "Multi-channel scheduling algorithms for fast aggregated convergecast in sensor networks," in *Mobile Adhoc and Sensor Systems, MASS, IEEE 6th International Conference on*, Oct. 2009, pp. 363–372.

[7] A. Ghosh, O. D. Incel, V. S. A. Kumar, and B. Krishnamachari, "Multichannel scheduling and spanning trees: Throughput; delay tradeoff for fast data collection in sensor networks," *Networking, IEEE/ACM Transactions on*, no. 99, 2011.

[8] P. Gupta and P. Kumar, "The capacity of wireless networks," *IEEE Trans. Inf. Theory*, vol. 46, no. 2, pp. 388–404, Mar. 2000.

[9] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *Automatic Control, IEEE Transactions on*, vol. 37, no. 12, pp. 1936 –1948, Dec 1992.

[10] J. Luo, C. Rosenberg, and A. Girard, "Engineering wireless mesh networks: Joint scheduling, routing, power control and rate adaptation," *IEEE/ACM Trans. Netw.*, vol. 18, no. 3, pp. 1387–1400, Oct. 2010.

[11] C. Florens, M. Franceschetti, and R. McEliece, "Lower bounds on data collection time in sensory networks," *IEEE J. Sel. Areas Commun.*, vol. 22, no. 6, pp. 1110–1120, Aug. 2004.

[12] L. Gargano and A. A. Rescigno, "Optimally fast data gathering in sensor networks," *Mathematical Foundations of Computer Science*, vol. 4162/2006, pp. 399–411, 2006.

[13] H. Choi, J. Wang, and E. Hughes, "Scheduling for information gathering on sensor network," *Wireless Networks*, vol. 15, pp. 127–140, 2009.

[14] P. Tiwari, "Lower bounds on communication complexity in distributed computer networks," *J. ACM*, vol. 34, no. 4, pp. 921–938, 1987.

[15] S. Kamath and D. Manjunath, "On distributed function computation in structure-free random sensor networks," in *IEEE ISIT*, Jul. 2008.

[16] N. Khude, A. Kumar, and A. Karnik, "Time and energy complexity of distributed computation of a class of functions in wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 7, no. 5, 2008.

[17] D. Mosk-Aoyama and D. Shah, "Fast distributed algorithms for computing separable functions," *IEEE Trans. Inf. Theory*, vol. 54, no. 7, pp. 2997–3007, Jul. 2008.

[18] X. Chen, X. Hu, and J. Zhu, "Minimum data aggregation time problem in wireless sensor networks," in *Mobile Ad-hoc and Sensor Networks*, ser. Lecture Notes in Computer Science, X. Jia, J. Wu, and Y. He, Eds. Springer Berlin / Heidelberg, 2005, vol. 3794, pp. 133–142.

[19] H. Li, Q. S. Hua, C. Wu, and F. C. M. Lau, "Minimum-latency aggregation scheduling in wireless sensor networks under physical interference model," in *the 13th ACM international conference on Modeling, analysis, and simulation of wireless and mobile systems*. New York, NY, USA: ACM, 2010, pp. 360–367.

[20] X.-Y. Li, X. Xu, S. Wang, S. Tang, G. Dai, J. Zhao, and Y. Qi, "Efficient data aggregation in multi-hop wireless sensor networks under physical interference model," in *Mobile Adhoc and Sensor Systems, 2009. MASS '09. IEEE 6th International Conference on*, oct. 2009, pp. 353 –362.

[21] P.-J. Wan, S. C.-H. Huang, L. Wang, Z. Wan, and X. Jia, "Minimum-latency aggregation scheduling in multihop wireless networks," in *the tenth ACM international symposium on Mobile ad hoc networking and computing*. New York, NY, USA: ACM, 2009, pp. 185–194.

[22] X. Xu, S. Wang, X. Mao, S. Tang, and X. Y. Li, "An improved approximation algorithm for data aggregation in multi-hop wireless sensor networks," in *the 2nd ACM international workshop on Foundations of wireless ad hoc and sensor networking and computing*. New York, NY, USA: ACM, 2009, pp. 47–56.

[23] L. Xiang, J. Luo, and A. Vasilakos, "Compressed data aggregation for energy efficient wireless sensor networks," in *the 8th IEEE SECON*, 2011, pp. 46–54.

[24] R. Maheshwari, J. Cao, and S. R. Das, "Physical interference modeling for transmission scheduling on commodity wifi hardware," *IEEE INFOCOM*, 2009.

[25] A. Iyer, C. Rosenberg, and A. Karnik, "What is the right model for wireless channel interference?" *IEEE Trans. Wireless. Commun.*, vol. 8, no. 5, pp. 2662–2671, 2009.

[26] L. Fratta, M. Gerla, and L. Kleinrock, "The flow deviation method: An approach to store-and-forward communication network design," *Networks*, vol. 3, no. 2, pp. 97–133, 1973.

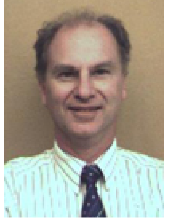
[27] J. Luo, A. Girard, and C. Rosenberg, "Efficient algorithms to solve a class of resource allocation problems in large wireless networks," in *the 7th international conference on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks, WiOPT*, 2009, pp. 313–321.

[28] A. A. Abbasi and M. Younis, "A survey on clustering algorithms for wireless sensor networks," *Computer Communications*, vol. 30, no. 1415, pp. 2826 – 2841, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0140366407002162

[29] O. Boyinbode, H. Le, A. Mbogho, M. Takizawa, and R. Poliah, "A survey on clustering algorithms for wireless sensor networks," in *Network-Based Information Systems (NBiS), 2010 13th International Conference on*, sept. 2010, pp. 358 –364.

Rajasekhar Sappidi graduated with B.Tech. in Electrical Engineering from Indian Institute of Technology, Bombay, India in 2007. He worked as a design engineer at Cypress Semiconductors, Hyderabad, India in 2007. Since 2008, he is pursuing Ph.D. in Electrical and Computer Engineering at the University of Waterloo, Canada. His research focus is on modelling and performance evaluation of wireless and wired networks with in-network computation.

Catherine Rosenberg was educated in France (Ecole Nationale Supérieure des Télécommunications de Bretagne, Diplôme d'Ingénieur in EE in 1983 and University of Paris, Orsay, Doctorat en Sciences in CS in 1986) and in the USA (UCLA, MS in CS in 1984), Dr. Rosenberg has worked in several countries including USA, UK, Canada, France and India. In particular, she worked for Nortel Networks in the UK, AT&T Bell Laboratories in the USA, Alcatel in France and taught at Purdue University (USA) and Ecole Polytechnique of Montreal (Canada). Since 2004, Dr. Rosenberg is a faculty member at the University of Waterloo where she now holds a Tier 1 Canada Research Chair in the Future Internet. Her research interests are broadly in networking with currently an emphasis in wireless networking and in smart energy systems. She has authored over 150 papers and has been awarded eight patents in the USA. She is a Fellow of the IEEE. More information can be found at http://ece.uwaterloo.ca/∼cath/.



André Girard received the Ph.D. degree in physics from the University of Pennsylvania, Philadelphia, in 1971. He is an Honorary Professor with INRS-EMT and an Adjunct Professor with École Polytechnique of Montréal, QC, Canada. His research interests all have to do with the optimization of telecommunication networks, and in particular with performance evaluation, routing, dimensioning, and reliability. He has made numerous theoretical and algorithmic contributions to the design of telephone, ATM, IP and wireless networks.