

Communication for Distributed Computation

Catherine Rosenberg

Dept. of Electrical and Computer Engineering
University of Waterloo, Waterloo, ON, Canada

Collaborators: Ravi R. Mazumdar, Raja Sappidi.

Outline

- Introduction
- Prior Art
- One Possible Problem Formulation: Structured Networks
 - The benchmark: case without aggregation
 - * The wired case
 - * The wireless case
 - From function to aggregation
 - The case with aggregation: $a(k) = 1$
 - * The wired case
 - * The wireless case
 - The case with aggregation: $a(k) \neq 1$
 - Conclusions and other challenges

Introduction

- Motivation for computational networks comes from many areas including vehicular (V2V), sensor, peer-to-peer, and adhoc networks.
- Nodes monitor some physical variables of interest like temperature, velocity, etc. either periodically or when an event is triggered.
- A particular node, called collector node (**or a group of nodes**) is interested in some **function** of these measurements.
- Nodes form a connected network (either wired or wireless, reliable or not, mobile or not).
- The network is multi-hop, hence each node is a possible relay.
- We could consider a single data collection campaign or multiple campaigns.

Introduction (Contd.)

- Two options need to be considered:
 - No processing at nodes: just store-and-forward functionality at the intermediate nodes.
 - Processing at the intermediate nodes: possibility for aggregation, this is really the distributed computation case.
- The objective is to design **distributed** communication and computation algorithms to compute the function of interest while
 - minimizing the total time taken (or maximizing the rate) to compute and communicate the value of the function to the sink.
 - minimizing the mean square error in the nodes' estimate of the desired function.

Challenges/Issues

- Wired: structured or unstructured, centralized versus distributed solution.
- Wireless: unicast or broadcast, MAC issues, energy.
- Information security.
- How much of the network topology is known at the collector node(s)?
- How structured is the network (e.g., addresses, synchronization, etc.)?

Challenges/Issues (Contd.)

- In a network where data can be duplicated or lost, how to aggregate/process partial information at intermediate nodes while allowing the function to be reconstructed correctly at the collector node(s).
- For example, even for a simple function like SUM, we do not know what kind of partial computation must be performed at the intermediate nodes. If the nodes simply add the data they receive, then the sink node may not be able to recover the correct value of the SUM even if we provide for every partial sum and the IDs of the nodes whose data was added.

Prior Art

- Cédric Florens, Massimo Franceschetti and Robert McEliece, Lower bounds on data collection time in sensory networks. *IEEE Journal on selected areas in communications*, 2004 .
- **Problem considered:** Minimum delay to collect all the data in a wireless sensor network at the sink.
- **Main Result:** They find the optimal routing and scheduling algorithm for a tree network that achieves minimum delay and also show that the minimum delay in any general graph is within a factor 2 of the minimum total delay in any shortest paths spanning tree of the graph.
- **Limitations:** They consider the interference protocol model. It is a centralized algorithm.

Prior Art

- A. Giridhar and P. R. Kumar, Computing and Communicating Functions Over Sensor Networks. *IEEE Journal on Selected Areas in Communications*, 2005.
- **Problem considered:** Maximum rate of sending data if the sink is interested in a symmetric function.
- **Main Result:** They give maximum rate of computation for two subclasses of symmetric functions.
- **Limitations:** They consider the interference protocol model. The results are applicable to a subclass of symmetric functions.

Prior Art

- S. Kamath and D. Manjunath, On distributed computation in structure-free random sensor networks. *In Proceedings of IEEE International Symposium on Information Theory (ISIT-2008), Toronto, Canada, 2008.*
- **Problem considered:** In-network computation of MAX in a structure free random sensor network (structure free = no addresses)
- **Main Result:** They give a protocol that could make MAX available at the sink within a constant factor of the time required by the best coordinated algorithm.
- **Limitations:** They consider the interference protocol model. The protocol they describe is applicable only to functions like MIN and MAX. It is not applicable for any other simple functions like SUM, etc.

Prior Art

- Damon Most-Aoyama and Devavrat Shah, Fast distributed algorithms for computing separable functions. *IEEE Transactions on Information Theory*, 2008.
- **Problem considered:** Distributed computation of functions that are linear combinations (separable functions) of data at all the nodes in the network.
- **Main Result:** They give a distributed randomized algorithm (based on the fact “The sum of exponential random variables is itself an exponential random variable with parameter as sum of the parameters of all the random variables involved”) for computing separable functions that uses a gossip algorithm as the communication protocol.
- **Limitations:** The time taken to send raw data may be much smaller than the time taken by their proposed protocol.

One Possible Problem Formulation: The Context

- **Problem:** In a network with N nodes and a collector node which is interested in a function of the data sensed by each node at $t = 0$, find the joint scheduling and routing that would minimize the time taken by the collector node to obtain the function.
- **Assumptions:** nodes are synchronized, time is slotted, solution is centralized (to start with), each node has an id, no broadcast, links are reliable and there are 2 options, one without aggregation at intermediate nodes (our benchmark) and one with aggregation. Hence there is **no duplication**, this makes the problem much simpler.

The benchmark: case without aggregation

- It is function insensitive as the sink computes the function only after receiving all the data.
- We seek to find the optimal joint scheduling and routing that would minimize the total delay in sending the data from all the nodes to the sink.
- Define the actions:

$$x_{i,j} = \begin{cases} 1 & \text{if link } (i,j) \text{ is active in slot } k \\ 0 & \text{otherwise} \end{cases}$$

- $x_{i,j}$ is defined only if the link (i,j) is in the network.
- Let $q_i(k)$ be the integer variable that represents the number of packets queued at node $i = 1 \dots N$ at the beginning of slot k .

The benchmark: case without aggregation

- We have the following constraints

$$q_i(k+1) = q_i(k) - \sum_j x_{i,j}(k) + \sum_j x_{j,i}(k) \quad i = 1 \dots N, k = 1 \dots T \quad (1)$$

$$\sum_j x_{i,j}(k) \leq q_i(k) \quad i = 1 \dots N, k = 1 \dots T \quad (2)$$

The initial state is given by $q_i(1) = 1$ for $1 \leq i \leq N$.

- The objective is to minimize T such that

$$\sum_{i=1}^N q_i(T) = 0 \quad (3)$$

over all possible binary variables (e.g., actions) $x_{i,j}(k)$.

The benchmark: case without aggregation

- Let T^* be the solution to this optimization problem.
- This problem in a wired network is an instance of the well-known “evacuation problem” or “quickest flow problem”. It is related to the “maximum dynamic flow” problem in a network. It was studied in a **centralized setting** by Ford and Fulkerson, by R. Burkard and B. Hoppe.
- **Objective:** To generalize their results to the aggregation case, to the wireless case, without and with aggregation. These results will serve as benchmarks for our distributed solutions.

The benchmark: The wired case

- Since the nodes that are one-hop neighbors of the sink form a bottleneck, we have the following bounds on T^*

$$\frac{N}{\delta} \leq T^* \leq N - \delta + 1$$

where δ is the degree of the sink node.

- For any given topology, this problem could be solved using the “time-expanded networks” technique.
- The time complexity of this algorithm is pseudo-polynomial but since we have an upper bound on T^* , it is a polynomial time algorithm.

The benchmark: The wired case

- This method is centralized. It would give us the value of T^* , as well as the action to be taken at each node in each time-slot. While it achieves the shortest possible delay, it is not robust to node failure (similar to computing the routing in a centralized manner).
- On the distributed front, we need to find a “simple and robust distributed algorithm” along the line of Bellman Ford or Dijkstra that will achieve a near-optimal solution. For example, a completely decentralized algorithm that uses shortest paths routing may not achieve the minimum delay (i.e., decoupling routing from scheduling does not work).

The benchmark: The wired case

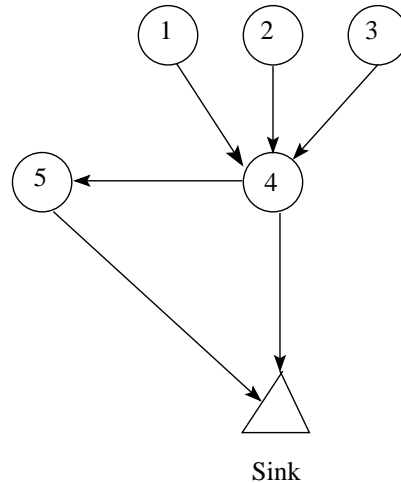


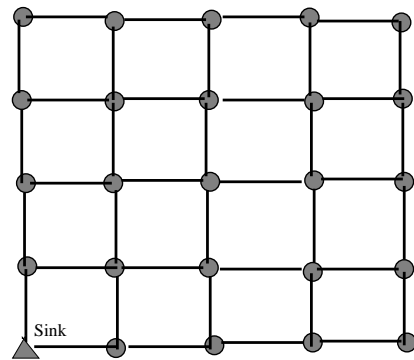
Figure 1: Example to show optimal strategy is a joint routing and scheduling strategy

- This topology shows that a simple shortest path routing is not optimal.
- A joint routing and scheduling strategy achieves the minimum delay.

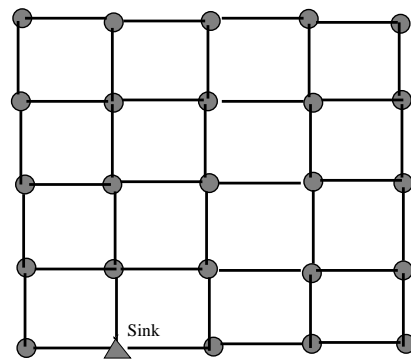
Special Cases

- The following are some results for special topologies
- For a line network and a tree network, routing and scheduling are straightforward as there is only route to the sink for every node and scheduling is to send the packets whenever a node has data, to its parent node.
- **Line Network:** T^* is equal to the total number of nodes in the line.
- **Tree Network:** T^* is equal to the total number of nodes in the heaviest branch rooted at the one-hop neighbors of the sink.

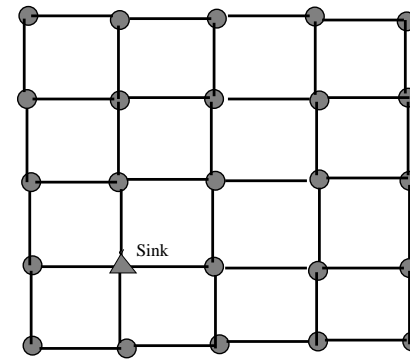
Special Cases



(a) Sink at the corner node



(b) Sink at an edge node



(c) Sink at an interior node

Figure 2: Different locations of the sink in the grid network

- **Grid Network:** Depending upon the location of the sink, we have different value of T^*

- For the kind of configuration in figure 2.a, we have

$$T^* = \left\lceil \frac{(M + 1)(N + 1) - 1}{2} \right\rceil$$

- For the kind of configuration in figure 2.b, we have

$$T^* = \left\lceil \frac{(M + 1)(N + 1) - 1}{3} \right\rceil$$

And for the kind of configuration in figure 2.c, we have

$$T^* = \left\lceil \frac{(M + 1)(N + 1) - 1}{4} \right\rceil$$

- The results in all these grid networks are based on the fact that we could form a *symmetric tree*.

Possible distributed algorithm

- We use shortest path routing as the basic routing and develop an heuristic that might achieve low delay (**not yet tested**).
- Let a_i be the number of next hop nodes for the shortest paths, b_i be the number of neighbours at the same hop distance as node i and q_i be the current number of packets at node i . Then
 - If $q_i \leq a_i$ then send q_i packets to the next hop nodes
 - or if $a_i < q_i \leq 2a_i$ then send a_i packets to the next hop nodes
 - or if $q_i > 2a_i$ then send a_i packets to the next hop nodes and also send $\min\{b_i, q_i - a_i\}$ to the neighbours at the same hop distance.

The benchmark: The wireless case

- Complexity increases in the case of wireless links as compared to wired because in addition to packets, links also need to be scheduled in order to avoid interference between them.
- Since the sink could receive from only one node at a time, we have $T^* \geq N$, where N is the total number of nodes in the network.
- A set of links is an independent set (Iset) if each link in the set can be activated at the same time without creating harmful interference at any of the receivers. We typically use an SINR-based interference model.

The benchmark: The wireless case

- Using the same notation as above, at the beginning of time-slot k , we would select a set $s(k) \in \mathcal{S}$ and then:

$$q_i(k) = q_i(k-1) - \sum_{l \in L_i^O} x_l(k) + \sum_{l \in L_i^I} x_l(k) \quad \forall 0 < i \leq N \quad \forall k > 1 \quad (4)$$

- with the following restrictions:

$$x_l(k) = 0 \quad \forall l \notin s(k) \quad (5)$$

$$\sum_{l \in L_i^O} d_l(k) \leq q_i(k-1) \quad \forall 0 < i \leq N \quad \forall k > 1 \quad (6)$$

$$q_i(0) = 1 \quad \forall 0 < i \leq N. \quad (7)$$

- Finally, we want to minimize T such that

$$\sum_{i=1}^N q_i(T) = 0 \quad (8)$$

over all possible binary variables (e.g., actions) $x_l(k)$ and all sets $s(k)$ subject to all the constraints.

From function to aggregation

- Most sensor networks perform specific functions and it is reasonable to exploit the nature of $f()$ in the design of efficient algorithms.
- Examples of some common functions: max, min, sum, mean and variance. Rarely do we see more complex functions in the literature.
- Objectives:
 - For any given function and a given topology, we would like to find the optimal routing and scheduling strategy as well as the type of intermediate node computation that would achieve the minimum delay.
 - We also want to quantify how more efficient (not necessarily the case) is the solution with aggregation and what is the complexity cost.

From function to aggregation

- We model the intermediate node computation by an aggregation factor $a(k)$, which is the number of output packets after computation, given k packets as input.
- The mapping between a function and its $a(k)$ is not always straightforward for an arbitrary given function. For example, if we consider the function SUM, $a(k)$ would depend on the actual values of the data (because of possible overflow) and not just on k .
- Also we create different types of packets in the network and it is not clear how they can be aggregated among themselves.

The special case of $a(k) = 1$

- An important case, corresponds to a MAX or MIN function for example.
- For a wired network, the optimal routing is to send along the shortest path and the optimal scheduling is to aggregate all the packets at the beginning of a time-slot and to send the resulting packet to the next hop on the shortest path to the sink.
- Multiple shortest-paths do not help in reducing T^* as every node has at most one packet to send at the beginning of any time-slot.
- For any general topology, T^* is equal to the length of the longest *shortest path* between any node and the sink.
- For a line network, aggregation doesn't help and T^* is still equal to the number of nodes in the line.

The special case of $a(k) = 1$

- For a tree network, T^* is equal to the distance between the sink and furthest leaf node.
- For a grid network of $(M + 1) \times (N + 1)$ structure with the sink node at (i, j) ,

$$T^* = \max\{M - i + N - j, i + j, i + N - j, M - i + j\}$$

- Hence, in a grid, without this aggregation, T^* is of the order of $M * N$, while with aggregation, it is only of the order of $M + N$. Hence, aggregation gives lower delay.
- For a wireless network, it is easy to see that $T^* \leq N$, where N is the total number of nodes in the network. Without aggregation, $T^* \geq N$, so $a(k) = 1$ always gives better performance in terms of delay.

The special case of the function SUM

- We could artificially get $a(k) = 1$ by making the data field large enough (by increasing it by $\log_2 N$ bits) so that the sum of $k \leq N$ values never overflows.
- For a fair comparison with the case without aggregation, we need to normalize the time-slots lengths.
- Let the length of the header be h bits, the length of each data be m bits and the length of the longest *shortest path* between the sink and any other node be L . If $L > \frac{N(m+h)}{\log_2 N + m + h}$, then the minimum time required to send the SUM to the sink using this method would be greater than the minimum time taken to send all the data to the sink.

The special case of the variance

- The sink could compute the variance if it knows the total number of nodes in the network, the sum of all the data and the sum of squares of all the data.
- So, the intermediate nodes should aggregate the data as sum and sum of squares packets. Hence three types of packets.
- If we assume that overflow doesn't occur when aggregating, then we have $a(k) \leq 2$.
- For a general topology of a wired network, the optimal strategy could be a complex joint routing and scheduling.
- However, if the network is a tree, the routing is fixed and the scheduling is that every node should send the packets whenever it has one, to its parent node and T^* could be computed by using a recursive algorithm.

The special case of the variance

- For a wireless network, we give an upperbound on T^* . The worst case occurs when only a single link could be active at a time. Hence

$$T^* \leq n_{\max} + 2 \sum_{i=1}^{\max-1} n_i = 2N - n_{\max}$$

where n_i is the number of nodes at the hop-distance of i from the sink.

Conclusions and other challenges

- What if the network is unstructured i.e. without addresses?
- What if the links are unreliable?
- What if we use broadcast?
- What if we use random access?