Lower Bounds for Information Gathering in Adversarial Systems (Extended Abstract)

Kishore Kothapalli
Department of Computer Science
Johns Hopkins University
3400 N. Charles Street
Baltimore, MD 21218, USA.
kishore@cs.jhu.edu

Christian Scheideler
Computer Science Department
Technische Universität München
D-85748 Garching
Germany.
scheideler@in.tum.de

Abstract

In this paper we consider the problem of routing packets to a common destination, also known as *information gathering*. Information gathering is an important communication primitive in sensor networks and allows an observer to collect information from the sensors. Because sensors usually do *not* move, they form a static topology of possible communication links, but since sensors may frequently be in sleep mode or their communication may be disrupted by interference or obstacles, communication links may be up and down in an unpredictable way. In order to ensure that algorithms work well in *any* scenario, we assume that the state of the edges and also packet injections are under the control of an adversary.

Recent studies of information gathering under adversarial models have shown that for simple topologies such as the line graph it suffices for an online algorithm to have a buffer size that is a logarithmic factor bigger than that of the optimal algorithm to achieve full throughput. In this paper, surprisingly, it will be shown that even small deviations to the line graph such as a fork graph have a huge overhead on the buffer size and therefore pose significant challenges for online algorithms in the worst case.

Our study suggests that one should either look at randomized algorithms or sacrifice a fraction of the throughput. In the latter context, we show that for the line graph the throughput improves exponentially as we allow a slightly higher overhead in the buffer size.

1 Introduction

Communication problems arising in sensor networks have attracted a lot of research attention in recent years. A large number of projects that study a variety of communication problems in sensor networks have been initiated and evaluated experimentally [1, 7, 10]. But much of the theoretical work remains to be done. However, especially for sensor networks, where human intervention is limited once the sensors are deployed, theoretical work is very important to ensure that the algorithms indeed work correctly and efficiently under *any* circumstances. In order to ensure a high efficiency the protocols must adapt themselves to dynamic network conditions so that resources, such as energy, can be used as efficiently as possible.

Communication in sensor networks falls under two modes: sensor-to-observer and observer-to-sensor. However, sensor-to-observer is by far the dominating mode of communication. The special case of a single observer is also called *information gathering* where all the injected packets have the same destination. For the case of information gathering, recent studies [8] have shown that there exist simple, local-control online strategies that are efficient with regard to throughput and buffer size when the network topology is either a line or a cycle. A natural question then is to find out whether such results exist also for other simple

topologies. In this paper, we answer this question in the negative by proving lower bounds on the buffer size required by online algorithms to compete with the optimal algorithm with regard to throughput.

1.1 Model

We model the sensor network as a graph $G = (V, E)$ where the nodes represent the sensors and the edges represent the potential communication links. All the injected packets have the same destination in G. We assume that time proceeds in synchronized steps. The graph G has a static topology but the edges may not be available for communication at all time steps. The edges are available for communication only during certain time steps and the edges that are available for communication during a time step are also called *active* edges. To achieve reliable communication over active edges, we assume that a MAC layer support along with virtual carrier sensing is available. During every time step at most one packet can be sent along each active edge and every packet needs one time unit to cross an edge. We consider only directed edges. This does not exclude the undirected edge case as, along with MAC layer support, each undirected edge can be seen as two directed edges, one in each direction. The edge activations and the packet injections are assumed to be under adversarial control.

Given nodes with buffer size B, we let the adversary inject an arbitrary number of packets at any node and activate (or deactivate) an arbitrary subset of edges at each time step so long as no packet has to be deleted while using an optimal routing algorithm. Thus, while using the optimal routing algorithm each node has at most B packets at any time step and all other packets have been successfully delivered to the destination. The throughput of an algorithm is measured as the number of packets successfully delivered.

To study the performance of online algorithms we use competitive analysis. For any sequence σ of edge activations and packet injections by the adversary such that an optimal algorithm with buffer size B never has to delete a packet, we say that algorithm A is c-competitive if algorithm A with buffer size cB also never has to delete any packet. In the following, B will always mean the buffer size of an optimal routing algorithm. The *height* of node v, denoted $\mathrm{ht}(v)$, is the number of packets stored in the buffer at that node.

For the case of a fixed buffer space, we are interested in studying the throughput achieved by the online algorithm. Algorithm A is said to be (r, c)–competitive if for any sequence σ of edge activations and packet injections, algorithm A with buffer size rB achieves a throughput of at least c times that of the throughput achievable by an optimal algorithm using buffer size B. The optimal algorithm is not allowed to delete any packets while the online algorithm may have to delete packets.

1.2 Related work

Communication problems under the presence of an adversary have been studied in several papers. In the context of queuing it has been studied in [5, 6, 12]. In the context of routing, the study of adversarial models has been presented in [4]. The special case of information gathering has been the study of several recent papers, for example [3, 8]. In [3], the authors consider an adversary that is as powerful as in our model and show a competitive ratio of $O(n)$ for any general network topology. However, it was left as an open question whether focusing on specific topologies would result in better algorithms. In [8], this question was positively answered for the line and cycle topologies.

1.3 Our Results

Our work considers various simple topologies and shows that for online algorithms performing information gathering, there exist adversarial scenarios which force the online algorithms to use a huge buffer space. A common feature in most of the topologies we consider, though they are simple extensions to the line graph, is that there exist nodes, also called *branching points*, where multiple paths meet. In our model, the online algorithms have to decide whether or not to move a packet along an active edge. Since edges are active

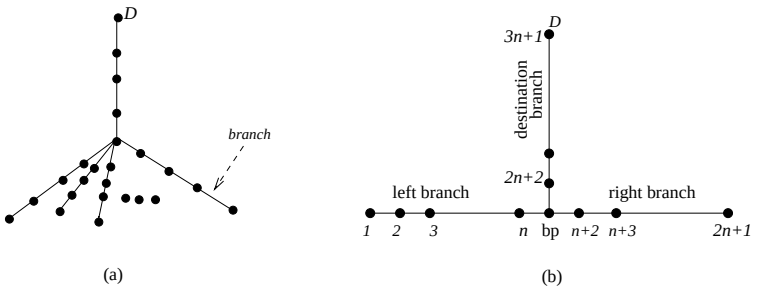

Figure 1: Figure (a) shows a star graph and (b) shows a fork graph.

only during certain adversarially selected time steps, packets moved across branching points by an online algorithm mistakenly may be hard to get back. As we will show, the adversary can exploit such situations often enough without violating the limitations imposed in the model so that the online algorithm requires a huge buffer size to store all the packets. Proofs omitted from this extended abstract can be found in [9].

In Section 2, we show that for the star graph on n nodes (see Figure 1(a)), *any* deterministic online algorithm has a lower bound of $\Omega(n)$. In Section 3, we show that for the fork graph of $3n + 1$ nodes (see Figure 1(b)), the competitive ratio of any T_c–bounded algorithm (defined in Section 3.1) has a lower bound of $\Omega(n^{\frac{1}{\log\log n}})$.

In Section 4, along the line of work in [2], we also investigate the throughput achievable on the line network when the online algorithm is allowed to use a buffer space that is only dependent on the buffer space of the optimal algorithm. We show that the throughput has an exponential improvement when the buffer size used by the online algorithms is twice that of the optimal algorithm (see also [11]).

2 Lower Bounds for the Star Graph

In this section, we show that for the star graph shown in Figure 1(a), *any* deterministic online algorithm has a huge lower bound. Each branch has the same number of nodes c, for some constant c. Let $m = \Theta(n)$ denote the number of branches.

The adversary is adaptive and works in stages. The steps of the adversary in each stage are described below. Assuming that the adversary has performed $k-1$ stages, we show how to proceed in stage k. At the beginning of stage k, the adversary has a set S_k of $m-k+1$ branches such that the average height of nodes in S_k is at least $(k-1)B/2$. The rest of the k branches are denoted by $\overline{S}_k$. Further, at the beginning of stage k, the optimal algorithm has no packets at any node. The adversary selects two branches A_1 and A_2 from S_k and does the following.

1. Inject B packets at each node in branch A_1. This injection increases the average height of nodes in A_1 to at least $(k+1)B/2$.
2. Activate edges so that all the packets in branch A_1 could be moved to branch A_2. At the end of all these activations, any deterministic algorithm has at least $ckB/2$ packets in one of the branches, which we refer to as the *heavier* branch and at most $ckB/2$ packets in one branch which we refer to as the *lighter* branch. The optimal algorithm has all the newly injected packets in the lighter branch only.
3. Activate edges so that all the packets in the *lighter* branch could be moved to the destination. The optimal algorithm has no packets in the lighter branch.
4. Now using the k branches in $\overline{S}_k$ along with the lighter branch, the adversary creates one branch C with the average node height at least $(k-1)B/2$. The set S_k is updated by replacing the lighter branch with branch C. The heavier branch is added to the set S_{k+1}.

Since at the end of the above procedure the number of branches in S_k reduces by 1, the adversary can use the procedure for $m-k$ times. At the end the set S_{k+1} contains $m-k$ branches such that the height

of nodes in S_{k+1} is at least $kB/2$ and the set $\overline{S}_{k+1}$ contains $k+1$ branches. Further, the optimal algorithm has no packets at any of the nodes. The adversary can thus perform $m-1$ such stages at the end of which there is one branch with at least $c(m-1)B/2$ packets. From the above discussion, the following theorem holds. Since for general graphs, the upper bound is shown to be $O(n)$ in [3], Theorem 2.1 gives a tight lower bound. The above approach of the adversary also extends to the case of trees of small degree, such as complete binary trees. The details can be found in [9].

Theorem 2.1. *Any deterministic online algorithm has a lower bound of $\Omega(n)$ on the star graph.*

3 Lower Bounds for Simple Fork Graphs

Since a star graph on n nodes has a node of degree $\Theta(n)$, intuitively it might be said that this presents too many opportunities for the adversary to create situations that are difficult to handle for online algorithms. However, in this section we show that even a single branching point in the topology is enough for the adversary to force the online algorithms to require a huge buffer space.

In this section, we consider the problem of information gathering in the fork graph shown in Figure 1(b). All packets have the same destination marked as node D. Each branch of the fork consists of n nodes and the node `bp` (for *b*ranching *p*oint) is the node where the three branches of the fork meet. The lower bound applies to the class of T_c–bounded algorithms, defined below.

3.1 T_c–bounded algorithms

We now define a class of deterministic online algorithms called the T_c–bounded algorithms for a fixed constant $T_c \geq 0$. We call T_c the *climbing threshold* of the online algorithm which denotes the height up to which packets may climb up along an active edge $u \rightarrow v$ even if $\mathrm{ht}(u) < \mathrm{ht}(v)$. More formally, when $\mathrm{ht}(u) \leq \mathrm{ht}(v)$, if $|\mathrm{ht}(u) - \mathrm{ht}(v)| \leq T_c$, the online algorithm ***may*** send a packet along an active edge $u \rightarrow v$. However, if $\mathrm{ht}(u) < \mathrm{ht}(v)$ and $|\mathrm{ht}(u) - \mathrm{ht}(v)| > T_c$ the online algorithm is **not** allowed to move a packet along edge $u \rightarrow v$. The above definition is not limiting since most online algorithms [3, 8] do not let packets climb above a certain threshold.

We now show the following lower bound for simple fork graphs for any T_c–bounded online algorithm.

Theorem 3.1. *Any T_c–bounded online algorithm has a lower bound of $\Omega(n^{1/\log\log n}\log n)$ on a fork graph of $3n+1$ nodes.*

Proof (sketch): The adversary works in phases and each phase consists of two stages namely, the *building stage* and the *spreading stage*. In phase $i \geq 0$, the adversary works with a consecutive set of nodes L_i, R_i and D_i in the left, right and the destination branch respectively. For $i = 0$, $L_i = \{1, 2, \cdots \mathtt{bp}\}$ and $R_i = \{n+2, n+3, \cdots, 2n+1\}$ and D_i similarly consists of all nodes in the destination branch.

In the *building stage* of phase i, the adversary uses the nodes in L_i and D_i to create a height of h_i at node `bp`. The height h_i is defined as $h_1 = B\log n - T_c$ and $h_{i+1} = 2h_i - T_c$. Due to its large height, the node `bp` acts as a barrier preventing the online algorithm to move any packets injected in the right branch to be moved towards destination. The building stage ends when all nodes in R_i have a height of h_{i+1}.

The *spreading stage* of phase i involves creating consecutive nodes L_{i+1}, R_{i+1} and D_{i+1} in each of the left, right and destination branches respectively. This time adversary uses the R_i nodes to create a height of h_{i+1} in the L_{i+1} and D_{i+1} nodes with $|L_{i+1}| = |L_i|/\log n$ and $|D_{i+1}| = |D_i|/\log n$. Here the adversary injects packets in the L_i (resp. D_i) nodes and activates edges to the R_i nodes. This causes the online algorithm to keep the packets at L_i(resp. D_i) nodes whereas the optimal algorithm can move the packets to the R_i nodes.

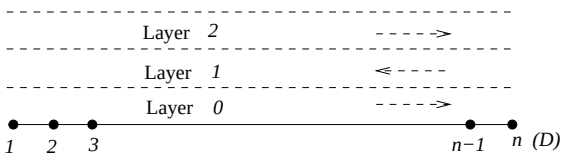

Figure 2: Line network with layers of alternating direction.

The adversary can perform $i = \Omega(\log n / \log\log n)$ such phases at the end of which there is at least one node in the online algorithm with a height of $2^{\Omega(\log n/\log\log n)} B \log n = \Omega(n^{1/\log\log n} B \log n)$. □

4 Bounds for the Line Graph with a Fixed Buffer

As shown in the previous sections, to achieve the same throughput as that of the optimal algorithm, the buffer size required by online algorithms could be as high as the size of the network except for line graphs and cycle graphs. Hence, as in [2], it is important to study the throughput achievable by online algorithms when only a buffer of fixed size, independent of the size of the network, is used.

We consider the line graph on n nodes with nodes numbered from left to right starting with 1 and node n being the destination of all packets (see Figure 2). It is easy to notice that when using the same buffer space as that of the optimal algorithm the throughput achievable is $\Omega(1/n)$. However, when just doubling the buffer space, we show the throughput improves exponentially to $\Omega(1/\sqrt{n})$ for the GuidedBalancing algorithm [8]. A brief review of the GuidedBalancing algorithm is provided below.

The GuidedBalancing Algorithm

In the GuidedBalancing algorithm each node in the network has a buffer that is used to store packets. Each buffer has *slots* numbered consecutively starting from 1 and each slot can store exactly one packet. The slots are grouped into layers where each *layer* is a consecutive set of B slots. The layers are numbered consecutively starting from 0 and the slots 1 to B belong to layer 0 and $iB + 1$ to $(i + 1)B$ belong to layer i, for $i \geq 0$. The direction of the layer is the direction in which packets can be moved by the algorithm in that layer. Thus, in an Always-Towards-Destination layer packets can only move towards the destination and in an Always-Away-Destination layer packets can only move away from the destination. The algorithm maintains layers that are alternating in direction starting with layer 0 being an Always-Towards-Destination layer, as shown in Figure 2. We now present the GuidedBalancing algorithm [8].

Algorithm GuidedBalancing
1. for every time step and every active edge (u, v)
2. if there exists a layer ℓ with the direction same as the direction of (u, v)
   so that u has at least 1 packet in layer ℓ and v has at most B packets in layer ℓ then
3. move a packet in layer ℓ from u to v
4. for every time step at every node u
5. accept the incoming packets and the injected packets and store them at the lowest
   empty slots available currently.

End Algorithm

We now prove a lower bound on the throughput achievable by the GuidedBalancing algorithm while using $2B$ slots. We do this by proving an upper bound on the number of deletions that the GuidedBalancing algorithm can suffer via an interesting *witnessing mechanism*. A matching upper bound on throughput is shown in [9].

Theorem 4.1. *On the line graph of n nodes, the GuidedBalancing algorithm has a competitive ratio of $(2, \Omega(1/\sqrt{n}))$ on the throughput.*

Proof (sketch): Consider assigning a witness for every packet deleted by the online algorithm. This witness is assigned as a pair of packets in the online algorithm. This process can be modeled as a graph $G_d = (V_d, E_d)$, the deletion graph, with vertices representing the packets in the online algorithm and edges representing deletions encountered by the online algorithm. For a node $u \in V_d$, let $P(u)$ denote the corresponding packet in the online algorithm. It can be shown that the deletion graph has the following property [9].

Lemma 4.2. *At all time steps, between any two nodes $u, v \in V_d$, there is at most one edge.*

The proof of the theorem follows from the above lemma. □

5 Conclusions

Our lower bounds show that relatively simple topologies already create huge challenges for information gathering in sensor networks. Thus, one should either use randomized algorithms or suffer a loss in throughput to improve the competitive ratio.

While theoretical lower bounds show the limit of what online algorithms can hope to achieve, the average case behavior of the online algorithms might still be much better. While our preliminary experimental results support this hypothesis, detailed experimental studies are required. It would also be interesting to see if relaxing the power of the adversary model considered would lead to better results.

References

[1] Pico project. Available at http://bwrc.eecs.berkeley.edu/Research/Pico_Radio/Default.htm.

[2] AIELLO, W., OSTROVSKY, R., KUSHILEVITZ, E., AND ROSÉN, A. Dynamic routing on networks with fixed-size buffers. In *ACM SODA* (2003), pp. 771–780.

[3] AWERBUCH, B., BERENBRINK, P., BRINKMANN, A., AND SCHEIDELER, C. Simple online strategies for adversarial systems. In *IEEE FOCS* (2001), pp. 158–167.

[4] AWERBUCH, B., AND LEIGHTON, T. Improved approximation algorithms for the multi-commodity flow problem and local competitive routing in dynamic networks. In *ACM STOC* (1994), pp. 487–496.

[5] BORODIN, A., KLEINBERG, J., RAGHAVAN, P., SUDAN, M., AND WILLIAMSON, D. Adversarial queuing theory. In *ACM STOC* (1996), pp. 376–385.

[6] GOEL, A. Stability of networks and protocols in the adversarial queueing model for packet routing. In *ACM SODA* (1999), pp. 911–912.

[7] INTANAGONWIWAT, C., GOVINDAN, R., AND ESTRIN, D. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proc. of the 6th ACM/IEEE Mobicom Conference* (2000), pp. 56–67.

[8] KOTHAPALLI, K., AND SCHEIDELER, C. Information gathering in adversarial systems: lines and cycles. In *ACM SPAA* (2003), pp. 380–389.

[9] KOTHAPALLI, K., AND SCHEIDELER, C. Lower bounds for information gathering in adversarial systems. Available at www.cs.jhu.edu/∼kishore/, 2005.

[10] LINDSEY, S., AND RAGHAVENDRA, C. PEGASIS: Power Efficient GAthering in Sensor Information Systems. In *Proc. of IEEE Aerospace Conference* (2002), pp. 3:1125–1130.

[11] REHRMANN, R., MONIEN, B., LULING, R., AND DIEKMANN, R. On the communication throughput of buffered multistage interconnection networks. In *ACM SPAA* (1996), pp. 152–161.

[12] SCHEIDELER, C., AND VÖCKING, B. From static to dynamic routing: efficient transformations of store-and-forward protocols. In *ACM STOC* (1999), pp. 215–224.