13 Sensor networks

Wireless sensor systems have a broad range of civil and military applications such as controlling inventory in a warehouse or office complex, monitoring and disseminating traffic conditions, or exploring regions affected by a disaster or military conflict. With recent advances in microelectromechanical systems (MEMS) technology and its associated interfaces, signal processing, and radio and infrared circuitry the focus has shifted away from limited macrosensors communicating with base stations to creating wireless networks of communicating microsensors that aggregate complex data to provide rich, multi-dimensional pictures of the environment. While individual microsensor nodes are not as accurate as their macrosensor counterparts, the networking of a large number of nodes enables high quality sensing networks with the additional advantages of easy deployment and fault-tolerance. These characteristics make microsensors ideal for deployment in otherwise inaccessible environments where maintenance would be inconvenient or impossible.

Communication within a sensor network can be classified into two categories: *application* and *infrastructure*. The network protocol must support both types of communication.

Infrastructure communication refers to the communication needed to configure, maintain, and optimize operation. Thus, infrastructure communication is needed to keep the network functional, ensure robust operation in dynamic environments, as well as optimize overall performance.

Application communication consists of two communication modes: sensor-to-observer and observer-to-sensor communication. Sensor-to-observer communication is the by far dominating mode. Hence, it is of highest priority to find flexible and efficient protocols for this mode. In general, the problem of forwarding information to a single point is called *gathering*. For the rest of this section, we will study how to come up with suitable routing algorithms for gathering. First, we study the problem of gathering in adversarial environments, and then we study the problem of gathering in a static environment.

13.1 Gathering in an adversarial environment

In order to compare our gathering algorithm with a best possible gathering algorithm, we will use competitive analysis.

Suppose that we have an adversary that has full control over the network topology and the injection of packets. Our aim will be to find an online algorithm for this case that can compete with a best possible algorithm. Each time a packet reaches one of its destinations, we count it as one *delivery*. The number of deliveries that is achieved by an algorithm is called its *throughput*. We are interested in maximizing the throughput together with a low storage overhead. In particular, since the adversary is allowed to inject an unbounded number of packets, we allow to drop (or merge) packets so that a high throughput can be achieved with a buffer size that is as small as possible.

Given any sequence of edge activations and packet injections σ, let $\mathrm{OPT}_B(\sigma)$ be the maximum possible throughput (i.e. the maximum number of deliveries) when using a buffer of size B in every node, and let $A_{B'}(\sigma)$ be the throughput achieved by some given online algorithm A with buffer size B'. We call an online algorithm A (t, s)-*competitive* if for all σ and all B, A can guarantee that

$$A_{s\cdot B}(\sigma) \geq t \cdot \mathrm{OPT}_B(\sigma) - r$$

for some $r \geq 0$ that is independent of σ (but may depend on s, B and n). $t \in [0, 1]$ denotes here the fraction of the best possible throughput that can be achieved by A and s denotes the space overhead

necessary to achieve this. If t can be brought arbitrarily close to 1, A is also called $s(\epsilon)$-*competitive* or simply *competitive*, where $s(\epsilon)$ reflects the relationship between s and ϵ with $t = 1 - \epsilon$. Obviously, it always holds that $s(\epsilon) \geq 1$, and the smaller $s(\epsilon)$, the better is the algorithm A.

In order to perform gathering in an adversarial environment, we use a balancing algorithm. Let $h_{v,t}$ denote the amount of packets in node v at the beginning of time step t. For the observer q, $h_{q,t} = 0$ for all t. In every time unit $t \geq 1$ the T-*balancing algorithm* performs the following operations.

1. For every active edge $e = (v, w)$, check whether $h_{v,t} - h_{w,t} > T$. If so, send a packet from v to w (otherwise do nothing).

2. Receive incoming packets and newly injected packets, and absorb those that reached the observer.

3. Eliminate (or merge) any packet that cannot be stored due to a full buffer.

Recall our competitive model above. We will show that T-balancing algorithm achieves the following result:

Theorem 13.1 ([2]) *Suppose an adversary is allowed to injected an unbounded number of packets and to generate an arbitrary network of maximum degree Δ in each time step. If $T \geq B + 2(\Delta - 1)$, then the T-balancing algorithm is $1 + (1 + (T + \Delta)/B)L/\epsilon$-competitive, where L is the average path length used by successful packets in an optimal solution with buffer size B.*

Proof. Let n be the number of sensor nodes in the system, and let node 0 represent the observer. As mentioned previously, the height of node 0 is always 0, since any packet reaching 0 will be absorbed. For each of the remaining nodes we assume that it has H slots to store packets. The slots are numbered in a consecutive way starting from below with 1. Every slot can store at most one packet. After every step of the balancing algorithm we assume that if a node holds h packets, then its first h slots are occupied. The *height* of a packet is defined as the number of the slot in which it is currently stored. If a new packet is injected, it will obtain the lowest slot that is available after all packets that are moved to that node from another node have been placed.

For every successful packet in an optimal algorithm a schedule can be identified. A schedule $S = (t_0, (e_1, t_1), \ldots, (e_\ell, t_\ell))$ is called *active* at time t if $t_0 \leq t \leq t_\ell$. The *position* of a schedule at time t is the node at which its corresponding packet would be at that time if it is moved according to S. An edge is called a *schedule edge* if it belongs to a schedule of a packet. Suppose that we want to compare the performance of the balancing algorithm with an optimal algorithm that uses a buffer size of B. Then the following fact obviously holds.

Fact 13.2 *At every time step, at most B schedules can have their current position at some node v.*

Next we introduce some further notation. We will distinguish between three kinds of packets: *representatives*, *zombies*, and *losers*. During their lifetime, the packets have to fulfill certain rules. (These rules will be crucial for our analysis. The balancing algorithm, of course, cannot and does not distinguish between these types of packets.) Every injected packet that does not have a schedule will initially be a zombie. Every other packet will initially be a representative. If a packet is injected into a full node, then the highest available loser will be selected to take over its role.

We want to ensure that a representative always stays with its schedule as long as this is possible. Two cases have to be considered for this when the adversary offers a schedule edge $e = (v, w)$:

1. A packet is sent along e: Then we always make sure that this packet is the representative belonging to e.

2. No packet is sent along e: If w has a loser, then the representative exchanges its role with the highest available loser in w. In this case we will also talk about a *virtual* movement. Otherwise, the representative is simply transformed into a loser. In this case, we will disregard the rest of the schedule (i.e. we will not select a representative for it afterwards and the rest of the schedule edges will simply be treated as non-schedule edges).

Furthermore, if a packet is sent along a non-schedule edge $e = (v, w)$, then we always make sure that none of the representatives is moved out of v but only a loser (which always exists if T is large enough).

The three types of packets are stored in the slots in a particular order. The lowest slots are always occupied by the losers, followed by the zombies and finally the representatives. Every zombie that happens to be placed in a slot of height at most $H - B$ will be immediately converted into a loser. Together with Fact 13.2 these rules immediately imply the following fact.

Fact 13.3 *At any time, the number of zombies and representatives stored in a node is at most B.*

Let $h_{v,t}$ be the *height* of node v (i.e. the number of packets stored in it) at the beginning of time step t, and let $h'_{v,t}$ be its height when considering only the losers. The *potential* of node v at step t is defined as $\phi_{v,t} = \sum_{j=1}^{h'_{v,t}} j = \binom{h'_{v,t}+1}{2}$ and the potential of the system at step t is defined as $\Phi_t = \sum_v \phi_{v,t}$. First, we study how the potential can change in a single step. Since no two packets can cross the same edge at the same time, we arrive at the following fact.

Fact 13.4 *At any time t, any edge activated at time t belongs to at most one schedule.*

Hence, an edge can only belong to one or no schedule. To simplify the consideration of these two cases, we consider the edges that are active in a time step one by one, starting with non-schedule edges and always assuming the worst case concerning previously considered edges. When processing these edges, we always use the (worst case) rule that if a loser is moved to some node w, it will for the moment be put on top of all old packets in w. This will simplify the consideration of edges that belong to a schedule. At the end, we then move all losers down to fulfill the ordering condition for the representatives, zombies, and losers. This will certainly only decrease the potential. Using this strategy, we can show the following result.

Lemma 13.5 *If $T \geq B + 2(\Delta - 1)$, then any non-schedule edge does not increase the potential of the system.*

Proof. Consider any fixed non-schedule edge $e = (v, w)$. If no packet is sent along e, the lemma is certainly true. Otherwise, it holds that $h_{v,t} - h_{w,t} > T$. If $T \geq B + 2(\Delta - 1)$, then even after $\Delta - 1$ removals of packets from v and the arrival of $\Delta - 1$ packets at w, there are still losers left in v, and the height of the highest of these is higher than the height of w. Hence, we can avoid moving any representative away from the position of its schedule and instead move a loser from v to w without increasing the potential. □

For schedule edges, only a slight increase in the potential is caused.

Lemma 13.6 *If $T \geq B + 2(\Delta - 1)$, then every schedule edge increases the potential of the system by at most $T + B + \Delta$.*

Proof. Consider some fixed schedule edge $e = (v, w)$. If e is selected for the transmission of a packet, then we can send the corresponding representative along e, which has no effect on the potential.

Otherwise, it must hold that $\delta_e = h_{v,t} - h_{w,t} \leq T$. In this case, the representative R for e has to be moved virtually or transformed into a loser.

First of all, note that our rule of placing new losers on top of the old packets makes sure that the height of the representative in v does not increase. Furthermore, there are two ways for w to lose losers before considering e: either an unused schedule edge to w forced a virtual movement of a representative to w, or a used non-schedule edge from w forced to move a loser out of w. Let s be the number of edges with the former property and ℓ be the number of edges with the latter property. If w had r representatives (and zombies) at the beginning of t, then it must hold that $r + s - (\Delta - \ell) \leq B$ to ensure that at the end of step t w has at most B representatives. Thus, $r + s + \ell \leq B + \Delta$. Hence, if there is still a loser left in w when considering e, the highest of these must have a height of at least $h_{w,t} - (B + \Delta)$. Therefore, if $h_{w,t} > B + \Delta$, then it is possible to exchange places between R and a loser in w so that the potential increases by at most

$$h_{v,t} - (h_{w,t} - (B + \Delta)) = \delta_e + B + \Delta \ .$$

Since $\delta_e \leq T$, this is at most $T + B + \Delta$.

If $h_{w,t} \leq B - \Delta$, then it may be necessary to convert R into a loser. However, since $h_{v,t} - h_{w,t} \leq T$, this increases the potential also by at most $T + B + \Delta$. □

In addition to edges, also injection events and the transformation of a zombie into a loser can influence the potential. This will be considered in the next two lemmata.

Lemma 13.7 *Every deletion of a newly injected packet decreases the potential by at least $H - B$.*

Proof. According to Fact 13.3, the highest available loser in a full node must have a height of at least $H - B$. Since the deletion of a newly injected packet causes this loser to be transformed into a representative or zombie, this decreases the potential by at least $H - B$. (Note that in case of a zombie, it might be directly afterwards converted back into a loser, but this will be considered in the next lemma.) □

If an injected packet is not deleted, this will initially not affect the potential, since it will either become a representative or a zombie. However, a zombie may be converted into a loser.

Lemma 13.8 *Every zombie can increase the potential by at most $H - B$.*

Proof. Note that zombies do not count for the potential. Hence, the only time when a zombie influences the potential is the time when it is transformed into a loser. Since we allow this only to happen if the height of a zombie is at most $H - B$, the lemma follows. □

Now we are ready to prove an upper bound on the number of packets that are deleted by the balancing algorithm.

Lemma 13.9 *Let σ be an arbitrary sequence of edge activations and packet injections. Suppose that in an optimal strategy, s of the injected packets have schedules and the other z packets do not. Let L be the average length of the schedules. If $H \geq B + 2(\Delta - 1)$, then the number of packets that are deleted by the balancing algorithm is at most*

$$s \cdot \frac{L(T + B + \Delta)}{H - B} + z \ .$$

Proof. First of all, note that only newly injected packets get deleted. Let p denote the number of schedule edges and d denote the number of packets that are deleted by the balancing algorithm. Since

- due to Lemma 13.5 non-schedule edges do not increase the potential,
- due to Lemma 13.6 every schedule edge increases the potential by at most $T + B + \Delta$,
- due to Lemma 13.7 every deletion of a newly injected packet decreases the potential by at least $H - B$, and
- due to Lemma 13.8 every zombie increases the potential by at most $H - B$,

it holds for the potential Φ after executing σ that

$$\Phi \leq p \cdot (T + B + \Delta) + z \cdot (H - B) - d \cdot (H - B) \ .$$

Since on the other hand $\Phi \geq 0$, it follows that

$$d \leq \frac{p \cdot (T + B + \Delta)}{H - B} + z \ .$$

Using in this inequality the fact that the average number of edges used by successful packets is at most L, and therefore the number of injected packets with a schedule, s, satisfies $s \geq p/L$, concludes the proof of the lemma. □

From Lemma 13.9 it follows that the number of packets that are successfully delivered to their destination by the balancing algorithm must be at least

$$s + z - \left(s \cdot \frac{L(T + B + \Delta)}{H - B} + z \right) - H \cdot n = s \cdot \left(1 - \frac{L(T + B + \Delta)}{H - B} \right) - H \cdot n \ ,$$

where n is the number of sensor nodes. For $H \geq L(T + B + \Delta)/\epsilon + B$ this is at least

$$(1 - \epsilon)s - r$$

for some value r independent of the number of packets successful in an optimal schedule. □

13.2 Gathering in a static environment

Next, we consider the case where the network and the injection pattern are fixed. For simplicity, we assume that instead of sending discrete packets, we can send arbitrary fractional flows along the edges as long as the total amount of flow traversing an edge is at most one. In this case, we use the following algorithm

Let $\bar{h}_{v,t} = h_{v,t}/\Delta$, where Δ is the maximum degree of the given network. In every time step $t \geq 1$ the (τ, Δ)-*balancing algorithm* performs the following operations.

1. For every edge $e = (v, w)$, check whether $\bar{h}_{v,t} - \bar{h}_{w,t} > \tau$. If so, send a flow of $\min[\bar{h}_{v,t} - \bar{h}_{w,t} - \tau, 1]$ from v to w (otherwise do nothing).

2. Receive incoming flow and newly injected flow, and absorb the flow that reached the observer.

The next theorem shows that as long as the injection pattern does not exceed the capacity of the network, the distribution of flow among the nodes when using the (σ, Δ)-algorithm will approach a *fixpoint*. A fixpoint is a point in which the amount of flow at every node remains unchanged, i.e. for every node the amount of outgoing flow is equal to the amount of incoming flow. Its proof is rather involved and has therefore been omitted here.

Theorem 13.10 ([1]) *For any static network, any sustainable static flow injection pattern, and any $\tau \geq 0$ the distribution of the flow will converge towards a fixpoint.*

Since flow can only move from higher nodes to lower nodes, this implies that the flow movements form fixed, connected, and loop-free paths. The sum of the flow values of these paths is equal to the amount of flow injected into the system in every time step. Let the (weighted) average path length L_{BAL} of these paths be defined as follows.

$$L_{BAL} = \tfrac{1}{\lambda} \sum_{\text{flow paths } p} \ell_p \cdot \lambda_p$$

where ℓ_p is the length of flow path p, λ_p is the amount of flow following path p, and $\lambda = \sum_{\text{flow paths } p} \lambda_p$. The question is how close L_{BAL} can be to a best possible average path length, L_{OPT}. The next theorem gives the answer to this question.

Theorem 13.11 *For any static network, any sustainable static flow injection pattern, and any $\tau > 0$ the (τ, Δ)-balancing algorithm ensures that, in the fixpoint, $L_{BAL} \leq (1 + 1/\tau) L_{OPT}$.*

Proof. We know that at the fixpoint of the balancing algorithm we have a fixed collection P of paths whose demands sum up to the total injection rate. Let Q be any collection of optimal paths for the given injection process. W.l.o.g. we can assume that P and Q have the same number of paths, and the ith path of P and Q has the same demand and connects the same source-destination pair (otherwise split the paths of P and Q into subpaths such that this property is fulfilled). Thus, let $p_1, \ldots, p_r$ be the paths in P, and let $q_1, \ldots, q_r$ be the paths in Q. Let ℓ_i denote the length of p_i and k_i denote the length of q_i. Furthermore, let λ_i be the flow value of path p_i and q_i. Suppose now that

$$\sum_i \lambda_i \ell_i > \frac{\tau + 1}{\tau} \sum_i \lambda_i k_i \; . \tag{1}$$

We will show via contradiction that this is not possible, which would prove the theorem.

For each $i \in \{1, \ldots, r\}$, let s_i denote the source of path p_i and q_i. Furthermore, for any node v let h_v denote the height of node v in the fixpoint, and for any edge $e = (v, w)$ let $\delta_e = h_v - h_w$. We define the *potential* of a collection C of paths p with demands λ_p as

$$\Phi(C) = \sum_{p \in C} \lambda_p \sum_{(v,w) \in p} (h_v - h_w) \; .$$

Then we obtain for any collection of paths C connecting the same set of endpoints as P and Q that

$$\Phi(C) = \sum_i \lambda_i h_{s_i} \ .$$

Thus, $\Phi(P) = \Phi(Q)$. For any edge e and path collection C, let $\lambda_e(C)$ denote the amount of capacity of e used by the paths in C. Furthermore, for any two path collections C_1 and C_2 let $\lambda_e(C_1 \cap C_2) = \min[\lambda_e(C_1), \lambda_e(C_2)]$ and $\lambda_e(C_1 \setminus C_2) = \max[\lambda_e(C_1) - \lambda_e(C_2), 0]$. Then we obtain from $\Phi(P) = \Phi(Q)$ that

$$\sum_{e \in E} \delta_e \lambda_e(P \cap Q) + \sum_{e \in E} \delta_e \lambda_e(P \setminus Q) = \sum_{e \in E} \delta_e \lambda_e(P \cap Q) + \sum_{e \in E} \delta_e \lambda_e(Q \setminus P)$$

and thus

$$\sum_{e \in E} \delta_e \lambda_e(P \setminus Q) = \sum_{e \in E} \delta_e \lambda_e(Q \setminus P) \ .$$

We know that for all edges e with $\lambda_e(P \setminus Q) > 0$ we have $\delta_e \geq \tau \cdot \Delta$, and for all edges e with $\lambda_e(Q \setminus P) > 0$ we have $\delta_e \leq (\tau + 1)\Delta$. Hence,

$$\tau \cdot \Delta \sum_{e \in E} \lambda_e(P \setminus Q) \leq \sum_{e \in E} \delta_e \lambda_e(P \setminus Q)$$

and

$$\sum_{e \in E} \delta_e \lambda_e(Q \setminus P) \leq (\tau + 1)\Delta \sum_{e \in E} \lambda_e(Q \setminus P)$$

and so

$$\sum_{e \in E} \lambda_e(P \setminus Q) \leq \frac{\tau + 1}{\tau} \sum_{e \in E} \lambda_e(Q \setminus P) \ .$$

On the other hand, (1) implies that

$$\sum_{e \in E} \lambda_e(P \cap Q) + \sum_{e \in E} \lambda_e(P \setminus Q) > \frac{\tau + 1}{\tau} \left(\sum_{e \in E} \lambda_e(P \cap Q) + \sum_{e \in E} \lambda_e(Q \setminus P) \right)$$

and therefore

$$\sum_{e \in E} \lambda_e(P \setminus Q) > \frac{\tau + 1}{\tau} \sum_{e \in E} \lambda_e(Q \setminus P) \ ,$$

which is a contradiction. □

The theorem implies that if every edge has unit cost concerning energy consumption, then the overall energy consumption of the flow solution used by the balancing algorithm is only a $(1 + 1/\tau)$ factor away from a minimum possible overall energy consumption. We believe that Theorem 13.11 can also be shown if the edges have non-uniform costs. However, in this case the thresholds used by the balancing algorithm have to be adapted to the cost of the edges.

References

[1] B. Awerbuch, P. Berenbrink, A. Brinkmann, and C. Scheideler. Simple routing strategies for adversarial systems. In *Proc. of the 42nd IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 158–167, 2001.

[2] B. Awerbuch, A. Brinkmann, and C. Scheideler. Anycasting and multicasting in adversarial systems. Unpublished manuscript, March 2002.