

Effective Scheduling for Coded Distributed Storage in Wireless Sensor Networks

Jan-Willem van Bloem, Richard J. Boucherie, Jasper Goseling, Maurits de Graaf, Geert Heijenk, Jan-Kees van Ommeren and Roel Schiphorst

Faculty of EEMCS, University of Twente, Enschede, The Netherlands
{J.W.H.vanbloem, R.J.Boucherie, J.Goseling, M.deGraaf, Geert.Heijenk, J.C.W.vanOmmeren, R.Schiphorst}@utwente.nl

Abstract

A distributed storage approach is proposed to access data reliably and to cope with node failures in wireless sensor networks. This approach is based on random linear network coding in combination with a scheduling algorithm based on backpressure. Upper bounds are provided on the maximum rate at which data can be reliably stored. Moreover, it is shown that the backpressure algorithm allows to operate the network in a decentralized fashion for any rate below this maximum.

1 Introduction

In wireless sensor networks, reliable data storage is essential for coping with failures. Wireless sensor networks consist of cooperating devices that have sensing capabilities to monitor environmental data. These sensor devices have limited computational power, buffer and storage capabilities. A common application of wireless sensor networks is environmental monitoring in remote and inaccessible areas to detect fires or floods, for example. In such harsh environments, the wireless sensor network is vulnerable to sensor device failures, and unreliable communication links.

In wireless sensor networks, it is not always possible to store the data immediately at a central server system. For this purpose, data must be stored reliably at the sensor devices in network. Since these sensor devices may eventually fail, it is essential that data is stored in a distributed fashion at the sensor devices in the network. Various strategies have been proposed to achieve distributed networked storage. One such scheme, based on uncoded random storage, is widely used in peer-to-peer systems. Another strategy is based on erasure codes, *e.g.*, Reed-Solomon or fountain codes, which are employed in many applications such as redundant arrays of independent disks (RAID) systems. These erasure codes have appealing fast decoding properties but require centralized coordination [3].

Recently, the distributed storage problem for wireless sensor networks was addressed by Dimakis *et al.* [3], in which the goal is to store data in such a way that decoding complexity is minimized when retrieving the data. However, the problem of distributing the data through the network has not been addressed in [3]. This means that aspects such as scheduling packet transmissions and queuing of packets at devices is not addressed. In the present work, the focus is on these scheduling aspects for distributed storage, by scheduling packets in a decentralized way so as to minimize bandwidth and avoid buffer overflows.

More precisely, we address the following problem. Data is generated by a single sensor device and needs to be stored in a distributed fashion in a network of N

sensor devices with storage capabilities. The goal is that, at any point in time, the data can be retrieved by querying any subset of K devices. We refer to the process of retrieving the data as a data collector connecting to these K devices. Hence, if $N - K$ sensor devices fail in the network, then the sensed data is required to be present on the remaining K sensor devices. Obviously, the rate at which the source generates data, determines the bandwidth that is required in the network to achieve distributed storage. Given a network topology and link capacities there is an upper bound to the source rate under which data can be stored in distributed fashion, without causing buffer overflows.

As was done in [2,3] we will consider the use of network coding. We make use of some of the ideas from [3], reducing the problem of distributed storage to a multicast communication problem. Note that, in a multicast problem each of several destinations needs to receive all information from a source. Network coding was introduced in [1], where it was demonstrated that by performing coding at intermediate devices in a network, throughput for multicast communication can be improved compared to traditional routing approaches in which data is only forwarded in the network. In [5], it was shown that random linear network coding is optimal for multicasting in networks modelled by directed graphs. Random operations allow devices to operate in a fully decentralized way. Each device is able to make decisions autonomously for coding and forwarding of data packets, based solely on information obtained locally from neighboring devices.

For multicast communication, it was shown in [6] that a scheduling algorithm exists to achieve the multicast capacity for a particular network. This algorithm is known as the backpressure algorithm, and was presented by Tassiulas *et al.* [9]. The backpressure algorithm makes decisions about resource allocation to forward the data efficiently in the network. The scheduling decisions are made at the beginning of each time slot, to dynamically route the data over the outgoing links. The use of the backpressure algorithm is appealing since it allows to operate in a fully decentralized fashion [4], [8]. Hence, each sensor device only needs local information to make scheduling decisions in order to forward the data over the outgoing links. Generally, the backpressure algorithm works by applying the following two steps at the start of each new time slot t :

1. For each link, determine the difference in queue length.
2. A scheduling decision is made based on the queue length difference. For each link, high link rates are offered to traffic that has a positive valued queue length difference; otherwise no link rate is offered.

By describing the distributed storage problem as a multicast problem, the backpressure algorithm is applicable. By applying the backpressure algorithm to the distributed storage problem, scheduling decisions can be made in a decentralized fashion. However, for the distributed storage model, central coordination is still required to let each node know in which subsets it participates. In addition, our proposed scheme is at the expense of higher decoding complexity at the data collector.

The remainder of the paper is structured as follows. Section 2 contains a description of the model and our assumptions. In Section 3, the optimization problem is addressed and an upperbound is derived for the sensing rates. Furthermore, in Section 4, a discussion follows about the scheduling aspects.

2 Model

We model the wireless network with sensor devices as a directed graph $G(V', E)$ where $V' = V \cup \{s\}$. Here, the set with nodes V represents the sensor devices in the network which have both storage and relay capabilities. The set with edges E corresponds to the links in the network, to which capacity values are assigned. Furthermore $N = |V|$, which makes that the total number of nodes in the network equals $N + 1 = |V \cup \{s\}|$. We assume a single source sensor device that generates packets. Without loss of generality, this can be extended to multiple sources, which is modelled as a single source injecting data at different places in the network [1]. In the graph, node s represents the source sensor device which generates the data.

At the source, the packet generation into the network occurs according to a ergodic process at a time-average rate R . These packets are stored in a redundant way on all subsets of K nodes in the network. We will reduce the distributed storage model to multicast by modeling the different subsets of nodes by virtual nodes. Hence, one virtual node is assigned for each $q \in T$ where the set $T = \{q \subset V; |q| = K\}$ contains all $\binom{N}{K}$ subsets. For a particular subset $q \in T$, $i \in q$ denotes one node. A node must know in advance to which subsets it belongs, which in turn requires centralized coordination. For this purpose, each node must know at least the scalars N and K in addition to its node ID. Note, a virtual node is not physically present in the network but allows the reduction to multicast. Therefore the capacities for links from nodes to the virtual nodes are modelled as infinitely large. In addition, we assume that the data is permanently stored at the nodes, and that the data collector's computational power is sufficient to decode the stored data.

In order to store the data in a distributed way in the network, the data packets are forwarded to the different subsets, in such a way to avoid buffer overflows and to minimize bandwidth. The assumptions related to the scheduling aspects are the following. We assume that time is slotted and that scheduling decisions are made at the beginning of each time slot. An integer number of packets is sent during each timeslot, with packets having fixed lengths. The transmission over a link (a, b) between nodes is point-to-point: a packet is transferred from the queue at node a to the queue at node b . By applying random linear network coding, one physical packet contains information for different virtual nodes. This is because packets are mixed to form one packet [1] where the newly formed packet is a linear combination of earlier received packets, by performing operations over a finite field \mathbb{F}_m , of size m . By choosing the coefficients randomly, it is shown in [5], that the probability of unsuccessful decoding decreases exponentially with the field size. We follow the approach from [6], *i.e.*, in order to keep track of the packets for virtual nodes, a node i maintains virtual queues Q_i^q for all virtual nodes $q \in T$. At node i , the length of a virtual queue is denoted by U_i^q . For subset q , the virtual queue backlog difference for one link (i, j) is denoted by $(U_i^q - U_j^q)$. Note, data packets can either be stored temporary in a buffer or stored permanently.

There is one important difference between [6] and our work. Let $q \in T$ and $i \in q$, if a packet arrives at Q_i^q , it will immediately be moved to the permanent storage at node i . Therefore, the virtual queue Q_i^q , with $i \in q$, will always be empty. This in contrast to the approach in [6], where arriving packets at Q_i^q find a non-empty queue. The reason therefore is the following. In their model, if node $i \neq q$, then packets are forwarded further into the network towards destination node q .

We define stability of the network in terms of buffer overflow functions [6,8]. For

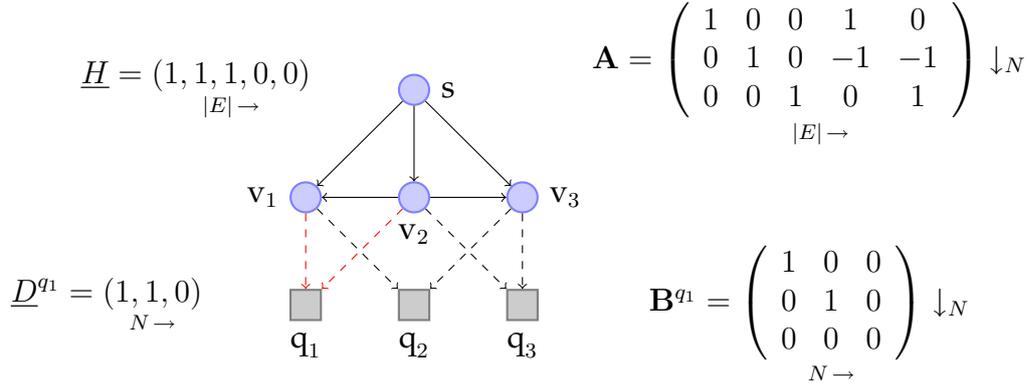


Figure 1: The network with source node s , nodes v_1, v_2, v_3 representing storage devices, and the virtual nodes q_1, q_2, q_3 . Here, $N = 3$ and $K = 2$. The links have unit capacity.

a virtual queue Q_i^q , let

$$\gamma_i^q(M) = \limsup_{t \rightarrow \infty} \frac{1}{t} \sum_{t'=0}^t \Pr\{U_i^q(t') > M\} \quad (1)$$

Furthermore, a network of queues is said to be stable if each individual queue is stable. Note that since the source is generating data at a constant rate, the amount of data stored in permanent storage at devices is increasing linearly over time. The queues Q_i^q , however, keep track only of the data that will be retransmitted in the network. Hence, if link capacities are sufficiently high, scheduling can be performed such that the Q_i^q are stable.

Finally, we introduce the vectors \underline{H} , \underline{D}^q and the matrices \mathbf{A} , \mathbf{B}^q to indicate how nodes are connected in the network (see figure 1). Both vectors have entries that are either 0 or 1. Here, the $1 \times |E|$ vector \underline{H} indicates how the source node is connected to the network. Each entry represents an edge $e \in E$. A 1 entry indicates if a node is connected to edge e . Otherwise a 0 entry is assigned. For the $1 \times N$ vector \underline{D}^q , a 1 entry at node i indicates that node i is assigned to store data for subset q . Furthermore, the infrastructure of the network is defined by matrix \mathbf{A} which is a $N \times |E|$ incidence matrix:

$$\mathbf{A}_{v,e} : \begin{cases} 1 & \text{if } e \text{ enters } v \\ -1 & \text{if } e \text{ leaves } v \\ 0 & \text{else} \end{cases} \quad (2)$$

The $N \times N$ matrix \mathbf{B}^q holds information about the nodes in subset $q \in T$. The diagonal matrix of \mathbf{B}^q corresponds to the \underline{D}^q vector.

Note that both \mathbf{B}^q and \underline{D}^q are specific for one subset $q \in T$, listing for each subset the set with K nodes.

3 Capacity

We characterize the capacity in terms of flow and capacity constraints for network-coded traffic to the different subsets by using the reduction to multicast for dis-

tributed storage. It was shown in [6] that capacity for multicast communication is defined by a set of constraints determined by the link capacities and by the requirement that flow conservation must hold at each node in the network. Moreover, the necessary condition for stability of multicast in a network is given with respect to the capacity as proven in [6]:

Theorem 1 (Ho *et al.* [6]). *A necessary condition for stability for multicasting the data to all storage subsets with intra-session network coding is $R \leq C$. Furthermore, the backpressure control policy stabilizes the network for any of these rates, where*

$$C = \max R \quad \text{subject to:} \quad (3)$$

Flow Constraints:

$$\mathbf{A} \cdot \underline{f}^q = \mathbf{B}^q \cdot \underline{r}^q \quad \forall q \in T \quad (4)$$

$$\underline{D}^q \cdot \underline{r}^q = R \quad \forall q \in T \quad (5)$$

$$R = \underline{H} \cdot \underline{f}^q \quad \forall q \in T \quad (6)$$

Capacity Constraints:

$$\underline{f}^q \leq \underline{c} \quad \forall q \in T \quad (7)$$

$$\underline{f}^q \geq \underline{0}, \underline{r}^q \geq \underline{0}, R \geq 0 \quad \forall q \in T \quad (8)$$

Here, the time-averaged rate over link (i, j) to subset $q \in T$ is denoted as f_{ij}^q where the superscript q denotes the destination virtual node. This corresponds to an entry in the $1 \times |E|$ vector \underline{f}^q for subset $q \in T$. The capacity of a link (i, j) is denoted by c_{ij} , which is written in vector form as the $1 \times |E|$ vector \underline{c} , containing entries for all links in the network. The capacity constraints relate directly to network coding. Instead of the traditional routing constraint defining capacity $\sum_{q \in T} f_{ij}^q \leq c_{ij}$, network coding allows us to state $f_{ij}^q \leq c_{ij}$ as shown in [5].

The conservation laws define the flow constraints. The first flow equation (4) defines virtual flow conservation for all nodes in the network. This states that all traffic that comes in at node i from upstream neighboring nodes $a \in V_i$ also goes out to downstream neighboring nodes $b \in V_i$. Here V_i denotes the set with neighboring nodes of node i . Rewriting the matrix constraints in equation (4) gives the following flow equation at node i :

$$\sum_{a \in V_i} f_{ai}^q - \sum_{b \in V_i} f_{ib}^q = r_i^q \quad \forall i \in V, q \in T \quad (9)$$

where r_i^q denotes the flow at which data is stored at node i . This corresponds to the i^{th} entry in the $1 \times |N|$ vector \underline{r}^q . Subset q data at node $i \in q$ can be conveyed out of the virtual queue U_i^q without a queuing delay, so that data packets can be stored immediately. When a node does not store, the data is relayed further into the network. In that case, the right-hand side of equation (9) is zero.

The second flow constraint in equation (5) represents the total flow for subset $q \in T$ by summing over all outgoing $r_i^q \forall i \in q$. This corresponds to the following equation:

$$\sum_{i \in q} r_i^q = R \quad \forall q \in T \quad (10)$$

The third flow constraint in equation (6) is obtained by assuming conservation of flow at the source.

How the max flow problem in equation (3) relates to the network's bottleneck is formulated in Theorem 2, where the network's bottleneck is defined as the min-cut.

Theorem 2. *Using network coding, the maximum multicast flow from source s to all possible subsets equals the min-cut to the subset $q \in T$ which is the minimum of all subsets's min-cuts.*

Proof. (Outline)

The maximum flow in the network cannot exceed the sum of the capacities in the bottleneck without buffer overflows taking place. To obtain a lower bound for the capacity, the dual problem of the flow maximization problem (equations (3)-(8)) is derived. Here the set of constraints define a convex region denoted by the polytope P_1 . The dual problem involves a minimization to compute the network's min-cut. Using combinatorial arguments, the minimization problem is decoupled into $\binom{N}{K}$ separate min-cut computations [7]:

$$C = \min_q \left[\min_{P_1} \left(\sum_{(i,j) \in E} W_{ij}^q c_{ij} \right) \right] \quad (11)$$

$$P_1 : \begin{cases} U_i^q - v_s^q + W_{si}^q \geq 0 & \forall (s, i) \in E, q \in T \\ U_j^q - U_i^q + W_{ij}^q \geq 0 & \forall (i, j) \in E, q \in T \\ U_i^q = 0 & \forall i \in q, q \in T \\ v_s^q = 1 & \forall q \in T \\ U_i^q \geq 0, W_{ij}^q \geq 0, & \forall (i, j) \in E, i \in V, q \in T \end{cases} \quad (12)$$

Here the Lagrange multipliers U_i^q, v_s^q play the role of unfinished work at the nodes in the network. For subset q data, the number of bits at node i is denoted as U_i^q , and at source node as v_s^q . Note that the amount of unfinished work U_i^q equals zero for all nodes in subset $q \in T$. Furthermore, the Lagrange multipliers W_{ij}^q indicate which links are in the min-cut. Solving the optimization problem stated in equation (11) yields the weight values in the network's min-cut for a particular subset $q \in T$. This min-cut is the minimum of all min-cuts in the network $\forall q \in T$. \square

4 Discussion

For the source sensor device, an upperbound for the maximum rate at which data can be send into the network is derived in section 3. Furthermore, a scheduling algorithm is needed to forward the coded data in a decentralized way to the $\binom{N}{K}$ different storage subsets without causing buffer overflows. For the described distributed storage model, the backpressure algorithm is an optimal scheduling algorithm which allows to operate the network in a decentralized fashion for any rate below this maximum rate. This is based on the following mapping, where the routing problem for distributed storage is reduced to a multicast problem. The mapping to a multicast model is made by storing the data for subset q immediately when it arrives at queue Q_i^q for $i \in q$. As a consequence, the virtual queues $Q_i^q \forall i \in q$ for subset q are always empty. This is in line with the description of the dual problem

provided in section 3. There the dual variables U_i^q , which play the role of the virtual queue backlogs, are set to $U_i^q = 0 \forall i \in q, q \in T$. Therefore, the distributed storage problem can be reduced to a multicast model where the data, generated by source s , is sent to storage subsets $\forall q \in T$. Note, that for multicast communication, capacity can be achieved using the backpressure algorithm. Hence, the backpressure algorithm is also applicable for forwarding data in an optimal way to the different storage subsets. Furthermore, the backpressure algorithm operates in a decentralized fashion and only requires queue backlog information from neighboring nodes. In addition, to operate in a fully decentralized way, the coding operation at the devices in the network must take place in a decentralized way. This is possible by using random linear network coding. Hence, combining backpressure scheduling and random linear network coding provides a way to forward packets without central coordination. However, a drawback of our approach is that central coordination is still required to let each node know in which subsets it participates. Moreover, the number of virtual queues scales exponential with the size of the network.

5 Conclusion

The main problem addressed in this paper is how to best schedule data for distributed storage using network coding. To cope with node failures in wireless sensor networks, the data is stored on all subsets of K storage devices in a network of N storage devices. Our contribution includes providing an upperbound on the maximum source rate at which data can be stored in a distributed way in a network of storage devices. By reducing the distributed storage problem to a multicast problem, it is shown that the backpressure algorithm can forward the data without causing buffer overflows, and in such a way that all the data can be stored on each subset of K devices in a network of N devices.

References

- [1] Rudolf Ahlswede, Ning Cai, Shuo-Yen Robert Li, and Raymond W. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, 2000.
- [2] A.G. Dimakis, P.B. Godfrey, M.J. Wainwright, and K. Ramchandran. Network coding for distributed storage systems. *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 2000–2008, may 2007.
- [3] Alexandros G. Dimakis, Vinod M. Prabhakaran, and Kannan Ramchandran. Decentralized erasure codes for distributed networked storage. *IEEE Transactions on Information Theory*, 52(6):2809–2816, 2006.
- [4] Leonidas Georgiadis, Michael J. Neely, and Leandros Tassiulas. Resource allocation and cross-layer control in wireless networks. *Foundations and Trends in Networking*, 1(1):1–144, 2006.
- [5] Tracey Ho, Muriel Médard, Ralf Koetter, David R. Karger, Michelle Effros, Jun Shi, and Ben Leong. A random linear network coding approach to multicast. *IEEE Transactions on Information Theory*, 52(10):4413–4430, 2006.

- [6] Tracey Ho and Harish Viswanathan. Dynamic algorithms for multicast with intra-session network coding. *IEEE Transactions on Information Theory*, 55(2):797–815, 2009.
- [7] Zongpeng Li and Baochun Li. Efficient and distributed computation of maximum multicast rates. *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, 3:1618 – 1628 vol. 3, march 2005.
- [8] Michael J. Neely, Eytan Modiano, and Charles E. Rohrs. Dynamic power allocation and routing for time-varying wireless networks. *IEEE Journal on Selected Areas in Communications*, 23(1):89–103, 2005.
- [9] Leandros Tassiulas and Anthony Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Transactions on Automatic Control*, 37(12):1936–1948, dec 1992.