Chapter 12

Network Coding

In Chapter 11, we have studied the token dissemination (or multi-message broadcast) problem in dynamic networks. There are k tokens (messages) which have to be sent to all the nodes of a network. So far, we considered so-called store-and-forward (or token-forwarding) algorithms in which at all times, each node can forward one or a subset of the tokens it possesses. A node can only learn a new token if the token is forwarded by one of its neighbors. It is for example not possible to send only parts of different tokens or to send combinations of tokens. Sending combinations of the pieces information that have to be disseminated is known as *network coding*. In some cases, being able to send combinations of the tokens can lead to a significant improvement. Figure 12.1 shows a simple standard example, where network coding helps to send strictly more information through a network than what can be achieved using store-and-forward approaches.

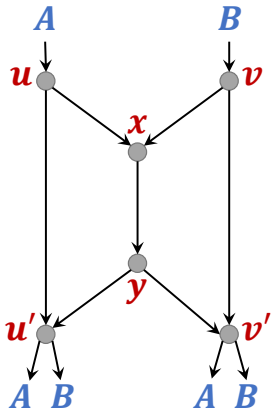

Figure 12.1: A simple directed network where network coding achieves a strictly better throughput than any store-and-forward algorithm.

Lemma 12.1. *Assume that in the (directed) network of Figure 12.1, nodes u and v both want to send a stream of tokens to both nodes u' and v'. If each message can only fit the size of a single token, the possible throughput (total*

number of tokens per time unit) of store-and-forward algorithms is at most 1*, whereas with network coding, one can achieve a throughput of* 2*.*

Proof. Assume that there is a stream A of tokens arriving at node u and a stream of tokens B arriving at node v. Further assume that nodes u' and v' need to receive both streams A and B. Let us first consider what can be achieved by using store-and-forward protocols. Note that in order to send the tokens of stream A from node u to nodes v', all the stream A tokens need to pass through the edge (x, y). Similarly, also all the stream B tokesn need to pass through the edge (x, y) in order to arrive at node u'. If we assume that each edge can be used for only one token per time unit, the streams A and B can only produce a token every second time step in order to not overload the network. Together, the two streams can therefore inject one token per time unit into the network and we can achieve a throughput of at most 1.

If we are allowed to use network coding, there is a simple protocol that allows to double the throughput of the network. Whenever two new token τ_A and τ_B arrive at nodes u and v, u sends its token τ_A to u' and to x and v sends its token τ_B to v' and to x. The node x now sends the xor $\tau_A \oplus \tau_B$ of the two tokens to node y and y forwards $\tau_A \oplus \tau_B$ to u' and v'. Node u' can now reconstruct the content of token τ_B as $\tau_B = (\tau_A \oplus \tau_B) \oplus \tau_A$ and node v' can similarly reconstruct the content of token τ_A. In this way, both streams can inject a new token per time unit without overloading the system and we therefore achieve an overall throughput of 2. □

In the following, we generalize the above idea to general networks. We study so-called *random linear network coding* to solve the token dissemination problem in general, possibly dynamic networks. The basic idea of random linear network coding is very simple: In each round, each node combines (“**xor**s”) a random subset of the messages received so far. As soon as enough different combinations of the tokens have beein received, a node can compute all of them. In order to analyze this algorithm, we need to phrase it more formally and we need to use some basic tools from linear algebra.

We assume that there are k tokens which need to be broadcast to all nodes in the network. For simplicity, we assume that the tokens are numbered from 1 to k and that the nodes which initially possess some tokens also know the numbers of these tokens.[1] We assume that the tokens are bit strings of length b. We can therefore interpret the token as b-dimensional vectors over $\mathbb{Z}_2$. We denote the tokens by $\underline{\tau}_1, \ldots, \underline{\tau}_k \in \mathbb{Z}_2^b$. Each message M sent by a token dissemination contains a linear combination of the vectors representing the tokens, i.e.,

$$M = \sum_{i=1}^{k} \mu_i \cdot \underline{\tau}_i, \tag{12.1}$$

where $\mu_i \in \mathbb{Z}_2$ are multipliers in $\mathbb{Z}_2$ (i.e., in $\{0, 1\}$). Note that in the above equation, all arithmetic operations are in $\mathbb{Z}_2$ and therefore componentwise modulo 2. Note that when considering tokens as bit vectors, such a linear combination corresponds to the bit-wise **xor** of the subset of tokens τ_i for which $\mu_i = 1$. If a node u sends a message M to a neighbor v, it does not suffice if u just

[1] Note that this is a strong assumption. In many cases, there are techniques to drop the assumption, however they go beyond what we can do in this chapter.

sends M as given in (12.1) to v (M is just the bit-wise **xor** of a subset of the tokens). In order to be able to use M, v also needs to know the multipliers μ_i, i.e., v needs to know how the bit-wise **xor** M was built. With every message M, it is therefore necessary to also send a k-bit header containing the vector $\underline{\mu} = (\mu_1, \ldots, \mu_k) \in \mathbb{Z}_2^k$. As we will see later, network coding is therefore most efficient if the possible message (token) size is relatively large.

As a result of (12.1), each message M can be uniquely represented by a coefficient vector $\underline{\mu} = (\mu_1, \ldots, \mu_k) \in \mathbb{Z}_2^k$. The space of all possible messages can then be identified with $\mathbb{Z}_2^k$. Note that since $\mathbb{Z}_2$ (using ordinary addition and multiplication modulo 2) is a field, $\mathbb{Z}_2^k$ is a k-dimensional vector space and we can use the techniques from linear algebra to study linear combinations of vectors in $\mathbb{Z}_2^k$. At all times, each node v can create a subset of the possible messages. If v initially possesses token $\underline{\tau}_i$, v can initially create the message corresponding to unit vector $\underline{e}_i = (e_{i,1}, \ldots, e_{i,k})$ with $e_{i,i} = 1$ and $e_{i,j} = 0$ for $j \neq i$. Assume that at a given time, a node v has received the messages $\mathcal{M}_v$ characterized by the coefficient vectors $\underline{\mu}_{v,1}, \ldots, \underline{\mu}_{v,p}$. For simplicity, we assume that the vectors $\underline{e}_i$ for tokens initially possessed by v are also part of the set of received vectors $\mathcal{M}_v$ of v. If v knows all vectors in $\mathcal{M}_v$, it can send any possible linear combination of the vectors in $\mathcal{M}_v$ as a message. Because the set of all coefficient vectors $\mathbb{Z}_2^k$ is a vector space, the set of all possible linear combinations of vectors in $\mathcal{M}_v$ is a subspace of $\mathbb{Z}_2^k$. In the following, we use S_v to denote this subspace. We will use the following basic properties from the theory of vector spaces about S_v:

(I) The dimension $\dim(S_v)$ of S_v is equal to the size of any maximal set of linearly independent vectors of $\mathcal{M}_v$. It always holds that $\dim(S_v) \in \{0, \ldots, k\}$.

(II) Any set of $\dim(S_v)$ linearly independent vectors of S_v is called a basis of S_v. In particular, any maximal set of independent vectors of $\mathcal{M}_v$ is a basis of S_v.

(III) Node v can compute all the tokens if and only if $\dim(S_v) = k$.

(IV) We define $S_v^\perp$ to be the set of vectors which are orthogonal to all vectors in S_v, i.e., $S_v^\perp$ contains the set of all vectors where the dot product (in $\mathbb{Z}_2$) with all vectors in $\mathcal{M}_v$ evaluates to 0. $S_v^\perp$ is called the orthogonal (dual) complement of S_v.

(V) $S_v^\perp$ is a vector space of dimension $\dim(S_v^\perp) = k - \dim(S_v)$. Hence, by Property (III), v can compute the value of all tokens if and only if $\dim(S_v^\perp) = 0$.

The last property shows that as the dimension of S_v grows (when v learns new information), the dimension of the orthogonal complement $S_v^\perp$ gets smaller. The subspace S_v known by v becomes everything ($\dim(S_v) = k$) when the dimension of $S_v^\perp$ becomes 0 (and thus $S_v^\perp$ becomes empty). It turns out that it is much easier to argue about how $S_v^\perp$ becomes smaller than arguing about how S_v grows. The key tool to analyze random linear network coding is given by the following definition.

Definition 12.2. *We say that a node v* ***knows about*** *a coefficient vector $\underline{\mu} \in \mathbb{Z}_2^k$ if and only if $\underline{\mu} \notin S_v^\perp$.*

Remarks:

- In other words, a node v knows about $\underline{\mu}$ iff there is a vector $\underline{\mu}' \in \mathcal{M}_v$ such that $\underline{\mu}$ is not orthogonal to $\underline{\mu}'$ ($\sum_{i=1}^{k} \mu_i \cdot \mu_i' \not\equiv 0 \pmod 2$).
- Note that the fact that v knows about $\underline{\mu}$ does not mean that v can construct the message corresponding to $\underline{\mu}$ (i.e., it does not mean that $\underline{\mu} \in S_v$). In fact, it is not even true that $\underline{\mu} \in S_v$ implies that v knows about $\underline{\mu}$. In the vector space $\mathbb{Z}_2^k$, it is for example possible that a vector $\underline{\mu}$ is orthogonal to itself. Hence, if S_v only consists of such a vector, S_v is a subspace of $S_v^\perp$.

The next lemma shows why the above definition is useful.

Lemma 12.3. *If a node v knows about all vectors $\underline{\mu} \in \mathbb{Z}_2^k$, node v has enough information to compute all the k tokens.*

Proof. If for all $\underline{\mu} \in \mathbb{Z}_2^k$, v knows about $\underline{\mu}$, it follows by Definition 12.2 that $S_v^\perp$ is empty. Therefore, the dimension of $S_v^\perp$ is 0. Using Property (V), from about, we can then conclude that $\dim(S_v) = k - \dim(S_v^\perp) = k$ and using Property (III), this implies that v can decode all k tokens. □

The next lemma shows that if node u knows about a vector $\underline{\mu}$ and a neighbor v does not know about $\underline{\mu}$ node v has a good chance (probability at least 1/2) to know about $\underline{\mu}$ after receiving a single message from u.

Lemma 12.4. *Let $\underline{\mu} \in \mathbb{Z}_2^k$ be a coefficient vector and assume that a node u knows about $\underline{\mu}$. A random linear combination of all the vectors in $\mathcal{M}_u$ (the initial tokens of u and all the message received by u so far), is not orthogonal to $\underline{\mu}$ with probability at least* 1/2.

Proof. Since we assume that u knows about $\underline{\mu}$, there is a vector $\underline{\nu} \in \mathcal{M}_v$ which is not orthogonal to $\underline{\mu}$. A random linear combination $\underline{\rho}$ of the vectors in $\mathcal{M}_v$ can be written as

$$\underline{\rho} = \sum_{\underline{\eta} \in \mathcal{M}_v} \beta_{\underline{\eta}} \cdot \underline{\eta} = \beta_{\underline{\nu}} \cdot \underline{\nu} + \underbrace{\sum_{\underline{\eta} \in \mathcal{M}_v \setminus \{\underline{\nu}\}} \beta_{\underline{\eta}} \cdot \underline{\eta}}_{\underline{\rho}'}.$$

In the above expression, the values of $\beta_{\underline{\eta}}$ are independent random bits such that $\Pr(\beta_{\underline{\eta}} = 1) = 1/2$. All operations are component-wise modulo 2 (i.e., in $\mathbb{Z}_2^k$). Consider the right-hand side of the above expression. First assume that the bits $\beta_{\underline{\eta}}$ for $\underline{\eta} \in \mathcal{M}_v \setminus \{\underline{\nu}\}$ are chosen such that the vector $\underline{\rho}'$ is not orthogonal to $\underline{\mu}$ In that case, choosing $\beta_{\underline{\nu}} = 0$ results in $\underline{\rho} = \underline{\rho}'$ and thus in a vector $\underline{\rho}$ which is not orthogonal to $\underline{\mu}$. If the vector $\underline{\rho}'$ is orthogonal to $\underline{\mu}$, $\underline{\rho}$ is not orthogonal to $\underline{\mu}$ if and only if $\beta_{\underline{\nu}} \cdot \underline{\nu}$ is not orthogonal to $\underline{\mu}$ Since we assumed that $\underline{\nu}$ is not orthogonal to $\underline{\mu}$, this happens if the random bit $\beta_{\underline{\nu}}$ is chosen to be 1. In both cases, the probability that $\underline{\rho}$ is not orthogonal to $\underline{\mu}$ is therefore at least 1/2. □

Remarks:

- The above lemma can also be shown using basic properties of the vector space $\mathbb{Z}_2^k$. First note, that a random linear combination of the vectors in $\mathcal{M}_v$ yields a uniformly random vector from the space S_v (proving this is a nice exercise). The set of all vectors in S_v which are orthogonal to $\underline{\mu}$ form a subspace of S_v and because not all vectors in S_v are orthogonal to $\underline{\mu}$, the dimension of this subspace is strictly smaller than $\dim(S_v)$. Since the number of vectors in a d-dimensional subspace of $\mathbb{Z}_2^k$ is exactly 2^d, the number of vectors in S_v orthogonal to $\underline{\mu}$ is at most half the number of vectors in S_v and a uniformly random vector from S_v is therefore not orthogonal to $\underline{\mu}$ with probability at least $1/2$.

The following proof requires the following Chernoff bound on the sum of independent 0/1 random variables. A proof of the bound can for example be found in [].

Theorem 12.5. *Let $X_1, \dots, X_n$ be independent Bernoulli random variables with $\Pr(X_i = 1) = p_i$ and $\Pr(X_i = 0) = 1 - p_i$. We further define $X := \sum_{i=1}^n X_i$ and $\mu := \mathbb{E}[X] = \sum_{i=1}^n p_i$. For every $\varepsilon > 0$, it holds that*

$$\Pr(X \leq (1-\varepsilon)\mu) \leq e^{-\varepsilon^2 \mu/2}.$$

Theorem 12.6. *Assume that we have a synchronous dynamic network which is connected in every round (as introduced in Chapter 11). Further, assume that each message fits a k-bit header and a linear combination of tokens. Then, using, random linear network coding, the k-token dissemination problem can be solved in time $O(n + k)$ with high probability.*

Proof. Consider any fixed vector $\underline{\mu} \in \mathbb{Z}_2^k$. As long as $\underline{\mu}$ is not the 0-vector, initially at least one node knows about $\underline{\mu}$ (actually any node possessing one of the tokens corresponding to any non-0 component of $\underline{\mu}$. Now, consider some round r and let $K(r, \underline{\mu})$ be the nodes which know about $\underline{\mu}$ at the beginning of round r. If $K(r, \underline{\mu}) \neq V$, the connectivity requirement guarantees that there is an edge $\{u, v\}$ between some node $u \in K(r, \underline{\mu})$ and $v \notin K(r, \underline{\mu})$. By Lemma 12.4, after round r, v knows about $\underline{\mu}$ with probability at least $1/2$. Hence, as long as not all nodes know about $\underline{\mu}$, in every round, the number of nodes which know about $\underline{\mu}$ grows with probability at least $1/2$.

Assume that we run the algorithm for T rounds. For round r, we define X_r to be a random variable, where $X_r = 1$ if a) in round r, the number of nodes which know about $\underline{\mu}$ grows or b) at the beginning of round r all nodes in V already know about $\underline{\mu}$. Independently of what happens in rounds $1, \dots, r-1$, we then have $\Pr(X_r = 1) \geq 1/2$. Further, all nodes know about $\underline{\mu}$ at the latest when there are $n - 1$ rounds r in which $X_r = 1$. Consider the random variable $X := \sum_{r=1}^T X_r$. By the above observation, X dominates a binomial random variable Y with parameters T and $1/2$, i.e., for all $x \geq 0$, $\Pr(X \geq x) \geq \Pr(Y \geq x)$. Assume that $T \geq 4n$. Using the Chernoff bound of Theorem 12.5, we then

have

$$\begin{aligned}
\Pr(X < n) &\leq \Pr(Y < n) \\
&\overset{(T \geq 4n)}{\leq} \Pr(Y < 1/2 \cdot T/2) \\
&= \Pr(Y < 1/2 \cdot \mathbb{E}[Y]) \\
&\overset{\text{(Thm. 12.5)}}{<} e^{-\mathbb{E}[Y]/8} = e^{-T/16}.
\end{aligned}$$

If we choose $T \geq 16(n+k)$, this probability becomes e^{-n-k}. As there are only $2^k \leq e^k$ different vectors $\underline{\mu}$, a union bound then implies that after $16(n+k)$ rounds, all nodes know about all vectors with probability at least $1-e^{-n}$. Using Property (V), this implies that all nodes can compute all the tokens. □

Remarks:

- In Chapter 11, we have seen that if $O(1)$ tokens can be packed in each message, the k-token dissemination problem can be solved in $O(nk)$ rounds using a simple store-and-forward (token-forwarding) algorithm. Further, for deterministic algorithms, we have seen that $\Omega(nk/\log(nk))$ is a lower bound on the time complexity of algorithms.

- To compare the behavior of random linear network coding with store-and-forward algorithms, let us assume that tokens are b bits long and that messages can fit $O(b)$ bits. If we assume that $b = O(\log n)$, we can only have a header size of $O(\log n)$ and therefore the above network coding algorithm only works if $k = O(\log n)$. If k is larger, we can partition the tokens into groups of size $O(\log n)$ which are then broadcast one after the other. The total running time in this case becomes $O(nk/\log n)$, which is by log-factor faster than the simple store-and-forward algorithm.

- Network coding is most efficient if the tokens are large. Assume that $k = O(b)$. In that case, we can have a k-bit header and therefore network coding solves the k-token dissemination problem in time $O(n+k)$. In general, the time complexity of the above algorithm when having $O(b)$ bit messages is $O(n + k + (nk \log n)/b^2)$.

- Instead of interpreting tokens as bit vectors in $\mathbb{Z}_2^b$, one can consider vectors over some other finite field $\mathbb{F}_q$. This improves the probability that a message contains useful information from $1/2$ to $1 - 1/q$. However, it also requires the coefficients to be from $\mathbb{F}_q$ and therefore increases the header size by a factor $\Theta(\log q)$.

Chapter Notes

The analysis of random linear network coding that we have discussed in this chapter is by Bernhard Haeupler [Hae11].

Bibliography

[Hae11] B. Haeupler. Analyzing network coding gossip made easy. In *Proc. 43rd ACM Symp. on Theory of Computing (STOC)*, pages 293–302, 2011.