17 Network coding in bi-directed and peer-to-peer networks

Zongpeng Li[†], Hong Xu[‡], and Baochun Li[‡]
[†]University of Calgary, [‡]University of Toronto

Network coding has been shown to help achieve optimal throughput in directed networks with known link capacities. However, as real-world networks in the Internet are bi-directed in nature, it is important to investigate theoretical and practical advantages of network coding in more realistic bi-directed and peer-to-peer (P2P) network settings. In this chapter, we begin with a discussion of the fundamental limitations of network coding in improving routing throughput and cost in the classic undirected network model. A finite bound of 2 is proved for a single communication session. We then extend the discussions to bi-directed Internet-like networks and to the case of multiple communication sessions. Finally, we investigate advantages of network coding in a practical peer-to-peer network setting, and present both theoretical and experimental results on the use of network coding in P2P content distribution and media streaming.

17.1 Network coding background

Network coding is a fairly recent paradigm of research in information theory and data networking. It allows essentially every node in a network to perform information coding, besides normal forwarding and replication operations. Information flows can therefore be "mixed" during the course of routing. In contrast to source coding, the encoding and decoding operations are not restricted to the terminal nodes (sources and destinations) only, and may happen at all nodes across the network. In contrast to channel coding, network coding works beyond a single communication channel, it contains an integrated coding scheme that dictates the transmission at every link towards a common network-wise goal. The power of network coding can be appreciated with two classic examples in the literature, one for the wireline setting and one for the wireless setting, as shown in Figure 17.1.

Next-Generation Internet Architectures and Protocols, ed. Byrav Ramamurthy, George Rouskas, and Krishna M. Sivalingam. Published by Cambridge University Press.

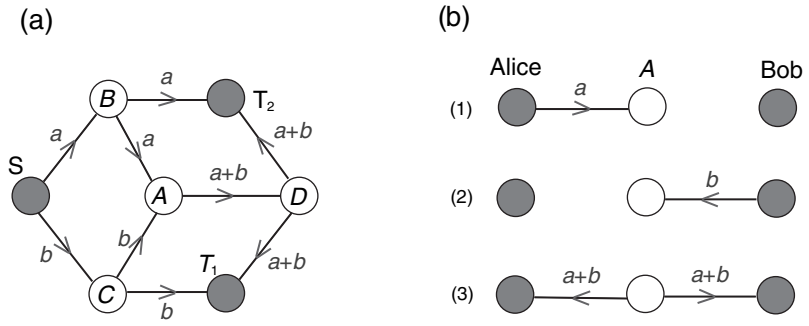

Figure 17.1 The power of network coding. (a) A wireline network where network coding help achieve a multicast throughput of 2 bps. (b) A wireless network where network coding help improve the data exchange rate between Alice and Bob.

Figure 17.1(a) shows a wireline network with a single multicast transmission, from a sender S to two receivers T_1 and T_2 simultaneously. These three terminal nodes are in black; the four white nodes are relay nodes. Each of the nine links in the network has the same unit capacity of 1 bps. With the assumption that link delays can be ignored, a multicast transmission scheme with network coding is depicted. Here a and b are two information flows each of rate 1 bps. Replication happens at nodes B, C and D. Encoding happens at relay node A, which takes a bit-wise exclusive-or upon the two incoming flows a and b, and generates $a + b$. The multicast receiver T_1 receives two information flows b and $a + b$, and can recover a as $a = b + (a + b)$. Similarly, T_2 receives a and $a + b$ and can recover b. Both receivers are therefore receiving information at 2 bps, leading to a multicast throughput of 2 bps. The reader is invited to verify that, without network coding, the throughput 2 bps can not be achieved.

Figure 17.1(b) shows a wireless network, in which Alice and Bob each operates a laptop computer and communicate to each other through the help of a relay A (a third laptop computer or a base station). Each of the three nodes is equipped with an omnidirectional antenna; Alice and Bob are too far away from each other for direct communication, but can both reach the relay A in the middle. Assume Alice and Bob wish to exchange a pair of files. In the depicted transmission scheme, the exchange of a pair of packets (a from Alice and b from Bob) is achieved within three rounds, without interference between concurrent transmissions. It is an easy exercise to verify that without coding at the relay node A, four rounds would have been necessary.

In both examples above, the coding operation is *bit-wise exclusive-or*, which can be viewed as coding over the finite field $GF(2)$. A larger field $GF(2^k)$ can be used in general network coding, with typical values for k being 8 or 16. Since the seminal work of Ahlswede *et al.* [2] published in 2000, the benefit of network coding has been identified in a rather diverse set of applications, including for example: improving network capacity and transmission rates, efficient multicast algorithm design, robust network transmissions, network security, and P2P file

dissemination and media streaming. Our focus in this chapter is on the potential for network coding to improve transmission throughput in various network models.

17.2 Network coding in bi-directed networks

Early research on network coding usually focuses on directed network models, where each link in the network has a prefixed direction of transmission. A fundamental result for network coding in directed networks generalizes the celebrated max-flow min-cut theorem from one-to-one unicast flows to one-to-many multicast flows:

Theorem 17.1. [2] *For a given multicast session in a directed network with network coding support, if a unicast rate x is feasible from the sender to each receiver independently, then it is feasible as a multicast rate to all the receivers simultaneously.*

This result changed the underlying structure of multicast algorithm design, from a tree packing perspective (without coding) to a network flow perspective (with coding), and consequently reduced the computational complexity of optimal multicast from NP-hard to polynomial-time solvable. Both changes apply in undirected as well as in directed networks. It has been shown that the *coding advantage*, the ratio of achievable throughput with coding versus without coding, can be arbitrarily high in directed networks. In this chapter, we reveal a different picture in undirected networks and bi-directed networks , which are closer to the reality of Internet topologies.

17.2.1 Single multicast in undirected networks

A single communication session can be in the form of a one-to-one unicast, one-to-many multicast or one-to-all broadcast. Among these, multicast is the most general. Unicast and broadcast can be viewed as special cases of multicast, where the number of receivers equals one and the network size, respectively. Hence, for the case of a single communication session, we focus on multicast. We use a simple graph $G = (V, E)$ to represent the topology of a network, and use a function $C : E \to Z^+$ to denote link capacities. The multicast group is $M = \{S, T_1, \ldots, T_k\} \subseteq V$, with S being the multicast sender. In our graphical illustrations, terminal nodes in the multicast group are black, and relay nodes are white.

We use $\chi(N)$ to denote the maximum throughput of a multicast network N. A linear programming formulation for $\chi(N)$, based on Theorem 1.1, is given below [14]. Here the objective function is χ, the multicast throughput; $N(u)$ is the set of neighbor nodes of u, f_i is a network flow from the multicast sender S to receiver T_i, c is a variable vector storing link capacities for an orientation

of the undirected network, and $\vec{T_iS}$ is a conceptual feedback link introduced for compact LP formulation.

$\chi(N) :=$ Maximize χ
Subject to:

$$\begin{cases} \chi \le f_i(\vec{T_iS}) & \forall i \\ f_i(\vec{uv}) \le c(\vec{uv}) & \forall i, \forall\, \vec{uv} \neq \vec{T_iS} \\ \sum_{v \in N(u)} (f_i(\vec{uv}) - f_i(\vec{vu})) = 0 & \forall i, \forall u \\ c(\vec{uv}) + c(\vec{vu}) \le C(uv) & \forall uv \neq T_iS \\ c(\vec{uv}), f_i(\vec{uv}), \chi \ge 0 & \forall i, \forall\, \vec{uv} \end{cases}$$

We compare $\chi(N)$ with two other parameters defined for a multicast network, the *packing number* and *edge connectivity*, and derive a bound on the coding advantage from the comparison results.

Packing refers to the procedure of finding pairwise edge-disjoint sub-trees of G, in each of which the multicast group remains connected. The packing number of a multicast network N is denoted as $\pi(N)$, and is equal to the maximum throughput without coding. The reason is that, each tree can be used to transmit one unit information flow from the sender to all receivers, therefore the packing number gives the maximum number of unit information flows that can be transmitted. When relaxing flow rates on trees to be fractional, the packing number can be defined using the following linear program. Here $\mathcal{T}$ is the set of all multicast tree, $f(t)$ is a variable representing the amount of information flow one ships along tree t.

$\pi(N) :=$ Maximize $\sum_{t \in \mathcal{T}} f(t)$
Subject to:

$$\begin{cases} \sum_{uv \in t} f(t) \le C(uv) & \forall uv \in E \\ f(t) \ge 0 & \forall t \in \mathcal{T} \end{cases}$$

Connectivity refers to the minimum edge connectivity between a pair of nodes in the multicast group, and is denoted as $\lambda(N)$. It is also the minimum size of a cut that separates the communication group. Figure 17.2 illustrates the concept of these parameters using an example network. We next prove a main theorem that categorizes the relation among them.

Theorem 17.2. *For a multicast transmission in an undirected network,* $N{=}\{G(V,E), C : E{\to}Z^+, M = \{S, T_1, \ldots, T_k\} \subseteq V\}$,

$$\frac{1}{2}\lambda(N) \le \pi(N) \le \chi(N) \le \lambda(N).$$

Proof: First, furnishing nodes with extra coding capabilities does not decrease the achievable throughput, hence $\pi(N) \le \chi(N)$. Furthermore, for a certain multicast throughput to be feasible, the edge connectivity from the sender to any receiver (unicast throughput) has to achieve at least the same value, therefore $\chi(N) \le$

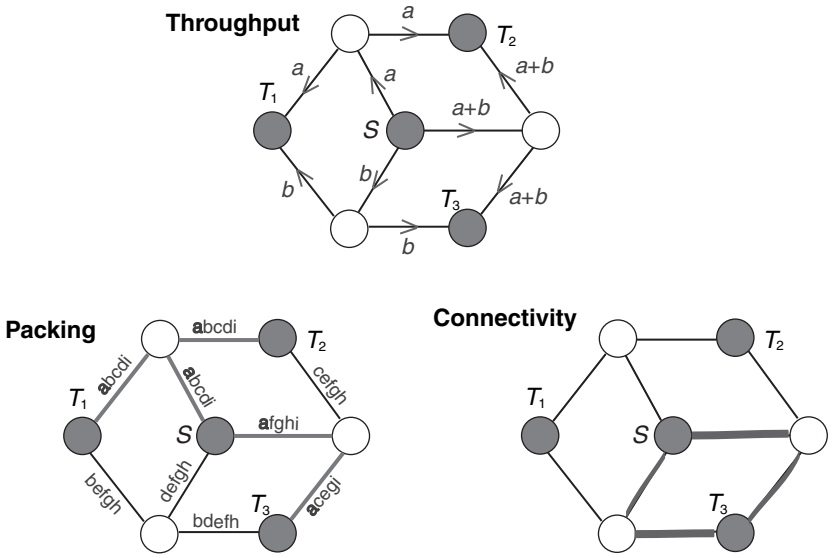

Figure 17.2 The three network parameters. In this particular network with unit capacity at each undirected link: $\pi(N) = 1.8$, nine trees (each labeled with a letter between 'a' and 'i') each of rate 0.2 can be packed; $\chi(N) = 2$, two unit information flows a and b can be delivered to all receivers simultaneously; $\lambda(N) = 2$, each pair of terminal nodes is 2-edge-connected.

$\lambda(N)$. We now have $\pi(N) \leq \chi(N) \leq \lambda(N)$, and will focus on the validity of $\frac{1}{2}\lambda(N) \leq \pi(N)$ in the rest of the proof. We first transform the multicast network into a broadcast one without hurting the validity of $\frac{1}{2}\lambda(N) \leq \pi(N)$, and then prove $\frac{1}{2}\lambda(N) \leq \pi(N)$ is true in the resulting broadcast network.

The transformation relies on Mader's Undirected Splitting Theorem [3]: *Let $G(V + z, E)$ be an undirected graph so that (V, E) is connected and the degree $d(z)$ is even. Then there exists a complete splitting at z preserving the edge-connectivity between all pairs of nodes in V.*

A split-off operation at node z refers to the replacement of a 2-hop path u-z-v by a direct edge between u and v, as illustrated in Figure 17.3. A complete splitting at z is the procedure of repeatedly applying split-off operations at z until z is isolated.

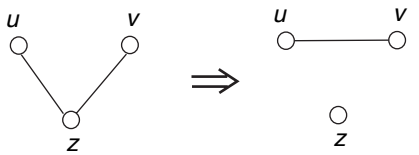

Figure 17.3 A split-off at node z.

The Undirected Splitting Theorem states that, if a graph has an even-degree non-cut node, then there exists a split-off operation at that node, after which

the pairwise connectivities among the other nodes remain unchanged; and by repeatedly applying such split-off operations at this node, one can eventually isolate it from the rest of the graph, without affecting the edge-connectivity of any node pairs in the rest of the graph.

Now, consider repeatedly applying one of the following two operations on a multicast network: (1) apply a complete splitting at a non-cut relay node, preserving pairwise edge connectivities among terminal nodes in M; or (2) add a relay node that is an M-cut node into the multicast group M, i.e., change its role from a relay node to a receiver. Here an M-cut node is one whose removal separates the multicast group into more than one disconnected components. Figure 17.4 illustrates these two operations with a concrete example.

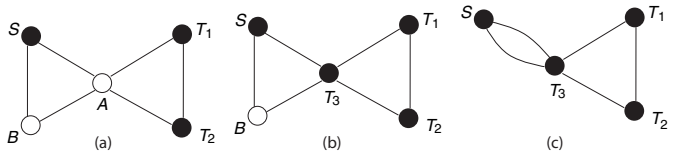

Figure 17.4 Transforming a multicast network into a broadcast network, where the validity of $\frac{1}{2}\lambda(N) \leq \pi(N)$ can be traced back. (a) The original multicast network, with unit capacity on each link. (b) The network after applying operation (2), moving the M-cut node A into the multicast group. Node A becomes receiver T_3. (c) The network after applying operation (1). A split-off was done at relay node B. A broadcast network is obtained.

In order to meet the even node degree requirement in the Undirected Splitting Theorem, we first double each link capacity in the input network, then scale the solution down by a factor of 1/2 at the end. Each node has an even degree after doubling link capacities, and a split-off operation does not affect the parity of any node degree in the network. Therefore the Undirected Splitting Theorem guarantees that as long as there are relay nodes that are not cut nodes, operation (1) is possible. Furthermore, operation (1) does not increase $\pi(N)$. Therefore, if $\frac{1}{2}\lambda(N) \leq \pi(N)$ holds after applying operation (1), it holds before applying operation (1) as well. Operation (2), applied to M-cut nodes, does not affect either $\pi(N)$ or $\lambda(N)$. So, again we can claim that for operation (2), if $\frac{1}{2}\lambda(N) \leq \pi(N)$ holds after applying the operation, it holds before applying the operation as well.

As long as there are relay nodes in the multicast network, at least one of the two operations can be applied. If both operations are possible, operation (1) takes priority. Since each operation reduces the number of relay nodes by one, eventually we obtain a broadcast network with terminal nodes only.

The fact that $\frac{1}{2}\lambda(N) \leq \pi(N)$ holds in a broadcast network can be proved by applying Nash-Willams' Weak Graph Orientation Theorem [3]: *a graph G has an α-edge connected orientation if and only if it is 2α-edge connected*. By the Weak Graph Orientation Theorem, we can orient an undirected multicast with connectivity $\lambda(N)$ into a directed one, so that the network flow rate from any node to

any other node (including in particular from the multicast sender to any multicast receiver) is at least $\frac{1}{2}\lambda(N)$. Then by Theorem 1.1, a multicast rate of $\frac{1}{2}\lambda(N)$ is feasible with network coding. Therefore we obtain $\chi(N) \geq \frac{1}{2}\lambda(N)$. Furthermore, Tutte–Nash-Williams' Theorem [20] on spanning tree packing implies that $\pi(N) = \chi(N)$ in any broadcast network, and hence we also have $\pi(N) \geq \frac{1}{2}\lambda(N)$.

Finally, note that we obtained an integral transmission strategy after doubling each link capacity. Therefore, after we scale the solution back by a factor of 1/2, the transmission strategy is half-integral. □

Corollary 17.1. *For a multicast transmission in an undirected network, the coding advantage is upper-bounded by a constant factor of 2, with either fractional routing or half-integer routing.*

Proof: By Theorem 3, $\frac{1}{2}\lambda(N) \leq \pi(N)$ and $\chi(N) \leq \lambda(N)$ as long as half integer routing is allowed. Therefore we conclude $\frac{1}{2}\chi(N) \leq \pi(N)$, i.e., the coding advantage $\chi(N)/\pi(N) \leq 2$. □

17.2.2 The linear programming perspective

We have just derived a bound of 2 for the coding advantage through a graph-theoretic approach. A linear programming perspective for studying the coding advantage turns out to be also interesting, and can lead to the same proven bound as well as other insights.

Table 17.1. min Steiner tree IP and min-cost multicast LP.

Minimize $\sum_e w(e)f(e)$ Subject to: $\begin{cases} \sum_{e\in\Gamma} f(e) \geq 1 \ \forall \text{ cut } \Gamma \\ f(e) \in \{0,1\} \quad \forall e \end{cases}$	Minimize $\sum_e w(e)f(e)$ Subject to: $\begin{cases} \sum_{e\in\Gamma} f(e) \geq 1 \ \forall \text{ cut } \Gamma \\ f(e) \geq 0 \quad \forall e \end{cases}$

Table 17.1 shows the linear integer program for the min Steiner tree problem on the left, where f is the variable vector and w is the constant link cost vector; Γ is a *cut*, or a set of links whose removal separates at least one receiver from the sender. The flow $f(e)$ can be assumed to be in the direction of originating from the sender component. On the right of the table is a linear program for min-cost multicast with network coding, with target throughput 1. The validity of this LP is based on Theorem 1.1. Note that different LP formulations for optimal multicast with network coding are possible and known, including link-based, path-based and cut-based ones. The first has a polynomial size and is practical to solve, while the later two are often convenient for theoretical analysis.

It is interesting that the min-cost multicast LP is exactly the LP relaxation of the min Steiner tree IP. Therefore the coding advantage for cost is equivalent to the integrality gap of the Steiner tree IP. Agarwal and Charikar [1] further

applied LP duality to prove that the maximum coding advantage for throughput is equivalent to the maximum integrality gap of the min Steiner tree IP. This reveals an underlying equivalence between the coding advantage in reducing cost and that in improving throughput, and also provides an interpretation of the power of network coding from the LP perspective: the flexibility of allowing arbitrary fractional flow rates. Furthermore, since it has been proven that the maximum integrality gap for the Steiner tree IP is 2 [1], one obtains an alternative proof for the bound 2 of the coding advantage; although it is not immediate whether this proof also works for half-integral flows.

17.2.3 Single multicast in Internet-like bi-directed networks

Given the fact that the coding advantage is finitely bounded in undirected networks but not so in directed ones, it is natural to ask which model is closer to real-world networks, and whether the coding advantage is bounded in such networks. A real-world computer network, such as the current generation Internet, is usually bi-directional but not undirected. If u and v are two neighbor routers in the Internet, the amount of bandwidth available from u to v and that from v to u are fixed and independent. At a certain moment, if the $u\rightarrow v$ link is congested and the $v\rightarrow u$ link is idling, it is not feasible to "borrow" bandwidth from the $v\rightarrow u$ direction to the $u\rightarrow v$ direction, due to the lack of a dynamic bandwidth allocation module. Therefore, the Internet resembles an undirected network in that communication is bidirectional, and resembles a directed network in that each link is directed with a fixed amount of bandwidth.

A better model for the Internet is a *balanced directed* network. In a balanced directed network, each link has a fixed direction. However, a pair of neighboring nodes u and v are always mutually reachable through a direct link, and the ratio between $c(\vec{uv})$ and $c(\vec{vu})$ is upper-bounded by a constant ratio $\alpha \geq 1$. In the case $\alpha = 1$, we have an absolutely balanced directed network. This is rather close to the reality in the Internet backbone, although last-hop connections to the Internet exhibit a higher degree of asymmetry in upstream/downstream capacities. Based on the constant bound developed in the previous section, we can show that the coding advantage in such an α-balanced network is also finitely bounded.

Theorem 17.3. *For a multicast session in an α-balanced bidirectional network, the coding advantage is upper-bounded by $2(\alpha + 1)$.*

Proof: We first define a few notations. Let $N_{1:\alpha}$ be the α-balanced network; let N_1 be an undirected network with the same topology as $N_{1:\alpha}$, where $c(uv)$ in N_1 is equal to the smaller one of $c(\vec{uv})$ and $c(\vec{vu})$ in $N_{1:\alpha}$; let $N_{\alpha+1}$ be the undirected network obtained by multiplying every link capacity in N_1 with $\alpha + 1$. Then we have:

$$\pi(N_{1:\alpha}) \geq \pi(N_1) \geq \frac{1}{\alpha+1}\pi(N_{\alpha+1}) \geq \frac{1}{\alpha+1}\frac{1}{2}\chi(N_{\alpha+1})$$
$$\geq \frac{1}{2(\alpha+1)}\chi(N_{1:\alpha}).$$

In the derivations above, the third inequality is an application of Theorem 1.1; the other inequalities are based on definitions. □

From Theorem 1.3, we can see that the more "balanced" a directed network is, a smaller bound on the coding advantage can be claimed. In the case of an absolutely balanced network, the bound is 4. In arbitrary directed networks, α may approach ∞, and correspondingly a finite bound on the coding advantage does not exist.

17.2.4 Towards tighter bounds

The constant bound of 2 for the coding advantage in undirected networks is not tight. So far, the largest coding advantage value observed is 9/8 for relatively small networks [14], and approaches 8/7 in a network pattern that grows to infinite size [1]. Closing the gap between 2 and 8/7 is an important open research direction. The significance here is twofold. First, it may provide a better understanding and more in-depth insights to network coding. Second, it may lead to advances in designing Steiner tree algorithms, which have important applications in operations research, VLSI design, and communication networks. Both the minimum Steiner tree problem and the Steiner tree packing problem are NP-hard, and it is known that for any constant $\alpha > 1$, a polynomial-time α-approximation algorithm exists for one of them if and only if it exists for the other. Note that, one may approximate the Steiner packing value $\pi(N)$ by computing the multicast throughput $\chi(N)$ instead. Such an approach yields a polynomial-time approximation algorithm, and the approximation ratio is precisely the tight upper-bound for the coding advantage. It is probable that the tight bound is closer to 8/7 than to 2. In that case, we might have obtained a better approximation algorithm for Steiner trees than the current best [18], which has an approximation ratio of 1.55. Initial progress has been made towards proving a tighter bound. In particular, it was proven that for a special set of combinatorial networks containing infinitely many network instances, the coding advantage is always bounded by 8/7 [19]. It is worth noting that so far, most known undirected network examples with > 1 coding advantage are closely related to the three-layer combinatorial networks.

17.2.5 Multiple communication sessions

Drastic changes occur when we switch the context from single to multiple communication sessions, where both intra-session and inter-session network coding are possible and need to be jointly considered. The complexity of deciding the optimal network coding scheme or the optimal throughput becomes NP-hard, and linear coding is not always sufficient. A fundamental tradeoff accompanies network coding across sessions: the ability to exploit the diversity of information flows brought by network coding, versus the onus of eliminating "noise" introduced by network coding on the receivers who no longer share the exact same interest in information reception. An understanding towards such a tradeoff is

only preliminary so far. In terms of demonstrating a higher coding advantage, no existing evidence shows that multiple sessions represent a better paradigm to be considered than single session.

For the special case where each session is a unicast, it is known that the coding advantage can be larger than 1 if either (a) the network is directed, or (b) integral routing is required. Due to the arbitrary asymmetry of connectivity between node pairs in opposite directions in (a), the gap can be as high as linear to the network size [10] and is therefore unbounded. In sharp contrast is the fact that so far, no example has been discovered for the case of an undirected network with fractional routing, where the coding advantage is larger than 1. It was conjectured in 2004 that network coding does not make any difference in this case [13, 9]. Arguments based on duality theory show that if the conjecture is false, then that implies a fundamental barrier on throughput-distance product in data communication can be broken by means of coding. The conjecture remains open today, with settlement obtained in special cases. It is worth noting that such settlement, even for the case of a 5-node fixed topology network, leverages tools from not only graph theory but also information theory, such as entropy calculus and information inequalities [12].

The case of multiple multicast sessions is most general and is even less understood. Furthermore, most existing results there pertain to directed networks only, and will not be discussed in this chapter. Due to space limitations, we have also chosen not to cover related studies of the coding advantage in other models, such as the case of average throughput and the case of wireless networks.

17.2.6 The source independence property of multicast

The source independence property refers to the fact that once the set of terminal nodes (including the sender and the receivers in the multicast group) is fixed in a given network, then the maximum achievable throughput is fully determined, regardless of which terminal node takes the sender role. Such a property is not true in directed networks, where bandwidth between neighbor nodes can be arbitrarily unbalanced. It holds in undirected networks for multicast without network coding, i.e., for tree packing. The definition of the packing number $\pi(N)$ does not specify which terminal node is the "sender" or the "root of the tree." The fact that source independence also holds in undirected networks for multicast with network coding is less obvious, but has been proven [14]. The proof is based on the observation that a valid multicast flow originating from one terminal node can be manipulated to construct a valid multicast flow of unchanged throughput that originates from another terminal node. More specifically, one just needs to reverse the network flow between the old sender and the new sender, as illustrated in Figure 17.5, the information content of the flow at each link does not need to be changed. It is interesting to observe that the definition of $\chi(N)$ relies on the selection of a special terminal node as the multicast sender, which should not be necessary by the source independence property. An equivalent, symmetrical

definition of $\chi(N)$ that does not isolate a single terminal node with a special role is open.

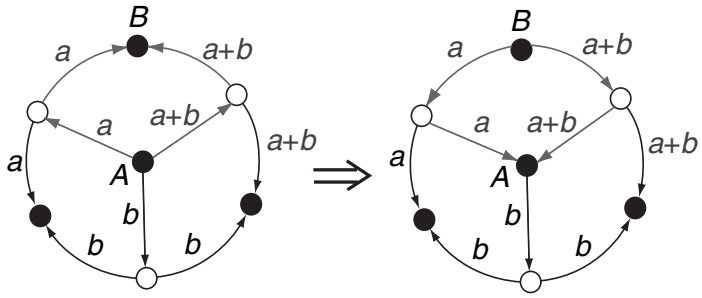

Figure 17.5 The source independence property of multicast with network coding: switching the source from a terminal A node to another terminal node B. The information flow between A and B are simply reversed.

We note that both the finite bound on the coding advantage and the source independence property hold in undirected networks but not in directed ones. We also showed that the finite bound holds in balanced directed networks. Now it is interesting to ask whether source independence also holds in an absolutely balanced directed network – we leave that as an exercise for the reader.

17.3 Network coding in peer-to-peer networks

Beyond theoretical studies, it is natural to assume a more pragmatic role and explore the feasibility of applying network coding to data communication over the Internet. It is intuitive that peer-to-peer (P2P) networks represent one of the most promising platforms to apply network coding, since end hosts on the Internet (called "peers") have the computational power to perform network coding, and are not constrained by existing Internet standards that govern most network switches at the core of the Internet. We now turn our attention to the advantages (and possible pitfalls) of using network coding in peer-to-peer networks, with a focus on two applications: *content distribution* and *media streaming*.

17.3.1 Peer-assisted content distribution with network coding

If the Internet is modeled as a balanced directed network, we have shown that the coding advantage is upper-bounded in theory. In reality, however, more pragmatic factors come into play when we consider the fundamental problem of multicast sessions: link capacities are not known *a priori*, and the optimal transmission strategy – including the linear code to be applied – needs to be computed. In the Internet, a multicast session corresponds naturally to a session of *content*

distribution, where information (such as a file) needs to be disseminated to a group of receivers.

Though it is feasible to use dedicated servers to serve content exclusively, it is wise to organize the receivers in a topology so that they serve one another, which captures the essence of *peer-assisted content distribution* in peer-to-peer networks. As peers assist each other by exchanging missing pieces of the file, they contribute upload bandwidth to the overall system of content distribution, thus alleviating the bandwidth load (and the ensuing costs) of dedicated content distribution servers.

In a peer-assisted content distribution session, the content to be disseminated is divided into blocks. Each peer downloads blocks that it does not have from other peers, and in turn, uploads blocks it possesses to others at the same time. Which block should a peer download, and from whom? Referred to as the *block scheduling* problem, this question needs to be addressed by designing protocols in a decentralized fashion, with local knowledge only. A poorly designed protocol may lead to the problem of *rare* blocks that are not readily available in the peer-to-peer network: those who have these rare blocks do not have the upload bandwidth to satisfy the demand for them.

To take advantage of network coding in peer-assisted content distribution, the first roadblock is the need to assign linear codes to network nodes. Randomized network coding, first proposed in [11], advocates the use of random linear codes, by assigning randomly generated coding coefficients to input symbols. With randomized network coding, receivers are able to decode with high probability, and no *a priori* knowledge of the network topology is assumed.

Gkantsidis *et al.* [8] have proposed to apply the principles of randomized network coding to peer-assisted content distribution systems. The basic concept may be best illustrated in the example of Figure 17.6. The file to be disseminated is divided into n blocks $b_1, b_2, \ldots, b_n$. The source first generates random coefficients $c_1, c_2, \ldots, c_n$, and then performs random linear coding on the original blocks b_i using these coefficients. All other peers follow suit, by generating coded blocks with random linear coding, using random coefficients on existing coded blocks it has received so far. All operations are to take place in a Galois field of a reasonable size, e.g., 2^{16}, to ensure the linear independence of encoded blocks [11].

It has been shown in [8] that the use of network coding introduces a substantial performance gain. We intuitively show such potential using the example in Figure 17.7. Assume that peer A has received blocks 1 and 2 from the source. If network coding is not used, node B can download block 1 or 2 from A with the same probability. At the same time, assume C independently downloads block 1. If B decides to retrieve block 1 from A, then both B and C will have the same block, and the link between them cannot be utilized. With network coding, A blindly transmits a linear combination of the two blocks to B, which is always useful to C because the combination contains block 2 as well.

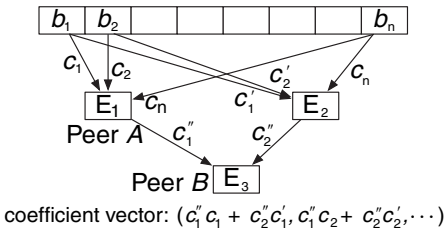

Figure 17.6 Peer-assisted content distribution with network coding.

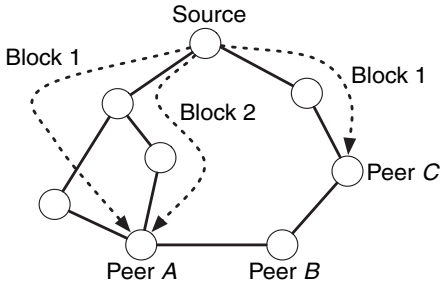

Figure 17.7 Advantages of network coding.

It is easy to see that random network coding offers three unique advantages to improve the performance of peer-assisted content distribution:

- First, it substantially increases the information contained in every block, as it transmits a random linear equation which may contain information about every block of the file. This has a high probability to be linearly independent to all existing blocks in the receivers, leading to a more efficient content dissemination process.
- Second, coding also greatly simplifies the problem of selecting the most appropriate (perhaps globally rarest) block to download, by blindly disseminating linear combinations of all existing blocks to downstream peers.
- Finally, it has been theoretically shown that network coding improves the resilience to highly dynamic scenarios of peer arrivals and departures [17]. This can be intuitively explained, since network coding eliminates the need to find rare blocks, the risk of "losing" these rare blocks when peers leave the system is no longer a concern.

To recover all n original blocks using n linearly independent coded blocks with high probability, a receiver needs to compute the inverse of the coefficient matrix, with a complexity of $O(n^3)$. As the number of blocks scales up with larger files, Chou *et al.* [4] proposed to divide the file into *generations*, and to perform network coding within the same generation. The performance of such generation-based network coding has been empirically evaluated in [7] and theoretically analyzed in [16], where network coding is shown to be highly practical for peer-assisted content distribution. Further, it has been shown [17] that generation-based network coding is still able to offer resilience to peer dynamics, even with just a small number of blocks in each generation.

17.3.2 Peer-assisted media streaming with network coding

Compared to content distribution, peer-assisted media streaming adds an additional requirement that the media content to be distributed needs to be played

back in real time as the media stream is being received. Similar to content distribution, we wish to conserve bandwidth on dedicated servers by maximally utilizing peer upload bandwidth. Different from content distribution, we also wish to maintain a satisfactory playback quality, without interruptions, especially during a "flash crowd" scenario when a large number of users wish to join around the same time.

To effectively stream media content with satisfactory real-time playback quality, it is important to deploy the most suitable peer topologies. Some argue that *tree-based push* protocols, which organize peers into topologies consisting of one or multiple trees, are the best for minimizing the delay from the source to receivers (e.g., [21]). However, trees may be difficult to construct and maintain when peers join and leave frequently. Most real-world peer-assisted streaming protocols, in contrast, use *mesh-based pull* protocols, which organize peers into mesh topologies, with each peer having an arbitrary subset of other peers as its neighbors (e.g., [24]). Such simplicity affords much better flexibility: there is no need to maintain the topology, as long as a sufficient number of neighbors is always available.

In particular, mesh-based pull protocols work as follows. For each streaming session, a finite buffer at a peer is maintained, with segments ordered according to playback sequence. Outdated segments after playback are deleted from the buffer immediately. After a new peer joins the system, it waits to accumulate a certain number of segments to start playback, the delay of which is referred to as the *initial buffering delay*. During playback, a peer concurrently sends requests for missing segments in the buffer, and downloads (or "pulls") these segments from those who have them. To update the knowledge of which neighbor has the missing segments, a peer would need to exchange availability bitmaps of its buffer with neighbors, referred to as *buffer map exchanges*.

Would network coding be instrumental in peer-assisted media streaming? Wang and Li [22] first evaluated the feasibility and effectiveness of applying network coding in live peer-assisted streaming sessions, with strict timing and bandwidth requirements. Generation-based random network coding has been applied as a "plug-in" component into a traditional pull-based streaming protocol without any changes. Gauss–Jordan elimination has been used to make it possible for peers to decode generations on the fly as blocks are being received, which fits naturally into streaming systems. It has been discovered that network coding provides some marginal benefits when peers are volatile with their arrivals and departures, and when the overall bandwidth supply barely exceeds the demand.

With such mildly negative results against the use of network coding, one would argue that the advantages of network coding may not be fully explored with a traditional pull-based protocol. In [23], Wang and Li proposed R^2, which uses random push with random network coding, and is designed from scratch to take full advantage of the benefits of network coding.

In R^2, the media stream is again divided into generations, and each generation is further divided into blocks. In traditional mesh-based pull protocols, missing

segments are requested by the downstream peer explicitly. Due to the periodic nature of buffer map exchanges, such requests can only be sent to those neighbors who have the missing segments in the previous round of exchanges. With network coding in R^2, however, since much larger generations are used, buffer maps that represent the state of generations – as opposed to blocks – can be much smaller. In addition, with larger generations than blocks, each generation takes a longer period of time to be received. As such, without additional overhead, buffer maps can be pushed to all neighbors as soon as a missing segment has been completely received, and there is no need to perform the exchange in a periodic fashion.

Since all buffer maps are up-to-date in R^2, a sender can simply select one of the generations – at random – still missing on the receiver. It then produces a coded block in this generation, and blindly *pushes* it to the receiver. An important advantage of network coding is "perfect collaboration": since all coded blocks within a generation are equally useful, multiple senders are able to serve coded blocks in the same missing generation to the same receiver, without the need for explicit coordination and reconciliation. An illustrative comparison between R^2 and traditional pull-based protocols is shown in Figure 17.8.

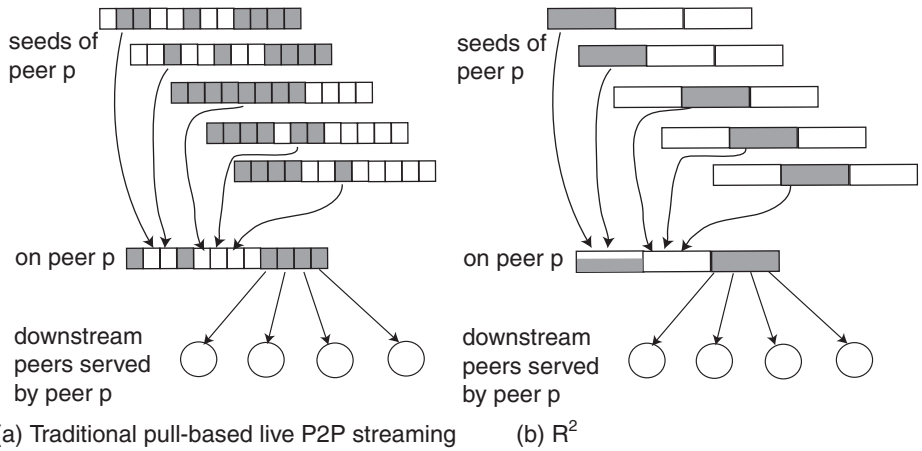

Figure 17.8 An illustrative comparison between a traditional pull-based streaming protocol and R^2.

Naturally, R^2 represents a simple design philosophy, rather than a strict protocol design. It allows for a flexible design space for more fine-tuned protocols. For example, the random push strategy can be tailored so that generations closer to playback deadlines are given higher priority to be served, in order to ensure timely playback. One possibility is to define and impose a *priority region*, which includes urgent segments immediately after the point of playback. Other prioritization and randomization strategies can also be incorporated. In this sense, R^2 is complementary to the usual algorithm design spaces, such as block selection and neighbor selection strategies.

With the use of generation-based network coding, R^2 enjoys two distinct advantages:

- First, it induces much less messaging overhead in buffer map exchange, leading to better performance in terms of playback quality and resilience. As buffer maps are pushed in real time when they change, neighboring peers receive timely feedback, and can proceed to serve missing segments with minimal delay. Indeed, Feng *et al.* [6] have theoretically corroborated the effectiveness of R^2, and pointed out that the lack of timely exchange of buffer maps may be a major factor that separates the actual performance of pull-based protocols from optimality.
- Second, equipped with random push and random network coding, R^2 offers shorter initial buffering delays and reduced bandwidth costs on dedicated streaming servers. This is due to the fact that multiple senders can serve the same receiver without the messaging overhead for explicit coordination purposes. With a stochastic analytical framework, Feng and Li [5] have analyzed both flash crowd and highly dynamic peer scenarios, and have shown that the design philosophy of using network coding in R^2 leads to shorter initial buffering delays, smaller bandwidth costs on servers, as well as better resilience to peer departures.

17.4 Conclusions

With network nodes performing coding operations on incoming messages, network coding is shown in theory to have the ability of improving throughput in multicast sessions within directed networks. When we think about applying the theory of network coding to the Internet, the most likely scenario is within the scope of peer-to-peer networks, since end hosts are computationally capable of performing such coding operations, and are not governed by stringent Internet standards. How likely is it for network coding to achieve similar theoretical gains in peer-to-peer networks?

In this chapter, we have started with a theoretical perspective: by extending from directed to undirected and bi-directed networks, we note that the coding advantage – the throughput gain compared to not using network coding – is finitely bounded in undirected and bi-directed networks. Instead of improving throughput, network coding makes it computationally feasible to compute optimal strategies for achieving optimal throughput: it changes the underlying structure of multicast algorithm design, from a tree packing perspective (without coding) to a network flow perspective (with coding), and consequently reduces the computational complexity of optimal multicast from NP-hard to polynomial-time solvable.

Assuming a more pragmatic role, we have presented known results in two applications in peer-to-peer networks: peer-assisted content distribution and media streaming. In peer-assisted content distribution, we have shown that network coding substantially simplifies the problem of selecting the most appropriate block to download (referred to as "the block selection problem"), and improves the resilience to highly dynamic scenarios of peer arrivals and departures. In peer-assisted media streaming, we have shown that protocols need to be designed from scratch to take full advantage of network coding; R^2, a collection of protocol design guidelines to incorporate network coding, has led to less messaging overhead in terms of exchanging buffer availability information, as well as shorter initial buffering delays.

It appears that the overall message is very optimistic. Despite the computational complexity of network coding (even with the use of generations), one would envision that network coding can be applied in the near-term future to peer-to-peer networks, which have consumed a substantial portion of the bandwidth available in the Internet today. Substantial savings in bandwidth costs on servers alone may be able to justify the additional computational complexity on end hosts. With Moore's Law that predicts increasingly abundant computational power, bandwidth, as a resource, will naturally be more scarce and needs to be carefully utilized. Network coding may very well be a useful tool to achieve more efficient bandwidth utilization in the Internet.

References

[1] Agarwal, A. and Charikar, M. (2004). On the Advantage of Network Coding for Improving Network Throughput. *Proc. IEEE Inform. Theory Workshop*, 2004.

[2] Ahlswede, R., Cai, N., Li, S. Y. R., and Yeung, R. W. (2000). Network Information Flow. *IEEE Trans. Inform. Theory*, 46(4):1204–1216, July 2000.

[3] Bang-Jensen, J. and Gutin G. (2009). Digraphs: Theory, Algorithms and Applications, 2nd edn., Springer.

[4] Chou, P. A., Wu, Y. and Jain, K. (2003). Practical Network Coding. *Proc. 42nd Annual Allerton Conference on Communication, Control and Computing*, 2003.

[5] Feng, C. and Li, B. (2008). On Large Scale Peer-to-Peer Streaming Systems with Network Coding. *Proc. ACM Multimedia*, 2008.

[6] Feng, C., Li, B., and Li, B. (2009). Understanding the Performance Gap between Pull-based Mesh Streaming Protocols and Fundamental Limits. *Proc. IEEE INFOCOM*, 2009.

[7] Gkantsidis, C., Miller, J., and Rodriguez, P. (2006). Comprehensive View of a Live Network Coding P2P System. *Proc. Internet Measurement Conference (IMC)*, 2006.

[8] Gkantsidis, C. and Rodriguez, P. (2005). Network Coding for Large Scale Content Distribution. *Proc. IEEE INFOCOM*, 2005.

[9] Harvey, N. J. A., Kleinberg, R. and Lehman, A. R. (2004). Comparing Network Coding with Multicommodity Flow for the k-Pairs Communication Problem. Technical Report, MIT CSAIL, November 2004.

[10] Harvey, N. J. A., Kleinberg, R., and Lehman, A. R. (2006). On the Capacity of Information Networks. *IEEE Trans. Inform. Theory*, 52(6):2345–2364, June 2006.

[11] Ho, T., Medard, M., Shi, J., Effros, M., and Karger, D. (2003). On Randomized Network Coding. *Proc. 41st Allerton Conference on Communication, Control, and Computing*, 2003.

[12] Jain, K., Vazirani, V. V., and Yuval, G. (2006) On The Capacity of Multiple Unicast Sessions in Undirected Graphs. *IEEE Trans. Inform. Theory*, 52(6):2805–2809, 2006.

[13] Li, Z. and Li, B. (2004). Network Coding: The Case of Multiple Unicast Sessions. *Proc. 42nd Annual Allerton Conference on Communication, Control, and Computing,* 2004.

[14] Li, Z., Li, B., and Lau, L. C. (2006). On Achieving Maximum Multicast Throughput in Undirected Networks. *IEEE/ACM Trans. Networking*, 14(SI):2467–2485, June 2006.

[15] Li, Z., Li, B., and Lau, L. C. (2009). A Constant Bound on Throughput Improvement of Multicast Network Coding in Undirected Networks. *IEEE Trans. Inform. Theory*, 55(3):997–1015, March 2009.

[16] Maymounkov, P., Harvey, N. J. A., and Lun D. S. (2006). Methods for Efficient Network Coding. *Proc. 44th Annual Allerton Conference on Communication, Control and Computing*, 2006.

[17] Niu, D. and Li, B. (2007). On the Resilience-Complexity Tradeoff of Network Coding in Dynamic P2P Networks. *Proc. International Workshop on Quality of Service (IWQoS)*, 2007.

[18] Robins, G. and Zelikovsky A. (2000). Improved Steiner Tree Approximation in Graphs. *Proc. 11th ACM-SIAM Symposium on Discrete Algorithms,* 2000.

[19] Smith, A., Evans, B., Li, Z., and Li, B. (2008). The Cost Advantage of Network Coding in Uniform Combinatorial Networks. *Proc. 1st IEEE Workshop on Wireless Network Coding,* 2008.

[20] Tutte, W. T. (1961). On the Problem of Decomposing a Graph into n Connected Factors. *Journal of London Math. Soc.*, 36:221–230, 1961.

[21] Venkataraman, V., Yoshida, K. and Francis, P. (2006). Chunkyspread: Heterogeneous Unstructured Tree-based Peer-to-Peer Muticast. *Proc. IEEE International Conference on Network Protocols (ICNP)*, 2006.

[22] Wang, M. and Li, B. (2007). Lava: A Reality Check of Network Coding in Peer-to-Peer Live Streaming. *Proc. IEEE INFOCOM*, 2007.

[23] Wang, M. and Li, B. (2007). R^2: Random Push with Random Network Coding in Live Peer-to-Peer Streaming. *IEEE J. Sel. Areas Commun.*, 25(9): 1655–1667, December 2007.

[24] Zhang, X., Liu, J., Li, B. and Yum, T.-S. P. (2005). CoolStreaming/ DONet: A Data-Driven Overlay Network for Efficient Live Media Streaming. *Proc. IEEE INFOCOM*, 2005.