Distrib. Comput.
DOI 10.1007/s00446-006-0014-9

SPECIAL ISSUE PODC 05

Building scalable and robust peer-to-peer overlay networks for broadcasting using network coding

Kamal Jain · László Lovász · Philip A. Chou

Received: 15 July 2005 / Accepted: 14 August 2006

Abstract We propose a scheme for building peer-to-peer overlay networks for broadcasting using network coding. The scheme addresses many practical issues such as scalability, robustness, constraints on bandwidth, and locality of decisions. We analyze the system theoretically and prove near optimal bounds on the parameters defining robustness and scalability. As a result we show that the effects of failures are contained locally, allowing the network to grow exponentially with server load. We also argue that adversarial failures are no more harmful than random failures.

1 Introduction

Consider a scenario where a server has content, such as a movie, that millions of clients would like to receive over the Internet, whether by downloading the content, or by streaming and playing the content in real time. The server has limited bandwidth. It has sufficient bandwidth to serve tens or perhaps hundreds of nodes, but not millions. In the absence of IP multicast, one solution is to form the server and clients into a peer-to-peer overlay network, and distribute the content using application layer multicast [7]. In multicast, the server sends the content to a collection of nodes, each node forwards the content to several other nodes, which in turn forward to several other nodes, and so forth. A problem with peer-to-peer application layer multicast is that the nodes are typically residential end-hosts, which are unreliable compared to routers, and furthermore do not have enough outgoing bandwidth to be able to forward the content to many other nodes. Moreover, a node has little incentive to forward the content to many other nodes. However, it is reasonable to assume that each node has enough bandwidth and incentive to forward the content to one other node. This would reduce the multicast distribution tree to a distribution "path". This could be an acceptable solution if the nodes were reliable. However, if there are a million nodes and the server is sending content directly to only one hundred nodes (the server's children), then there are nodes that are getting the content through approximately ten thousands hops. Even if there is a small probability that any particular node fails, the probability that any one of the upstream nodes fails is significant.

There has been a fair amount of work on this problem. (See for example [4,13] and the references therein.) The past work suggests that a node should be getting data from a small number of other nodes, that is, from more nodes than just a single parent, and that it should send data to an equal number (or approximately equal number) of child nodes. This honors the constraint of approximately equal input and output bandwidths, while allowing far shorter paths from the server. The data may be encoded with erasure codes (e.g., Reed-Solomon

A preliminary version of this paper appeared in *Proc. ACM Symp. Principles of Distributed Computation* (*PODC*), Las Vegas, July 2005.

K. Jain (✉) · L. Lovász · P. A. Chou
Microsoft Corporation, One Microsoft Way,
Redmond, WA 98052-6399, USA
e-mail: kamalj@microsoft.com

L. Lovász
e-mail: lovasz@microsoft.com

P. A. Chou
e-mail: pachou@microsoft.com

codes) or multiple description codes, so that it is not necessary for a node to get data successfully from all its parents. This solution improves the robustness over the previous solution, but reliability still degrades as the network gets larger if the number of connections between a node and its parents stays fixed. Moreover the building and maintenance of the overlay network can become complex if routing structures need to be maintained.

We propose a simple scheme for building overlay networks. Whenever a new node joins the network, it is connected randomly with other nodes (as detailed in Sect. 3). This results in a random graph as the network topology. Intuitively, random graphs have good connectivity. If every node has d parents, then most likely every node has connectivity d (i.e., d edge-disjoint paths) from the server. Random graphs also have good expansion properties. So if every node has d parents, then most likely every node will have about d^2 grandparents. If a node loses one of its parents due to a failure, then it will clearly suffer a loss of connectivity from the server. However, good expansion implies that if a node loses one of its grandparents, then most likely it will not suffer a loss of connectivity from the server. We show formally in Theorem 4 that if node failures are iid with probability p, then the probability of a working node's overall loss of connectivity from the server is about the same as the probability of the node's loss of connectivity from the server due to its parents' failures only, namely, about pd. So in effect, the impact of a node failure is localized. If a node fails then only its immediate children—not its grandchildren or other nodes—suffer a loss of connectivity from the server. The probability that a working node loses connectivity from the server does not increase as the size of the network grows.

Once we have such a strong guarantee on connectivity, in theory we can use Edmonds' matroid partition theorem [9] to do optimal multicast using multiple multicast trees. However, this theoretical solution is quite impractical to implement, though it achieves significantly better throughput than previously proposed solutions. As with many of the previously proposed solutions, it will need to recompute, when a node fails, the partition of the overlay network into multicast trees. This is not feasible if the node failures are short-lived events, such as packet loss or momentary congestion.

A more practical approach is to apply the idea of network coding [1,6,12,16]. Ahlswede et al. [1] showed that it is possible to broadcast information to all nodes at a rate equal to the minimum of the nodes' maximum rates of flow from the server, by mixing the packets of data entering each node using random functions. Li et al. [12] showed further that it is sufficient to use only linear functions. So if a node receives two packets A and B of the same size and it needs to forward one packet, then the node can use a time-varying linear function f to send a packet $f(A, B)$ having the same default size. Koetter and Médard [11] later showed that f can be time-invariant. It is worth stressing that the possibility of broadcasting at this maximal rate, which is implied by the theorem of Alswede et al., is not novel here, because in our case all the nodes are receivers and an existing theorem, namely Edmonds' theorem [9], already proves that it is possible to broadcast at the maximal rate in this case. The novelty here is randomization of the function f. Ho et al. [10] and independently Sanders et al. [17] showed that it is sufficient to choose f randomly. Shortly thereafter, Chou et al. [6] showed how randomization can be used to obtain practical network coding systems. In their scheme, the intermediate network nodes buffer packets, produce new packets as random linear combinations of the buffered packets, and send within each new packet a small set of coefficients that expresses the packet as a linear combination of an original set of packets. As each new packet carries within it the coefficients that are ultimately needed to decode or recode the packet, packets are decodable even if the network topology changes or components fail. Further, they showed through simulations using real network data that the throughput obtained using practical network coding is typically close to the optimum broadcast capacity. Finally, our collaborators Rodriguez and Gkantsidis show [16] (contemporaneously with this paper) that network coding can be used for large scale file distribution using a practical implementation based on [6] similar to the scheme we describe in this paper.

In this paper, we consider multicast in the overlay network from a general point of view: communication of common information from a source to a set of nodes. The communication may be realtime, or *synchronous*, as when broadcasting a live or pre-recorded television event to a set of receivers at nearly the same time, or it may be non-realtime, or *asynchronous*, as when downloading a file to a set of receivers at possibly different times. Asynchronous communication can be regarded as synchronous communication in which transmission of bits across a link can be deferred by buffering, allowing an arbitrary, non-deterministic transmission schedule. Otherwise they are quite similar. In this paper we choose to use the terminology of synchronous communication with the understanding that the two types of communication have corresponding concepts. For example, in synchronous communication *bit rate* refers to a number of bits per second, while in asynchronous communication the corresponding concept is simply a number of bits (per download task). Bit rate is often called *bandwidth* in this paper.

2 Setup and problem definition

We denote by N the number of nodes. Each node, or *user* has the same upload and download bandwidths. If the download bandwidth is more than the upload bandwidth, which is true for many DSL and cable modems, then we simply use the download bandwidth to the extent of the upload bandwidth. (For asynchronous communication this corresponds to transmitting the same number of bits that we receive for a file.) If the upload bandwidth is more than the download bandwidth then it only helps our system, though for simplicity, in this case we use only the upload bandwidth to the extent of the download bandwidth. Again for notational simplicity we assume that all users have the same nominal bandwidth. One can easily generalize our system to the case where different users have different nominal bandwidths. The actual bandwidths available to a user may be temporarily reduced below its nominal bandwidth due to congestion and competing traffic. Such bandwidth reductions can be treated as temporary failures, and will be briefly addressed below.

We decompose the bandwidth of each user into d equal "units" of bandwidth. In terms of this unit of bandwidth, the server bandwidth is denoted by k. (Thus the server would be able to support $\lfloor k/d \rfloor$ users through *unicast* connections.) Each user is allowed to join the system at any time and leave the system at any time. When joining the system the user is asked to follow a hello protocol and when leaving the user is asked to follow a good-bye protocol. Such a leave is called a *graceful leave*. If a node leaves otherwise (e.g., due to a system crash or killed application) the leave is considered a *non-ergodic failure*. A failure can also be due to a temporary, unannounced outage such as packet loss, network congestion, or other processes using the communication link. Such a temporary outage is called an *ergodic failure*. When failures occur, communication suffers. (For synchronous communication the bandwidth decreases, while for asynchronous communication the download time increases.) For non-ergodic failures, the server and the other affected nodes engage in repairs, basically to perform the steps that the leaving node was supposed to do in the good-bye protocol. Let p be the probability that a node fails non-ergodically within the repair interval, or fails ergodically. We expect p to be quite small, and we will put an upper bound on p in Sect. 4, when we analyze the system. Non-ergodic failures may also be adversarial. We deal with adversarial failures in Sect. 5.

The problems are how to build and maintain the overlay network (i.e., the hello protocol, the good-bye protocol, and the repair procedure) as well as the how to use the overlay network so that a large amount of data can be broadcast to a large number of users with high probability of success. We propose a solution to the former problem in this paper and we refer to the network coding literature for the latter problem. Network coding has been shown to be a good solution both theoretically [1,12] and practically [6,16]. In this paper, in addition to proposing how to build and maintain our overlay network, we prove that our overlay network has good theoretical properties. We believe that our network will also have good properties in practice, since we have made no impractical assumptions while proving the theoretical results (e.g., we did not take limits). Simulation results indicate that this will be the case [16].

3 Building and maintaining the network

We assume that the server emits (up to) k streams of unit bandwidth, and that each client node receives d streams of unit bandwidth and sends (up to) d streams of unit bandwidth. Imagine that the server is a curtain rod with k threads hanging, each thread representing a stream. When a node joins the network it picks d threads at random and clips them together. The clip represents the node, the threads entering the clip represent the streams entering the node, and the threads leaving the clip represent the streams leaving the node. Packets in the stream entering the node are mixed (using random linear combinations) and the mixed packets are sent in the streams leaving the node. Thus streams are mixed at each clip. We assume that newly arriving nodes clip the threads at the bottom. That is, nodes that come later in time receive streams from those nodes that came earlier in time. When a node leaves gracefully it just unclips the threads. That means the node basically matches each of its d children to one of its d parents. At all times there are k threads freely hanging from the bottom of the curtain. These threads represent a pool of slots, or unserved streams, to which a new node can connect.

A data structure maintained by the server (or some other centralized authority) mirrors the structure of the resulting network. The data structure is a matrix M of size $N' \times k$, where N' is the number of users currently in the system. Each row of the matrix corresponds to a node and each column corresponds to a thread. Each row of the matrix has exactly d ones and $k - d$ zeros. When a new node joins, the server creates a new row at the end of the matrix M with exactly d ones and $k - d$ zeros, selecting the locations of the d ones at random. When a node leaves gracefully, the server deletes

the corresponding row from the matrix M. The network topology captured by M is as follows. Assume that the matrix M contains an additional row at the top of the matrix, corresponding to the server, consisting of k ones. There is an edge from node i to node j if row i appears before row j in the matrix and there is a column containing a one in row i, a one in row j, and zeros in all the intervening rows. Whenever there is an edge from node i to node j we say that i is a *parent* of j and j is a *child* of i.

So when a new node wishes to join the network, it contacts the server. The server generates a new row at random and asks the indicated parents to begin sending streams to the new node. When an old node wishes to leave the network, it again contacts the server. The server asks the old node's parents to redirect their streams to the old node's children, and then deletes the old node's row. When a node fails (non-ergodically), then eventually the children of the failed node complain to the server. The server then asks the failed node's parents to redirect their streams to the failed node's children, and deletes the failed node's row. Persistent ergodic failures such as packet loss or network congestion can be dealt with in a similar way. However, there are usually more appropriate ways of dealing with the situation such as reducing the failing node's bandwidth [14]. Section 5 deals with such failures, as well as failures from adversarial attacks, and shows how to take more appropriate steps than simply deleting the node.

Our scheme is similar in spirit to a number of other schemes for building and maintaining peer-to-peer overlay networks (e.g., [3,8,15]). For example, in BitTorrent [8], a newly arriving node contacts the server for a random list of peers. The new node uses the list to find multiple parent nodes, from which it can receive streams with total bandwidth d. The new node then becomes part of the network, and may be called upon to serve streams with total bandwidth d to later-arriving nodes. The fact that BitTorrent has been used to deliver the 1.77 GByte Linux Redhat 9 distribution to well over a hundred thousand clients testifies to the scalability of such a central protocol. However, it is possible also to have a distributed protocol, as in [15], which uses a gossip mechanism for a newly arriving node to find its parents. The specifics of the protocol are less important than the topological structure of the resulting overlay network. Our protocol is an abstraction of the above (and similar) protocols sufficient to allow analysis of the performance of network coding within the resulting overlay network. Yet, our protocol is also sufficiently concrete that a practical protocol can be derived from it if the network properties must be guaranteed.

4 Analysis

In this section we prove bounds for robustness and stability in the face of failure and we relate these bounds to the number of threads emitted by the server and the number of threads subscribed to (and offered) by each user.

Let us recall our notation from the previous sections. We also make some additional assumptions. We let k be the number of threads hanging from the server, and we let d be the number of incoming and outgoing threads for each user, where $d \geq 2$. We assume $k \geq cd^2$, for some large enough constant c. We let p be the probability that a user fails in the repair interval. We assume $pd \leq \delta$, for some small enough constant δ. Failure is different from a graceful leave. As argued earlier, when a user leaves gracefully we effectively remove the corresponding row from M. This means the probability distribution of M, over all possible matrices of the same size, is the same as if the user had not had even joined the network. This gives us the following lemma.

Lemma 1 *When a node leaves gracefully then the probability distribution of the network over all possible networks is the same as if the node had not had even joined the network.*

Also as argued earlier, when a node fails, then after the repair, its corresponding row from M is removed by the server. So the failures that have been taken care of satisfy the above lemma too. This means that we can estimate the properties of M by building it sequentially. For the proof, we build M top down. Initially M has one row corresponding to the server. We will call an addition of a row in M a *step* or a *unit of time*. Note that as argued earlier, M contains the complete information about the network except for the failures. We put an additional tag on each row of M, denoting whether the corresponding node is a failed node or a working node. So, M together with the tag represents the complete information about the network, which we consider a directed acyclic graph on the working nodes.

Consider a node. According to the network coding theorem [1], it can receive the broadcast at the rate equal to its edge connectivity from the server. The connectivity should ideally be d. Its connectivity from the server will be affected by the failures of its immediate predecessors. The probability of failure of one of its immediate predecessors is at most dp. There is a possibility that its connectivity is affected by the failures of other nodes too. We show that this adds negligible probability to dp. Formally, we show that the probability that its connectivity will be affected is $(1+\epsilon)dp$. So in essence a node essentially feels only the effect of the failures of

its immediate predecessors and is not affected by the failures of other nodes in the network.

At any point in the network there are k threads hanging. A new node picks d of them at random. Conceptually we interchange the time ordering of two events: a node joining the network and it failing (or not). Instead we assume that the node tosses a coin before joining and thereby joins the network as a failed node with probability p and as a working node with probability $1-p$. When it joins it picks a random a set of d hanging threads.

Let $\mathcal{N}^t$ denote the network after t nodes have joined (including the information on which of these are failed nodes). Let B_j^t be the number of d-tuples of hanging threads in the network $\mathcal{N}^t$ that have edge-connectivity $d-j$ from the server (i.e., if a new node picks this d-tuple then its edge-connectivity from the server will be $d-j$). Clearly

$$\sum_{j=0}^{d} B_j^t = A = \binom{k}{d}$$

is the total number of d tuples of hanging threads. We are interested in the number of "total defects" measured by $B^t = 1\cdot B_1^t + 2\cdot B_2^t + \cdots + d\cdot B_d^t$. Note that the number of defective or "bad" d-tuples is $B_1^t + \cdots + B_d^t \le B^t$. The numbers B^t are random variables, where B^t depends only on the network $\mathcal{N}^t$.

Lemma 2 *Suppose a new node joins at time $t+1$. The probability of it picking a bad d-tuple is the expected proportion of bad d-tuples after time t, i.e., $E[B_1^t + \cdots + B_d^t]/A$.*

Proof The probability of it picking a bad d-tuple after time t is

$$\begin{aligned}\sum_i \frac{i}{A}\,\mathrm{prob}\left(B_1^t + \cdots + B_d^t = i\right) &= \frac{\sum_i i\,\mathrm{prob}\left(B_1^t + \cdots + B_d^t = i\right)}{A} \\ &= \frac{E[B_1^t + \cdots + B_d^t]}{A}\end{aligned}$$

□

Lemma 3 *Suppose a new node joins at time $t+1$. The expected loss in its bandwidth is $E[B^t]/A$.*

Proof Elementary. Follows from the network coding theorem [1]. □

Let us give an informal description of this process. If $B^t/A < pd$, then B^{t+1} will be larger than B^t in expectation; if $pd < B^t/A < 1-\epsilon$, then B^{t+1} will be smaller than B^t in expectation; finally, if B^t/A is close to 1, then B^{t+1} will again be larger than B^t in expectation. Thus the fraction of bad d-tuples has a drift toward the small value pd as long as it does not get to close to 1. If it gets close to 1, it will drift to 1, and the system will collapse. Our goal will be to show that this collapse will not happen (except with a negligibly small probability) for a time that is exponential in k/d^3. Let us also point out that such a collapse cannot be avoided: with some probability all nodes that join for a while will fail until no thread survives. The time before this happens is exponential in k.

Theorem 4 *Before the system collapses, $E[B^t]/A \le (1+\epsilon)pd$.*

Once we prove this theorem we will need to show that the system does not collapse for time exponential in k. More exactly,

Theorem 5 *The expected number of steps before the collapse is at least $\frac{1}{\xi_1} e^{\frac{\xi_2 k}{d^3}}$, where ξ_1 and ξ_2 are two appropriately chosen constants.*

The proof of this theorem will take several lemmas and is implied by Corollary 9.

Let us focus on the arrival of the t-th node first. For notational convenience, we suppress the superscript t, and write $B = B^t$, $B' = B^{t+1}$. When a failed node arrives, B tends to increase, and when a working node arrives B tends to decrease. Consider the t-th arriving node; suppose it picked a d-tuple denoted by D. The following lemma puts an upper bound on the maximum effect of this node on B.

Lemma 6

$$|B' - B| \le \frac{d^2}{k} A.$$

This bound cannot be improved in general; it is attained by the arrival of a single failed node at the beginning.

Proof Let T be any d-tuple of threads with $|T \cap D| = j$. Then the maximum change in the connectivity of T (up or down) is j. The number of such d-tuples is $\binom{d}{j}\binom{k-d}{d-j}$ and their effect on B is at most $j\binom{d}{j}\binom{k-d}{d-j} = d\binom{d-1}{j-1}\binom{k-d}{d-j}$. Summing this over all j gives

$$d\sum_j \binom{d-1}{j-1}\binom{k-d}{d-j} = d\binom{k-1}{d-1} = \frac{d^2}{k}A.$$

□

If the arriving node is a failure then B can increase by at most $(d^2/k)A$. If the arriving node is a working node then we show that the decrease of B is substantial, at least in expectation. The following lemma is the heart of our analysis.

Lemma 7 *If the total defect before a given step is B, and the new node is a working node, then the total defect after this step satisfies*

$$E[B'] \le B - B\frac{d}{k}\left(1 - \frac{d^2}{k} - \left(\frac{B}{A}\right)^{\frac{d-1}{d}}\right).$$

Proof We start by noting that the defects of d-tuples do not increase. All the probabilities below will be conditional on $\mathcal{N}_t$ and the event that the new node is working.

Consider a d-tuple F with connectivity $d-j$ from the server, where $j \ge 1$. Consider a $(d-j)$-element edge-cut separating F from the server; among all such cuts, consider one for which the side of the cut containing F is maximum. It is well known that this maximum is unique. Let T denote the set of hanging threads on the same side of the cut as F, and let $t = |T|$. Any d-tuple chosen from these t threads has connectivity at most $d-j$ from the server; hence we get

$$\binom{t}{d} \le B.$$

We can choose a subset $X \subset F$ with $|X| = j$ so that $F \setminus X$ has $d-j$ edge-disjoint paths to the server. F gets an additional connectivity when the arriving node picks at least one thread from X and at least one thread from outside T. We call this event a *good* event and the complement of this a *bad* event. A bad event is when the threads picked are either all from T or none from X. Using inclusion–exclusion, the probability of the bad event is

$$\frac{\binom{t}{d}}{\binom{k}{d}} + \frac{\binom{k-j}{d}}{\binom{k}{d}} - \frac{\binom{t-j}{d}}{\binom{k}{d}}.$$

Hence the probability of the good event is

$$\frac{\left(\binom{k}{d} - \binom{k-j}{d}\right) - \left(\binom{t}{d} - \binom{t-j}{d}\right)}{\binom{k}{d}}.$$

Let us try to lower bound the first term of the numerator,

$$\binom{k}{d} - \binom{k-j}{d} = \sum_{i=1}^{j} \binom{k-i}{d-1}.$$

We achieve a lower bound on this by bounding the ratio of the i-th term with the first term on the right hand side,

$$\begin{aligned}\frac{\binom{k-i}{d-1}}{\binom{k-1}{d-1}} &= \prod_{l=1}^{i-1} \frac{k-d-l+1}{k-l} \ge \left(\frac{k-d-i+2}{k-i+1}\right)^{i-1} \\ &= \left(1 - \frac{d-1}{k-i+1}\right)^{i-1} \ge 1 - \frac{(i-1)(d-1)}{k-i+1} \\ &\ge 1 - \frac{(d-1)^2}{k-d+1}.\end{aligned}$$

Since we know that $k \ge d^2$, we can lower bound the final term of the above inequality with $1 - (d^2/k)$. All together this yields

$$\begin{aligned}\binom{k}{d} - \binom{k-j}{d} &\ge \left(1 - \frac{d^2}{k}\right) \sum_{i=1}^{j} \binom{k-1}{d-1} \\ &= j\left(1 - \frac{d^2}{k}\right)\binom{k-1}{d-1}.\end{aligned}$$

It is easy to show that

$$\binom{t}{d} - \binom{t-j}{d} = \sum_{i=1}^{j} \binom{t-i}{d-1} \le j\binom{t-1}{d-1}.$$

Both together give a lower bound on the probability that the defect of F decreases,

$$\begin{aligned}&\frac{\left(\binom{k}{d} - \binom{k-j}{d}\right) - \left(\binom{t}{d} - \binom{t-j}{d}\right)}{\binom{k}{d}} \\ &\ge j\left(\frac{\left(1 - \frac{d^2}{k}\right)\binom{k-1}{d-1} - \binom{t-1}{d-1}}{\binom{k}{d}}\right).\end{aligned}$$

We also know that $\binom{t}{d} \le B$. Using this we want to upper bound $\binom{t-1}{d-1}$ in terms of B. For convenience let us upper bound the $\binom{t-1}{d-1}/\binom{k-1}{d-1}$ in terms of B/A. We claim that

$$\frac{\binom{t-1}{d-1}}{\binom{k-1}{d-1}} \le \left(\frac{B}{A}\right)^{\frac{d-1}{d}}. \tag{1}$$

Indeed,

$$\frac{\binom{t-1}{d-1}}{\binom{k-1}{d-1}} = \prod_{i=1}^{d-1} \frac{t-i}{k-i} \le \left(\frac{t}{k}\right)^{d-1},$$

and hence

$$\left(\frac{\binom{t-1}{d-1}}{\binom{k-1}{d-1}}\right)^d \le \left(\frac{t}{k}\frac{\binom{t-1}{d-1}}{\binom{k-1}{d-1}}\right)^{d-1}$$
$$= \left(\frac{\frac{t}{d}\binom{t-1}{d-1}}{\frac{k}{d}\binom{k-1}{d-1}}\right)^{d-1} = \left(\frac{\binom{t}{d}}{\binom{k}{d}}\right)^{d-1} \le \left(\frac{B}{A}\right)^{d-1}.$$

Using (1), the probability that the defect of F decreases can be bounded from below by

$$j\left(\frac{\left(1-\frac{d^2}{k}\right)\binom{k-1}{d-1} - \binom{t-1}{d-1}}{\binom{k}{d}}\right)$$
$$\ge \frac{jd}{k}\left(1 - \frac{d^2}{k} - \left(\frac{B}{A}\right)^{\frac{d-1}{d}}\right).$$

Hence the expected decrease in the total defect is at least

$$\sum_j B_j \frac{jd}{k}\left(1 - \frac{d^2}{k} - \left(\frac{B}{A}\right)^{\frac{d-1}{d}}\right)$$
$$= B\frac{d}{k}\left(1 - \frac{d^2}{k} - \left(\frac{B}{A}\right)^{\frac{d-1}{d}}\right).$$

This proves Lemma 7. □

Let $b = \frac{B}{A}$, $b' = \frac{B'}{A}$. We want to compare $E[b']$ with b (conditioning on $\mathcal{N}_t$). By Lemma 7,

$$E[b'] - b \le \frac{pd^2}{k} - \frac{(1-p)d(k-d^2)}{k^2}b + \frac{(1-p)d}{k}b^{2-\frac{1}{d}}.$$

Let $f(b)$ denote the right hand side as a function of b. It is straightforward to check that f is convex and has a minimum at $a_0 = \frac{1-d^2/k}{2-1/d} \approx \frac{1}{2}$. Furthermore, the minimum value of f is less than $-d/8k$.

We also need information about the roots of f. The above discussion implies that f has two roots $0 < a_1 < 1/2 < a_2 < 1$ in the interval $[0,1]$, and it is not hard to see that

$$a_1 = \frac{pd}{(1-p)(1-\frac{d^2}{k})}(1+\epsilon), \quad \text{where}$$
$$0 < \epsilon < (2pd)^{1-1/d}.$$

The other root (which is less interesting for us) satisfies

$$a_2 = 1 - \left(\frac{pd}{1-p} + \frac{d^2}{k}\right)(1+\epsilon),$$
$$\text{where } |\epsilon| < 2\left(\frac{1}{d} + \frac{d^2}{k}\right).$$

The first root proves Theorem 4. This is because when the total defect (i.e., B^t)is between a_1A and a_2A, total defect tends to decrease, whereas if the total defect is below a_1A, it tends to increase. This keeps the expected total defect at a_1A over all the networks which could have been evolved at this point and not necessarily for the particular network which has actually evolved. Note that Theorem 4 remains valid as long as we do keep the total defect less than a_2A. If we get a network with total defect at least a_2A then our network may collapse. For Theorem 4 to be meaningful we should prove that the expected time for the collapse to happen is exponentially large. Toward this goal let us solve the equation $f(b) \le -c_1$, where $c_1 > 0$. For these values of b, B' tends to be significantly smaller than B. Again one can show that the equation $f(b) \le -c_1$ has two roots. Let us call them b_1 and b_2. One can also show that the difference between them is at least a constant, say δ_1, for sufficiently small c_1. Here, $c_1 = \delta_2 d/k$ works for a sufficiently small constant δ_2.

Now let us construct an infinite graph in which each node represents a network. The node set V of the infinite graph is partitioned into set of vertices V_t, where V_t is the set of all possible networks we could encounter after t steps. We also partition the node set V in two more ways. In one, $V = U_0 \cup U_1 \cup \cdots \cup U_A$, where U_i is the set of all networks with total defect i. In another, $V = W \cup X \cup Y \cup Z$, where $W = U_0 \cup \cdots \cup U_{\lfloor b_1 A \rfloor}$, $X = U_{\lceil b_1 A \rceil} \cup \cdots \cup U_{\lfloor (b_1 + (d^2/k))A \rfloor}$, $Y = U_{\lceil (b_1+(d^2/k))A \rceil} \cup \cdots \cup U_{\lfloor b_2 A \rfloor}$ and $Z = U_{\lceil b_2 A \rceil} \cup \cdots \cup U_A$. For convenience put $b = b_2 - (b_1 + (d^2/k))$. We put an edge from a node u to a node v if the network corresponding to u can become a network corresponding to v by the arrival of a single network node. The weight of the edge is the corresponding probability of u becoming v in one step. Now we start a random walk from a node in U_0 corresponding to the network with one node, i.e., a server. Let us compute the expected number of steps to reach a node in Z. Since Lemma 6 put a bound on the maximum jump this random walk can make, it is sufficient to estimate the probability of crossing Y.

Lemma 8 *Start a new random walk at some node in X. The probability that the random walk reaches Z before reaching X or W is at most $\xi_1 e^{-\frac{\xi_2 k}{d^3}}$, where ξ_1 and ξ_2 are appropriately chosen constants.*

Proof Our random walk is not a martingale but it resembles a submartingle if we follow the subscript of U's. Our proof also resembles the proof of Azuma's inequality in martingales. Let X_i be a random variable, which measures the change in the subscript of U's in the i-th step (i.e., X_i is the total defect after the i-th step.) Lemma 6

tells us that $X_i < (d^2A)/k$. Let t be a positive integer. We want to find:

$$\begin{aligned}
&\text{prob}\,(X_1 + X_2 + \cdots + X_t \geq bA) \\
&\quad = \text{prob}\left(\mathrm{e}^{(\beta(X_1+X_2+\cdots+X_t)-\beta bA)} \geq 1\right) \\
&\quad \leq E\left[\mathrm{e}^{(\beta(X_1+X_2+\cdots+X_t)-\beta bA)}\right] \\
&\quad = \mathrm{e}^{-\beta bA} E\left[\mathrm{e}^{\beta(X_1+X_2+\cdots+X_t)}\right] \\
&\quad = \mathrm{e}^{-\beta bA} E\left[\mathrm{e}^{\beta(X_1+X_2+\cdots+X_{t-1})} E\left[\mathrm{e}^{\beta X_t}\right]\right].
\end{aligned}$$

Here β is some positive constant to be optimized later. The first inequality follows by Markov's inequality. The nested expectation is conditioned on the network obtained after the $t-1$ steps. Note that we terminate our random walk if we reach Z. We also terminate our random walk if we are in X or W after the start. So we are in Y. For networks in Y, we know that $E[X_t] \leq -(A\delta_2 d)/k$. We also know the maximum change. Using this and convexity of e^x we get

$$E\left[\mathrm{e}^{\beta X_t}\right] \leq \frac{d+\delta_2}{2d}\mathrm{e}^{-\frac{\beta d^2 A}{k}} + \frac{d-\delta_2}{2d}\mathrm{e}^{\frac{\beta d^2 A}{k}}.$$

We choose β so that $\mathrm{e}^{\frac{\beta d^2 A}{k}} = \sqrt{(d+\delta_2)/(d-\delta_2)}$. With some simplifications this gives

$$\begin{aligned}
&\text{prob}\,(X_1 + X_2 + \cdots + X_t \geq bA) \\
&\quad \leq \left(\sqrt{\frac{1-\frac{\delta_2}{d}}{1+\frac{\delta_2}{d}}}\right)^{\frac{kb}{d^2}} \left(\sqrt{1-\frac{\delta_2^2}{d^2}}\right)^t.
\end{aligned}$$

This implies that the probability that the random walk reaches Z before returning back to W or X is bounded above by

$$\frac{\left(\sqrt{\frac{1-\frac{\delta_2}{d}}{1+\frac{\delta_2}{d}}}\right)^{\frac{kb}{d^2}}}{1-\sqrt{1-\frac{\delta_2^2}{d^2}}}.$$

Choosing two new constant ξ_1 and ξ_2 we can upper bound the above by

$$\xi_1 \mathrm{e}^{-\frac{\xi_2 k}{d^3}}.$$

□

Corollary 9 *The probability of collapse within t steps is at most* $t\xi_1 \mathrm{e}^{-\frac{\xi_2 k}{d^3}}$.

Corollary 9 implies the Theorem 5. We start the random walk in W. We do not count the number of steps until it reaches some node in X. Once it reaches X, we count it one megastep when the random walk starting from X returns back to X or W or reaches Z. We call a megastep a *success* if it reaches Z. We know that the probability that a megastep is a success is exponentially small. So the random walk must make exponentially many megasteps before it has a reasonable probability of success. This proves Theorem 5.

5 Adversarial and other kinds of failures

In earlier sections we assume that a node fails with probability p. Such models are usually based on theoretical assumptions made to prove a practical scenario. We would like to avoid this assumption. Instead we would like to say that fraction p of the nodes can fail. This then also covers the case when adversaries join the overlay network with an aim to ruin the system. An adversary can ruin the system in two ways, first by injecting an incorrect data packet and second by not doing its share of forwarding data to other nodes. We do not deal with the first scenario in this section and assume that such behavior can be handled by either hardware security (tamper proof hardware distributed to users), software security (an incorrect data packet cannot be authenticated) and/or legal security (user signs a contract to not engage in such behavior). On the other hand the second scenario cannot be handled by security means. The reason is that the second scenario can happen naturally, e.g., due to congestion or real failures. So a set of adversaries may try to ruin the system by intentionally failing, perhaps simultaneously. For example, a set of adversaries may join the system, cut-off the power from their hardware at the same time, and later claim that it was only a coincidence.

Suppose the adversaries do not have any control over their arrival time, i.e., they cannot join together at the same time hoping to be logically close to each other in the overlay network. Precisely, let us assume an adversary joins the network at a random time. In other words, the set of adversaries is a uniformly chosen random subset of users of cardinality at most the p-fraction of users. This scenario is almost the same as the scenario when a user is an adversary with probability p (with high probability that at most fraction $p+\epsilon$ of users will fail). So the proofs and claims in Sect. 4 remain valid. Thus we can handle adversarial failures by reducing an oblivious schedule of node arrivals to an randomized schedule of node arrivals.

So the only thing we need to enforce is the random ordering of the arrival times of users. We can achieve the equivalent of this by slightly modifying the system defined in Sect. 3. On the arrival of new user, the server

will not append a corresponding row at the end of the matrix M but instead will insert the corresponding row randomly into the matrix M.

A facility to insert rows into the matrix will also allow us to handle other situations like congestion. Suppose a node becomes congested on either its incoming or outgoing links and would like to reduce its load. The node picks a child and a parent and joins them directly. Of course for logistical reasons the node must contact the server too. When the node sees that its congestion is gone for a sufficient length of time, it tries to increase its rate of obtaining data. It contacts the server, which makes one of the zeros in the corresponding row in matrix M into a one at random.

The last point we would like to make is that the proofs assume equal bandwidth for all the nodes. However, the design of the system does not use this fact anywhere. This allows us to have users with heterogeneous network connections, e.g., some users could have DSL connections and others could have T1 connections. The ability to handle heterogeneous users allows priority encoding transmission [2] or other means for users with higher bandwidth connections to get higher resolution broadcasts. Priority encoding transmission also allows graceful degradation of quality with network failures, as described in [6].

6 Delay versus cycles

The scheme proposed by our paper for constructing and maintaining the overlay network ensures that as users join and leave the network its topology remains acyclic. A consequence of this invariant is that for network coding there will be no loss of throughput due to the delay spread inherent in cyclic networks [6]. On the other hand, the resulting communication delay from the source to a user, on average, will grow linearly with the number of users. If we are willing to suffer a small loss of throughput, by tolerating cycles we can reduce the average communication delay to a factor logarithmic in the number of users, using for example a topology induced by the union of trees as in [4,13]. In [4,13], the nodes are arranged in d balanced d-ary trees, such that each node is an internal node in exactly one tree and is a leaf in the remaining trees. Hence each node has exactly d parents and at most d children. Furthermore, the communication delay from the server to a user, on average, is logarithmic in the number of users. Moreover, the server only needs to support d outgoing threads (one to each tree). On the other hand, cycles are created, which can cause some loss of throughput using network coding [6].

Random graphs exhibit similar characteristics. In this case, each new user selects d random edges in the existing network, and inserts itself at these edges. Since random graphs are expanders with high probability, the communication delay is again logarithmic. Moreover, since the graph is an expander, the load on the server can be reduced by supplying directly only a few child nodes, supporting the entire population through these nodes. In fact in the file download scenario it may be possible eventually for the server to disconnect itself completely from the network after the content has been delivered to a small fraction of the population. In the random graph model we do not even need to assume that the upload bandwidth of a node is the same as its download bandwidth, though we need to assume that the average upload bandwidth is no less than the average download bandwidth.

7 Discussion and open issues

Our scheme for building peer-to-peer overlay networks for broadcasting is based on network coding. Network coding has been shown to be an optimal broadcasting scheme when we are given a network. It has not been clear whether network coding would be as useful when we have the option of designing the network topology as well. This paper shows that when we use network coding, which is a more general way to broadcast than routing, we can simplify the design of the network itself. In particular, we show that effective peer-to-peer overlay networks can be designed and maintained with a very small data load on the server. In our scheme the server and clients follow a very simple hello, good-bye, and repair protocol. In corresponding practical schemes (e.g., [15]), the role of the server can be decreased still further or even eliminated.

Lemma 3 and Theorem 4 show that the expected loss in connectivity for any user is about pd. Note that pd is essentially the best possible here, because every node has d parents and the expected loss of connectivity from each parent is p. We further show that this remains true until the number of nodes in the system is exponential in k/d^3 (Theorem 5). We believe that well before this limit is reached, the system may be self-sustaining (without requiring bandwidth connectivity all the way from the source) if the scenario is a download scenario and each node is required to reliably transmit as many bytes as it consumes. This possibility is an open issue for further study.

Our theorems hold for $d \geq 2$. For distribution of content at a fixed bandwidth, the bandwidth is divided into d equal units of bandwidth. As d increases, the bandwidth

carried on each thread decreases inversely with d. Hence the expected fraction of bandwidth lost is essentially p, independent of d. Thus all choices of d are essentially equivalent in terms of expected loss of bandwidth. It is an open issue to show that the variance of the fraction of bandwidth lost decreases inversely with d. This would imply that if one wants a more consistent bandwidth (e.g., for Internet radio or video on demand), then a larger d would be a better choice. If one cares only about expected bandwidth over the long term (e.g., for long file downloads), then $d = 2$ would be sufficient.

Note that given a fixed server bandwidth, k is proportional to d, so the number of nodes we can add into the system is exponential in the server bandwidth divided by d^2. We do not believe that this is the best possible exponent. One approach to finding a better exponent is to bound the second moment of the loss in connectivity. In this paper, we estimate only the probability of losing one thread of connectivity, which we find to be about the same as the probability of losing a parent. We conjecture that the probability of losing $\kappa << d$ threads of connectivity must be about the same as the probability of losing κ parents. This would truly imply that the effect of failures is locally contained. An approach to tackle this conjecture could be to handle $B_1^t, B_2^t, \ldots, B_d^t$ separately.

Another open issue is how to guard against certain kinds of malicious attacks. As mentioned in Sect. 5, our system is fairly robust to *failure attacks* by malicious users who join the system in order to fail. Our system is also fairly robust, at least in the short term, to *entropy destruction attacks* by malicious users who join the system in order to simply pass on trivial linear combinations of packets. (In the longer term, entropy destruction attacks are worse than failure attacks because they are more difficult to detect.) However, our system is not robust to *jamming attacks* by malicious users who join the system in order to inject random information into the system by randomly corrupting packets. This attack is particularly serious because the random packets have the potential, after network coding, of contaminating almost every packet that almost every user receives, thus jamming the system. The standard way to prevent an attack by a malicious user injecting bad data into the system is to require that valid senders sign their packets cryptographically. Assuming no mixing occurs in the network, the invalid packets can then be immediately detected and discarded. The difficulty is that, in network coding, the packets are mixed in the network by the user nodes, none of which can be trusted. Thus, to prevent a jamming attack in an open system that uses network coding, one would need a signature scheme such that the signature of a mixed packet can be easily derived from the signatures of the packets contributing to the mixture. In an earlier version of the paper we posed this as an open question whether such a scheme is possible. Recently Charles et al. [5] answered this question with affirmative. If such a scheme is used in network coding, then a malicious user would not be able to inject a random packet without immediate detection. It is worth pointing out that [5] scheme require all the Network Coding operation to be performed on large fields thereby possibly slowing the system considerably. The next open problem we like to pose whether such a signature scheme, where only the signatures are done on large fields but basic network coding is done on small fields, is possible?

Acknowledgment We would like to thank Jeong Han Kim for a discussion about Azuma's inequalities.

References

1. Ahlswede, R., Cai, N., Li, S.-Y.R. Yeung, R.W.: Network information flow. IEEE Trans. Inf. Theory **IT-46**, 1204–1216 (2000)
2. Albanese, A., Blömer, J., Edmonds, J., Luby, M., Sudan, M.: Priority encoding transmission. IEEE Trans. Inf. Theory **42**, 1737–1744 (1996)
3. Byers, J.W., Considine, J., Mitzenmacher, M., Rost, S.: Informed content delivery across adaptive overlay networks. In: Proc. SIGCOMM, Pittsburg. ACM, (2002)
4. Castro, M., Druschel, P., Kermarrec, A.-M., Nandi, A., Rowstron, A., Singh, A.: Splitstream: high-bandwidth content distribution in a cooperative environment. In: Proceedings of the International Workshop on Peer-to-Peer Systems (2003)
5. Charles, D., Jain, K., Lauter, K.: Signature for network coding. Princeton University, Princeton (2006)
6. Chou, P., Wu, Y., Jain, K.: Practical network coding. In: Proceedings of Allerton Conf. Communications, Control, and Computing, Monticello (2003)
7. Chu, Y., Rao, S.G., Zhang, H.: A case for end system multicast. In: Joint Int'l Conf. Measurement and Modeling of Computer Systems (SIGMETRICS) (2000)
8. Cohen, B.: Incentives build robustness in BitTorrent. http://bitconjurer.org/BitTorrent/bittorrentecon.pdf (2003)
9. Edmonds, J.: Edge-disjoint branchings. In: Rustin, R. (eds.) Combinatorial Algorithms. New York Academic, pp. 91–96 (1973)
10. Ho, T., Koetter, R., Médard, M., Karger, D.R., Effros, M.: The benefits of coding over routing in a randomized setting. In: Proceedings of the Int'l Symp. Information Theory, Yokohama, IEEE, (2003)
11. Koetter, R., Médard, M.: An algebraic approach to network coding. IEEE ACM Trans Netw **11**, 782–795 (2003)
12. Li, S.-Y.R., Yeung, R.W., Cai, N.: Linear network coding. IEEE Trans. Inf. Theory **IT-49**, 371–381 (2003)
13. Padmanabhan, V.N., Wang, H.J., Chou, P.A.: Resilient peer-to-peer streaming. In Proceedings of the Int'l Conf. Network Protocols, Atlanta (2003)
14. Padmanabhan, V.N., Wang, H.J., Chou, P.A.: Supporting heterogeneity and congestion control in peer-to-peer

multicast streaming. In: Proceedings of the Int'l Workshop on Peer-to-Peer Systems, San Diego (2004)
15. Rejaie, R., Stafford, S.: A framework for architecting peer-to-peer receiver-driven overlays. In Proceedings of the Int'l Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV), Kinsale (2004)
16. Rodriguez, P., Gkantsidis, C.: Revolutionising content distribution. In: Proceedings of the Conf. Computer Communications (INFOCOM), Miami. IEEE, (2005) (Submitted)
17. Sander, P., Egner, S., Tolhuizen, L.: Polynomial time algorithms for network information flow. In: Symposium on Parallel Algorithms and Architectures (SPAA), pp. 286–294. San Diego ACM (2003)