

# $r$ -Regular P2P Broadcast Networks: Optimal Delay and Throughput Using Network Coding

Amy Fu and Parastoo Sadeghi

The Australian National University, Canberra, 0200, Australia

Emails:{amy.fu;parastoo.sadeghi}@anu.edu.au

**Abstract**—We introduce a homogeneous  $r$ -regular network model for a peer-to-peer (P2P) video broadcast network. Such networks are simple to construct and allow the implementation of fairness strategies. We use our model to show why the greedy and rarest first push-based strategies give the suboptimal performance often observed in the literature. We propose a novel network coding based transmission strategy and prove that it results in optimal playback delay and throughput performance.

## I. INTRODUCTION

### A. Motivation and Background

Recently there has been considerable research on how to effectively implement peer-to-peer video broadcast networks over the Internet. Key performance criteria are high bandwidth efficiency, low delay and continuity of the data stream [1], but there are several challenges in meeting these requirements. As the number of peers increases, coordinating their transmissions to maintain high efficiency and low delay becomes an increasingly difficult task. Any coordination overhead must be *scalable* so that it will not increase substantially with the number of peers. Ideally a peer should not be connected to too many *neighbours*, as establishing and maintaining connections also incurs overhead. Good performance must be maintained even under high *churn* where peers join and leave the network at whim [2].

A variety of approaches have been adopted in designing data propagation through the network. They can generally be classified as either *tree-based*, where data is distributed based on the structure of the network, or *data-driven* where decisions are based on what data peers have already received [1]. While tree-based strategies generally have lower delay, they are extremely vulnerable to changes in the network structure and so are not suited to networks with high churn. By comparison data-driven strategies are less dependent on the network structure, but peers also require information on what their neighbours have received.

Because peer-to-peer video broadcast networks have high churn and require low latency, data-driven *push-based* transmission strategies are the most suitable, where a peer's neighbour will decide what to send [3]. Two well-known strategies that fall into this category are the *greedy* and *rarest first* strategies [4]. While neither performs particularly well on its own, a compromise between the two has been experimentally shown to give good performance [4], [5]. To our knowledge the reasons behind this have not been studied in much detail in the context of a real network structure.

More advanced push-based transmission strategies have also been proposed. In [6], *randomised network coding* [7] is combined with push-based transmission for improved continuity and resilience to churn. Network coding allows good coordination between peers, but incurs a *decoding delay*, since enough chunks must be received to recover the original data. By carefully deciding which chunks to code together it is possible to limit this delay. For example, [6] only permits network coding within each segment of the video stream, so every segment can be decoded once all its constituent parts have been received. But one might ask whether there is an even better way for peers to decide which data to code together.

### B. Contributions

In this paper we investigate how different data propagation strategies affect the bandwidth efficiency and delay of a network. We demonstrate a data-driven push-based transmission strategy that uses network coding to simultaneously achieve optimal throughput and minimum delay.

We study a static homogeneous  $r$ -regular network, where each peer is randomly connected to exactly  $r$  bidirectionally linked neighbours. This type of network has already been implemented commercially for applications such as collaborative engineering [8], but to our knowledge has not been studied in the context of peer-to-peer video streaming. There are several reasons  $r$ -regular networks are useful to consider. For analytical purposes they have some useful statistical properties that make them relatively easy to study. They can achieve maximum throughput for the bandwidth available, and the delay across the network can be characterised. For the transmission schemes studied, we show that certain simplifications are possible which allow the performance of these schemes to be understood in greater detail. In practical terms,  $r$ -regular networks are simple to construct and reconnect under churn. They also allow peers to implement a simple fairness strategy to ensure cooperation from their neighbours.

In this paper we explain why under this network model, the rarest first and greedy transmission strategies result in suboptimal performance. We propose a novel scheme that utilises network coding and prove that it achieves the maximum throughput and minimum delay possible for the network.

## II. MODEL

A server node wishes to broadcast a video stream  $S$ , consisting of equal sized *chunks* of data, to a homogeneous

network of  $n - 1$  identical peers, each of which is assigned an index  $0 < i < n$  ( $i = 0$  for the server). Our peer-to-peer network model is a topology where each peer is connected to  $r$  neighbours by bidirectional links, so that every link  $l_{i,j}$  from peer  $i$  to  $j$  is mirrored by another link  $l_{j,i}$ . The capacity of these links is normalised to one chunk per unit time. The streaming rate of the broadcasted video is set to  $r$  chunks per unit time, the maximum achievable rate of the network.

The bidirectional nature of links between peers means that, unlike strategies such as [9], during typical operation a peer expects to send and receive roughly the same number of chunks from each of their neighbours. Therefore, peers can employ a simple tit-for-tat strategy to ensure their neighbours are contributing fairly.

### A. Constructing the Network

The network formed by the server and peers can be represented by  $n$  nodes, each of which is linked to exactly  $r$  neighbours to form an  $r$ -regular graph. A nice property of  $r$ -regular graphs is that the minimum cut between any two nodes, or equivalently a peer and the server, is almost surely  $r$  [10]. The nodes of the graph are connected at random based on the *configuration model* [11], which generates all possible  $r$ -regular graphs with equal probability.

To construct a random network we begin with a set of  $n$  unconnected vertices, to each of which  $r$  *half-links* are attached. These half-links are paired with each other uniformly at random to form the links of a network. A small fraction of the connections formed in this manner result in self-loops or multiple edges between two nodes. These redundant links would impact badly on the throughput of the network, so for practical purposes configurations containing them are avoided. We can correct these undesired links by severing an existing link at random, and connecting each of its ends to the ends of the redundant link to be removed. The result is an  $r$ -regular network where each node has  $r$  distinct neighbours.

1) *Propagating a single chunk*: As the number of peers  $n$  increases, the furthest distance  $D$  between any two nodes in an  $r$ -regular graph has been shown to lie between the values

$$\begin{aligned} D_{\min} &= \lfloor \log_{r-1} n \rfloor + \lfloor \log_{r-1} \log n - \log_{r-1} (6r/(r-2)) \rfloor + 1 \\ D_{\max} &= \lceil \log_{r-1} (2rn \log n) \rceil \end{aligned} \quad (1)$$

with high probability [10]. For example if  $n = 500$  and  $r = 4$ ,  $5 \leq D \leq 10$ . For practical purposes, the maximum distance between a peer and the server falls in a much narrower range: a sample of 2000 randomly generated graphs gave distances between 7 and 8. This is, therefore, the time required for the server to broadcast a single chunk to the network.

### B. Stream Transmission

The chunks are numbered  $c_1, c_2, \dots$  according to their viewing order in the video stream. Each peer  $i$  stores these chunks in a buffer  $B_i$ . To begin with, we do not consider a playback deadline and assume an infinite buffer size. We represent  $b_i$ , the binary state of the  $i^{\text{th}}$  peer's buffer, by setting  $b_i(k) = 1$  if it contains  $c_k$ , and  $b_i(k) = 0$  if it does not.

To transmit the video stream  $S$ , we first divide it into  $r$  substreams. If the original stream consists of chunks  $S = \{c_1, c_2, c_3, \dots\}$ , then the  $i^{\text{th}}$  substream  $S_i$  will be

$$S_i = \{c_i, c_{r+i}, c_{2r+i}, \dots\} \quad (2)$$

The server designates one of these substreams to each of its  $r$  neighbours. Since the delay is equal on every link, we can further simplify the model by treating time as being slotted. At each time slot  $t = k$ , the server transmits the  $k^{\text{th}}$  chunk of each substream to its designated neighbour.

In any time slot a peer will receive up to  $r$  chunks from its neighbours. It must then decide what information to send to each neighbour in the following time slot. In the remainder of this paper we will analyse the performance of the network under three different data-driven push-based transmission strategies: the existing greedy and rarest-first strategies, as well as a novel network coding and decoding scheme.

## III. GREEDY AND RAREST FIRST

We begin our analysis by examining how the greedy and rarest first strategies perform under our network model. For simplicity, we assume that all peers have perfect information about the contents of their neighbours' buffers.

A peer using the *greedy* strategy sends the chunk closest to the playback deadline that the neighbour has not already received. If a peer  $i$  and its neighbour  $j$  have buffers  $B_i$  and  $B_j$  respectively,  $i$  will send  $j$  the chunk  $c_k$ , where

$$\min k : b_i(k) \overline{b_j(k)} = 1 \quad (3)$$

The *rarest first* strategy does the opposite, selecting the chunk corresponding to

$$\max k : b_i(k) \overline{b_j(k)} = 1 \quad (4)$$

### A. Substream Contiguity

It is important to note that under our network model, both schemes result in the *contiguous* transmission of substreams to every peer. This means that at time  $t$  and for every peer  $i$  and every substream  $s$ , there is some integer  $m_{i,s}(t)$  for which

$$b_i(kr + s) = \begin{cases} 1, & 0 \leq k \leq m_{i,s}(t) \\ 0, & k > m_{i,s}(t) \end{cases} \quad (5)$$

We use induction to prove this is always the case. At time  $t = 0$  it is certainly true since no peer has received any chunks and by (2) the server also maintains substream contiguity. Under the greedy strategy, if all peers in the network already satisfy (5) at time  $t$ , then by (3) the next chunk chosen for transmission to any peer  $i$  must be chunk  $m_{i,s}(t) + 1$  for one of its substreams, unless none of its neighbours has the next chunk of some substream and the buffer state of peer  $i$  remains unchanged. This will still satisfy (5), therefore substream contiguity will be maintained under the greedy strategy.

For the rarest first scheme we also prove by induction that a link between two peers remains exclusive or dedicated forever to the transmission of single substream  $s$ . That is, if at time  $t$  peer  $i$  uses link  $l_{i,j}$  for the first time to send chunk  $c_s$  to peer  $j$ ,

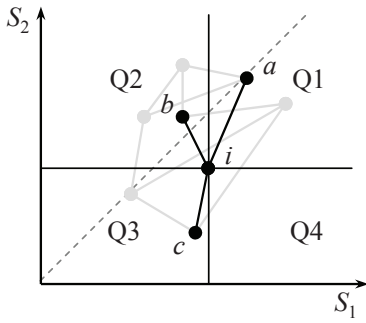


Fig. 1. Mapping peers by their buffer status shows how effectively the peers' bandwidth can be utilised. For peer  $i$ , neighbour  $a$  can only send chunks, while  $c$  can only receive. Therefore both  $a$  and  $c$  waste half their bandwidth.

then at time  $t+1$  it will send chunk  $c_{s+r}$  on the same link and so on. This is certainly true at  $t=0$ , since the links between the server and its neighbours have been exclusive and no other link has been used in the network. The induction claim is that if all substreams of every peer have been contiguous up to  $t$  and all links have been used exclusively for the transmission of substreams in the network, then this is also true for time  $t+1$ . For peer  $j$ , let  $\mathcal{R}_j^t$  be the subset of  $\{1, 2, \dots, r\}$ , for which  $m_{j,s}(t) \neq 0$  for each  $s \in \mathcal{R}_j^t$ . Hence, there exists a link  $l_{i,j}$  which has been exclusively used for the contiguous transmission of substream  $s$ :  $c_s, c_{r+s}, \dots, c_{m_{j,s}(t)r+s}$  to  $j$ . We first prove that for all  $s \in \mathcal{R}_j^t$ ,  $m_{j,s}(t) = m_{j,s}(t-1) + 1$ . That is, the next chunk of substream  $s$  must arrive at time  $t$  at peer  $j$ . Let us assume that this is not the case and there exists an  $s \in \mathcal{R}_j^t$  for which  $m_{j,s}(t) = m_{j,s}(t-1)$ . That is, link  $l_{i,j}$  has been *frozen*. Then no other substream  $s' \neq s$  at peer  $i$  can satisfy (4) due to the assumption of exclusiveness of  $l_{i,j}$  up to time  $t$ , which means that  $m_{i,s}(t) = m_{i,s}(t-1)$ . That is, the incoming link to peer  $i$  carrying substream  $s$  has been frozen too. Tracing this all the way back to the server leads to contradiction. Having proved  $m_{j,s}(t) = m_{j,s}(t-1) + 1$  for all  $s \in \mathcal{R}_j^t$ , we conclude that if substream  $s$  was exclusively transmitted up to time  $t$  by peer  $j$  to peer  $n_s$  via link  $l_{j,n_s}$ , substream contiguity and link exclusiveness is maintained at time  $t+1$  according to (4) on  $l_{j,n_s}$ . Since this is true for all peers  $j$ , the induction claim is true at time  $t+1$  for all nodes and links in the network.

### B. Network Performance

The contiguity property (5) greatly simplifies the study of data flow through the network, since we can assess their performance purely based on  $m_{i,s}(t)$ , the latest chunk that a peer  $i$  has received from each substream  $s$  by time  $t$ . The delay  $d_{i,s}(t)$  of substream  $s$  relative to the server at time  $t$  is

$$d_{i,s}(t) = t - m_{i,s}(t) \quad (6)$$

and the stream is continuous for the first  $m_i(t)r$  chunks, where

$$m_i(t) = \min(m_{i,1}(t), \dots, m_{i,r}(t)) \quad (7)$$

As  $t \rightarrow \infty$ , the overall throughput efficiency  $\mu$  is given by

$$\mu = \frac{1}{rt} \sum_{k=1}^r m_{i,k}(t) \quad (8)$$

At each time  $t$  it is useful to visualise the progress of each peer  $i$  in  $r$ -dimensional space, by mapping it to the point  $P(m_{i,1}(t), \dots, m_{i,r}(t))$ . The higher a peer's position in each dimension, the better its performance for the corresponding substream. For illustrative purposes, Fig. 1 depicts a possible network state for a somewhat unrealistic case where there are only two substreams,  $S_1$  and  $S_2$ .

A peer  $i$  can only transmit chunks from a substream  $s$  to their neighbour  $j$  if  $D_{i,j}(s) = 1$ , where

$$D_{i,j}(s) = I(m_{i,s}(t) > m_{j,s}(t)) \quad (9)$$

and  $I(\cdot) = 1$  if the inside statement is true and is otherwise 0. This is equivalent to dividing the  $r$ -dimensional space into  $2^r$  regions, each corresponding to a value of the  $r$ -dimensional binary vector  $\mathbf{D}_{i,j}$ . Any links to neighbours lying in the region for which  $\mathbf{D}_{i,j} = \mathbf{0}$ , which in Fig. 1 is given by Q3, cannot be utilised to transmit information and results in a loss of efficiency. The higher the probability of a peer's neighbours lying in Q1 or Q3 relative to one another, the more likely it is that these links will be underutilised. Therefore by examining how the selection strategies affect the peers' distribution in  $r$ -dimensional space, we can determine how the different data transmission strategies affect the performance of the network.

The greedy strategy selects the chunk that will increment (7) most effectively. This causes peers to preferentially equalise  $m_{j,s}$  values for every neighbour  $j$  wherever possible, pulling them towards the diagonal line  $(k, \dots, k)$ . However, if all peers in the network behave in this manner, it dramatically increases the probability of any peer  $i$  lying in the region  $\mathbf{D}_{i,j} = \mathbf{0}$  relative to its neighbour. This is captured in Fig. 2, which shows a trail of badly performing peers dotted along the diagonal line, those that have been caught in the low efficiency configuration. Over time, this tail lags further and further behind the server, affecting more and more of the peers.

By contrast, the rarest first strategy performs relatively well for the majority of peers who receive the full transmission rate with a fixed delay from the server. However, in Fig. 2 we observe a clump of yellow-coloured peers who have not yet received one of the streams. This occurs because the rarest first strategy dictates that a peer selects a single direction to pull each neighbour in, making no attempt to minimise the difference between  $m_{j,s}(t)$  values for different substreams. This works well under the assumption that the remaining streams can be obtained from other neighbours. But situations commonly arise where two neighbours wait indefinitely, each hoping to obtain the final missing substream from the other.

## IV. NETWORK CODING

In this section we demonstrate how, in contrast to the greedy and rarest first schemes, using network coding can give maximum throughput and minimal delay. On the surface, our coding scheme appears to be similar to existing schemes. However, coding and decoding are fundamentally different from network coding based solutions such as [6], [12] in that 1) peers are not restricted a priori to coding within predefined sections of the video stream, 2) a peer makes no attempt to

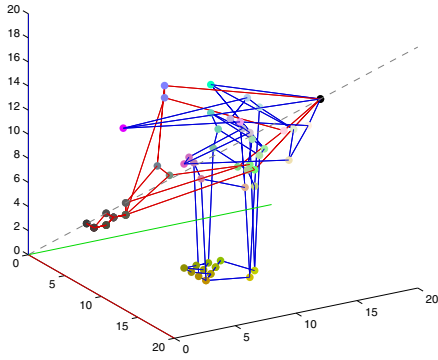


Fig. 2. A comparison of peers' buffer maps under the greedy and rarest first strategies for a network with  $n = 40$ ,  $r = 3$ , at  $t = 15$ . The black point represents the state of the server. Greedy is marked in red and rarest first in blue. Peers are coloured by their position in each substream.

decode received chunks before relaying them to its neighbours and 3) the decoding fully exploits the network structure and coded contiguity, which is defined shortly. We will show that chunks can be received and decoded in optimal time while maintaining maximum throughput and a finite buffer size.

#### A. Transmission

In our *network coding* scheme each peer sends a linear combination of all chunks received from its neighbours in the previous time slot. At each time  $t$  a peer  $i$  receives the vector  $\mathbf{y}_i[t]$  of network coded chunks from its neighbours  $n_1, \dots, n_r$ . It uses this to determine the vector of chunks to be transmitted in the next time slot,

$$\mathbf{u}_i[t+1] = \alpha_i \mathbf{y}_i[t] \quad (10)$$

sending the  $k^{\text{th}}$  term of  $\mathbf{u}_i[t+1]$  to neighbour  $n_k$ . In the manner of [13], the  $(j, k)$  term of the  $r \times r$  matrix  $\alpha_i$  is the coefficient assigned to the connection from  $l_{n_k, i}$  to  $l_{i, n_j}$ . Since the network itself is static, we use static coefficients chosen uniformly at random from the field  $\mathbb{F}_q$ . The size required for this field will be discussed in Section IV-D. An additional term may also be included from the buffer, the function of which will be discussed later on.

#### B. Coded Contiguity

As it turns out, network coding exhibits its own form of contiguity. We say that network coding maintains *coded contiguity* if  $\mathbf{y}_i[t]$ , the vector of the  $r$  coded chunks received by some peer  $i$  at time  $t$ , can be expressed as

$$\mathbf{y}_i[t] = \sum_{k=0}^t \mathbf{A}_{i,k} \mathbf{x}[t-k] \quad (11)$$

where  $\mathbf{x}[k]$  is a vector of the  $k^{\text{th}}$  chunk from each substream, and  $\mathbf{A}_{i,k}$  is the matrix of coefficients for any  $\mathbf{x}[0]$  terms in the chunks received at time  $k$ . The received chunks are contiguous in the sense that subsequent chunks hold identical information about subsequent chunks from each of the streams, which is clear from rewriting (11) in the form

$$\mathbf{y}_i[t+1] = \mathbf{A}_{i,t+1} \mathbf{x}[0] + \sum_{k=0}^t \mathbf{A}_{i,k} \mathbf{x}[t-k+1] \quad (12)$$

It is interesting to note the resemblance that (11) bears with convolution. This is because the network links essentially act as a linear channel between the server and the peers.

We can prove that all transmissions under our network coding scheme take the form of (11). At  $t = 0$  this is certainly true, with  $\mathbf{A}_{i,0} = \mathbf{I}_r$  for the server and  $\mathbf{0}_{r \times r}$  for the peers. The chunks received by every peer  $i$  at time  $t+1$  are

$$\mathbf{y}_i[t+1] = \sum_{k=1}^r \beta_k \mathbf{u}_{n_k}[t+1] \quad (13)$$

where  $\beta_k$  selects the appropriate chunk within  $\mathbf{u}_{n_k}[t+1]$  from neighbour  $n_k$ . From (10) it is straightforward to prove that if all transmissions at time  $t$  can be written in the form of (11), then (13) results in an equation of the same form at time  $t+1$ .

#### C. Decodability

A peer cannot use any of the information they have received until they can decode the original chunks. Therefore, it is important that the received chunks can be decoded as quickly as possible. Here we will make use of the coded contiguity property to prove that optimal time decodability is possible under our network coding scheme.

We first define the lower bound on the playback delay that we are trying to achieve. For a peer  $i$  we take the minimum playback delay  $E_i$  to be the first time that the full streaming rate  $r$  is possible. This corresponds to the first time that the minimum cut between the server and peer  $i$  is achievable. We assume that the field size  $q$  is large enough that edge disjoint paths provide linearly independent information. Therefore,  $E_i$  is the first time all  $r$  neighbours of peer  $i$  transmit coded chunks that are linearly independent of all other chunks the peer has received up to that time.

From (12) it is clear that if peer  $i$  can solve  $\mathbf{x}[0]$  after receiving coded chunks  $\mathbf{y}_i[0], \dots, \mathbf{y}_i[E_i]$ , it can solve any subsequent  $\mathbf{x}[k]$  after receiving  $\mathbf{y}_i[0], \dots, \mathbf{y}_i[E_i+k]$ . To prove that continuous playback is possible at time  $E_i$ , it suffices to show that by this time peer  $i$  can obtain  $r$  independent equations describing  $\mathbf{x}[0]$ , one from each of its  $r$  neighbours.

To prove this we first need to provide a few definitions. For some time  $0 \leq t \leq E_i$ , the *contributors* of peer  $i$  are the set of its neighbours who have started providing information that is linearly independent of all other neighbours in that set. The neighbours are ordered according to the time they first become a contributor so that the first  $r^{(t)}$  neighbours have contributed by time  $t$ , and the remainder are noncontributors. A chunk received from any of the noncontributors is described as *independent* if it is linearly independent of all chunks received from all identified contributors up to and including that time. So if we use the matrices

$$\mathbf{K}^{(t)} = \left[ \begin{array}{c|c} \mathbf{0}_{r^{(t)} \times r^{(t)}} & \mathbf{0}_{r^{(t)} \times r - r^{(t)}} \\ \hline \mathbf{0}_{r - r^{(t)} \times r^{(t)}} & \mathbf{I}_{r - r^{(t)}} \end{array} \right] \quad (14)$$

$$\mathbf{L}^{(t)} = \mathbf{I}_{r \times r} - \mathbf{K}^{(t)} \quad (15)$$

to isolate the chunks received from noncontributing and contributing neighbours respectively, we can describe the relationship between contributors and noncontributors as follows.

*Definition:* At time  $0 \leq t \leq E_i$ , the relationship between  $r^{(t)}$  contributors and  $r - r^{(t)}$  noncontributors is defined through

$$\mathbf{K}^{(t)}\mathbf{y}_i[t] - f_t(\mathbf{L}^{(t)}\mathbf{y}_i[0], \dots, \mathbf{L}^{(t)}\mathbf{y}_i[t]) = \mathbf{0}_{r \times 1} \quad (16)$$

where  $f_t(\mathbf{z}_0, \dots, \mathbf{z}_t)$  is a unique function defined through  $k$  unique  $r \times r$  matrices  $\lambda_k$ , as follows

$$f_t(\mathbf{z}_0, \dots, \mathbf{z}_t) = \sum_{k=0}^t \lambda_k \mathbf{z}_k \quad (17)$$

*Claim:* At each time  $t$ , every new contributor provides a new, linearly independent description of  $\mathbf{x}[0]$ .

*Proof:* At  $t = 0$ , (11) gives  $r^{(0)} = \text{rank}(\mathbf{A}_{i,0})$  independent equations describing  $\mathbf{x}[0]$  from the first  $r^{(0)}$  neighbours, our starting set of contributors. By (16)

$$\mathbf{K}^{(0)}\mathbf{y}_i[0] - f_0(\mathbf{L}^{(0)}\mathbf{y}_i[0]) = \mathbf{0}_{r \times 1} \quad (18)$$

If we take (16) and replace all  $\mathbf{y}_i[k]$  with  $\mathbf{y}_i[k+1]$ , applying (12) we find that at time  $t+1$

$$\begin{aligned} & \mathbf{K}^{(t)}\mathbf{y}_i[t+1] - f_t(\mathbf{L}^{(t)}\mathbf{y}_i[1], \dots, \mathbf{L}^{(t)}\mathbf{y}_i[t+1]) = \\ & \mathbf{K}^{(t)}\mathbf{A}_{i,t+1}\mathbf{x}[0] - f_t(\mathbf{L}^{(t)}\mathbf{A}_{i,1}\mathbf{x}[0], \dots, \mathbf{L}^{(t)}\mathbf{A}_{i,t+1}\mathbf{x}[0]) \\ & \triangleq \mathbf{K}^{(t)}\mathbf{A}_i^{(t+1)}\mathbf{x}[0] \end{aligned} \quad (19)$$

We first set  $r^{(t+1)} = r^{(t)}$ . For each noncontributor  $k > r^{(t)}$ , there are now two cases to consider. If it has transmitted an independent chunk, then it cannot be expressed in terms of chunks from the contributors so far. Since all expressions for  $\mathbf{x}[0]$  have so far been derived from the chunks of the contributors, it follows that this new expression must be linearly independent of all previous expressions of  $\mathbf{x}[0]$  obtained. Therefore,  $k$  is a new contributor and we can increment  $r^{(t+1)}$  by 1. If  $k$  did not transmit an independent chunk, it means that it can be expressed in terms of the chunks received from the previous contributors and  $r^{(t+1)}$  does not change. ■

At time  $E_i$ ,  $r^{(E_i)} = r$  for peer  $i$ , which is the minimum delay possible dictated by the network. Let us define  $E_{\max} = \max_i \{E_i\}$ . Therefore, at time  $E_{\max}$ ,  $r^{(E_{\max})} = r$  for all peers in the network. Therefore, all streams will be decodable within  $E_{\max}$  time slots of being transmitted by the server.

#### D. Buffer and Field Size

We must also consider the buffer size that peers will require to decode all chunks. Since all peers can receive and decode each chunk within  $E_{\max}$  time units of transmission by the server, the playback delay can be set to  $E_{\max}$  and the buffer must store at least  $rE_{\max}$  chunks. For this to be sufficient, peers need to ensure that no chunks more than  $E_{\max}$  time units old are included from each substream. At every time  $t \geq E_{\max}$ , a peer eliminates the newly decoded chunk  $\mathbf{x}[t - E_{\max}]$  from the coded chunks it transmits to its neighbours.

To ensure that the full streaming rate can be achieved by time  $E_{\max}$ , the field size  $q$  must be large enough to give high probability of linear independence between the  $r$  streams. For example, for 100 randomly generated networks of size  $n = 100$  nodes and  $r = 4$ , we found through simulations that with a field size of  $q = 2^8$ , all nodes were able to decode  $\mathbf{x}[0]$  after 8 time slots.

## V. CONCLUSION AND FUTURE WORK

In this paper we have introduced a homogeneous  $r$ -regular network model for a peer-to-peer video broadcasting network. These networks are practical to consider, having a number of good properties such as fairness and simple construction. In the context of this model we showed that the greedy strategy is fundamentally flawed in that, while it ensures continuous playback for all peers, it results in low bandwidth efficiency by reducing the probability that links in the network can be used to transmit new data. By comparison, the rarest first strategy was found to give optimal throughput for the majority of peers, but a subset could never achieve continuous playback. We showed that instead, network coding could be utilised to achieve optimal playback rate and minimum delay.

The next step is to find whether it is possible to extend this model to take into account more features of real networks, such as variable bandwidth and delay, churn and fluctuating network size. If our new network coding scheme can be adapted to perform well under these conditions, it could lead the way towards more efficient, lower delay peer-to-peer video broadcast over the Internet.

## ACKNOWLEDGMENT

This work was supported under Australian Research Council Discovery Projects funding scheme (project no. DP0984950).

## REFERENCES

- [1] J. Liu, S. Rao, B. Li, and H. Zhang, "Opportunities and challenges of peer-to-peer internet video broadcast," *Proceedings of the IEEE*, vol. 96, no. 1, pp. 11–24, 2008.
- [2] K. Sripanidkulchai, A. Ganjam, B. Maggs, and H. Zhang, "The feasibility of supporting large-scale live streaming applications with dynamic application end-points," *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 4, pp. 107–120, 2004.
- [3] M. Zhang, J. Luo, L. Zhao, and S. Yang, "A peer-to-peer network for live media streaming using a push-pull approach," in *Proc. of ACM Multimedia*, Nov. 2005, pp. 287–290.
- [4] Y. Zhou, D. M. Chiu, and J. Lui, "A simple model for analyzing P2P streaming protocols," in *Proc. IEEE Int. Conf. on Network Protocols (ICNP)*, Oct. 2007, pp. 226–235.
- [5] M. Zhang, C. Chen, Y. Xiong, Q. Zhang, and S. Yang, "Optimizing the throughput of data-driven based streaming in heterogeneous overlay network," *Advances in Multimedia Modeling*, vol. 4351/2006, pp. 475–484, 2006.
- [6] M. Wang and B. Li, " $R^2$ : Random push with random network coding in live peer-to-peer streaming," *IEEE J. on Selected Areas in Commun.*, vol. 25, no. 9, pp. 1655–1666, Dec. 2007.
- [7] T. Ho, M. Médard, J. Shi, M. Effros, and D. Karger, "On randomized network coding," in *Proc. of the Annual Allerton Conference on Communication Control and Computing*, vol. 41, 2003, pp. 11–20.
- [8] V. Bourassa and F. Holt, "SWAN: Small-world wide area networks," in *Proc. of Int. Conf. on Advances in Infrastructures (SSGRR 2003w)*, L'Aquila, Italy, 2003.
- [9] J. Jiang, S. Zhang, M. Chen, and M. Chiang, "Minimizing streaming delay in homogeneous peer-to-peer networks," in *IEEE Intl. Symp. Info. Theory (ISIT)*, Jun. 2010, pp. 1783–1787.
- [10] B. Bollobás, *Random graphs*. Cambridge Univ Press, 2001.
- [11] M. E. J. Newman, "The structure and function of complex networks," *SIAM Review*, vol. 45, no. 2, pp. 167–256, 2003.
- [12] P. Chou, Y. Wu, and K. Jain, "Practical network coding," in *Proc. of the Annual Allerton Conference on Communication Control and Computing*, vol. 41, 2003, pp. 40–49.
- [13] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 782–795, Oct. 2003.