

Abstract—In this paper, we demonstrate a network coded transmission scheme that allows the implementation of multicast video on demand using a single server channel. A receiver is permitted to join the broadcast at any time, where after an initial startup delay, it is able to deliver packets at its channel rate. This transmission scheme is the first to allow in-order packet delivery while providing innovative information to all receivers at all times. We illustrate how this feature can be achieved by taking advantage of the early decoding properties of an existing network coding transmission scheme.

I. INTRODUCTION

Much research has been devoted to the implementation of video on demand [1]. Solutions must allow the video to be viewable from the beginning in a timely manner, and ensure that the packets of the video are delivered from the beginning in the correct order. High bandwidth efficiency is also an important feature.

Numerous methods for implementing multicast video on demand have been proposed in the literature [1]. Video on demand systems generally make use of a number of server channels to distribute each video.

In server initiated algorithms, for example [2], [3], the video is split up across different server channels. Each server channel sequentially transmits the packets in their designated portion of the video, looping back to the start once it reaches the end of its portion. Receivers are able to view the video by subscribing to the appropriate channels. Any number of receivers can subscribe to a given video, however upon joining, receivers often experience a startup delay, as they must wait until the relevant server channel loops back to the start of the stream before they can begin watching the video.

In contrast, client initiated algorithms such as [4], [5] assign server channels dynamically, according to users’ requests for videos. Making use of batching techniques, these client initiated algorithms can provide instantaneous playback, as seen in [4], however the number of channels required to support this service increases with the number of users.

In this paper, we will demonstrate that, if receivers are willing to tolerate some delay, multicast video on demand can be implemented using only a single server channel. By taking advantage of the heterogeneity of different receivers’ packet losses, it is possible to serve multiple receivers through a single channel. To our knowledge, all multicast video on demand solutions proposed in the literature rely on at least two server channels, but often more, for implementation.

A. Network coding

Our methods are based on linear network coding (network coding, for short), first introduced in [6]. Under network coding, rather than transmitting one packet at a time, the sender combines multiple packets together for transmission using Galois field arithmetic [7]. Each packet is multiplied by a chosen coding coefficient, before being summed together to form a network coded packet. These coefficients are usually transmitted along with the packet, in the manner of [8]. As a result, network coding has the advantage of being able to simultaneously provide innovative information to a number of receivers, even if they are waiting for information about different packets.

Broadly speaking, network coding strategies can be classified in one of two ways. Under random linear network coding (RLNC), the sender selects the coding coefficients at random from a large field size. This has been shown [9] to provide new information to all receivers, with high probability. Although it allows for extremely efficient, low overhead transmission, one weakness of RLNC is that a receiver cannot decode any new packets until it has received sufficient information to decode all packets transmitted by the sender so far. Deterministic network coding on the other hand relies on feedback from the receivers to allow the sender to determine the network coding coefficients. Although this is more computationally expensive for the sender, some deterministic network coding schemes allow receivers to decode sooner than is permitted under RLNC [10]. We believe this paper is the first to exploit early decoding to achieve full rate packet delivery.

B. Related work and Contributions

In this paper we demonstrate that it is possible to use network coding to provide video on demand for a video stream using a single multicast server channel. We analyse an adaptation of the network coding scheme introduced in [11], and show how early decoding can allow all receivers to deliver packets at full rate, even if they join the broadcast at different times. This transmission scheme has the distinction of being transmission optimal, meaning that every received transmission provides innovative information to every receiver.

In [12], we showed that, using deterministic network coding, receivers can consistently achieve early decoding. It was demonstrated that under a lossy single hop network, the same set of packets could be simultaneously delivered to a set of non-uniform receivers at multiple rates. However, the assumptions that were required in [12] meant that results were limited to the case where receivers had strictly different channel rates and joined the transmission at the same time. It did not make any speculations on the possibility of joining the broadcast at different times, nor what kind of delivery
II. SYSTEM MODEL

A single sender aims to transmit an infinite backlog of data packets \(p_1, p_2, \ldots\) in the correct order to a set of \(R\) receivers, \(R_x_1, R_x_2, \ldots, R_x_R\). Time is slotted, denoted by \(t = 1, 2, \ldots\) and the sender broadcasts packets at the rate of one network coded packet per time slot. The receivers are connected to the sender via independent erasure channels with channel rate \(\mu\), so they successfully receive transmissions independently with probability \(\mu\) at each time slot. Each receiver \(R_x_j\) joins the broadcast at time \(t_j\). Without loss of generality we set \(t_1 = 0\), and order the receivers so that \(t_1 \leq t_2 \leq \cdots \leq t_R\).

Each receiver stores the packets it receives, along with the corresponding coding coefficients, in its own buffer. To decode the original packets, receivers apply Gaussian elimination \([7]\), using information from the coding coefficients to recover the original uncoded packets. After each packet transmission, the receiver sends an acknowledgement if the packet was successfully received, or a negative acknowledgement if the packet was discarded due to an erasure. For the purposes of this work we assume the sender detects these acknowledgements without error or delay. The sender uses this information to record which packets each receiver has stored in its buffer. Based on this information, a transmission scheme is used to determine the packet combinations the sender will transmit.

A. Definitions

Before we begin outlining the transmission scheme we will use, we provide a list of some of the terminology that will be essential throughout this paper.

\(\diamond\) The packet \(p_n\) has a packet index \(n\).
\(\diamond\) A packet \(p_n\) is newer than a packet \(p_m\) if its packet index is higher, i.e. \(n > m\). The packet \(p_n\) is older if its packet index is lower.
\(\diamond\) A packet is delivered to a receiver \(R_x_r\) when it and all older packets have been decoded.
\(\diamond\) A transmission is innovative to a receiver \(R_x_r\) if it provides new information about an undecoded packet.\(^1\)
\(\diamond\) At any time \(t\), the next needed packet \(N_n(t)\) of a receiver \(R_x_r\) is the oldest packet \(p_n\) which has not been delivered to that receiver.
\(\diamond\) A receiver \(R_x_r\) is faster than another receiver \(R_x_s\) if its next needed packet is newer. \(R_x_s\) is then said to be slower than \(R_x_r\).\(^2\)
\(\diamond\) At a given time \(t\), the fastest receiver is called the leader.

B. Transmission scheme

The sender uses Algorithm 1, adapted from \([11]\),\(^3\) to determine each transmission \(s(t)\). At each time slot, the sender orders the receivers from fastest to slowest. Starting with

\(^1\)In other words, it is linearly independent of previously received packets.

\(^2\)Note that due to fluctuations in the receivers’ channel conditions, their relative progress may change over the course of the broadcast. This differs from \([12]\) where it is assumed that the receivers maintain a fixed ordering from fastest to slowest.

\(^3\)Note that, unlike \([11]\), we do not take the usual step of restricting the sender’s transmission rate to be less than the channel rate.

\(s(t) = 0\), the sender transmission is sequentially updated to ensure that \(s(t)\) is innovative to each receiver. If the transmission so far is non-innovative to a particular receiver \(R_x_r\), \(s(t)\) is updated to include that receiver’s next needed packet. Packets are only added if they are necessary to ensure \(s(t)\) is innovative, so Algorithm 1 ensures that the sender transmits parsimonious combinations of the next needed packets of the receivers.

Algorithm 1 has a few simple properties which will be useful throughout this paper.

Property 1. Each transmission \(s(t)\) is coded from the next needed packets of the receivers.

Property 2. The next needed packet of a receiver \(R_x_r\) is only coded into \(s(t)\) if it allows \(R_x_r\) to decode the packet from \(s(t)\) upon successful reception.

Property 3. The next needed packet of the leader will always be coded into each transmission.

Algorithm 1 Coding algorithm based on \([11]\).

1: Organise receivers \(R_x_m, \ldots, R_x_R\) into groups \(G_j\), so that \(G_j\) contains all receivers \(R_x_i\) whose next needed packet \(N_i(t) = p_j\). Let \(G\) be the set of all groups \(G_j\) which contain at least one receiver.
2: Initialise \(s(t) = 0\).
3: for each group \(G_j \in G\), from high to low \(j\), do
4: Initialise the empty veto list \(v_j = \{\}\).
5: for each receiver \(R_x_i \in G_j\) do
6: Calculate \(r_i\), the result of performing Gaussian elimination on \(s(t)\) with the packet in \(R_x_i\)'s buffer.
7: if \(r_i = 0\) then
8: Add 0 to \(v_j\).
9: else if \(r_i = \alpha p_j\) for some field element \(\alpha\) then
10: Add \(\alpha\) to \(v_j\).
11: end if
12: end for
13: if 0 \(\in v_j\) then
14: Set \(a_j\) to the smallest value not listed in \(v_j\).
15: Set \(s(t) = s(t) + a_j p_j\).
16: end if
17: end for

C. Buffer model

To help the reader understand the packet delivery process, we make use of a buffer model to describe which packet combinations are known by each of the receivers. Imagine that each receiver has an infinite buffer that has one space allocated for each packet in the sender’s backlog. The \(n^{th}\) position of this buffer corresponds to \(p_n\) in the sender’s backlog, as illustrated in Fig. 1.

\(\diamond\) If position \(n\) in a receiver’s buffer is filled, then the corresponding packet \(p_n\) is said to have been seen by that receiver.\(^4\)

Gaussian elimination is performed on the received packets as they are stored in the receivers’ buffers. Each packet is

\(^4\)The concept of seen packets is inspired by \([11]\).
stored in a receiver’s buffer according to the newest packet coded into the transmission, after Gaussian elimination has been applied. One useful property of this buffer model is that once packets $p_1, \ldots, p_n$ have been seen by a receiver, they have also been delivered. This is the case because once a receiver has obtained $n$ coded combinations of $n$ unknown packets, it has a sufficient amount of information to decode all packets.

D. Organisation of the paper

In the remainder of this paper, we will study the packet delivery behaviour of the receivers under the transmission scheme we have outlined. To achieve this, we divide the problem into two parts. In Section III, we estimate the probability of knowledge differential transmissions; the means by which early packet delivery is made possible. Based on these results, in Section IV we find both the delivery rate and the expected number of undelivered packets at each receiver.

III. Knowledge differential transmissions

In this section we will estimate the probability of knowledge differential transmissions for any given receiver.

\( \diamond \) For a receiver $R_x$, a transmission $s(t)$ is classified as a knowledge differential transmission if $R_x$ can decode its next needed packet, $N_r(t)$, from $s(t)$.

A. Approximations

The behaviour of Algorithm 1 is highly complex, due to the presence of feedback-induced recursive dependencies among receivers. To obtain tractable and intuitively useful results, it is necessary to make some simplifying approximations.

In practice, any number of network coded combinations could potentially be stored in the receivers’ buffers. However we can simplify our representation of the receivers’ buffers by making the following approximation.

**Approximation 1.** All packets stored in a receiver’s buffer are uncoded.

This statement is in fact true for the vast majority of packets stored in the receiver’s buffer. Algorithm 1 ensures that the first transmission of a new packet will be uncoded, therefore, at least $\mu$ of the sender’s transmissions must be uncoded. Furthermore, Gaussian elimination applied inside the receivers’ buffers allows the decoding of some of the coded transmissions received. As a consequence of Approximation 1, we obtain the following property.

**Property 4.** A receiver $R_x$ can decode its next needed packet $N_r(t)$ from $s(t)$ if they have seen all other packets coded into $s(t)$.

It is not immediately clear how likely it is for a packet to have been seen, since sender transmissions can be stored nearly anywhere in a receiver’s buffer, depending on the effects of Gaussian elimination. Therefore, we introduce Approximation 2.

\( \diamond \) Let $p_n$ be the newest packet coded into a transmission $s(t)$. Then $s(t)$ is said to be a leader transmission of the packet $p_n$.

**Approximation 2.** A receiver $R_x$, has seen a packet $p_n$, iff $R_x$, has received a leader transmission of the packet $p_n$.

If $p_n$, the newest packet coded into $s(t)$, has not yet been seen, then it will be stored at position $n$. This is the most typical scenario, accounting for more than $\mu$ of the received transmissions. Using Approximation 2, the likelihood of a packet being seen then depends on the number of leader transmissions of the packet.

The two approximations we have introduced allow us to determine a receiver’s probability of a knowledge differential transmission by answering three questions. Firstly, what is the probability that there have been $k$ leader transmissions of a packet? Secondly, given this information, what is the probability that a packet has been seen? And finally, what is the likelihood that a particular set of packets is coded into $s(t)$? We will answer each of these questions in Sections III-B to III-D, using the results to find the probability of a knowledge differential transmission in Section III-E.

B. Leader transmissions

Our first task is to determine how many leader transmissions there have been of any given packet in the buffer. By Property 3, $s(t)$ is a leader transmission of the current leader $Rx_\mu$’s next needed packet, $N_l(t)$. It follows that leader transmissions of a packet $p_n$ only occur while $N_l(t) = p_n$. The following property explains how $N_l(t)$ evolves over time.\(^5\)

**Property 5.** The leader $Rx_\mu$’s next needed packet $N_l(t)$ remains the same while $Rx_\mu$ is in erasure, but will progress to the next packet every time the leader receives a packet.

We can prove Property 5 as follows. The leader can always decode its next needed packet from $s(t)$, since it has already delivered all other packets which might be coded into $s(t)$. Therefore the leader’s next needed packet remains the same while it is in erasure (since it receives no new information), but will progress with every successful reception. The leader’s next needed packet is always the newest packet to have been transmitted by the sender so far, so it is not possible for the leader to ‘skip’ any next needed packets.

We can use Property 5 to determine the probability that there have been $k$ leader transmissions of a given packet $p_n$.

If $p_n$ is older than $N_l(t)$, this is the probability it took exactly $k$ leader transmissions of $p_n$ for it to be received,

$$L(k) = \begin{cases} 0, & k = 0, \\ \frac{1}{\mu^{k-1}}, & k > 0. \end{cases} \quad (1)$$

This is the probability that, when leader transmissions of $p_n$ began, $Rx_t$ experienced $k - 1$ erasures, followed by a

\(^5\)We assume here the leader is a fixed receiver which does not change. In reality it is possible for other receivers to take the place of the leader and affect the number of leader transmissions of a packet. However from simulations, we observe that the leader changes very infrequently, and therefore have a negligible effect on the observed performance.
successful reception. Note that \( k \) cannot be zero, since a packet must be transmitted at least once for \( \text{Rx}_1 \) to receive it.

On the other hand, if the packet \( p_n = N_1(t) \) is still in the process of being transmitted to \( \text{Rx}_1 \), then the likelihood \( L^s(k) \) of there having been \( k \) transmissions of \( p_n \) so far is given by

\[
L^s(k) = \mu^k.
\]

This is the likelihood that the last change of next needed packet occurred exactly \( k \) time slots ago.

Packets \( p_n \) that are newer than \( N_1(t) \) have not yet been transmitted by the sender, so in this case \( k = 0 \).

C. Probability that a packet is seen

We now find the probability that a packet \( p_n \), corresponding to the next needed packet of another receiver, has been seen by a receiver \( \text{Rx}_r \) who is not the leader. \( D_s, D_f \) and \( D_l \) denote the probability of having seen the next needed packet of a slower, faster and leading receiver respectively.

Figures 3 and 4 illustrate how uncommon it is for two receivers to have exactly the same next needed packet. We therefore introduce Approximation 3.

**Approximation 3.** Each receiver \( \text{Rx}_r \)'s next needed packet \( N_r(t) \) is distinct.

If \( p_n \) is the next needed packet of a slower receiver, then \( \text{Rx}_r \) has already delivered, and therefore seen, the packet. So

\[
D_s = 1. \tag{3}
\]

By Approximation 2, undelivered packets can only be seen from leader transmissions. Therefore, the probability that a packet \( p_n \) has been seen by \( \text{Rx}_r \) is given by the probability that they received at least one of the leader transmissions of the packet.

If \( p_n \) is the next needed packet of the leader,

\[
D_l = \sum_{k=0}^{\infty} (1 - \bar{\mu}^k)L^s(k) = \frac{\mu}{1 + \mu}. \tag{4}
\]

If \( p_n \) is the next needed packet of a faster non-leading receiver, then

\[
D_f = \sum_{k=0}^{\infty} (1 - \bar{\mu}^k)L(k) = \frac{1}{1 + \mu}. \tag{5}
\]

D. Codings

From Property 4, the probability of a knowledge differential transmission depends on which other packets are coded into \( s(t) \). This is represented using a coding \( C \), defined as follows.

The **coding** \( C \) is a binary vector of length \( R \) that indicates which of the receivers’ next needed packets, **ordered from fastest receiver to slowest**, have been coded into the transmission \( s(t) \). The \( r \)-th element of \( C \), \( C(r) = 1 \) if the \( r \)-th fastest receiver’s next needed packet is coded into the transmission \( s(t) \); otherwise it is 0.\(^7\)

From Property 3 we can establish that

\[
\Pr(C(1) = 1) = 1. \tag{6}
\]

In practice, the probability that a given packet \( p_n \) has been seen is correlated among receivers. For example, \( p_n \) is more likely to have been seen by receivers if there were four leader transmissions of \( p_n \), compared with only one. However this correlation makes coding probabilities more difficult to calculate. Approximation 4 simplifies our calculations, in exchange for a small loss of accuracy.

**Approximation 4.** The probability of a packet \( p_n \) being seen is independent for each receiver.

Using Approximation 4, the probability that a coded packet \( p_n \) has been seen by a non-leading receiver \( \text{Rx}_r \) depends only on which receiver’s next needed packet it is. If it is the next needed packet of the leader, the probability is \( D_l \). If \( p_n \) belongs to a faster receiver who is not the leader, the probability is \( D_f \). If \( p_n \) belongs to a slower receiver, then it has already been seen, since \( D_s = 1 \).

Say that \( C_{r-1} \), the first \( r - 1 \) elements of \( C \), are already determined. Then let \( F_r = \sum_{i=3}^{r} C(i) \) denote how many of the \( r \)-fastest receivers’ next needed packets are coded into \( s(t) \), excluding the leader. Making use of (6), the likelihood that the \( r \)-th fastest receiver is coded into \( s(t) \) can be written as

\[
\Pr(C(r) = 1|C_{r-1}) = D_l D_f^{F_r-1}. \tag{7}
\]

Combining (6) and (7), we can determine the probability of a coding \( C \),

\[
\Pr(C) = \left( \prod_{r \in v_1(C)} D_l D_f^{F_r-1} \right) \left( \prod_{r \in v_0(C)} (1 - D_l D_f^{F_r-1}) \right), \tag{8}
\]

where \( v_1(C) \) is the set of values \( r \) for which \( C(r) = 1 \), and \( v_0(C) \) is the set of values \( r \) for which \( C(r) = 0 \).

E. Probability of a knowledge differential transmission

We now derive a receiver \( \text{Rx}_r \)'s probability of a knowledge differential transmission \( K_r \).

By Property 2, knowledge differential transmissions almost always result from the receiver’s next needed packet being included in \( s(t) \). To exclude the small possibility of knowledge differential transmissions occurring by other means, we introduce Approximation 5.

**Approximation 5.** A knowledge differential transmission occurs for a receiver \( \text{Rx}_r \) iff its next needed packet \( N_r(t) \) is coded into \( s(t) \).

Then the probability a transmission \( s(t) \) is a knowledge differential transmission for the \( r \)-th fastest receiver is given by the likelihood that \( C(r) = 1 \),

\[
K_r = \sum_{C} C(r) \Pr(C). \tag{9}
\]

Note that this equation differs from a similar result in [12].

\(^7\)Note that we still make use of Approximation 3, so receivers’ next needed packets are assumed to be distinct.

\(^8\)Theoretically it is also possible for a next needed packet to be decoded via Gaussian elimination, however this probability is extremely small.
in that we do not assume any fixed ordering of the receivers from fastest to slowest.

IV. Delivery Rate Analysis

In this section, we find the receivers’ expected delivery rates. We can divide packet delivery into two stages: the startup phase and delivery phase. Fig. 2 illustrates a receiver’s buffer storage distribution during each of these phases.

Startup phase: The startup phase applies to non-leading receivers. During the startup phase, a receiver Rx_l first joins the broadcast with an empty buffer. However the sender’s transmissions will allow the receiver to see some of the next needed packets of the leader. As a result of receiving these transmissions, at any time \( t > t_r \), a fraction of the packets in the range \( N_l(t_r), \ldots, N_l(t) \) will have been seen by \( Rx_l \).

In addition to this, the receiver will occasionally receive knowledge differential transmissions, which allow the receiver to slowly begin delivering its next needed packets. Higher delivery rates are only possible once the receiver is close to completing the startup phase. Once the receiver has delivered all the packets up to \( N_l(t_r) \), the startup phase is complete. Fig. 3 illustrates the small, nonzero delivery rate experienced by receivers during the startup phase.

Delivery phase: The delivery phase begins once \( Rx_l \) has delivered all packets up to \( N_l(t_r) \). The receiver is still able to make use of knowledge differential transmissions, but seen packets already stored in its buffer allow the \( Rx_l \) to deliver packets at a higher rate than during the startup phase.

A. Delivery phase analysis

We now calculate the average delivery rate of the \( r \)th fastest receiver during the delivery phase. All sender transmissions can be classified as either:

- Knowledge differential transmissions, which account for \( K_r \) of the sender’s transmissions. These deliver the receiver’s next needed packets.
- Non-knowledge differential transmissions, which account for the remaining \( 1 - K_r \) fraction of the sender’s transmissions. These transmissions are distributed evenly through the shaded portion of the buffer, as illustrated in Fig. 2 (b) and (c).

Non-knowledge differential transmissions are stored uniformly across a receiver’s buffer, corresponding to the shaded regions in Fig. 2. Knowledge differential transmissions deliver the receiver’s next needed packets, filling in any missing packets from oldest to newest. Therefore, the delivery rate is given by the average rate at which knowledge differential transmissions can fill in the packets which were missed by non-knowledge differential transmissions.

The buffer density is the average fraction of packets in the shaded region that have been seen from non-knowledge differential transmissions. An average of \( \mu(1 - K_r) \) non-knowledge differential transmissions are received per unit time. Since the leader progresses at rate \( \mu \), an average of \( 1/\mu \) time slots will be dedicated to each position in the buffer. Therefore, the buffer density is given by

\[
B_r = 1 - K_r. \tag{10}
\]

On average, \( \mu K_r \) knowledge differential transmissions are received per time slot. Therefore, the delivery rate of the \( r \)th fastest receiver is given by

\[
P_r^d = \frac{\mu K_r}{1 - B_r} = \mu. \tag{11}
\]

We therefore come to the interesting conclusion that, so long as there is a nonzero probability of knowledge differential transmissions, the receiver’s average delivery rate during the delivery phase will match that of the leader. In Fig. 3, once receivers have delivered up to \( N_l(t_r) \), the delivery phase begins. A sudden increase in the delivery rate is observed, and receivers’ delivery progress runs roughly parallel to that of the leader. Note however that packet delivery is not always smooth, since packet delivery is dependent upon sometimes rare knowledge differential transmissions.
Markov state is packet in the buffer density region has been seen. If a receiver’s $\mu$ at the leader and any other receiver can be predicted by its C. Expected number of undelivered packets causing slower receivers to overtake and replace the leader. At a receiver can fluctuate dramatically – in some cases even of how many packets are undelivered at a particular receiver. known, our analysis so far does not give us any indication

![Fig. 4. Packets delivered at the receivers as a function of time. $R = 4$, $\mu = 0.8$, $t_1 = 0$, $t_2 = 10$, $t_3 = 20$, $t_4 = 30$. We can observe that fluctuations in the receivers’ delivery rates can cause changes in the receivers’ ordering.](image)

B. Undelivered packets

Although the average delivery rate of the receivers is now known, our analysis so far does not give us any indication of how many packets are undelivered at a particular receiver. As we observed in Fig. 4, the number of undelivered packets at a receiver can fluctuate dramatically – in some cases even causing slower receivers to overtake and replace the leader.

C. Expected number of undelivered packets

The difference between the number of packets delivered at the leader and any other receiver can be predicted by its Markov state. A receiver $Rx_k$’s Markov state is the difference between the number of innovative packets stored at the leader and $Rx_k$.

From (5) we have an estimate of $D_f$, the probability that a packet in the buffer density region has been seen. If a receiver’s Markov state is $k$, then there are $k$ unseen packets in the buffer. This corresponds to the $1 - D_f$ fraction of seen packets. Therefore, if a receiver’s Markov state is $k$, we can expect that the number of undelivered packets $U$ is given by

$$U \approx \frac{1}{1 - D_f} k \approx \left(1 + \frac{1}{\mu} \right) k.$$  \hspace{1cm} (12)

Fig. 5 compares the average number of undelivered packets at each receiver against their Markov state. As we can observe, (12) is shown to be a fairly accurate approximation of these values.

V. Conclusion

In this paper, we have demonstrated that a deterministic network coding scheme can be used to enable multicast video on demand using a single server channel. Receivers joining at different times are able to simultaneously deliver at full rate, after a startup delay has been incurred.

Some weaknesses of the transmission scheme in its current form include stringent feedback requirements, increased memory and processing at both the sender and receivers, and the potential for unacceptably large startup delays. It is anticipated that some of these weaknesses can be alleviated by hybridising our proposed single server channel transmission scheme with other existing video on demand algorithms. Used in conjunction with server push algorithms, it could allow for more forgiving join times. Combined with client initiated algorithms, it could reduce the number of secondary server channels required to bring receivers up to speed with their designated batch.

REFERENCES