Multi-Source Coded Downloads

Patrik J. Braun*†, Derya Malak*, Muriel Médard*
*Research Laboratory of Electronics (RLE)
Massachusetts Institute of Technology
Cambridge, MA 02139 USA
{pbraun, deryam, medard}@mit.edu

Péter Ekler†
†Department of Automation and Applied Informatics
Budapest University of Technology and Economics
Budapest, 1111 Hungary
{patrik.braun, peter.ekler}@aut.bme.hu

Abstract—In this paper, we propose a selective-repeat (SR) automatic repeat-request (ARQ) model for multi-source download scenarios and analyze their useful throughput that we refer to as goodput. The multi-source scenario comprises a set of transmitters that send packets to a receiver. We characterize the forward channels from the transmitters to the receiver via a general hidden Markov model (HMM) and assume that the reverse channels from the receiver to the transmitter are lossless. To find the average goodput of the network, we exploit the probability-generation function. We consider different packet transmission schemes, including uncoded random, network coded and sliding window-based network coded packets, and contrast their performance. Our calculations show that using network coding in a multi-source scenario can increase the average goodput, while sliding window-based coding may also archive the theoretical maximum goodput. We show that our multi-source approach avoids the straggler problem, therefore adding more transmitters to the network increases its throughout and the system does not get limited by the weakest transmitter. We also verify our analytic results with extensive simulations.

Index Terms—Network coding, selective-repeat (SR), Automatic Repeat-reQuest (ARQ), Hidden Markov model (HMM), Multi-source network, Throughput.

I. INTRODUCTION

Automatic Repeat reQuest (ARQ) is a widely used error control method for data transmissions. It uses timeouts and acknowledgments (ACKs) to achieve reliable transmission over an unreliable channel and has several well known types including Stop-and-wait ARQ, Go-Back-N ARQ, and selective-repeat (SR) ARQ. In case of SR ARQ, the transmitter sends packets without waiting for their ACK and only the lost packets are selectively retransmitted. ARQ has been applied in modern networks to boost their throughput and reliability [1],[2] and there are detailed analytical models to calculate its throughput: it has been shown that if the average packet-error rate is ϵ, the throughput of SR ARQ with reliable feedback is $1 - \epsilon$ [3]. Y. J. Cho and C. K. UN analyzed different ARQ models with forward and backward channels memory [4] and showed that error bursts have a significant impact on throughput. In [5], Ausavapattanakun and Nosratinia suggested a more versatile, hidden Markov model (HMM) based approach for analyzing SR ARQ with a discrete channel model.

We have recently extended the work of Ausavapattanakun and showed that using erasure coding, e.g.: random linear network coding (RLNC) on ARQ channels models may increase

the throughput by up to 40% [6]. M. Tömösközi et al. showed their coded sliding window approach outperforms the Reed-Solomon and other RLNC approaches in per-packet delay [7]. J. K. Sundararajan et al. introduced a network coded (NC) approach to transmission control protocol (TCP) and showed that their scheme achieves a much higher throughput compared to TCP over a lossy link [8].

Most of the ARQ approaches work on a point-to-point basis that can be used in single-receiver single-transmitter networks, but they do not support multi-source scenarios. Multi-source download has a huge potential in future 5G networks, where users are using mobile networks to access bandwidth and delay intensive services, like video streaming. It has been shown through measurements that multi-source video streaming may help to meet this bandwidth and delay constraints, since it increases download throughput and reliability, and thereby the quality of service [9]. Furthermore, using network coded shared file system for multi-source download with four commercial cloud solutions may achieve up to five-fold increase in download speed compared to single-source download [10]. M. Sipos showed a six-fold increase in download speed by using four commercial clouds and a custom network coded protocol [11]. While these works show huge potential of multi-source download, they mainly do it through measurement results and lack a rigorous analytical model.

In this paper, we propose an SR ARQ model to analyze the multi-source networks, inspired by the point-to-point model in [5] and [6]. The analysis focuses on goodput, the useful throughput of the network. Our model contains N transmitters (with N orthogonal channels) and one receiver. Our forward link is modeled by a hidden Markov model (HMM). We consider not only the conventional uncoded transmission schemes but also the rateless coded and sliding window-based coding methods. We show that the sliding window-based coding may reach optimal goodput. The uncoded scheme also converges to the optimal goodput with the increase of the window size on the transmitter. Our results also show that applying rateless codes on the transmitted data may further increase goodput. Furthermore, the straggler problem is a huge challenge in distributed systems [12]. Results also show that our approach avoids the straggler problem, thus increasing the number of transmitters, increases goodput without getting limited by the weakest transmitter. We also compare our analysis with simulation results. To the best of our knowledge, this paper

is the first to consider an HMM-based channel model, which also incorporates RLNC in a multi-source network scenario.

II. System model

We focus on multi-source networks, where there are N transmitters and only one receiver. Each transmitter has its own channel, but all transmitters have the same source data, i.e. the set of original packets: $\mathcal{L} = \{p_1, \ldots, p_L\}$, where L is the total number of packets. The receiver aims to collect the set $\mathcal{L}$. We consider that the receiver has an infinite receive-side window. While each transmitter has access to all L packets, it also maintains a w-sized window, where $w < L$. An overview about the proposed multi-source system is shown in Fig. 1.
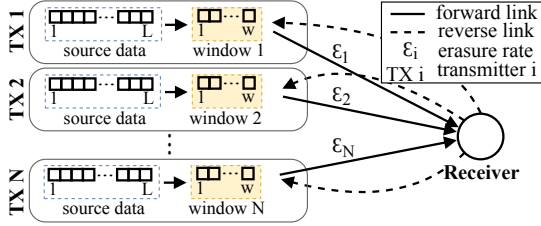

Fig. 1. Multi-source system overview.

A. Channel model

Each transmitter has an unreliable forward link (the channel from the transmitter to the receiver) and a lossless reverse link (the channel from the receiver to the transmitter) that does not interfere with other links. All links are delayed: we assume the round trip time (RTT) is fixed and is equal for each transmitter and given by $\kappa_c = \kappa_{ct \to r} + \kappa_{cr \to t}$, where $\kappa_{ct \to r}$ is the latency between the transmitter and the receiver, while $\kappa_{cr \to t}$ is the latency between the receiver and the transmitter.

We model the erasures on the forward link with a hidden Markov model to make our solution applicable to different types of channels, similarly to the work of Ausavapattanakun and Nosratinia [5]. At every time slot, a transmitter sends a packet that may be delivered or lost due to an erasure. The outcome of a transmission through channel i, denoted by $X_t^{(i)}$, is a Bernoulli random variable, taking values from $\mathcal{X}^{(i)} = \{0, 1\}$, where 0 and 1 correspond to an erroneous and an error-free transmission, respectively. The channel condition is model by a multistate Markov chain $S_t^{(i)}$, in which the states are $S^{(i)} = \{1, \ldots, K^{(i)}\}$, and its probability transition matrix is $P_{(i)}$. Each state $S_t^{(i)} = j, j \in S^{(i)}$ has a different error probability $\epsilon_j^{(i)}$. We denote the set of these channel error probabilities by $\epsilon^{(\mathbf{i})} = \{\epsilon_1^{(i)}, \ldots, \epsilon_{K^{(i)}}^{(i)}\}$. The process $X_t^{(i)}$, which is driven by the Markov process $S_t^{(i)}$ is a hidden Markov process and can be characterized by $\{S^{(i)}, \mathcal{X}^{(i)}, P_{(i)}, \epsilon^{(\mathbf{i})}\}$. Furthermore $P_{L,(i)} = P_{(i)} \cdot \text{diag}\{\epsilon^{(\mathbf{i})}\}$ and $P_{R,(i)} = P_{(i)} \cdot \text{diag}\{1 - \epsilon^{(\mathbf{i})}\}$ are the probabilities of losing and receiving a packet, respectively. Note that $P_{L,(i)} + P_{R,(i)} = P_{(i)}$.

Furthermore, our model does not use an explicit channel coding, but it can be applied on the top of a network that uses channel coding. We assume that the underlying layers use some channel coding that can indicate if a packet was lost.

B. Protocol Description

In our model, the source of a packet is not important as long as the receiver receives that packet. Thus to avoid the race condition in a parallel multi-source system and make the analysis simpler, we assume for our analysis that the transmitters are scheduled in a round-robin fashion. In every time slot, only one transmitter sends a packet. The RTT for this round-robin model will be: $k = N\kappa_c$ and also $k_{r \to t} = N\kappa_{cr \to t}$ and $k_{t \to r} = N\kappa_{ct \to r}$. As a result of round-robin scheduling of the transmitters in ascending order, a packet received at time slot t is sent by transmitter:

$$s(t) = \begin{cases} N \text{ if } (t \bmod N) = 0 \\ (t \bmod N) \text{ otherwise,} \end{cases} \tag{1}$$

and transmitter $s(t)$ sends:

$$pkt(t) = \text{packet arrives or gets lost at the receiver at time } t. \tag{2}$$

The life cycle of a packet is the following:
1) *packet scheduling and sending*: In every time slot, a transmitter selects a packet from their w-sized window and sends it over their channel.
2) *packet arrives or gets lost*: Receiver sends a feedback $\kappa_{ct \to r}$ time slots after the transmitter sent the packet, independent of whether the packet got lost or arrived at the receiver.
3) *receiving the feedback*: $\kappa_{cr \to t}$ later the feedback arrives at the transmitter, which updates its window content based on the feedback.

Since a transmitter sends a packet in every time slot and the reverse link is perfect, transmitters receive an ACK or NACK in every slot as well. A transmitter selects a packet to send based on a pre-determined scheduling method that is the same for every transmitter. We detail the different scheduling methods in Section IV.

We do not consider conventional SR ARQ protocol in our analysis since not all lost packets need to be retransmitted automatically: We use cumulative feedback that contains all previously received packets at the receiver (from all transmitters). If a subset of the channels wants to transmit packet $p_l \in \mathcal{L}$ and it gets lost on some of the channels, but received through at least one of the channels, all transmitters will receive an ACK corresponding to packet p_l. Therefore, it is not necessary and also redundant to retransmit packet p_l on any of the channels. Fig. 2 gives an example of our round-robin transmission model.
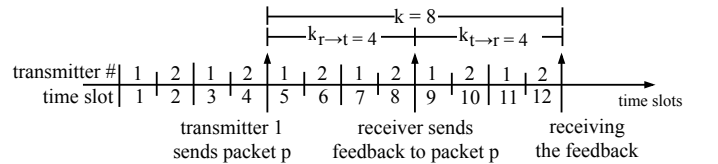

Fig. 2. Timeline example for serialized model with $N = 2, k = 8$.

In our analysis, we assume that the transmitters cannot communicate with each other, which makes the packet scheduling

challenging. We measure the receiver status with its Degrees of Freedom (DoF). DoF at the receiver increases if it receives a new, useful packet that contains new information. Due to the lack of cooperation, several transmitters may schedule the same packet for transmission, and the receiver may receive duplicate packets that do not increase its DoF.

Data download in our system has a push fashion instead of a centralized, receiver-driven pull fashion, because of the cumulative feedback and the lack of cooperation. Due to this push fashion, a transmitter can schedule any not yet acknowledged packet without depending on other transmitters. Therefore the system is not limited by the weakest transmitter and avoids the straggler problem.

We focus on estimating the goodput of a multi-source system in our analysis. We define goodput as the number of DoF increases at the receiver per sent packet. We distinguish goodput $\eta_{(i)} \in [0, 1]$ for channel i and goodput $\eta \in [0, N]$ for the whole system.

III. ANALYSIS

In this section we describe a method for analyzing the overall and per channel average goodput of a system with N transmitters. First, we detail the possible outcomes of packet transmission.

In the forward channel, during transmission, a packet can get:

1) $E_{\mathrm{L},(i)}$ (*lost*): the event that a packet is lost with $P_{\mathrm{L},(i)}$ probability on channel i,
2) $E_{\mathrm{R},(i)}$ (*received*): the event that a packet is received with $P_{\mathrm{R},(i)}$ probability on channel i.

During scheduling time, a transmitter might schedule a packet that is:

1) $E_{\mathrm{pU},(i)}$ (*potentially useful*): the event that given $E_{\mathrm{R},(i)}$, the packet will increase the DoF at the receiver,
2) $E_{\mathrm{pD},(i)}$ (*potentially duplicate*): the event that given $E_{\mathrm{R},(i)}$, the packet will not increase the DoF at the receiver.

If the packet is received, it might be

3) $E_{\mathrm{U},(i)}$ (*useful*): the event that a packet is successfully received on channel i and increases the DoF at the receiver,
4) $E_{\mathrm{D},(i)}$ (*duplicate*): the event that a packet is successfully received on channel i, but does not increase the DoF at the receiver.

Event $E_{\mathrm{L},(i)}$ and $E_{\mathrm{D},(i)}$ are equivalent, since in both cases receiver does not receive new DoFs in that time slot. Therefore, these two events can be combined into a single event:

5) $E_{\mathrm{F},(i)}$ (*fail*): packet was lost, or it was received on channel i, but does not increase the DoF at the receiver.

Using these events, we define the following two main probabilities:

$$\mathcal{P}_{\mathrm{U},(i)} = \mathrm{P}(E_{\mathrm{U},(i)})$$
$$\mathcal{P}_{\mathrm{F},(i)} = \mathrm{P}(E_{\mathrm{F},(i)}) \qquad (3)$$

Based on (3), we construct a signal-flow graph [13] to model the goodput of individual channels. We use matrix branch

gains in the graph, since each link has multiple states because we use HMM to model them. A signal-flow graph is a diagram of directed branches between nodes to visually represent a system of equations. Nodes are variables of the equations, while the branches are the relationships between the variables. Basic equivalences, like parallel, series, self-loop can be used to simplify a flow graph [14]. A signal-flow graph with matrix branch transmissions and vector node values is a matrix signal-flow graph (MSFG).

We construct the MSFG in such a way that branch gains appear as pz^x, where x is the random variable of interest and p is a probability. Thereby the graph represents an equation system that is polynomial in z with coefficients that are the probabilities of a given value of x. This system of equations is the $\mathbb{E}[z^n]$, the probability generation function (PGF) for x.

Fig. 3 shows the matrix flow graph of our transmission model. In the figure, state $\mathrm{I}_{(i)}$ represents the transmission of a new packet, while at state $\mathrm{O}_{(i)}$, the feedback of event $E_{\mathrm{U},(i)}$ is received at the transmitter i and the transmitter can update its window accordingly.
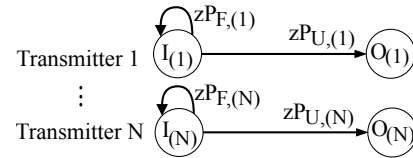

Fig. 3. Matrix signal-flow graph for goodput analysis of our serialized model.

Next, we calculate the transmission time τ that we define as the number of transmitted packets per DoF increase at the receiver. τ can be calculated by using the matrix-generating function $\Phi_{\tau}(z)$. We get $\Phi_{\tau,(i)}(z)$ by applying basic node reduction on the MSFG, similarly to [5]:

$$\Phi_{\tau,(i)}(z) = (\mathbf{I} - z\mathcal{P}_{\mathrm{F},(i)})^{-1} z\mathcal{P}_{\mathrm{U},(i)}, \qquad (4)$$

where $\mathbf{I}$ is the identity matrix.

To calculate the PGF, we need to express $\pi_{I(i)}$, the probability vector of event $E_{\mathrm{U},(i)}$. In this case, it is $\pi_{I(i)} = \pi_{(i)}\mathcal{P}_{\mathrm{U},(i)}$, where $\pi_{(i)}$ is the stationary vector of $P_{(i)}$ and can be found by solving:

$$\begin{aligned} \pi_{(\mathbf{i})}P_{(i)} &= \pi_{(\mathbf{i})} \\ \pi_{(\mathbf{i})}\mathbf{1} &= 1, \end{aligned} \qquad (5)$$

where $\mathbf{1}$ is the column vector of ones. Furthermore, let $\epsilon_{\mathrm{F},(i)}$ be the packet-failure rate: $\epsilon_{\mathrm{F},(i)} = \pi_{(i)}\mathcal{P}_{\mathrm{F},(i)}\mathbf{1}$. Then PGF of $\phi_{\tau}(z)$ can be calculated by pre- and post-multiplying $\Phi_{\tau}(z)$ with a row and a column vector, respectively:

$$\phi_{\tau(i)}(z) = \frac{\pi_{I(i)}\Phi_{\tau,(i)}(z)\mathbf{1}}{\pi_{I(i)}\mathbf{1}} = \frac{1}{1 - \epsilon_{\mathrm{F},(i)}}\pi_{(i)}\mathcal{P}_{\mathrm{U},(i)}\Phi_{\tau,(i)}(z)\mathbf{1}. \qquad (6)$$

The average transmission time of transmitter i, $\tau_{(i)}$ can be obtained by evaluating the first derivative of PGF $\phi_{\tau(i)}(z)$ at $z = 1$. The goodput, $\eta_{(i)}$ of channel i is the reciprocal is the average transmission time, i.e., $\eta_{(i)} = 1/\tau_{(i)}$.

A. Calculating the probability of sending a useful packet $\mathcal{P}_{U,(i)}$ and a packet failure $\mathcal{P}_{F,(i)}$

Whether a packet p_t received at time t is potentially useful depends only on the last k time slots: Packet p_t is sent at time $t_s = t - k_{t \to r}$, since the transmitter-receiver latency is $k_{t \to r}$. Transmitter $s(t_s)$ has a feedback that contains information from time $t_s - k_{r \to t} = t - k$, since the receiver-transmitter latency is $k_{r \to t}$ (i). Furthermore, transmitters can also keep records of previously sent packets (ii). Since the transmitters may not cooperate, a transmitter may only use information (i) and (ii) to schedule a packet for transmission.

Using the feedback from time $t - k$, it is guaranteed that a transmitter will not send a packet that would be a duplicate of packets before time $t - k$, but it has no information about the packets after that time. Therefore it may schedule duplicates with them. We assume that a transmitter does not schedule packets that are duplicates with its previously sent packets[1]. Thus, a packet at time t will not be useful only if it has the same information as any of the useful packets in the last k time slots. There may be $u \in [0, k - \frac{N}{k}]$ useful packets[2] sent by transmitters j, $j \neq s(t)$ between time slots $t - k$ and t.

We next investigate the number of potentially duplicates sent by transmitter $s(t)$. If the packet from transmitter $s(t)$ is a potentially duplicate of a useful packet from any transmitter j, $j \neq s(t)$, then the probability is higher that the packet at time t is useful (since if a duplicate packet was already transmitted by transmitter $s(t)$ in the last k time, it will not retransmit that packet. Thus it is more likely to choose a useful packet).
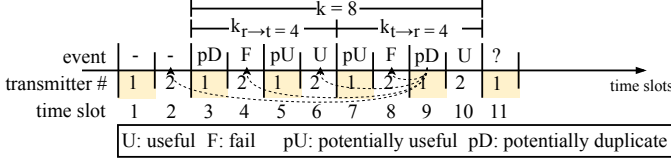

Fig. 4. Example realization to calculate $\mathcal{P}_{U,(i)}$ and $\mathcal{P}_{F,(i)}$, $N = 2$, $k = 8$.

To better understand our methodology, let us consider the following example for $N = 2$, $k = 8$, as shown in Fig. 4. In this example, we are interested in calculating the probability that the packet received at time $t = 11$ from transmitter 1 is useful. We know that the receiver obtained $u = 2$ packets from transmitter 2 in the last $k = 8$ time slots. The packet at time $t = 11$ may be a duplicate of any of those 2 useful packets. The Packet at time $t = 9$ from transmitter 1 is a potentially duplicate with any of the packets from transmitter 2 between time slots [2,8]. If it is a duplicate of the packet at time $t = 6$, our investigated packet at time $t = 11$ may only be a duplicate (if it is a duplicate at all) with packet at time 10.

Rest of this section uses this methodology to express $\mathcal{P}_{U(i)}$ and $\mathcal{P}_{F,(i)}$ as a function of t through several steps. At every step, we express the probability of a packet being useful or to fail (is duplicate or lost) based on a given condition and

also the probability of that given condition. To obtain $\mathcal{P}_{U(i)}$ and $\mathcal{P}_{F,(i)}$, we define the following quantities at time t:

$$\mathbf{v} \in \{0,1\}^t, \, v_l = \begin{cases} 1 & \text{if } E_{R,(s(l))} \\ 0 & \text{otherwise,} \end{cases} \quad (7)$$

where $\mathbf{v}$ represents a possible channel outcome between time slots $[0,t]$.

By $E_r(\mathbf{v})$, we denote the event that $\mathbf{v}$ is the channel outcome between $[0,t]$ time slots. We define the probability of $\mathbf{v}$ as the channel outcome, $\mathcal{P}_r(\mathbf{v}) = P(E_r(\mathbf{v}))$. The packet at time t is being useful or fail, conditioned on $\mathbf{v}$:

$$\begin{aligned} \mathcal{P}_{cU}(\mathbf{v}) &= P(E_{U,(s(t))} \text{ at time } t \mid E_r(\mathbf{v})) \\ \mathcal{P}_{cF}(\mathbf{v}) &= P(E_{F,(s(t))} \text{ at time } t \mid E_r(\mathbf{v})) \end{aligned} \quad (8)$$

Note that all probabilities with $\mathbf{v}$ as parameter, now implicitly depends on the transmitter, since $\mathbf{v} = [v_1 \ldots v_t]$ is the input of the function and only transmitter $i = s(t)$ may transmit at time t. Therefore, we can also omit the transmitter from $\mathcal{P}_{U(i)}(t)$ and $\mathcal{P}_{F,(i)}(t)$ and express them as follows:

$$\begin{aligned} \mathcal{P}_U(t) &= \sum_{v \in \{0,1\}^t} \mathcal{P}_{cU}(\mathbf{v}) \mathcal{P}_r(\mathbf{v}) P_{R,(s(t))} v_t \\ \mathcal{P}_F(t) &= \sum_{v \in \{0,1\}^t} \mathcal{P}_{cF}(\mathbf{v}) \mathcal{P}_r(\mathbf{v}) P_{R,(s(t))} v_t + P_{L,(s(t))} |1 - v_l| \end{aligned} \quad (9)$$

where $\mathcal{P}_r(\mathbf{v})$ is:

$$\mathcal{P}_r(\mathbf{v}) = \prod_{i=1}^{N} \prod_{\substack{l=i \\ l=l+N}}^{t-1} \pi_{(i)} P_{R,(i)}^{v_l} P_{L,(i)}^{|1-v_l|} \mathbf{1}. \quad (10)$$

The probabilities $\mathcal{P}_{cU}(\mathbf{v})$ and $\mathcal{P}_{cF}(\mathbf{v})$ depend on the probability of a packet being potentially useful or duplicate:

$$\begin{aligned} \mathcal{P}_{pU}(\mathbf{v}) &= P(E_{pU,(s(t))} \text{ at time } t \mid E_r(\mathbf{v})) \\ \mathcal{P}_{pD}(\mathbf{v}) &= P(E_{pD,(s(t))} \text{ at time } t \mid E_r(\mathbf{v})) \end{aligned} \quad (11)$$

and they may be expressed the following way:

$$\begin{aligned} \mathcal{P}_{cU}(\mathbf{v}) &= \mathcal{P}_{pU}(\mathbf{v})^{v_t} + 0^{|1-v_t|} \\ \mathcal{P}_{cF}(\mathbf{v}) &= \mathcal{P}_{pD}(\mathbf{v})^{v_t} + 1^{|1-v_t|}. \end{aligned} \quad (12)$$

As (12) shows, a packet is useful with probability $\mathcal{P}_{pU}(\mathbf{v})$ if it is received and fails with probability $\mathcal{P}_{pD}(\mathbf{v})$ if received or with probability 1 if lost.

To calculate $\mathcal{P}_{pU}(\mathbf{v})$ and $\mathcal{P}_{pD}(\mathbf{v})$, the following quantities need to be expressed:

$$\mathbf{a} \in \{0,1\}^k, \, a_j = \begin{cases} \text{'x'} & \text{if } s(t-j) = s(t) \\ 1 & \text{else if } E_{U,(s(t-j))} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{b} \in \{0,1\}^k, \, b_i = \begin{cases} \text{'x'} & \text{if } s(t-i) \neq s(t) \\ 1 & \text{else if } E_{pD,(s(t-i))}, \text{ and} \\ & \exists j, 0 < i \leq k, E_{U,(s(t-j))}, \\ & pkt(t-i) \equiv pkt(t-j) \\ 0 & \text{otherwise,} \end{cases} \quad (13)$$

where x means the value at that position will not be used during our calculations, but to simplify our formulas, we assume its value to be 0. $pkt(i) \equiv pkt(j)$ means that two packets are interchangeable, i.e., they increase the DoF at the receiver by at most one. Vector $\mathbf{a}$ represents the useful packets received from transmitter $j, j \neq s(t)$ between time slots $[t-k,t]$, $\mathbf{b}$ represents the potentially duplicates with any useful packet between $[t-k,t]$ and received from transmitter $s(t)$ between time slots $[t-k,t]$.

We define the probability that there is a useful packet with a given $\mathbf{a} = [a_i, \ldots, a_k]$ for every potentially duplicate packet in $\mathbf{b} = [b_i, \ldots, b_k]$:

$$E_{\mathrm{dp}}(\mathbf{a},\mathbf{b}) = \forall i, b_i = 1 : \exists j, j < i, a_j = 1$$
$$pkt(t-i) \equiv pkt(t-j) \tag{14}$$
$$\mathcal{P}_{\mathrm{dp}}(\mathbf{a},\mathbf{b}) = \mathrm{P}(E_{\mathrm{dp}}(\mathbf{a},\mathbf{b})).$$

We also define $\mathcal{P}_{\mathrm{outcome}}(\mathbf{v},\mathbf{a},\mathbf{b})$ as the probability of $\mathbf{a}$ and $\mathbf{b}$ is the outcome between time slots $[t-k,t]$:

$$\mathcal{P}_{\mathrm{outcome}}(\mathbf{v},\mathbf{a},\mathbf{b}) = \mathrm{P}(\forall a_j = 1, E_{\mathrm{U},(s(t-j))},$$
$$\forall a_j = 0, E_{\mathrm{D},(s(t-j))}, \tag{15}$$
$$\forall b_i = 1, E_{\mathrm{pD},(s(t-i))} \mid E_{\mathrm{dp}}(\mathbf{a},\mathbf{b}), E_{\mathrm{r}}(\mathbf{v}) \,)$$

Furthermore, we define $\mathcal{P}_{\mathrm{sU}}(\mathbf{a},\mathbf{b})$ and $\mathcal{P}_{\mathrm{sD}}(\mathbf{a},\mathbf{b})$, the probabilities of a packet at time t being useful or duplicate, respectively, conditioned on $\mathbf{a}$ and $\mathbf{b}$:

$$\mathcal{P}_{\mathrm{sU}}(\mathbf{a},\mathbf{b}) = \mathrm{P}(E_{\mathrm{U},(t)} \mid E_{\mathrm{dp}}(\mathbf{a},\mathbf{b}) \,)$$
$$\mathcal{P}_{\mathrm{sD}}(\mathbf{a},\mathbf{b}) = \mathrm{P}(E_{\mathrm{D},(t)} \mid E_{\mathrm{dp}}(\mathbf{a},\mathbf{b}) \,) \tag{16}$$

Using eqs. (13) to (16), we can express $\mathcal{P}_{\mathrm{pU}}(\mathbf{v})$:

$$\mathcal{P}_{\mathrm{pU}}(\mathbf{v}) = \sum_{u=0}^{k-\frac{k}{N}} \sum_{d=0}^{\frac{k}{N}} \sum_{\substack{\sum a_j = u \\ \sum b_i = d}} \mathcal{P}_{\mathrm{dp}}(\mathbf{a},\mathbf{b}) \mathcal{P}_{\mathrm{sU}}(\mathbf{a},\mathbf{b}) \mathcal{P}_{\mathrm{outcome}}(\mathbf{v},\mathbf{a},\mathbf{b})$$
$$\tag{17}$$

where u is the number of useful packets sent by the transmitter $j, j \neq s(t)$. d is the number of packets that are sent by the transmitter $s(t)$ and are potentially duplicate packets with the useful packets in the last k time slots. Similarly to $\mathcal{P}_{\mathrm{pU}}(\mathbf{v})$, we can also express $\mathcal{P}_{\mathrm{pD}}(\mathbf{v})$ by using $\mathcal{P}_{\mathrm{sD}}(\mathbf{a},\mathbf{b})$, instead of $\mathcal{P}_{\mathrm{sU}}(\mathbf{a},\mathbf{b})$.

$\mathcal{P}_{\mathrm{dp}}(\mathbf{a},\mathbf{b})$ and $\mathcal{P}_{\mathrm{outcome}}(\mathbf{v},\mathbf{a},\mathbf{b})$ can be expressed in the following way:

$$\mathcal{P}_{\mathrm{dp}}(\mathbf{a},\mathbf{b}) = \prod_{\substack{l=1 \\ s(t-l)=s(t)}}^{k} \frac{\sum_{j=1}^{l} a_j - \sum_{i=1}^{l} b_i}{k - \frac{k}{N}}$$

$$\mathcal{P}_{\mathrm{outcome}}(\mathbf{v},\mathbf{a},\mathbf{b}) = \prod_{\substack{j=1 \\ s(t-j) \neq s(t)}}^{k} \mathcal{P}_{\mathrm{cU}}(\mathbf{v}_{t-j})^{a_j} \mathcal{P}_{\mathrm{cF}}(\mathbf{v}_{t-j})^{|1-a_j|}. \tag{18}$$

$$\prod_{\substack{i=1 \\ s(t-i)=s(t)}}^{k} \mathcal{P}_{\mathrm{pD}}(\mathbf{v}_{t-i})^{b_i} (\mathcal{P}_{\mathrm{pU}}(\mathbf{v}_{t-i}) + \mathcal{P}_{\mathrm{pD}}(\mathbf{v}_{t-i}))^{|1-b_i|}$$

$$\mathbf{v}_{t-l} = [v_1 \ldots v_{t-l}].$$

The presented equations in this section do not depend on the method how a transmitter selects a packet for transmission, but to calculate $\mathcal{P}_{\mathrm{sU}}(\mathbf{a},\mathbf{b})$ and $\mathcal{P}_{\mathrm{sD}}(\mathbf{a},\mathbf{b})$, one also has to consider the applied packet scheduling method. We detail that in the next section. Furthermore, our matrix-flow graph approach to calculate the average goodput is only applicable if $\lim_{t \to \infty} \mathcal{P}_{\mathrm{U}}(t)$ and $\lim_{t \to \infty} \mathcal{P}_{\mathrm{F}}(t)$ exist.

IV. SCHEDULING METHODS

In this section, we enumerate several packet scheduling strategies. We calculate $\mathcal{P}_{\mathrm{sU}}(\mathbf{a},\mathbf{b})$ and $\mathcal{P}_{\mathrm{sD}}(\mathbf{a},\mathbf{b})$, that are required to calculate the average goodput in (17), corresponding to a given scheduling method.

As described in Section II, transmitters maintain a w-sized window. We consider a *moving* window instead of a sliding window that we define the following way: If a packet gets removed from the window, the next available packet will be picked from the L source data to fill the window. Therefore, the window constantly contains w packets[3]. We assume L is large enough, so that there are always enough packets to fill the window, which is the case in a streaming scenario.

A. Sufficient genie scheme

We introduce a *sufficient genie* scheduling strategy to find the optimal goodput of a system with the given channel properties. It is not a *full genie*, since it only focuses on sending the perfect packet regarding usefulness, but packets might be lost on the channel. Therefore, $\mathcal{P}_{\mathrm{sU}}(\mathbf{a},\mathbf{b}) = 1$ and $\mathcal{P}_{\mathrm{sD}}(\mathbf{a},\mathbf{b}) = 0$.

Using a genie, the transmitter-channel pairs can be decoupled and analyzed independently. The average goodput of transmitter i only depends on the loss probability $P_{\mathrm{L},(i)}$. Following the steps in [5], the average goodput of a channel i is:

$$\eta_{(i)} = 1 - \epsilon_{(i)} = 1 - \pi_{(i)} P_{\mathrm{L},(i)} \mathbf{1}. \tag{19}$$

The overall average goodput of the system for N transmitters is $\eta = \sum_{i=1}^{N} \eta_{(i)}$.

B. Uncoded random scheme

In this approach, transmitters select a not-in-flight[4] packet uniformly at random from their send window for transmission. We can express $\mathcal{P}_{\mathrm{sU}}(\mathbf{a},\mathbf{b})$ and $\mathcal{P}_{\mathrm{sD}}(\mathbf{a},\mathbf{b})$ in the following way:

$$\mathcal{P}_{\mathrm{sU}}(\mathbf{a},\mathbf{b}) = 1 - \mathcal{P}_{\mathrm{sD}}(\mathbf{a},\mathbf{b}) = \frac{w - \frac{k}{N} - (\sum_{j=1}^{k} a_j - \sum_{i=1}^{k} b_i)}{w - \frac{k}{w}} \tag{20}$$

where $\frac{k}{N}$ is the number of packets in flight from one transmitter, and the summation gives how many useful packets were sent by transmitter $j, j \neq s(t)$ in the last k time slot in such a way that the transmitter $s(t)$ has not sent any potentially duplicate packet to those packets.

---

[3]The packet in our window may not be consecutively chosen and there is no limit on the maximum time a packet can spend in the window.

[4]A packet is in flight when it is sent, but feedback has not been received.

C. Rateless RLNC coded schemes

RLNC creates linear combinations of original packets with randomly chosen coefficients. It may be applied to the transmitted data to reduce the probability of receiving duplicate packets. RLNC has recoding ability and can work as a rateless code over a fixed set of packets [11] or as a sliding window code over a changing set of packets [15].

In this scheme, we use RLNC in a rateless coding way: packets are grouped into *generations*, creating altogether $G \in \mathbb{Z}^+$ *generations* with $g \in \mathbb{Z}^+$ packets in each. Network coding is applied to each of the *generations*. Each transmitter groups the packets in the same way, but uses a different random seed to generate the linear combinations. In our analysis, we assume that the field size used is high enough such that the probability of two encoded packets being linearly dependent goes to zero [16]. The receiver feedback contains the rank of a *generation* instead of information about an individual packet, where the rank equals the DoF of a given *generation*. The transmitter window contains $G_w = \frac{w}{g}$ *generations*[5]. In every time slot, a transmitter chooses one *generation* from its window to create an encoded packet from and sends it over the channel. The selection of a *generation* may be based on different approaches. In this paper, we investigate a random and a *rarest first* generation selection schemes.

In both cases, $\mathcal{P}_{sU}(\mathbf{a}, \mathbf{b})$ and $\mathcal{P}_{sD}(\mathbf{a}, \mathbf{b})$ depend on the probability of transmitter $s(t)$ choosing the *generation* γ for transmission and its rank at time slot t. Calculating these probabilities is not part of this paper. We instead show the goodput of applying network coding in a multi-source environment through simulations in Section V.

1) Random generation selection scheme: Transmitters choose a *generation* for transmission uniformly at random.

2) Rarest first generation selection scheme: Transmitters approximate the rank of the *generations* an the receiver and choose the one that hase the least rank. The approximation is based on two components: 1) the feedback that represents the receiver state $k_{r \to t}$ time slots ago, 2) the sent packets by that given transmitter. We call this strategy *rarest generation first* strategy, referring to the *rarest piece first* algorithm in *BitTorrent* [17].

One should note two special cases that apply for both generation selection approaches: 1) if $g = 1$, the goodput will be identical with the *uncoded random* schemes. 2) if $L = w = g$, the goodput will be identical with the *sufficient genie* scheme, since all received packet will be useful.

D. Coded sliding window scheme

In case of the network coding sliding window [15] scheme, a transmitter encodes all the packets in its window with RLNC. The receiver feedback contains information about the successfully decoded packets. The probability of receiving a useful packet is the following:

---

[5]To keep the analysis simple, we assume $L \bmod g = w \bmod g = 0$.

$$\mathcal{P}_{sU}(\mathbf{a}, \mathbf{b}) = 1 - \mathcal{P}_{sD}(\mathbf{a}, \mathbf{b}) = \begin{cases} 1 & \text{if } (t \bmod k) < w \\ 0 & \text{otherwise} \end{cases}. \quad (21)$$

Note that if $k \leq w$, all received packets will be useful, therefore the strategy would have the same goodput as the *sufficient genie* scheme.

Comparing this solution to the rateless RLNC coded strategies, sliding window achieves optimal performance with coding less or equal packets together, thereby using less CPU cycles, since we usually have $k << L$. On the other hand, with rateless coding the random seed can be shared between the transmitter and the receiver, while with sliding window the coefficient vector needs to travel in the packet payload.

V. NUMERICAL RESULTS

We computed the numerical results for our model by using a two state Gilbert-Elliot (GE) channel model [18] for the forward link of the transmitters. The state-transition matrix of the channel is given by:

$$P_{(i)} = \begin{bmatrix} 1 - q_{(i)} & q_{(i)} \\ r_{(i)} & 1 - r_{(i)} \end{bmatrix}, \quad (22)$$

where the first row corresponds to the good (G) state and the second to the bad (B) state. The channel error probability is $\epsilon^{(i)} = \{\epsilon_G^{(i)}, \epsilon_B^{(i)}\} = \{0, 1\}$. The packet loss rate $\epsilon_{F,(i)}$ can be calculated from $\epsilon^{(i)}$ and the stationary vector of $P_{(i)}$ as shown in Section III.

We use our simulator testbed to analyze the goodput of our data scheduling schemes. Each simulation was run 1000 times, and an average is calculated from them. We compare our simulations and numerical results and they show similar trends.
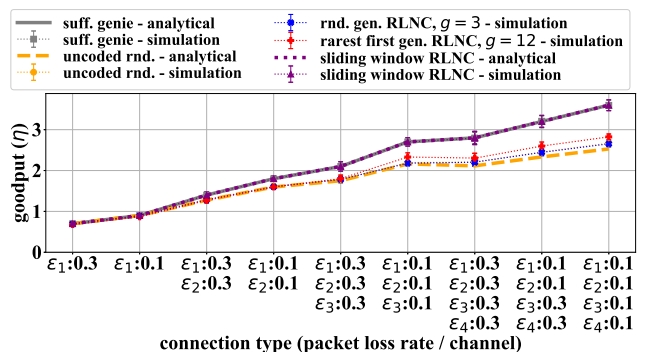

Fig. 5. Goodput for transmitters $N = [1, 2, 3, 4]$, RTT $\kappa_c = 3$, window size $w = 24$ and burst rate $r = 0.3$.

Fig. 7 shows that apart from the *sufficient genie* and the *coded sliding window* scheme, that have the achievable maximum goodput, window size has a high impact on goodput: small window size causes a significant goodput decrease, since the transmitters have a smaller set of packets to choose from. As the figure also shows, in case *rarest first generation selection* scheme, goodput also depends on the combination of the window size and the generation size.

Higher RTT values have a negative impact on goodput, as Fig. 6 shows. *Rarest first generation selection* scheme may perform better compared to the *uncoded random*, but the gain depends on both generation size and RTT. With low RTT, the bigger generation size, while with high RTT the smaller generation size performs better.

Increasing the number of transmitters increases overall average goodput, but increases the chance of sending duplicate packets for the *uncoded random* scheme or the rateless RLNC coded schemes, as Fig. 5 shows, since the difference between the achievable maximum and the actual throughput increases.
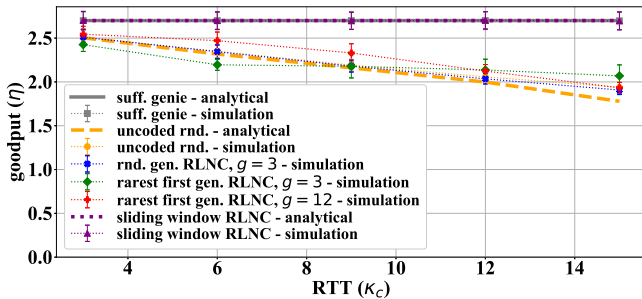

Fig. 6. Goodput for transmitters $N = 3$, packet loss rate $\epsilon_{F,(i)} = 0.1$, window size $w = 24$ and burst rate $r = 0.3$.
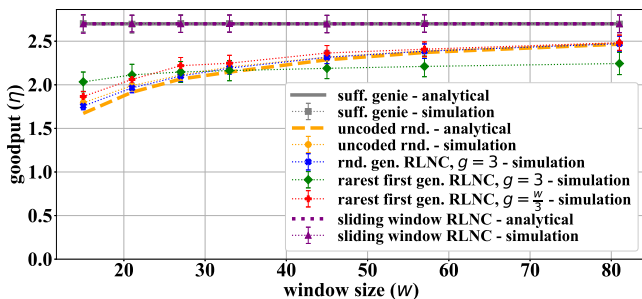

Fig. 7. Goodput for transmitters $N = 3$, RTT $\kappa_c = 4$, packet loss rate $\epsilon_{F,(i)} = 0.1$ and burst rate $r = 0.3$.

VI. CONCLUSION

In this paper, we proposed an SR ARQ model for multi-source single-receiver download. The model uses lossy forward links that are modeled with a hidden Markov process. We used a matrix signal-flow graph approach to calculate the probability generation function of the goodput, and to analyze the average goodput of a multi-source download system.

We compared numerical results with simulation results for several packet scheduling approaches, including the uncoded and network coded approaches. Our results show that rateless network coding techniques can boost goodput, while network coded sliding window may achieve optimal performance. We also showed that our multi-source approach avoids the straggler problem, therefore adding new transmitters to the network increases the goodput.

In this paper, we analyzed a moving window approach does not set any constraints on the packet delay. As future work, we plan to investigate a more flexible window approach that has a constraint on the delay and we would like to also consider further packet scheduling schemes.

REFERENCES

[1] 3GPP, "Study on new radio access technology Physical layer aspects," 3rd Generation Partnership Project (3GPP), Technical report (TR) 38.802, 09 2017. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3066

[2] S. R. Khosravirad and H. Viswanathan, "Analysis of Feedback Error in Automatic Repeat reQuest," *CoRR*, vol. abs/1710.00649, 2017.

[3] D.-L. Lu and J.-F. Chang, "Performance of ARQ protocols in nonindependent channel errors," *IEEE Transactions on Comm.*, vol. 41, no. 5, pp. 721–730, May 1993.

[4] Y. J. Cho and C. K. Un, "Performance analysis of ARQ error controls under Markovian block error pattern," *IEEE Transactions on Comm.*, vol. 42, no. 234, pp. 2051–2061, FEBRUARY 1994.

[5] K. Ausavapattanakun and A. Nosratinia, "Analysis of Selective-Repeat ARQ via Matrix Signal-Flow Graphs," *IEEE Transactions on Comm.*, vol. 55, no. 1, pp. 198–204, Jan 2007.

[6] M. M. D. Malak and E. M. Yeh, "Analysis of Coded Selective-Repeat ARQ via Matrix Signal-Flow Graphs," *CoRR*, vol. abs/1801.10500, 2018.

[7] M. Tömösközi, F. H. P. Fitzek, D. E. Lucani, M. V. Pedersen, and P. Seeling, "On the Delay Characteristics for Point-to-Point Links using Random Linear Network Coding with On-the-Fly Coding Capabilities," in *20th European Wireless Conf.*, May 2014, pp. 1–6.

[8] J. K. Sundararajan, D. Shah, M. Médard, S. Jakubczak, M. Mitzenmacher, and J. Barros, "Network Coding Meets TCP: Theory and Implementation," *Proceedings of the IEEE*, vol. 99, no. 3, pp. 490–512, March 2011.

[9] J. Bruneau-Queyreix, M. Lacaud, D. Negru, J. M. Batalla, and E. Borcoci, "MS-Stream: A multiple-source adaptive streaming solution enhancing consumer's perceived quality," in *14th IEEE CCNC*, Jan 2017, pp. 427–434.

[10] C. W. Sørensen, D. E. Lucani, and M. Médard, "On network coded filesystem shim: Over-the-top multipath multi-source made easy," in *IEEE ICC*, May 2017, pp. 1–7.

[11] M. Sipos, F. H. P. Fitzek, D. E. Lucani, and M. V. Pedersen, "Distributed cloud storage using network coding," in *IEEE 11th CCNC*, Jan 2014, pp. 127–132.

[12] M. A. M. Songze Li and A. S. Avestimehr, "A Unified Coding Framework for Distributed Computing with StragglingServers," *CoRR*, vol. abs/1609.01690, 2016.

[13] S. J. Mason and H. J. Zimmermann, "Electronic circuits, signals, and systems," pp. xviii, 616 p., companion volume to Electronic circuit theory, by H.J. Zimmermann and S.J. Mason.

[14] R. A. Howard, *Dynamic probabilistic systems: Markov models*. Courier Corporation, 2012, vol. 1.

[15] S. Wunderlich, F. Gabriel, S. Pandi, and F. H. P. Fitzek, "We don't need no generation - a practical approach to sliding window RLNC," in *2017 Wireless Days*, March 2017, pp. 218–223.

[16] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, "A Random Linear Network Coding Approach to Multicast," *IEEE Transactions on Inf. Theory*, vol. 52, no. 10, pp. 4413–4430, Oct 2006.

[17] A. Legout, G. Urvoy-Keller, and P. Michiardi, "Rarest First and Choke Algorithms Are Enough," in *6th ACM SIGCOMM Conf. on Internet Measurement*, ser. IMC '06. New York, NY, USA: ACM, 2006, pp. 203–216.

[18] E. O. Elliott, "Estimates of error rates for codes on burst-noise channels," *The Bell System Tech. Journal*, vol. 42, no. 5, pp. 1977–1997, Sept 1963.