Goodput Analysis for Multi-Source Coded Downloads

Patrik J. Braun, Derya Malak, Muriel Médard, Péter Ekler

Abstract

Multi-source download comprise a set of sources that send packets to a receiver. In this paper, we focus on a scenario, where sources cannot communicate with each other, which makes the packet scheduling challenging: sources may send the same packet to the receiver, which results in a throughput drop. Applying coding, such as Random Linear Network Coding (RLNC) on the transmitted packets may help to overcome this issue. Instead of collecting individual packets, the receiver collects RLNC encoded packets that have a higher chance to increase the Degree of Freedom (DoF) at the receiver's end, as they are the combination of multiple packets.

In this paper, we propose a selective-repeat (SR) Automatic Repeat-reQuest (ARQ) model for multi-source download. We introduce an analytical framework to analyze six multi-source protocols, including uncoded, rateless RLNC, and sliding-window based RLNC protocols and contrast their goodput (application-level throughput). Our analysis shows that employing RLNC in a multi-source network improves goodput, in particular sliding window-based RLNC protocol achieves the maximum theoretical goodput. In addition, we show that our multi-source approach avoids the straggler problem, therefore adding more sources to the network increases its goodput. We also verify our analytical results with extensive simulations.

Index Terms

Automatic Repeat-reQuest (ARQ), Goodput, Hidden Markov model (HMM), Selective-Repeat (SR), Network Coding, Multi-source network.

Patrik J. Braun, Derya Malak and Muriel Médard are with the Network Coding and Reliable Communications Group, RLE, MIT, Cambridge, Massachusetts, 02139 USA, e-mail: {pbraun, deryam, medard}@mit.edu.

Patrik J. Braun, Péter Ekler are with the Department of Automation and Applied Informatics, VIK, Budapest University of Technology and Economics, Budapest, 1117 Hungary, e-mail: {braun.patrik, ekler.peter}@aut.bme.hu.

This work has been done, when Patrik J. Braun was a visiting student researcher at MIT.

This research was supported by the BME-Artificial Intelligence FIKP grant of EMMI (BME FIKP-MI/SC), by the János Bolyai Research Fellowship of the Hungarian Academy of Sciences and by the Fulbright and Rosztoczy programs.

I. Introduction

Multi-source download involves a single receiver requesting the same data from multiple data sources. It has high potential in information-centric networking (ICN) [1], especially in an Internet of Things (IoT) or a Vehicle-to-Vehicle (V2V) communication environment [2]. In both IoT and V2V setup, the network contains a set of data sources that potentially have the same data, and a receiver aims to obtain that data. An example of this scenario is as follows: Consider a group of cars traveling on the highway and sharing all of their sensor information with each other. When a new car joins the group and wants to download that sensor information as well, the most efficient way of doing so is to request this data from multiple sources.

There have been several works on multi-source data transfer: Hashmeni and Bohlooli modeled multi-source content delivery through multi-path transmission in ICNs [1]. They estimated virtual round-trip time (VRTT) between the receiver and a group of sources in their proposed model. They selected this VRTT as a key parameter of performance evaluation that can be used to calculate the network throughput. A congestion control mechanism for Content-Centric Networking (CCN) with multi-source content retrieval was proposed by Miyoshi et al. [3]. Their solution uses end-to-end flow control to regulate the transmission only on the congested paths. Thomas et al. designed a multi-source and multipath File Transfer Protocol (mmFTP) for ICN networks [4]. They showed through measurements that mmFTP has 37% throughput increase compared to a single-source download, while it avoids congested paths or sources. In a video streaming environment, multi-source download also improves network bandwidth and delay, and thereby the quality of service [5]. Our solution differs from previous works by providing an analytical framework for estimating the goodput (i.e., the useful throughput of the network) of different multi-source protocols.

In a multi-source scenario, managing the packet scheduling at the different sources is a challenging task, especial in a loosely orchestrated scenario, like in V2V communication, where the set of available sources are continuously changing. In these environments, multiple sources may schedule the same data packet that does not contain new information to the receiver. To avoid this issue, coding can be applied to the source packets.

An early version of coded data transfer was rateless codes like fountain codes that often did not use feedback [6]. However, to have flow control in a transmission, feedback is required. Malak et al. analyzed Selective-Repeat (SR) Automatic Repeat reQuest (ARQ) channel models that use

erasure coding (like Random Linear Network Coding (RLNC)) with feedback [7]. They showed that RLNC increases the throughput by up to 40% in SR ARQ channels. Liu et al. presented a solution for network coded multi-source transmissions in ICN [8]. They collaborated in-network caching with network coding to increase transmission efficiency. We also proposed a system for downloading YouTube videos from multiple sources that support uncoded, and RLNC encoded multi-source protocols [9]. Our measurement results showed that our uncoded protocol could outperform a simple parallel HTTP approach, while applying coding, it is possible to reach up to a three-fold goodput increase. It has also been shown that using a 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]. Furthermore, Sipos showed a six-fold increase in download speed by using four commercial clouds and a custom network coded protocol [11]. In this work, we propose RLNC based multi-source protocols and contrast their performance with uncoded multi-source protocols by using our analytical framework.

Apart from achieving flow control, feedback can also be used to improve the performance of coding. Sundararajan et al. used feedback to acknowledge degrees of freedom (DoF) (the number of useful packets arrived at the receiver) instead of original decoded packets [12]. Using this feedback, they proposed a network coding method that can be performed in an online manner, without the need for grouping packets into batches of generations. They also introduced a network coded approach to transmission control protocol (TCP) and showed that their scheme achieves a ten fold throughput increase compared to TCP on a link with 1% loss rate [13]. They presented a sliding window network coding approach, where they used the feedback to adjust the window of packets that they coded on. Kim et al. presented a model to analyze the performance of TCP with network coding [14]. They showed that network coding could prevent TCP's performance degradation that is often observed in lossy networks. Tömösközi et al. showed that RLNC coded sliding window approach outperforms the Reed-Solomon, and other rateless RLNC approaches in terms of per-packet delay [15]. In this paper, we use the RLNC coded sliding window approach in a multi-source download scenario and contrast its goodput with uncoded and rateless RLNC encoded multi-source protocols by using our analytical framework.

In a distributed system, the straggler problem is also a challenge [16] [17]. If the client has to wait for a packet that is unusually late to arrive, the network throughput may drop. In this paper, we propose a solution that avoids the straggler problem.

In this paper, we extend our previous work on coded multi-source download [18]. We build on singe-source singe-receiver SR ARQ model of Ausavapattanakun and Nosratinia [19], and the RLNC encoded SR ARQ model of Malak et al. [7] and extend them to support multi-source downloads. We propose an analytical framework to investigate the performance of six multi-source protocols, including uncoded and RLNC encoded protocols. The main contributions of this paper can be summarized as follows:

- Section II proposes an SR ARQ model to analyze a general multi-source network, inspired by the point-to-point model in [7] and [19]. Our model contains N data sources (with N orthogonal forward and backward links) and one receiver. We use a hidden Markov model (HMM) to model the transmission states on the forward links. We assume that feedback is perfect
- Section III presents an analytical framework to analyze our multi-source network model and calculate its goodput. We construct our analysis in a way that it is independent of the used packet scheduling strategy.
- Section IV enumerates six packet scheduling strategies for multi-source protocols, including approaches where sources send uncoded, rateless RLNC and sliding window based RLNC encoded packet. Furthermore, we also consider a *sufficient genie* scheduling strategy that has the theoretical maximum goodput in a multi-source network. In Section IV, we also propose two *moving window* approaches on the sender side that better fit for a multi-source scenario than the conventional sliding window solution: one *moving window* without packet delay constraint and the other one is with the constraint. We also apply our goodput analysis from Section III to these scheduling strategies with both *moving window* approaches and calculate their goodput.
- Section V presents our analytical results. We also verify our analysis with extensive simulations.
- Section VI summarizes our work and suggests further research.

The significance of our work is that we show that RLNC encoded sliding window-based approach may reach the optimal goodput. The uncoded protocol also converges to the optimal goodput with the increase of the window size on the sources. Our results also show that applying rateless RLNC encoding on the transmitted data may further increase goodput. Furthermore, results also show that our approach avoids the straggler problem, thus increasing the number of

sources, increases goodput without getting limited by the weakest sources. To the best of our knowledge, this paper is the first to consider an analytical model for multi-source networks that also incorporates RLNC.

II. System model

We focus on multi-source networks, where there are N sources and only one receiver. Each source has its own link (channel), but all sources have the same original 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 source has access to all L packets, it also maintains a w-sized window, where $w \leq L$. An overview of the proposed multi-source system is shown in Fig. 1.

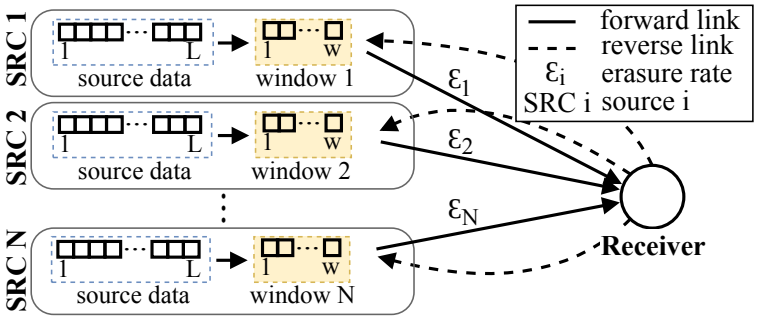

Fig. 1. Multi-source system overview.

A. Link model

Each data source has an unreliable forward link (the channel from the source to the receiver) and a lossless reverse link (the channel from the receiver to the source) 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 sources and given by $\kappa_c = \kappa_{cs\to r} + \kappa_{cr\to s}$, where $\kappa_{cs\to r}$ is the number of time slots to transmit a packet from the source to the receiver, while $\kappa_{cr\to s}$ is the number of time slots to send an ACK from the receiver to the source.

We model the transmission status with a hidden Markov model that is driven by a multi-state Markov process to make our solution applicable to different types of links, similarly to the work of Ausavapattanakun and Nosratinia [19]. At every time slot, a source sends a packet that may be delivered or lost due to erasure. The outcome of a transmission through link 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 error-free and an erroneous transmission, respectively. The link condition is modeled 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 $\boldsymbol{P}_{(i)}$. Each state $S_t^{(i)} = j, j \in S^{(i)}$ has a different error probability $\epsilon_j^{(i)}$. We denote the set of these link error probabilities by $\boldsymbol{\epsilon}^{(\mathbf{i})} = \{\epsilon_1^{(i)}, \ldots, \epsilon_{K^{(i)}}^{(i)}\}$. 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 $\{S^{(i)}, \mathcal{X}^{(i)}, \boldsymbol{P}_{(i)}, \boldsymbol{\epsilon}^{(\mathbf{i})}\}$. Furthermore $\boldsymbol{P}_{\mathrm{L},(i)} = \boldsymbol{P}_{(i)} \cdot \mathrm{diag}\{\boldsymbol{\epsilon}^{(\mathbf{i})}\}$ and $\boldsymbol{P}_{\mathrm{R},(i)} = \boldsymbol{P}_{(i)} \cdot \mathrm{diag}\{1 - \boldsymbol{\epsilon}^{(\mathbf{i})}\}$ are the probabilities of losing and receiving a packet, respectively. Note that $\boldsymbol{P}_{\mathrm{L},(i)} + \boldsymbol{P}_{\mathrm{R},(i)} = \boldsymbol{P}_{(i)}$.

In or model we consider an abstract link layer, provided on the network that can detect packet losses. The link and the the underlying network layers may or may not use additional channel coding, but we do not take that into consideration in our calculations.

B. Protocol Description

In a multi-source scenario, sources send a packet in every time slot. In out model, the receiver sends feedback to all sources, and the reverse link is perfect. Therefore, sources receive an ACK or NACK in every slot. The life cycle of a packet is the following:

1) *packet scheduling and sending*: In every time slot, a source selects a packet from their w-sized window and sends it over their link.
2) *packet arrives or gets lost*: Receiver sends feedback to all sources $\kappa_{\mathrm{cs}\to\mathrm{r}}$ time slots after the source sent the packet, independent of whether the packet gets lost or arrives at the receiver.
3) *receiving the feedback*: $\kappa_{\mathrm{cr}\to\mathrm{s}}$ time slots later, the feedback arrives at the source, which updates its window content based on the feedback.

A source selects a packet to send based on a pre-determined scheduling method that is the same for every source. 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 sources). If a subset of the sources wants to transmit packet $p_l \in \mathcal{L}$ and it gets lost on some of the links but received through at least one of the links, all sources will receive an ACK corresponding to packet p_l. Therefore, it is redundant to retransmit packet p_l on any of the links.

In our analysis, we assume that the sources 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 sources 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 since the sources decide which packet to send and not the receiver requests them. Furthermore, due to the cumulative feedback and the lack of cooperation, a source can schedule any not yet acknowledged packet without depending on other sources. Since the packet scheduling at a source is independent of the other sources, introducing a new source to the system will not limit the other sources. Thus the system avoids the straggler problem.

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

The notation of this paper is summarized in TABLE I.

III. General Analysis Approach

In this section, we describe a general framework for analyzing the overall and per link goodput of a system with N sources. We construct this framework such a way that it is independent of the applied multi-source protocol. Then, in Section IV we enumerate several multi-source protocols and use this framework to compute their goodput.

A. Round-robin source scheduling

Due to the multi-source scenario, the receiver may receive up-to N packets in each time slot (at most one from each source, but some might get lost). If two or more packets are the same (i.e., they increase the DoF at the receiver only by one), at most one of them may be useful, and the rest is duplicate. A packet is duplicate or useful depending on the order that the receiver processes them. In our model, the source of a packet is not important as long as the receiver successfully 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 sources are scheduled in a round-robin fashion. Hence, in every time slot, only one source sends a packet. A further benefit of this round-robin approach is that we can analyze each packet separately when

TABLE I

SYMBOLS LIST

Symbols definition	
N	number of sources (or servers, we use the two terms interchangeably)
k	round trip time (RTT), measured in time slots
$k_{\mathrm{r}\to\mathrm{s}}$	latency from receiver to source in time slots
$k_{\mathrm{s}\to\mathrm{r}}$	latency from source to receiver in time slots
L	number of packets
w	window size in packets
g	generation size in packets
G	number of generations
G_w	number of generations in the window
η	goodput: number of useful packets per time slots
$\eta_{(i)}$	goodput of link i
$s(t)$	transmitting server at time t
$pkt(t)$	packet arrives or gets lost at the receiver at time t
$\epsilon_{\mathrm{F}}^{(i)}$	packet loss rate on link i
$r_{(i)}$	burst error rate on link i
Events	
$E_{\mathrm{R},(t)}$	packet Received at time t
$E_{\mathrm{L},(t)}$	packet Lost at time t
$E_{\mathrm{U},(t)}$	packet is Useful at time t
$E_{\mathrm{D},(t)}$	packet is Duplicate at time t
$E_{\mathrm{F},(t)}$	packet Fail at time t
$E_{\mathrm{pU},(t)}$	packet Potentially Useful at time t
$E_{\mathrm{pD},(t)}$	packet Potentially Duplicate at time t
$\mathbf{v} \in \{0,1\}^t$	represents a series of events, where packets are received (1) or not (0) between time slots $[0,t]$
$\mathbf{a} \in \{0,1\}^k$	represents the useful packets from source $j, j \neq s(t)$ between time slots $[t-k, t-1]$
$\mathbf{b} \in \{0,1\}^k$	represents the potentially duplicate packets from source $s(t)$ between time slots $[t-k, t-1]$
$\mathbf{c} \in \{0,1\}^k$	represents the potentially duplicates in $\mathbf{b}$ that have a pair in $\mathbf{a}$.
Probabilities	
$\mathcal{P}_{\mathrm{U},(i)}$	sources i sends a useful packet
$\mathcal{P}_{\mathrm{F},(i)}$	sources i sends a duplicate packet or it gets lost
$\mathcal{P}_{\mathrm{U}}(t)$	packet at time slot t is useful
$\mathcal{P}_{\mathrm{F}}(t)$	packet at time slot t failed (duplicate or lost)
$\mathcal{P}_{\mathrm{pU}}(\mathbf{v})$	packet at time slot t is potentially useful
$\mathcal{P}_{\mathrm{pD}}(\mathbf{v})$	packet at time slot t is potentially duplicate
$\mathcal{P}_{\mathrm{past}}(\mathbf{v})$	$\mathbf{v}$ is the series of events
$\mathcal{P}_{\mathrm{outcome}}(\mathbf{v},\mathbf{a},\mathbf{b})$	$\mathbf{a}$ and $\mathbf{b}$ are the series of events
$\mathcal{P}_{\mathrm{paired}}(\mathbf{a},\mathbf{b},\mathbf{c})$	duplicates in $\mathbf{c}$ have a pair in $\mathbf{a}$
$\mathcal{P}_{\mathrm{sU}}(\mathbf{a},\mathbf{b},\mathbf{c})$	packet is useful conditioned on $\mathbf{a},\mathbf{b}$ and $\mathbf{c}$
$\mathcal{P}_{\mathrm{sD}}(\mathbf{a},\mathbf{b},\mathbf{c})$	packet is duplicate conditioned on $\mathbf{a},\mathbf{b}$ and $\mathbf{c}$

calculating the goodput. Thus a goodput increase in our model is not the result of having higher bandwidth (through more sources) but result of sending packets in a more efficient way. The RTT for this round-robin model will be: $k = N\kappa_{\mathrm{c}}$ and also $k_{\mathrm{r}\to\mathrm{s}} = N\kappa_{\mathrm{c}_{\mathrm{r}\to\mathrm{s}}}$ and $k_{\mathrm{s}\to\mathrm{r}} = N\kappa_{\mathrm{c}_{\mathrm{s}\to\mathrm{r}}}$. Fig. 2 gives an example of our round-robin transmission model.

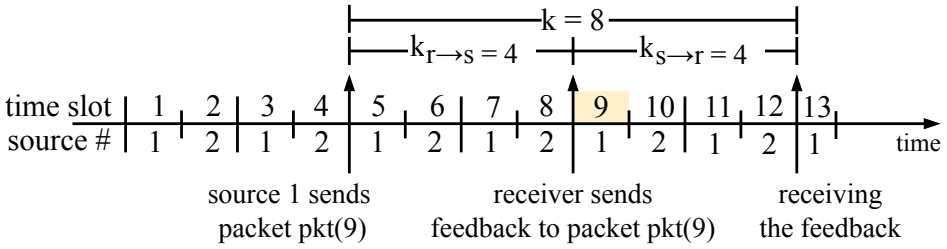

Fig. 2. Timeline example for serialized model with $N = 2$ sources and, RTT $k = 8$.

As a result of round-robin scheduling of the sources in ascending order, a packet received at time slot t is sent by source:

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

and source $s(t)$ sends:

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

B. Taxonomy of transmission events

To calculate the goodput of a multi-source system, we detail the possible outcomes of packet transmission first. In the forward link, during transmission, a packet can get:

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

During scheduling time, a source 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 received on link i and increases the DoF at the receiver,
4) $E_{\mathrm{D},(i)}$ (*duplicate*): the event that a packet is received on link 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) $E_{\mathrm{F},(i)}$ (*failed*): packet was lost, or it was received on link i, but does not increase the DoF at the receiver.

C. Technical approach

Using these events, we define the probability of a packet is useful or the transmission failed:

$$\begin{aligned} \mathcal{P}_{\mathrm{U},(i)} &= \mathrm{P}(E_{\mathrm{U},(i)}) \\ \mathcal{P}_{\mathrm{F},(i)} &= \mathrm{P}(E_{\mathrm{F},(i)}) \end{aligned} \tag{3}$$

Based on (3), we construct a signal-flow graph [20] to model the goodput of individual links. 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 that 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 [21]. 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 $\mathbb{E}[z^n]$, 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 source i and the source can update its window accordingly.

Next, we calculate the transmission time τ that we define as the number of sent packets per DoF increase at the receiver. τ can be calculated by using the matrix-generating function $\Phi_{\tau,(i)}(z)$. We get $\Phi_{\tau,(i)}(z)$ by applying basic node reduction on the MSFG, similarly to [19]:

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

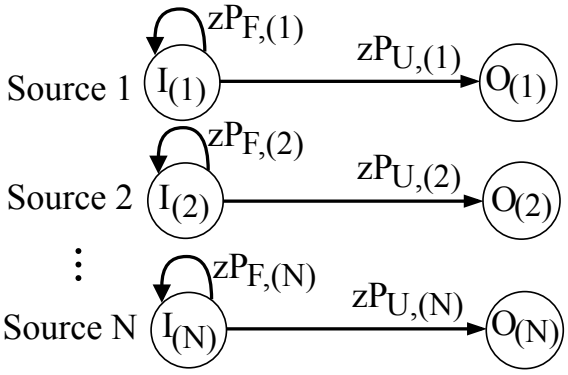

Fig. 3. Matrix signal-flow graph for goodput analysis of our serialized model.

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 $\boldsymbol{P}_{(i)}$ and can be found by solving:

$$\begin{aligned} \pi_{(\mathbf{i})}\boldsymbol{P}_{(i)} &= \pi_{(\mathbf{i})} \\ \pi_{(\mathbf{i})}\mathbf{1} &= 1, \end{aligned} \tag{5}$$

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 of $\pi_{I(i)}$ and $\mathbf{1}$, respectively:

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

The average transmission time of source i, $\overline{\tau}_{(i)}$ can be obtained by evaluating the first derivative of PGF $\phi_{\tau(i)}(z)$ at $z = 1$. The goodput, $\eta_{(i)}$ of link i is the reciprocal is the average transmission time, i.e., $\eta_{(i)} = 1/\overline{\tau}_{(i)}$[1].

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

Whether a packet $pkt(t)$ is received at time t is potentially useful depends only on the last k time slots: Packet $pkt(t)$ is sent at time $t_s = t - k_{\mathrm{s}\to\mathrm{r}}$, since the source-receiver latency is $k_{\mathrm{s}\to\mathrm{r}}$. (i) The Source, that sends $pkt(t)$, has a feedback that contains information from time $t_s - k_{\mathrm{r}\to\mathrm{s}} = t - k$, since the receiver-source latency is $k_{\mathrm{r}\to\mathrm{s}}$. (ii) Furthermore, sources can also

[1]This is indeed a lower bound due to the Jensen's inequality [22] and the convexity of $1/\overline{\tau}_{(i)}$.

keep records of previously sent packets. Since the sources may not cooperate, a source may only use information (i) and (ii) to schedule a packet for transmission.

Using the feedback from time slot $t-k$, it is guaranteed that a source will not send a packet that would be a duplicate of packets before time slot $t-k$, but it has no information about the packets after that time (i.e., packets sent between $[t-k, t-1]$). Therefore it may schedule duplicates with them. We assume that a source does not schedule packets that are duplicates with its previously sent packets[2]. Thus, a packet at time t will be duplicate only if it has the same information as any of the useful packets in the last k time slots. There may be $u \in [0, k - \frac{k}{N}]$ useful packets[3] sent by sources j, $j \neq s(t)$ between time slots $[t-k, t-1]$.

We next investigate the number of potentially duplicates sent by source $s(t)$ between time slots $[t-k, t-1]$. If a packet from source $s(t)$ is a potentially duplicate of a useful packet from any source 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 source $s(t)$ in the last k time, it will not retransmit that packet. Thus it is more likely to choose a useful packet).

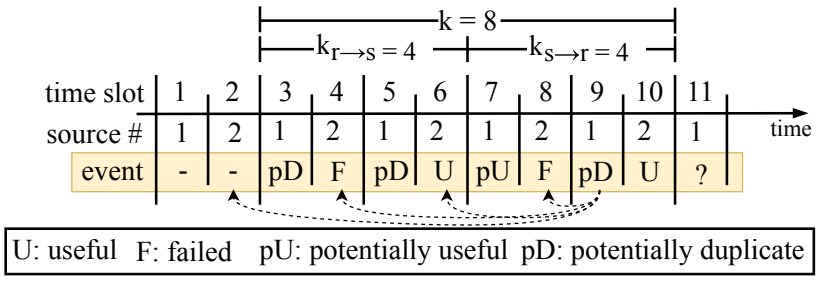

Fig. 4. Example realization to calculate $\mathcal{P}_{\mathrm{U},(i)}$ and $\mathcal{P}_{\mathrm{F},(i)}$, $N = 2$ sources and RTT $k = 8$.

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 source 1 is useful. We know that the receiver obtained $u = 2$ packets from source 2 in the last $k = 8$ time slots. The packet at time $t = 11$ may be a duplicate of any of those two useful packets. The packet at time $t = 9$ from source 1 is a potentially duplicate with any of the packets from source 2 between time slots [2,8]. If it is a

[2]Throughout our analysis, we do not use forward error correction. Therefore a source reschedules a packet only if it has received a NACK for that packet.

[3]Since $k = N\kappa_c$, $\kappa_c \in \mathbb{Z}^+$, thus $(k \bmod N) = 0$

duplicate of the packet at time $t = 6$, our investigated packet at time $t = 11$ may be a duplicate (if it is a duplicate at all) with the packet at time slot 10.

Rest of this section uses this methodology to express $\mathcal{P}_{\mathrm{U}(i)}$ and $\mathcal{P}_{\mathrm{F},(i)}$ as a function of t through several steps. At every step, we express the probability of a packet to be useful or failed. To obtain $\mathcal{P}_{\mathrm{U}(i)}$ and $\mathcal{P}_{\mathrm{F},(i)}$, we define vector $\mathbf{v} = [v_1 \dots v_t]$, the series of event, where packet are received or lost between time slots $[1, t]$. We express $\mathbf{v}$ as follows:

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

Using $\mathbf{v}$, we can define the probabilities that a packet is potentially useful or duplicate at time:

$$\begin{aligned} \mathcal{P}_{\mathrm{pU}}(\mathbf{v}) &= \mathrm{P}(E_{\mathrm{pU},(s(t))} \text{ at time } t \mid \mathbf{v}\ ) \\ \mathcal{P}_{\mathrm{pD}}(\mathbf{v}) &= \mathrm{P}(E_{\mathrm{pD},(s(t))} \text{ at time } t \mid \mathbf{v}\ ) \end{aligned} \tag{8}$$

Furthermore, we define $\mathcal{P}_{\mathrm{past}}(v_1 \dots v_{t-1})$ as the probability of $[v_1 \dots v_{t-1}]$ is the series of events between time slots $[1, t-1]$:

$$\begin{aligned} \mathcal{P}_{\mathrm{past}}(v_1 \dots v_{t-1}) &= \mathrm{P}(v_1 \dots v_{t-1}) \\ &= \prod_{i=1}^{N} \prod_{\substack{l=i \\ (l-i \bmod N)=0}}^{t-1} \pi_{(i)} \boldsymbol{P}_{\mathrm{R},(i)}^{v_l} \boldsymbol{P}_{\mathrm{L},(i)}^{|1-v_l|} \mathbf{1}, \end{aligned} \tag{9}$$

where $\pi_{(i)}$ is the stationary vector of $\boldsymbol{P}_{(i)}$ and $\mathbf{1}$ is a column vector of ones, as defined above.

Note that all probabilities that are conditioned on $\mathbf{v}$, now implicitly depend on the source, since $\mathbf{v}$ is the input of the function and only source $s(t)$ may transmit at time t. Therefore, we will omit the source index from $\mathcal{P}_{\mathrm{U}(i)}(t)$ and $\mathcal{P}_{\mathrm{F},(i)}(t)$ and express them as follows:

$$\begin{aligned} \mathcal{P}_{\mathrm{U}}(t) &= \sum_{v \in \{0,1\}^t} \mathcal{P}_{\mathrm{pU}}(\mathbf{v}) \mathcal{P}_{\mathrm{past}}(v_1 \dots v_{t-1}) \boldsymbol{P}_{\mathrm{R},(s(t))} v_t \\ \mathcal{P}_{\mathrm{F}}(t) &= \sum_{v \in \{0,1\}^t} \mathcal{P}_{\mathrm{pD}}(\mathbf{v}) \mathcal{P}_{\mathrm{past}}(v_1 \dots v_{t-1}) \boldsymbol{P}_{\mathrm{R},(s(t))} v_t + (1 - v_t) \boldsymbol{P}_{\mathrm{L},(s(t))}, \end{aligned} \tag{10}$$

where we consider all possible series of packet lose or receive events and calculate $\mathcal{P}_{\mathrm{U}}(t)$ as the probability of a packet is potentially useful and it was received. We get $\mathcal{P}_{\mathrm{F}}(t)$ by calculating the probability of a packet potentially duplicate, and it was received, or it was lost.

As shown previously in Fig. 4, to calculate if a packet is potentially useful or duplicate at time t, we need to consider the events in the last k time slots, therefore we define the following quantities:

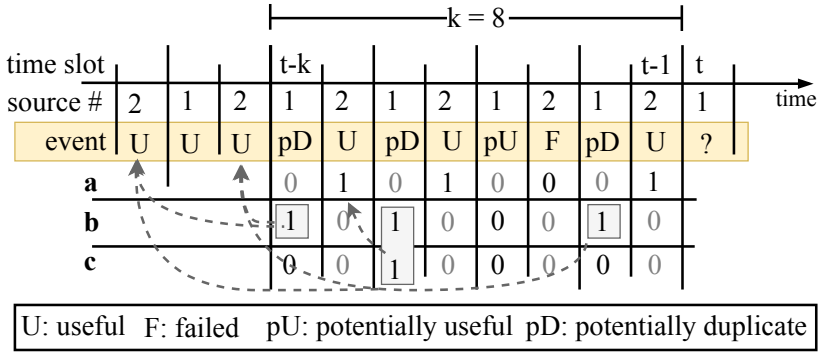

Fig. 5. Example configuration of vectors $\mathbf{a}, \mathbf{b}$ and $\mathbf{c}$ with $N = 2$ sources and RTT $k = 8$.

1) $\mathbf{a} \in \{0,1\}^k$ (useful vector): indicates if a packet from any source $j, j \neq s(t)$ was useful or failed in the last k time slots.
2) $\mathbf{b} \in \{0,1\}^k$ (duplicates vector): indicates if source $s(t)$ sent a potentially duplicate packet in the last k time slots. Note that $\mathbf{b}$ does not specify if the duplicate packet is duplicate with a useful packet from time slots $[k-t, t-1]$ or before that time.
3) $\mathbf{c} \in \{0,1\}^k$ (matched duplicates vector): indicates if source $s(t)$ sent a potentially duplicate packet in the last k time slots and that potentially duplicate packet is a duplicate with a useful packet from time slots $[k-t, t-1]$ (i.e., for every 1 in $\mathbf{c}$, there is a unique pair in $\mathbf{a}$). Note that c_i is 1 if and only if $b_i = 1$. Fig. 5 shows a possible configuration of vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}$.
4) $\mathcal{P}_{\text{paired}}(\mathbf{a}, \mathbf{b}, \mathbf{c})$ represents the probability that $\mathbf{a}, \mathbf{b}, \mathbf{c}$ are valid, i.e.: all potentially duplicate packets in $\mathbf{b}$ have a useful packet pair, while duplicates in $\mathbf{c}$ has a useful packet pair in $\mathbf{a}$.
5) $\mathcal{P}_{\text{outcome}}(\mathbf{v}, \mathbf{a}, \mathbf{b})$ represents probability that events $\mathbf{a}$ and $\mathbf{b}$ occur, conditioned on $\mathbf{v}$.
6) $\mathcal{P}_{\text{sU}}(\mathbf{a}, \mathbf{b}, \mathbf{c})$ and $\mathcal{P}_{\text{sD}}(\mathbf{a}, \mathbf{b}, \mathbf{c})$ represent the probabilities of a packet is useful or duplicate, respectively, conditioned on events $\mathbf{a}, \mathbf{b}$ and $\mathbf{c}$.

We define these quantities formally as follows:

$$
\begin{aligned}
\mathbf{a} \in \{0,1\}^k,\, a_j &= \begin{cases} \text{'undefined'} & \text{if } s(t-j) = s(t) \\ 1 & \text{else if } E_{\mathrm{U},(s(t-j))} \\ 0 & \text{otherwise} \end{cases} \\
\mathbf{b} \in \{0,1\}^k,\, b_i &= \begin{cases} \text{'undefined'} & \text{if } s(t-i) \neq s(t) \\ 1 & \text{else if } E_{\mathrm{pD},(s(t-i))} \\ 0 & \text{otherwise} \end{cases} \\
\mathbf{c} \in \{0,1\}^k,\, c_i &= \begin{cases} \text{'undefined'} & \text{if } s(t-i) \neq s(t) \\ 1 & \text{else if } E_{\mathrm{pD},(s(t-i))}, \text{ and} \\ & \exists j, 0 \leq j < i,\, E_{\mathrm{U},(s(t-j))}, \\ & pkt(t-i) \equiv pkt(t-j) \\ 0 & \text{otherwise,} \end{cases}
\end{aligned}
\tag{11}
$$

where *'undefined'* means that the source at that time slot is not active, 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., receiving both packets would only increase the DoF at the receiver by at most one.

We formally define $E_{\mathrm{paired}}(\mathbf{a}, \mathbf{b}, \mathbf{c})$, the event that there is a useful packet with a given $\mathbf{a} = [a_1, \ldots, a_k]$ for every potentially duplicate packet in $\mathbf{b} = [b_1, \ldots, b_k]$, as follows:

$$
\begin{aligned}
E_{\mathrm{paired}}(\mathbf{a}, \mathbf{b}, \mathbf{c}) = \forall i, c_i = 1 : \exists j, j < i, a_j = 1 \\
pkt(t-i) \equiv pkt(t-j).
\end{aligned}
\tag{12}
$$

Using $E_{\mathrm{paired}}(\mathbf{a}, \mathbf{b}, \mathbf{c})$, we define its probability, $\mathcal{P}_{\mathrm{paired}}(\mathbf{a}, \mathbf{b}, \mathbf{c})$ as follows:

$$
\mathcal{P}_{\mathrm{paired}}(\mathbf{a}, \mathbf{b}, \mathbf{c}) = \mathrm{P}(E_{\mathrm{paired}}(\mathbf{a}, \mathbf{b}, \mathbf{c})). \tag{13}
$$

We also define $\mathcal{P}_{\mathrm{outcome}}(\mathbf{v}, \mathbf{a}, \mathbf{b})$, the probability of events $\mathbf{a}$ and $\mathbf{b}$ occur between time slots $[t-k, t-1]$, conditioned on $\mathbf{v}$:

$$
\begin{aligned}
\mathcal{P}_{\mathrm{outcome}}(\mathbf{v}, \mathbf{a}, \mathbf{b}) = \mathrm{P}(&\forall a_j = 1,\, E_{\mathrm{U},(s(t-j))}, \\
&\forall a_j = 0,\, E_{\mathrm{D},(s(t-j))}, \\
&\forall b_i = 1,\, E_{\mathrm{pD},(s(t-i))} \mid E_{\mathrm{paired}}(\mathbf{a}, \mathbf{b}, \mathbf{c}), \mathbf{v}\ )
\end{aligned}
\tag{14}
$$

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

$$\begin{aligned}\mathcal{P}_{\mathrm{sU}}(\mathbf{a},\mathbf{b},\mathbf{c}) &= \mathrm{P}(E_{\mathrm{U},(t)} \mid E_{\mathrm{paired}}(\mathbf{a},\mathbf{b},\mathbf{c}) \ )\\ \mathcal{P}_{\mathrm{sD}}(\mathbf{a},\mathbf{b},\mathbf{c}) &= \mathrm{P}(E_{\mathrm{D},(t)} \mid E_{\mathrm{paired}}(\mathbf{a},\mathbf{b},\mathbf{c}) \ )\end{aligned} \tag{15}$$

Using eqs. (11) and (13) to (15), we can express the probabilities of a packet at time t is potentially useful or duplicate, respectively:

$$\begin{aligned}\mathcal{P}_{\mathrm{pU}}(\mathbf{v}) &= \sum_{u=0}^{k-\frac{k}{N}} \sum_{d=0}^{\frac{k}{N}} \sum_{m=0}^{\min(d,u)} \sum_{\substack{\sum a_j = u \\ \sum b_i = d \\ \sum c_i = m,\, c_i \le b_i}} \mathcal{P}_{\mathrm{paired}}(\mathbf{a},\mathbf{b},\mathbf{c})\, \mathcal{P}_{\mathrm{sU}}(\mathbf{a},\mathbf{b},\mathbf{c}) \mathcal{P}_{\mathrm{outcome}}(\mathbf{v},\mathbf{a},\mathbf{b})\\ \mathcal{P}_{\mathrm{pD}}(\mathbf{v}) &= \sum_{u=0}^{k-\frac{k}{N}} \sum_{d=0}^{\frac{k}{N}} \sum_{m=0}^{\min(d,u)} \sum_{\substack{\sum a_j = u \\ \sum b_i = d \\ \sum c_i = m,\, c_i \le b_i}} \mathcal{P}_{\mathrm{paired}}(\mathbf{a},\mathbf{b},\mathbf{c})\, \mathcal{P}_{\mathrm{sD}}(\mathbf{a},\mathbf{b},\mathbf{c}) \mathcal{P}_{\mathrm{outcome}}(\mathbf{v},\mathbf{a},\mathbf{b}).\end{aligned} \tag{16}$$

$\mathcal{P}_{\mathrm{paired}}(\mathbf{a},\mathbf{b},\mathbf{c})$ in (16) can be expressed in the following way:

$$\mathcal{P}_{\mathrm{paired}}(\mathbf{a},\mathbf{b},\mathbf{c}) = \prod_{\substack{l=1 \\ s(t-l)=s(t) \\ b_l=1}}^{k} \left| c_l - \frac{\sum_{j=1}^{l} a_j - \sum_{i=1}^{l-N} c_i}{k - \frac{k}{N}} \right|. \tag{17}$$

We get every l, where $b_l = 1$ and we calculate the probability that every duplicate ($c_i = 1$) has a pair in $\mathbf{a}$, ($a_j = 1,\ j < i$), such that $a_j = 1$ was not paired previously. Furthermore if $c_i = 0$, but $b_i = 1$, we consider the probability of packet $pkt(t-(k-i))$ is duplicate with a useful packet between time slots $[1, t-k-1]$.

$\mathcal{P}_{\mathrm{outcome}}(\mathbf{v},\mathbf{a},\mathbf{b})$ in (16) can be expressed in the following way:

$$\begin{aligned}\mathcal{P}_{\mathrm{outcome}}(\mathbf{v},\mathbf{a},\mathbf{b}) = &\prod_{\substack{j=1 \\ s(t-j)\neq s(t)}}^{k} \left(\mathcal{P}_{\mathrm{pU}}(\mathbf{v}_{t-j}) v_{t-j}\right)^{a_j} \left(\mathcal{P}_{\mathrm{pD}}(\mathbf{v}_{t-j}) v_{t-j} + (1 - v_{t-j})\right)^{|1-a_j|}\\ &\prod_{\substack{i=1 \\ s(t-i)=s(t)}}^{k} \mathcal{P}_{\mathrm{pD}}(\mathbf{v}_{t-i})^{b_i} \mathcal{P}_{\mathrm{pU}}(\mathbf{v}_{t-i})^{|1-b_i|}\\ \mathbf{v}_{t-l} = &[v_1 \dots v_{t-l}],\end{aligned} \tag{18}$$

Where the first row of the equation calculates the probability of source $j, j \neq s(t)$ sent a useful packet or the transmission failed. The second row calculates the probability of source $s(t)$ sent a potentially useful or duplicate packet.

To calculate $\mathcal{P}_{\mathrm{sU}}(\mathbf{a},\mathbf{b},\mathbf{c})$ and $\mathcal{P}_{\mathrm{sD}}(\mathbf{a},\mathbf{b},\mathbf{c})$, one also has to consider the applied packet scheduling strategy. We detail that in the next section.

Furthermore, our matrix-flow graph approach to calculate the 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. Multi-source protocols

In this section, we enumerate several packet scheduling strategies for the multi-source protocols. We calculate $\mathcal{P}_{\mathrm{sU}}(\mathbf{a},\mathbf{b},\mathbf{c})$ and $\mathcal{P}_{\mathrm{sD}}(\mathbf{a},\mathbf{b},\mathbf{c})$, that are required to calculate the potential duplicate and useful probabilities in (16), corresponding to a given scheduling method.

As described in Section II, sources maintain a w-sized window. Most of the SR ARQ models use a sliding window approach to send packets. Packets with the lowest packet id are chosen from the window for transmission. If a packet with the lowest id gets successfully transmitted, it can be removed from the window, and the window can slide to include new packets. In a multi-source scenario, we cannot use this conventional sliding window, since in that case, all servers would send the same packet. Therefore, in this paper, we consider a *moving window* setup. In a *moving window* setup, any packet can be chosen for scheduling (i.e., the transmitted packets do not have to be in consecutive order). We differentiate between *strict moving window* and *sparse moving window*. Fig. 6 shows an example of the two *moving window* approaches.

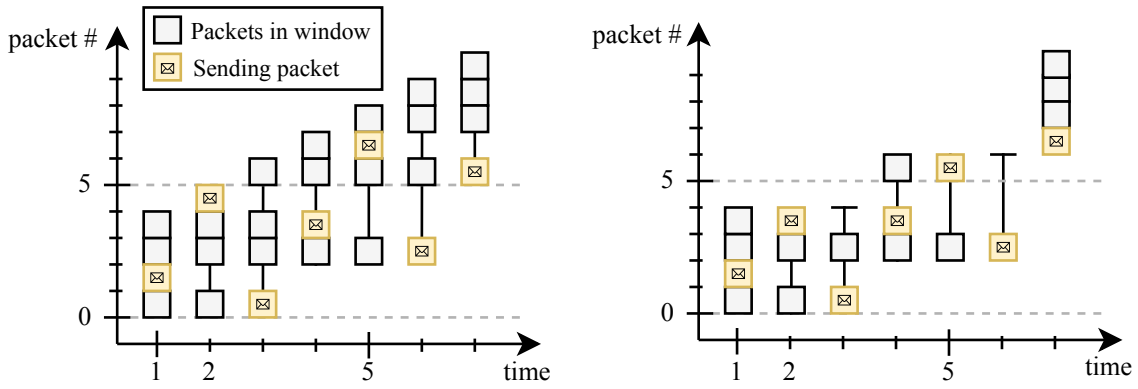

Fig. 6. Example of sparse moving window (left) and strict moving window (right).

a) Sparse moving window: 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[4]. 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. Note that this window does not have any constraint on the packet delay.

b) Strict moving window: To introduce a constraint on packet delay to *sparse moving window*, we define $W(t)$, the set of packets in a *strict moving window* at time t the following way:

$$\begin{aligned} W(t) &= \{i \in \mathcal{L} \mid w_{\min} \leq i \leq w_{\min} + w - 1\} \\ \mathcal{L} &= \{0, \ldots, L\} \\ \mathcal{L}_{\text{acked}}(t) &= \{i \in \mathcal{L} \mid \text{ packet } i \text{ was acknowledged by time } t\} \\ w_{\min} &= \min(\mathcal{L} \backslash \mathcal{L}_{\text{acked}}(t)), \end{aligned} \tag{19}$$

where $\mathcal{L}$ is the set of the original data packets, $\mathcal{L}_{\text{acked}}(t)$ is the set of all successfully received and acknowledged packets by time t and $w_{\min}$ is a not-yet-acknowledged packet with the lowest index.

Using our window models, we consider the following scheduling approaches: 1) *sufficient genie* that gives the optimal scheduling with the given link conditions, 2) *uncoded sequential*, which is a simple approach for packet scheduling, 3) *uncoded random*, that aim is to exploit the multi-source network in a more efficient way than the *uncoded sequential*, 4) *RLNC rateless coded* that applies network coding to a group of source packets and those coded packets travel the network, 5) *RLNC sliding window* that uses network coding in a sliding window fashion to create coded packets.

A. *Sufficient genie*

We introduce a *sufficient genie* scheduling strategy to find the optimal goodput of a system with the given link properties. It is not a *full genie* since it only focuses on sending the perfect packet (i.e., the most useful packet), but packets might get lost on the link. The goodput is the same for both *moving* window approaches:

$$\begin{aligned} \mathcal{P}_{\text{sU}}(\mathbf{a}, \mathbf{b}, \mathbf{c}) &= 1 \\ \mathcal{P}_{\text{sD}}(\mathbf{a}, \mathbf{b}, \mathbf{c}) &= 0. \end{aligned} \tag{20}$$

[4]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.

Using a genie, the source-link pairs can be decoupled and analyzed independently. The goodput of source i only depends on the loss probability $\boldsymbol{P}_{\mathrm{L},(i)}$. Following the steps in [19], the goodput of a link i is:

$$\eta_{(i)} = 1 - \epsilon_{\mathrm{F}}^{(i)} = 1 - \pi_{(i)}\boldsymbol{P}_{\mathrm{L},(i)}\mathbf{1}. \tag{21}$$

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

B. Uncoded sequential

In the case of *uncoded sequential*, a source sends packets in a sequential fashion. If a packet gets lost on all the links, each source reschedules that packet. This scheduling strategy uses the window in a first in, first out (FIFO) fashion. The first packet of the window gets scheduled, while if a packet is lost it is added to the end of the window. Since we assume that the sources are scheduled in a round-robin fashion, every source transmits the same packet between time slots $[x, x+N]$, $x \in \mathbb{Z}^+$, $(x \bmod N) = 0$. Therefore, only one of the packets may be useful in that interval. A packet is useful with probability 1 if no useful packet was received in the last $s(t) - 1$ time slots:

$$\mathcal{P}_{\mathrm{sU}}(\mathbf{a}, \mathbf{b}, \mathbf{c}) = 1 - \mathcal{P}_{\mathrm{sD}}(\mathbf{a}, \mathbf{b}, \mathbf{c}) = \left| (1 - \min(\sum_{j=k-s(t)-1}^{k} a_j, 1)) \right|, \tag{22}$$

where $|x|$ is the absolute value of x. This also means that if $s(t) = 1$ the probability is always 1. Therefore the maximum goodput of the *uncoded sequential* approach is $\eta = \sum_{i=1}^{N} \eta_{(i)} \leq 1$.

Uncoded sequential behaves similarly with both window approaches. In the case of *strict moving window*, the window may get empty. In our model we assume that the window never gets empty. Therefore, (22) will not hold for *strict moving window*. The window can only get empty if the same packet gets lost repeatedly from all the sources, while all the other packets are transmitted. This has a low probability. Thus we can use (22) to estimate the goodput of the *uncoded sequential* approach with a *strict moving window*. We use simulations to verify this estimation in Section V.

C. Uncoded random

In this approach, sources select a not-in-flight[5] packet uniformly at random from their send window for transmission. The probabilities $\mathcal{P}_{\text{sU}}(\mathbf{a},\mathbf{b},\mathbf{c})$ and $\mathcal{P}_{\text{sD}}(\mathbf{a},\mathbf{b},\mathbf{c})$ depend on the window model:

a) Sparse moving window: The probabilities can be expressed the following way:

$$\begin{aligned}\mathcal{P}_{\text{sU}}(\mathbf{a},\mathbf{b},\mathbf{c}) = 1 - \mathcal{P}_{\text{sD}}(\mathbf{a},\mathbf{b},\mathbf{c}) &= \frac{w - (\frac{k}{N} - (d-m)) - (u-m)}{w - (\frac{k}{N} - (d-m))} \\ u &= \sum\nolimits_{j=1}^{k} a_j \\ d &= \sum\nolimits_{i=1}^{k} b_i \\ m &= \sum\nolimits_{i=1}^{k} c_i \end{aligned} \tag{23}$$

where $\frac{k}{N}$ is the number of in-flight packets from source $s(t)$. u is the number of useful packets sent by source j, $j \neq s(t)$. $d \leq \frac{k}{N}$ is the number of duplicates sent by source $s(t)$, and $m < d$ is the number of duplicates sent by source $s(t)$ and these packets are duplicates with packets in the last k time slot. $\frac{k}{N}$ number packets are *locked* in the window as they cannot scheduled since the source is waiting for a feedback about them. $(d-m)$ number of packets are duplicates with useful packets between time slots $[1, t-k-1]$. The source already has a feedback about those useful packets. Therefore, the source already know that $(d-m)$ of these *locked* packets can be removed from the window and new packets can be added to the window. Finally, a packet at time t can be duplicate with any of the useful packets in $\mathbf{a}$ that does not have a duplicate in $\mathbf{c}$. There are $(u-m)$ such packets in total.

b) Strict moving window: While the number of packets in the window is constant and equals to w in the case of *sparse moving window*, in the case of *strict moving window*, the number of packets might change over time: $w(t) \in \{1,\ldots,w\}$. Since our analysis assumes that $\mathcal{P}_{\text{sU}}(\mathbf{a},\mathbf{b},\mathbf{c})$ and $\mathcal{P}_{\text{sD}}(\mathbf{a},\mathbf{b},\mathbf{c})$ only depend on the last k time slots and not all t time slots, our model cannot be applied for the *strict moving window*. Therefore, we can only estimate the goodput in this case. We use simulations to verify our estimation in Section V.

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

We estimate the $\mathcal{P}_{\text{sU}}(\mathbf{a},\mathbf{b},\mathbf{c})$ and $\mathcal{P}_{\text{sD}}(\mathbf{a},\mathbf{b},\mathbf{c})$ probabilities by assuming that the number of packets in the window is independent of t and can be represented with a random variable $\mathcal{W}$:

$$\begin{aligned}
&\mathcal{P}_{\text{sU}}(\mathbf{a},\mathbf{b},\mathbf{c}) = 1 - \mathcal{P}_{\text{sD}}(\mathbf{a},\mathbf{b},\mathbf{c}) = \\
&\qquad \sum_{w_a=w_{min}}^{w} P(\mathcal{W}=w_a)\frac{1}{\sum_{w'_a=w_{min}}^{w} P(\mathcal{W}=w'_a)}\frac{w_a-(\frac{k}{N}-(d-m))-(u-m)}{w_a-(\frac{k}{N}-(d-m))} \\
&\qquad u = \sum\nolimits_{j=1}^{k} a_j \\
&\qquad d = \sum\nolimits_{i=1}^{k} b_i \\
&\qquad m = \sum\nolimits_{i=1}^{k} c_i \\
&\qquad w_{min} = max\left(1, u, (\frac{k}{N}-(d-m))+(u-m)\right),
\end{aligned} \tag{24}$$

where w_{min} is the minimum value that the window may have. It has to be at least 1 (otherwise it would move and include new packets). $w_{min} \geq u$ as, sources $j, j \neq s(t)$ were able to send u useful packets from the same window. $w_{min} \geq (\frac{k}{N}-(d-m))+(u-m)$ as the probability should be between $[0,1]$.

The distribution of $\mathcal{W}$ can be obtained by representing the *strict moving window* model as a Markov process. The states of the process are the possible window configurations. Using the transition probability matrix of the process, the stationary distribution can be calculated. This stationary distribution can be used to calculate the probability that the window contains $w_a \in [1,w]$ packets. The detailed analysis of the distribution of the *strict moving window* is left as future work. Based on our window model, we build a simulator, and we use empirical distribution for our calculations that we demonstrate in Section V.

One should note, that if $w = 1$, *uncoded random* and *sequential* approaches have the same goodput for both window types. Furthermore, as the *sufficient genie* provides an upper bound of the goodput, the *uncoded sequential* is a lower bound for the *uncoded random* approach.

D. Rateless RLNC coded

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 [23].

In this protocol, 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 source 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 [24]. 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 source window contains $G_w = \frac{w}{g}$ *generations*[6]. In every time slot, a source chooses one *generation* from its window to create an encoded packet and sends it over the link. In this paper, we investigate a *random* generation selection strategy and a *rarest first* generation selection strategy. In both cases, $\mathcal{P}_{\mathrm{sU}}(\mathbf{a}, \mathbf{b}, \mathbf{c})$ and $\mathcal{P}_{\mathrm{sD}}(\mathbf{a}, \mathbf{b}, \mathbf{c})$ depend on the probability of source $s(t)$ 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 RLNC in a multi-source environment through simulations in Section V.

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

2) Rarest first generation selection: Sources approximate the rank of the *generations* at the receiver and choose the one that has the least rank. The approximation is based on two components: 1) the feedback that represents the receiver state $k_{\mathrm{r}\to\mathrm{s}}$ time slots ago, 2) the sent packets by that given source. We call this strategy *rarest generation first* strategy, referring to the *rarest piece first* algorithm in *BitTorrent* [25].

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* strategy. 2) if $L = w = g$, the goodput will be identical with the *sufficient genie* approach, since all received packet will be useful.

E. Coded sliding window

In the case of the network coded sliding window [23] approach, a source 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:

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

$$\mathcal{P}_{\mathrm{sU}}(\mathbf{a},\mathbf{b},\mathbf{c}) = 1 - \mathcal{P}_{\mathrm{sD}}(\mathbf{a},\mathbf{b},\mathbf{c}) = \begin{cases} 1 & \text{if } (t \bmod k) < w \\ 0 & \text{otherwise} \end{cases}. \tag{25}$$

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

Comparing this solution to the rateless RLNC coded approaches (IV-D), sliding window achieves optimal performance with coding less or equal packets together, thereby using fewer CPU cycles since we usually have $k << L$. On the other hand, with rateless coding, the random seed can be shared between the source and the receiver, while with the sliding window the coefficient vector needs to travel in the packet payload. Furthermore, with rateless coding, the generation size can vary between $[2, w]$, while in case of the sliding window, coding is done over the w-sized window, which should be $w \geq k$. In the case of high RTT κ_{c} or high number of sources, N, k is high which can lead to high computational overhead.

V. Numerical results

We computed the numerical results for our model by using a two-state Gilbert-Elliot (GE) link model [26] for the forward link of the sources. The state-transition matrix of the link is given by:

$$\boldsymbol{P}_{(i)} = \begin{bmatrix} 1 - q_{(i)} & q_{(i)} \\ r_{(i)} & 1 - r_{(i)} \end{bmatrix}, \tag{26}$$

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

We use our simulator testbed to analyze the goodput of our multi-source protocols, presented in Section IV. Each simulation was run 1000 times, and an average is calculated from them. We compare our simulations, and numerical results both for sparse and strict moving window approaches, and our results demonstrate that they show similar trends. Furthermore, we also compare our results to our previous measurement-based empirical results [9]. They also show similar trends.

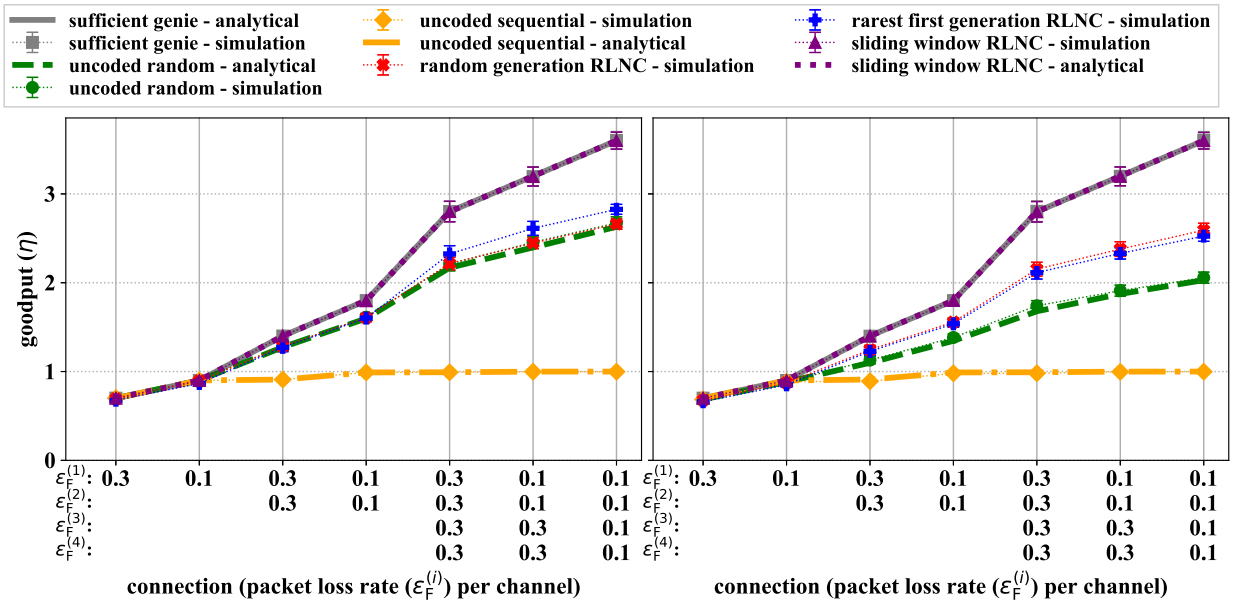

Fig. 7. Goodput for sparse moving (left) and strict moving (right) window for sources $N \in \{1,2,4\}$, RTT $\kappa_c = 3$, window size $w = 24$, generation size $g = 12$ and burst rate $r = 0.3$.

a) Per source count: Increasing the number of sources increases the overall goodput with both window models. As described in Section IV-B, the maximum goodput for the *uncoded sequential* approach is $\eta = 1$ (but it also cannot exceed the goodput of the *sufficient genie*). The *uncoded sequential* already approaches its theoretical maximum $\eta = 1$ goodput in case of using only $N = 2$ sources.

In the case of *uncoded random* and the *rateless RNLC coded* approaches, the number of duplicates increases with the number of sources. Therefore, these protocols cannot achieve the goodput of the *sufficient genie* for sources $N \geq 2$. Using sparse moving window, the performances of *uncoded random* and *rateless RNLC coded* approaches follow similar trends. In contrast to that, using a strict moving window to restrict the network delay, *uncoded random* approach has a significant goodput drop compared to the *rateless RNLC coded* approach. These results show that RLNC coding can decrease the packet delay in a multi-source network (or increase the goodput, while having the same delay constraints as the *uncoded random*). Our results suggest that we can achieve optimal goodput by using *RLNC coded sliding window* approach.

b) Per window size: Fig. 8 shows the impact of the window size on the goodput. *Sufficient genie* and the *coded sliding window* have the achievable maximum goodput that is independent of

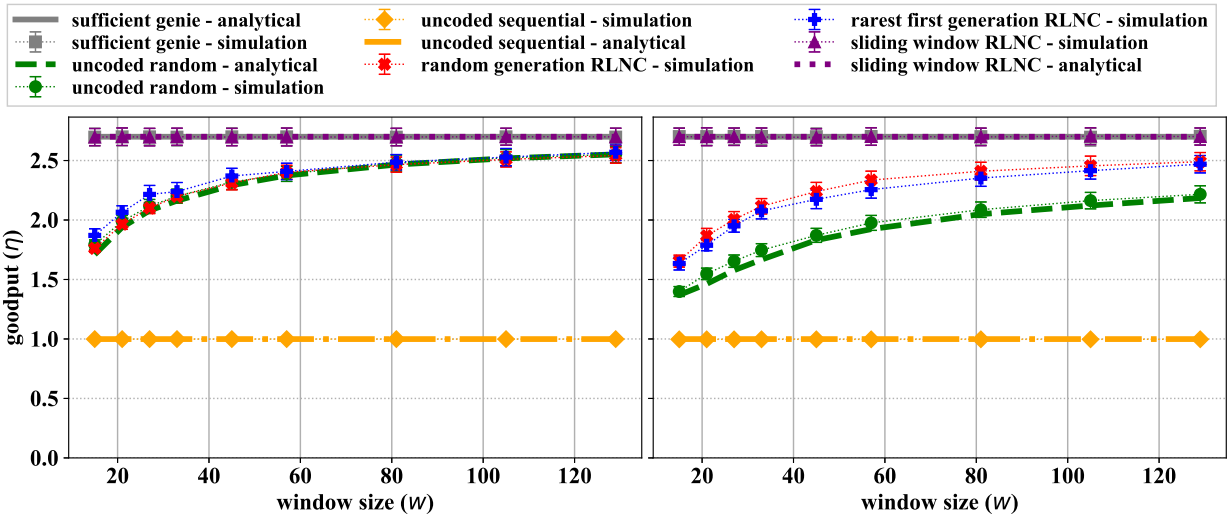

Fig. 8. Goodput for sparse moving (left) and strict moving (right) window for $N = 3$ sources, packet loss rate $\epsilon_{\mathrm{F}}^{(i)} = 0.1$, generation size $g = \frac{w}{3}$, burst rate $r = 0.3$ and RTT $\kappa_{\mathrm{c}} = 4$.

the window size (assuming that $w \geq k$). *Uncoded sequential* is also independent of the window size and provides a lower-bound for the other downloading approaches. *Uncoded random* and *rateless RLNC coded* approaches tend to increase goodput as the window size increases since the sources have a larger set of packets to choose from. Fig. 8 also shows that RLNC coding can decrease the packet delay in a multi-source network or increase goodput while having the same delay constraint. As the window size increases the goodput of the *uncoded random* and *rateless RLNC coded* approaches go to the maximum achievable goodput and the advantage of the *rateless RLNC coded* compared to the *uncoded random* also disappears.

c) Per generation size: Fig. 9 presents the goodput of the network with regard to generation size. It also includes the non-RLNC scheduling approaches for reference. The figure shows that with the increase of the generation size the goodput also increases. This trend is more clear in the case of using a strict moving window. *Rarest generation first* approach with sparse moving window has a local maximum at $g = N$ and a minimum at $g = N + 1$. We interpret this as if $g = N$; every source sends exactly one packet from each generation. In contrast to that, if $g = N + 1$, first every source sends exactly one packet from each generation then competes for the last packet. This competition increases the probability of sending a duplicate packet.

d) Per RTT: Higher RTT values have a negative impact on goodput, as Fig. 10 shows. *Rateless RLNC* and *uncoded random* approaches have decreased goodput with higher RTT

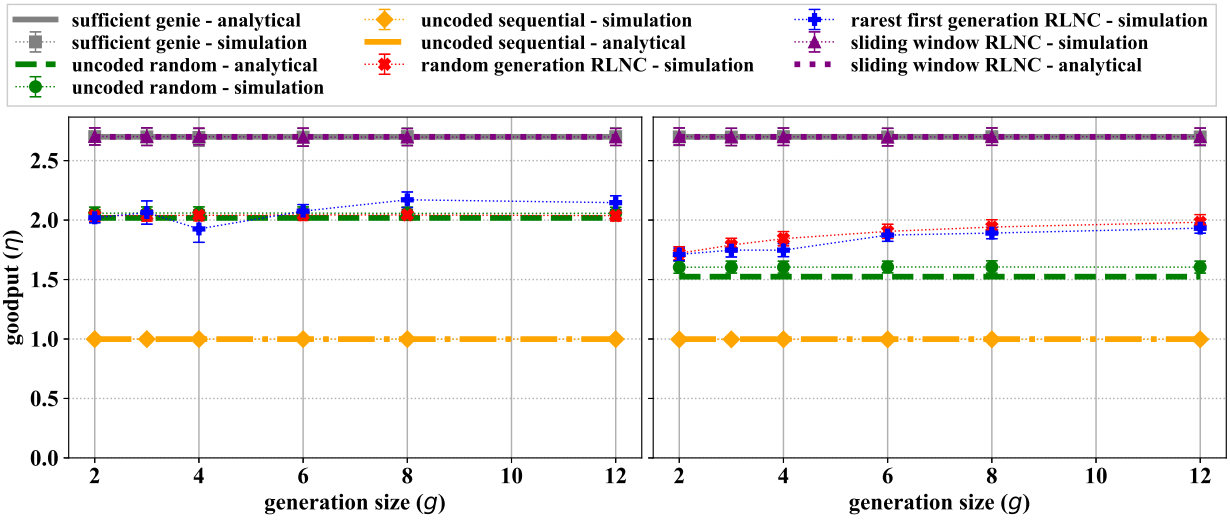

Fig. 9. Goodput for sparse moving (left) and strict moving (right) window per generation size for $N = 3$ sources, packet loss rate $\epsilon_{\mathrm{F}}^{(i)} = 0.1$, window size $w = 24$, burst rate $r = 0.3$ and RTT $\kappa_{\mathrm{c}} = 4$.

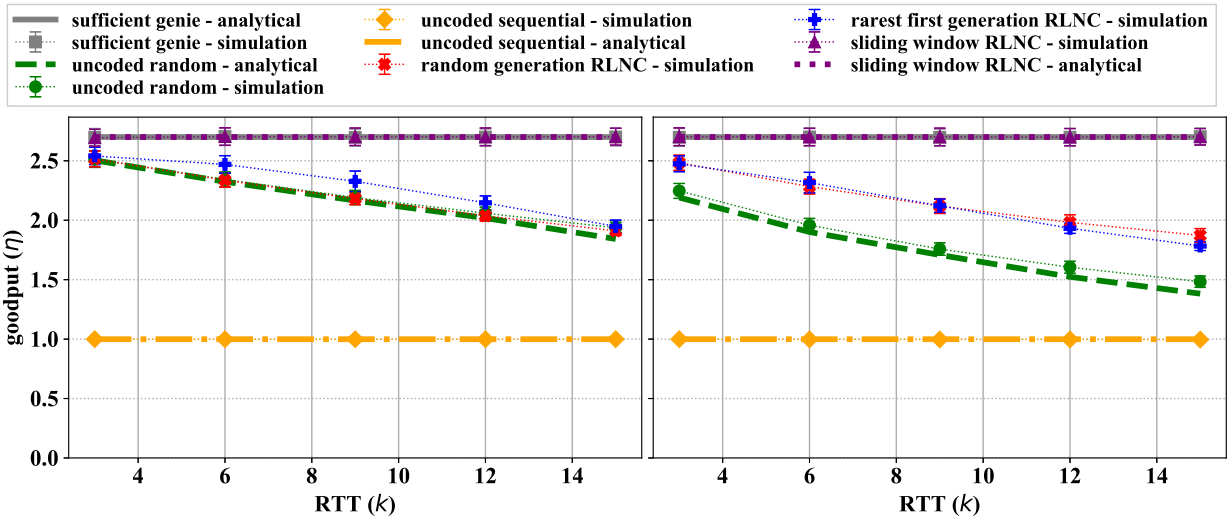

Fig. 10. Goodput for sparse moving (left) and strict moving (right) window for $N = 3$ sources, packet loss rate $\epsilon_{\mathrm{F}}^{(i)} = 0.1$, window size $w = 24$, generation size $g = 12$ and burst rate $r = 0.3$.

values. Adding a delay constraint to the network with a strict moving window, the goodput decrease is even more significant. The *rateless RLNC* approaches have a similar goodput, but in the case of a strict moving window, *rarest generation* based scheduling slightly outperforms the *random generation* based scheduling. In contrast to that, using sparse moving window, the *random generation* approach has a better performance. The value of RTT does not influence the

uncoded sequential scheduling (as long as $w \geq k$).

e) Error burst: We have also investigated the impact of the error bursts (i.e., using low $r_{(i)}$ values in (26)). As shown by D. L. Lu and J. F. Chang in [27], the error burst does not effect the goodput of a single-link SR-ARQ network if the feedback is reliable. According to our observations, this holds in our multi-source network model as well.

Fig. 11 shows simulations results for networks with lossy feedback links. We used the same GE link setup for both forward and reverse link. Sources also tracked the send time of each packet. After T (timeout) time slots, if no feedback was received for a packet, the source reschedules the packet.

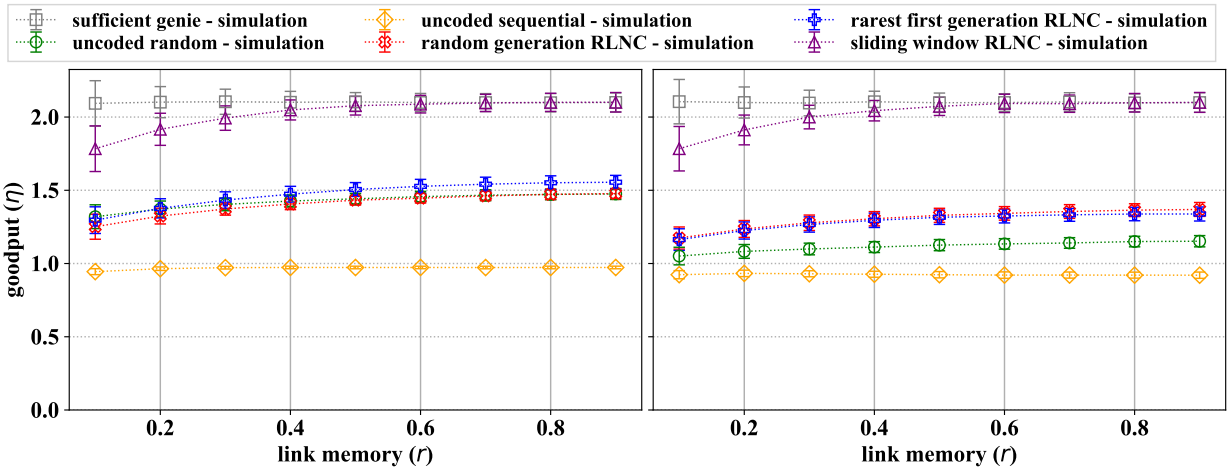

Fig. 11. Goodput with lossy feedback link for sparse moving (left) and strict moving (right) window for $N = 3$ sources, packet loss rate $\epsilon_{\mathrm{F}}^{(i)} = 0.3$, window size $w = 24$, generation size $g = 12$, timeout $T = 15$ and RTT $\kappa_{\mathrm{c}} = 4$.

Fig. 11 shows that *genie* is not effected by the burst error. *Uncoded sequential* stays also constant with respect to $r_{(i)}$, but it can never reach its theoretical maximum value 1. *Coded sliding window* has a significant performance drop on high burst errors ($r_{(i)}$ has a small value). Due to feedback losses, the window cannot slide. Therefore the source keeps sending linear combinations of the same packets. The lack of information, because of the missing feedback leads to a goodput drop in the case of *uncoded random* and *rateless RLNC coded* approaches as well.

Our simulations can be used to obtain insight on the goodput, and it shows that losses on the reverse link may have a significant impact on the goodput. Our analysis assumes a perfect

feedback link. Therefore, it cannot be utilized to analyze networks with feedback losses. It is part of our future work to extend our model to also support losses on the reverse link.

VI. Conclusion

In this paper, we propose an SR ARQ model for multi-source single-receiver download. The model uses lossy forward links. The transmission status of these links is modeled as a hidden Markov process. We propose a general framework for analyzing the goodput of different multi-source protocols. In our analysis, we serialize the sources to ensure that a goodput increase is not the result of increased bandwidth (coming from the increased number of sources), but the result of a more efficient packet scheduling approach.

We compare numerical results with simulation results for six multi-source protocols, including uncoded and network coded approaches. We also consider two windowing approaches for the data sources: 1) a *sparse moving window* that has no constraint on packet delay, and 2) a *strict moving window* that introduces a constraint on the delay. Our results show that rateless network coding techniques can boost goodput, while network coded sliding window achieves optimal performance. On the other hand, depending on the network environment, a network coded sliding window may introduce significantly higher computation overhead, since it codes over more packets. Using our simulator, we also show that burst errors can significantly drop the goodput in the network if we allow losses on the reverse link. Furthermore, we show that our multi-source approach avoids the straggler problem, therefore adding new sources to the network increases the goodput.

Our simulations have already shown that losses on the reverse link may have a significant impact on the goodput, but we could not use our model for analysis as it assumes a perfect reverse link. As a future project, we plan to extend our analysis to also be able to support losses on the reverse link.

Our work shows the potential of coded multi-source downloads that has high applicability in information-centric networking (ICN) [1] and distributed systems [28].

References

[1] S. N. S. Hashemi and A. Bohlooli, "Analytical modeling of multi-source content delivery in information-centric networks," *Computer Networks*, vol. 140, pp. 152 – 162, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128618302056

[2] M. Amadeo, C. Campolo, and A. Molinaro, "Multi-source data retrieval in iot via named data networking," in *Proceedings of the 1st ACM Conference on Information-Centric Networking*, ser. ACM-ICN '14. New York, NY, USA: ACM, 2014, pp. 67–76. [Online]. Available: http://doi.acm.org/10.1145/2660129.2660148

[3] J. Miyoshi, S. Kawauchi, M. Bandai, and M. Yamamoto, "Multi-source congestion control for content centric networks," in *Proceedings of the 3rd ACM Conference on Information-Centric Networking*, ser. ACM-ICN '16. New York, NY, USA: ACM, 2016, pp. 205–206. [Online]. Available: http://doi.acm.org/10.1145/2984356.2985235

[4] Y. Thomas, C. Tsilopoulos, G. Xylomenos, and G. C. Polyzos, "Multisource and multipath file transfers through publish-subscribe internetworking," in *Proceedings of the 3rd ACM SIGCOMM Workshop on Information-centric Networking*, ser. ICN '13. New York, NY, USA: ACM, 2013, pp. 43–44. [Online]. Available: http://doi.acm.org/10.1145/2491224.2491238

[5] 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 *2017 14th IEEE Annual Consumer Communications Networking Conference (CCNC)*, Jan 2017, pp. 427–434.

[6] P. Pakzad, C. Fragouli, and A. Shokrollahi, "Coding schemes for line networks," in *Proceedings. International Symposium on Information Theory, 2005. ISIT 2005.*, Sep. 2005, pp. 1853–1857.

[7] D. Malak, M. Médard, and E. M. Yeh, "Tiny Codes for Guaranteeable Delay," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 4, pp. 809–825, April 2019.

[8] W.-X. Liu, S.-Z. Yu, G. Tan, and J. Cai, "Information-centric networking with built-in network coding to achieve multisource transmission at network-layer," *Comput. Netw.*, vol. 115, no. C, pp. 110–128, Mar. 2017. [Online]. Available: https://doi.org/10.1016/j.comnet.2015.05.009

[9] P. J. Braun, D. Malak, M. Médard, and M. Ekler, "Enabling multi-source coded downloads," in *2019 IEEE International Conference on Edge Computing (EDGE)*, July 2019.

[10] C. W. Sørensen, D. E. Lucani, and M. Médard, "On network coded filesystem shim: Over-the-top multipath multi-source made easy," in *2017 IEEE International Conference on Communications (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 *2014 IEEE 11th Consumer Communications and Networking Conference (CCNC)*, Jan 2014, pp. 127–132.

[12] J. K. Sundararajan, D. Shah, M. Médard, and P. Sadeghi, "Feedback-based online network coding," *IEEE Transactions on Information Theory*, vol. 63, no. 10, pp. 6628–6649, Oct 2017.

[13] 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.

[14] M. Kim, T. Klein, E. Soljanin, J. a. Barros, and M. Médard, "Modeling network coded tcp: Analysis of throughput and energy cost," *Mob. Netw. Appl.*, vol. 19, no. 6, pp. 790–803, Dec. 2014. [Online]. Available: http://dx.doi.org/10.1007/s11036-014-0556-1

[15] 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.

[16] M. A. M. Songze Li and A. S. Avestimehr, "A Unified Coding Framework for Distributed Computing with Straggling Servers," *CoRR*, vol. abs/1609.01690, 2016. [Online]. Available: http://arxiv.org/abs/1609.01690

[17] G. Ananthanarayanan, S. Kandula, A. Greenberg, I. Stoica, Y. Lu, B. Saha, and E. Harris, "Reining in the outliers in map-reduce clusters using mantri," in *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 265–278. [Online]. Available: http://dl.acm.org/citation.cfm?id=1924943.1924962

[18] P. J. Braun, D. Malak, M. Médard, and M. Ekler, "Multi-Source Coded Downloads," in *2019 IEEE International Conference on Communications (ICC)*, May 2019, pp. 1–7.

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

[20] S. J. Mason and H. J. Zimmermann, "Electronic circuits, signals, and systems," pp. xviii, 616 p., 1960, companion volume to Electronic circuit theory, by H.J. Zimmermann and S.J. Mason.

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

[22] I. S. Gradshteyn and I. M. Ryzhik, *Table of integrals, series, and products*, 7th ed. Elsevier/Academic Press, Amsterdam, 2007.

[23] 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.

[24] 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 Information Theory*, vol. 52, no. 10, pp. 4413–4430, Oct 2006.

[25] 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. [Online]. Available: http://doi.acm.org/10.1145/1177080.1177106

[26] 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.

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

[28] N. Anjum, D. Karamshuk, M. Shikh-Bahaei, and N. Sastry, "Survey on peer-assisted content delivery networks," *Computer Networks*, vol. 116, pp. 79 – 95, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128617300464