Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Automation and Applied Informatics

DESIGN AND PERFORMANCE EVALUATION OF CONTENT SHARING AND STREAMING TECHNIQUES IN DISTRIBUTED COMMUNICATION NETWORKS

TARTALOM MEGOSZTÁSI ÉS STREAMING TECHNOLÓGIÁK TERVEZÉSE ÉS TELJESÍTMÉNY VIZSGÁLATA ELOSZTOTT KOMMUNIKÁCIÓS HÁLÓZATOKBAN

Ph.D. Thesis Booklet

Patrik János Braun

Supervisor: Dr. Ekler Péter
Advisor: Prof. Frank H. P. Fitzek

Budapest
2020.

Ph.D. Thesis Booklet

Patrik János Braun

Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Automation and Applied Informatics

1117 Budapest, Magyar Tudósok körútja 2. QB-207.

e-mail:	braun.patrik@aut.bme.hu
tel:	+36(1)4631668
fax:	+36(1)4633478

Supervisor: Dr. Ekler Péter
Advisor: Prof. Frank H. P. Fitzek

1 Introduction

With the miniaturization of chips and the development of wireless networks, there are more device online than ever. Ericcson estimates to reach 8.3 billion mobile subscriptions by the end of 2024 [Eri18]. The network requirements of these subscriptions are diverse. On the one hand, half of the mobile subscriptions are estimated to be used for Internet of Things (IoT) connections. The challenge of IoT networks is not their bandwidth requirement but the massive number of online nodes. On the other hand, with the increasing popularity of streaming services like YouTube or Netflix, users steam video while they commute to work. This streaming generates a huge traffic load: video traffic accounted for 70% of the mobile Internet traffic in 2018, according to Ericcson. Finally, new application scenarios, like Vehicle-to-Vehicle (V2V) communication, require a low latency network. 5G intends to give a solution to this demand by aiming < 1 ms latency. However, reaching sub-millisecond latency cannot be achieved by relying solely on hardware solutions as packets cannot travel faster on the network than the speed of light. Therefore, smart software solutions are also required to reduce the server-client physical distance.

In this work, I analyze these three requirements in detail and investigate the potential of using distributed networks and network coding to give solutions to them. Compared to server-client based setups, distributed networks are better to scale, so it can serve more nodes with higher throughput. Furthermore, having the possibility to choose a node or server to download from can reduce the network latency as nodes in proximity can be selected. Network coding increases cache hit ratio and reduces packet errors.

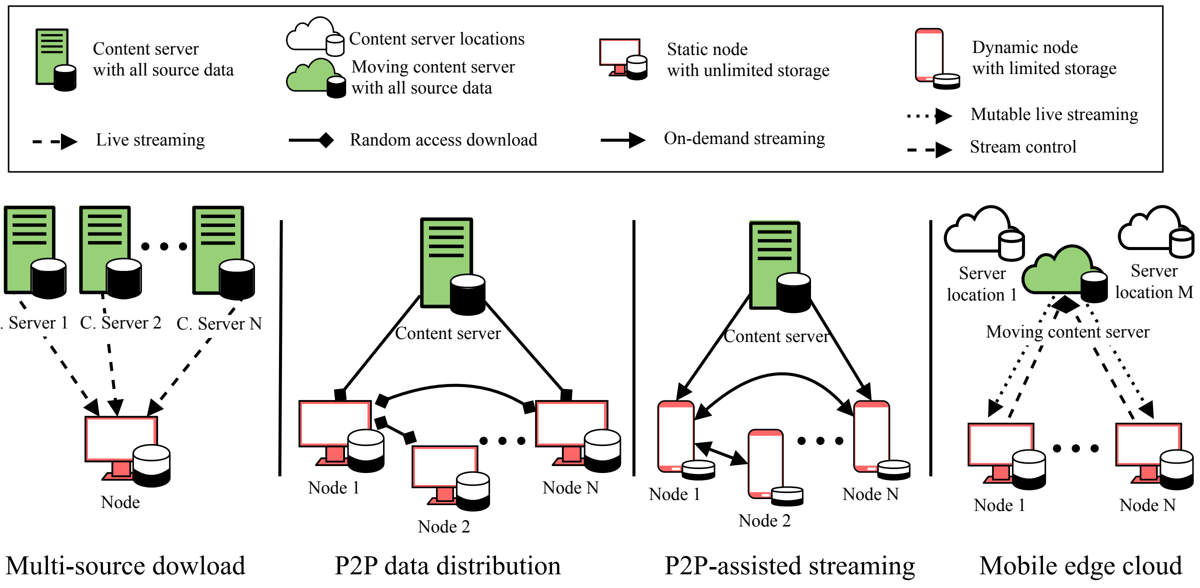

Figure 1: Networks overview

In my work, I have identified four distributed scenarios in interest: *Multi-source download*, *Peer-to-Peer (P2P) data distribution*, *mobile P2P-assisted streaming* and *mobile edge cloud* as Figure 1 shows. I proposed several protocols for multi-source and P2P download, including general P2P data distribution and P2P-assisted streaming. I investigated the possibilities of applying coding, namely Random Linear Network Coding (RLNC) [Ho+03], on the distributed data for both multi-source and P2P downloads. Furthermore, I proposed protocols and software architecture for mobile edge cloud to bring the computation power and data close to the client to the edge of the network.

2 Problem statement

This work focuses on distributed communication networks that comprise several nodes that I define the following way:

Definition 2.1 (Node)**.** Node is a single entity in the network that has storage, connects to other nodes, and can upload and download from them.

Connected node exchange the source data:

Definition 2.2 (Source data)**.** Source data is the original data that the nodes aim to download. It consists of L packets.

I differentiate four key properties that influence the behavior of a node. A node can be *static* that never leaves the network or *dynamic*. A node has *limited* $A \leq L$ or *unlimited* $A = L$ storage size. It can download the data by *random access* or *streaming*. Streaming can be *on-demand*, *immutable* or *mutable live streaming*. With mutable live streaming, nodes can influence the source data.

To simplify some of my notations, I further define:

Definition 2.3 (Content Server)**.** Static node with unlimited storage that possesses all L packets.

Definition 2.4 (Moving content server)**.** A content server that can change its location.

Figure 1 shows an overview of the four environments in interest that I define the following way:

Definition 2.5 (Multi-source network)**.** Multi-source networks contain N content servers and a static node with unlimited storage. The node is connected to the content servers with a lossy data link and a perfect reverse link. Links have a delay where the round trip time (RTT) takes up k time slots. The node sends commutative acknowledgment to the servers. There are L source packets that all the content servers posses. The node aims to gather all L packets in a streaming fashion by downloading from all contents servers simultaneously.

Definition 2.6 (Peer-to-Peer (P2P) data distribution)**.** P2P data distribution network contains a content server and N static nodes with unlimited storage. Content server behaves as a regular node, therefore, the nodes do not know which node is the content server before they connect to it. There are L source packets that the content server posses. Nodes can connect to any node to collect all L packets through on-demand download.

Definition 2.7 (P2P-assisted streaming)**.** P2P-assisted streaming systems contain a content server and N dynamic nodes with A-sized limited storage. There are L source packets that the content server posses. Nodes always have a connection to the content server and they can also connect to other nodes. Nodes aim to gather all L packets in a streaming fashion.

Definition 2.8 (Mobile edge cloud network)**.** Mobile edge cloud network has N static nodes, a single content server, and several server locations, where the server can migrate. The content server posses all L source packets and able to change its location. Other nodes live stream the data from the content server. Nodes also have a control channel back to the content server that they can use to alter the streamed content. If a node alters the stream, it will be changed for all other nodes.

3 New Theoretical results

Thesis I: Multi-source coded download

I have maximized throughput of multi-source protocols based on different network parameters by designing an analytical framework and investigating multi-source download in loosely orchestrated networks. I have proposed six protocols, including uncoded, rateless RLNC, and sliding-window based RLNC protocols and contrast their goodput. I have shown that employing RLNC in a multi-source network improves goodput, in particular sliding window-based RLNC protocol achieves the maximum theoretical goodput. I have shown that RLNC can significantly increase the network goodput in practical systems by designing a application that uses the uncoded and rateless RLNC protocols to download YouTube videos from multiple sources. To verify my findings, I have run an extensive measurement campaign over eleven months and analyzed 1,300,000 log records.

Publications related to this thesis: [1], [4], [5], [6], [7]
Patents related to this thesis: [21]
Talks and Demos related to this thesis: [22]

In this thesis, content servers and the static node are also referred to as sources and receiver, respectively. I have focused on a loosely orchestrated environment where there is no central controller that orchestrates the download scheduling. Thus, servers can only use the information from the commutative feedback to schedule packets for transmission.

Subthesis I.1: Proposing SR-ARQ-based model for multi-source download

I have proposed a modified SR-ARQ-based model that supports lossy forward and perfect reverse links to analyze multi-source downloads. I have modeled the transmission status with a hidden Markov model that is driven by a multi-state Markov process to make my solution applicable to different types of links. I have measured the receiver status with its Degrees of Freedom (DoF) to make the model support both coded and uncoded protocols.

I proposed a time-slotted model, where 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 is a Bernoulli random variable, denoted by $\mathcal{X}^{(i)} = \{0, 1\}$. The link condition is modeled by a multistate Markov chain $S^{(i)} = \{1, \ldots, K^{(i)}\}$ with probability transition matrix $\boldsymbol{P}_{(i)}$. 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 $\epsilon^{(\mathbf{i})} = \{\epsilon_1^{(i)}, \ldots, \epsilon_{K^{(i)}}^{(i)}\}$. The Markov process $S_t^{(i)}$ driven hidden Markov process $X_t^{(i)}$ can be characterized by $\{S^{(i)}, \mathcal{X}^{(i)}, \boldsymbol{P}_{(i)}, \epsilon^{(\mathbf{i})}\}$. Furthermore $\boldsymbol{P}_{\mathrm{L},(i)} = \boldsymbol{P}_{(i)} \cdot \mathrm{diag}\{\epsilon^{(\mathbf{i})}\}$ and $\boldsymbol{P}_{\mathrm{R},(i)} = \boldsymbol{P}_{(i)} \cdot \mathrm{diag}\{1 - \epsilon^{(\mathbf{i})}\}$ are the probabilities of losing and receiving a packet, respectively.

The receiver sends feedback to all sources, and the reverse link is perfect. Therefore, sources receive an ACK or NACK in every slot. Furthermore, 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.

Subthesis I.2: Introducing a general analysis approach

I have investigated the performance of different multi-source protocols by proposing an analytical framework. I have expressed the goodput of networks with N sources through constructing a matrix signal-flow graph and using probability generation function. I have shown that it is sufficient to consider the last k time slots to calculate the goodput. I have used my framework to contrast the performance of several multi-source protocols and to maximize the goodput of multi-source networks.

I define the seven packet events: A packet can get 1) $E_{\mathrm{L},(i)}$ (*lost*) or 2) $E_{\mathrm{R},(i)}$ (*received*) on link i. A packet can be 3) $E_{\mathrm{pU},(i)}$ (*potentially useful*) or 4) $E_{\mathrm{pD},(i)}$ (*potentially duplicate*). It can be 5) $E_{\mathrm{U},(i)}$ (*useful*) or 6) $E_{\mathrm{D},(i)}$ (*duplicate*) if it is received and it was potentially useful or duplicate, respectively. As $E_{\mathrm{L},(i)}$ and $E_{\mathrm{D},(i)}$ are equivalent in terms of receiving new packet, thus I merge them into 7) $E_{\mathrm{F},(i)}$ (*failed*) event.

The receiver sends cumulative feedback to the sources. Therefore each source has information about the packets at the receiver. This information is at most k time slot old. Therefore, it is sufficient to consider only the events in time slots between $[t-k, t-1]$ to calculate the probability of a packet is useful at time t.

To calculate the probabilility of $E_{\mathrm{U},(i)}$, I have proposed a matrix signal-flow graph for multi-source systems [AN07] that I used to express the probability generation function (PGF) of the transmission time:

PGF of the transmission time for link i

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

where $\epsilon_{\mathrm{F}}^{(i)}$ is the packet-failure rate, $\pi_{(i)}$ is the stationary vector of $\boldsymbol{P}_{(i)}$, $\mathcal{P}_{\mathrm{U},(i)}$ and $\mathcal{P}_{\mathrm{F},(i)}$ are the probabilities of a packet is useful or failure. $\mathbf{I}$ is the identity matrix and $\mathbf{1}$ is the column vector of ones.

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].

Calculating the probability of sending a useful packet $\mathcal{P}_{\mathbf{U},(i)}$ and a packet failure $\mathcal{P}_{\mathbf{F},(i)}$ at time slot t

$$\begin{aligned} \mathcal{P}_{\mathrm{U}}(t) &= \sum_{v\in\{0,1\}^t} \mathcal{P}_{\mathrm{pU}}(\mathbf{v}) \mathcal{P}_{\mathrm{past}}(v_1 \dots v_{t-1}) \boldsymbol{P}_{\mathrm{R},(s(t))} v_t \\ \mathcal{P}_{\mathrm{F}}(t) &= \sum_{v\in\{0,1\}^t} \mathcal{P}_{\mathrm{pD}}(\mathbf{v}) \mathcal{P}_{\mathrm{past}}(v_1 \dots v_{t-1}) \boldsymbol{P}_{\mathrm{R},(s(t))} v_t + (1-v_t) \boldsymbol{P}_{\mathrm{L},(s(t))}, \end{aligned} \tag{2}$$

where vector $\mathbf{v} = [v_1 \dots v_t]$ is the series of event that packets are received or lost between time slots $[1, t]$. $\mathcal{P}_{\mathrm{pU}}(\mathbf{v})$ and $\mathcal{P}_{\mathrm{pD}}(\mathbf{v})$ are the probabilities that a packet is potentially useful or duplicate at time t. $\mathcal{P}_{\mathrm{past}}(v_1 \dots v_{t-1})$ is the probability of $[v_1 \dots v_{t-1}]$ is the series of events between time slots $[1, t-1]$.

[1] This is indeed a lower bound due to the Jensen's inequality [GR07] and the convexity of $1/\overline{\tau}_{(i)}$.

Calculating $\mathcal{P}_{\mathbf{past}}(v_1 \dots v_{t-1})$ at time slot t

$$\mathcal{P}_{\text{past}}(v_1 \dots v_{t-1}) = \prod_{i=1}^{N} \prod_{\substack{l=i \\ (l-i \bmod N)=0}}^{t-1} \pi_{(i)} \boldsymbol{P}_{\text{R},(i)}^{v_l} \boldsymbol{P}_{\text{L},(i)}^{|1-v_l|} \mathbf{1}. \tag{3}$$

Probability of a packet is potentially useful $\mathcal{P}_{\mathbf{pU}}(\mathbf{v})$ or duplicate $\mathcal{P}_{\mathbf{pD}}(\mathbf{v})$

$$\mathcal{P}_{\text{pU}}(\mathbf{v}) = \sum_{u=0}^{k-\frac{k}{N}} \sum_{d=0}^{\frac{k}{N}} \sum_{m=0}^{\min(d,u)} \sum_{\substack{\sum a_j = u \\ \sum b_i = d \\ \sum c_i = m,\, c_i \le b_i}} \mathcal{P}_{\text{paired}}(\mathbf{a}, \mathbf{b}, \mathbf{c}) \mathcal{P}_{\text{sU}}(\mathbf{a}, \mathbf{b}, \mathbf{c}) \mathcal{P}_{\text{outcome}}(\mathbf{v}, \mathbf{a}, \mathbf{b}) \tag{4}$$

where $\mathcal{P}_{\text{paired}}(\mathbf{a}, \mathbf{b}, \mathbf{c})$ is the probability of $\mathbf{a}, \mathbf{b}$ and $\mathbf{c}$ are the events in the last k time slots, while $\mathcal{P}_{\text{outcome}}(\mathbf{v}, \mathbf{a}, \mathbf{b})$ is the probability of $\mathbf{a}, \mathbf{b}$ and $\mathbf{c}$ are the events, conditioned on $\mathbf{v}$. $\mathcal{P}_{\text{sU}}(\mathbf{a}, \mathbf{b}, \mathbf{c})$ and $\mathcal{P}_{\text{sD}}(\mathbf{a}, \mathbf{b}, \mathbf{c})$ are the probability that a source chooses a useful or duplicate packet for transmission, respectively, based on applied packet scheduling strategy. $\mathcal{P}_{\text{pD}}(\mathbf{v})$ can be expressed similarly to $\mathcal{P}_{\text{pU}}(\mathbf{v})$ by using $\mathcal{P}_{\text{sD}}(\mathbf{a}, \mathbf{b}, \mathbf{c})$ instead of $\mathcal{P}_{\text{sU}}(\mathbf{a}, \mathbf{b}, \mathbf{c})$.

My matrix-flow graph approach to calculate the goodput is only applicable if $\lim_{t\to\infty} \mathcal{P}_{\text{U}}(t)$ and $\lim_{t\to\infty} \mathcal{P}_{\text{F}}(t)$ exist.

Subthesis I.3: Proposing multi-source protocols

I have provided the theoretical upper and lower bound for the goodput of multi-source downloads by proposing sufficient genie and uncoded sequential protocols. To maximize the goodput, I have further designed four practical protocols, including RLNC-based protocols. I have introduced a constraint on the packet delay by proposing two windowing techniques. I have shown that rateless RLNC techniques can boost goodput, while network coded sliding window achieves optimal performance with both window technique.

I have proposed two moving window model for multi-source data transfer:

Sparse moving window contains w packets. Once a packet is transmitted from the window, the subsequent one from the available packets is added to the window.

Strict moving window for source i represented with a set of packets $\mathcal{W}^{(i)}(t)$:

$$\begin{aligned} \mathcal{W}^{(i)}(t) &= \{l \in \mathcal{L} \mid l_{\text{dMin}} < l \le l_{\text{dMin}} + w\} \\ \mathcal{L}_{\text{down}}^{(i)}(t) &= \{l \in \mathcal{L} \mid \text{packet } l \text{ was transmitted and acknoledged by time slot } t\}, \end{aligned} \tag{5}$$

where $\mathcal{L} = \{1, \dots, L\}$ is the set of packets, $l_{\text{dMin}} = \min(\mathcal{L} \backslash \mathcal{L}_{\text{down}}^{(i)}(t))$ is the packet with least ID that has not been transmitted or acknowledged by time t.

While sparse window always keeps w packets in the window, the strict window has a more strict limit on the packet delay.

I have proposed six protocols and investigated their goodput:

Protocol 3.1 (Sufficient genie)**.** The Sufficient genie transits only DoF-increasing packets to the receiver.

As packets may get lost on the link, the sufficient genie only gives the optimal scheduling with the given link conditions.

$\mathcal{P}_{\mathbf{sU}}(\mathbf{a},\mathbf{b},\mathbf{c})$ **for sufficient genie** independently of the applied windowing mechanism:

$$\mathcal{P}_{\mathrm{sU}}(\mathbf{a},\mathbf{b},\mathbf{c}) = 1 - \mathcal{P}_{\mathrm{sD}}(\mathbf{a},\mathbf{b},\mathbf{c}) = 1. \tag{6}$$

Protocol 3.2 (Uncoded sequential)**.** Uncoded sequential transmit the packets in a sequential fashion, using the window in a first-in, first-out (FIFO) fashion.

$\mathcal{P}_{\mathbf{sU}}(\mathbf{a},\mathbf{b},\mathbf{c})$ **for uncoded sequential** independently of the applied windowing mechanism:

$$\mathcal{P}_{\mathrm{sU}}(\mathbf{a},\mathbf{b},\mathbf{c}) = 1 - \mathcal{P}_{\mathrm{sD}}(\mathbf{a},\mathbf{b},\mathbf{c}) = \left|(1 - \min(\sum_{j=k-s(t)-1}^{k} a_j, 1))\right|. \tag{7}$$

Protocol 3.3 (Uncoded random)**.** Uncoded random selects a packet uniformly at random from its window for transmission.

$\mathcal{P}_{\mathbf{sU}}(\mathbf{a},\mathbf{b},\mathbf{c})$ **for uncoded random** with sparse moving window:

$$\mathcal{P}_{\mathrm{sU}}(\mathbf{a},\mathbf{b},\mathbf{c}) = 1 - \mathcal{P}_{\mathrm{sD}}(\mathbf{a},\mathbf{b},\mathbf{c}) = \frac{w - (\frac{k}{N} - (d-m)) - (u-m)}{w - (\frac{k}{N} - (d-m))}, \tag{8}$$

where $u = \sum_{j=1}^{k} a_j$, $d = \sum_{i=1}^{k} b_i$, $m = \sum_{i=1}^{k} c_i$.

$\mathcal{P}_{\mathbf{sU}}(\mathbf{a},\mathbf{b},\mathbf{c})$ **for uncoded random** with strict moving window:

$$\begin{aligned}\mathcal{P}_{\mathrm{sU}}(\mathbf{a},\mathbf{b},\mathbf{c}) &= \sum_{w_a=w_{min}}^{w} P(\mathcal{W}=w_a)\frac{w_a - (\frac{k}{N} - (d-m)) - (u-m)}{\sum_{w'_a=w_{min}}^{w} P(\mathcal{W}=w'_a)(w_a - (\frac{k}{N} - (d-m)))}\\ w_{min} &= max\left(1, u, (\frac{k}{N} - (d-m)) + (u-m)\right)\end{aligned} \tag{9}$$

Protocol 3.4 (Rateless RLNC coded with random generation selection)**.** The protocol groups the packet in the window into g-sized generation and applies RLNC to them. A generation for transmission is chosen uniformly at random.

Protocol 3.5 (Rateless RLNC coded with rarest-first-generation selection)**.** The protocol groups the packet in the window into g-sized generation and applies RLNC to them. A generation with the least received packet is chosen for transmission.

Protocol 3.6 (Coded sliding window)**.** The coded sliding window encodes all the packets in the window with RLNC.

$\mathcal{P}_{\mathbf{sU}}(\mathbf{a},\mathbf{b},\mathbf{c})$ **for coded sliding window** independently of the applied windowing mechanism:

$$\mathcal{P}_{\mathrm{sU}}(\mathbf{a},\mathbf{b},\mathbf{c}) = 1 - \mathcal{P}_{\mathrm{sD}}(\mathbf{a},\mathbf{b},\mathbf{c}) = \begin{cases}1 & \text{if } (t \bmod k) < w\\ 0 & \text{otherwise}\end{cases}. \tag{10}$$

As Figure 2 shows, rateless RLNC techniques increase goodput, while coded sliding window achieves optimal performance. On the other hand, the coded sliding window may introduce significantly higher computation overhead, as it codes over more packets. I have also shown that burst errors can significantly drop the goodput if we allow losses on the reverse link. Furthermore, I showed that our multi-source approach avoids the straggler problem.

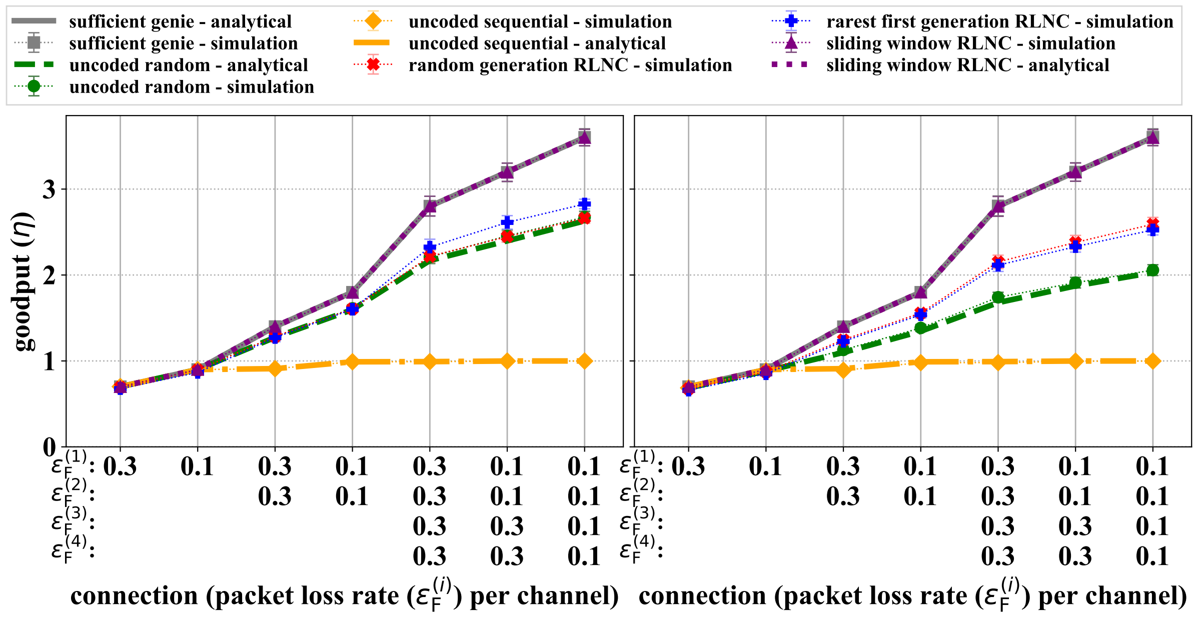

Figure 2: Goodput for sparse (left) and strict (right) moving window for sources $N \in \{1, 2, 4\}$, RTT $\kappa_c = 3$, window size $w = 24$, generation size $g = 12$ and burst rate $r = 0.3$.

Subthesis I.4: Designing a multi-source system

I have shown the practical applicability of multi-source coded download by designing and implementing a system that uses multi-source protocols to download data from multiple sources. I have designed Uncoded and Coded MUTP that implements the uncoded random rateless RLNC coded protocol over WebRTC, respectively. I have shown that MUTP protocols can outperform HTTP, reaching an up to three-fold goodput increase by presenting measurement results collected over eleven months.

Protocol 3.7 (MUlti-source Transmission Protocol (MUTP))**.** MUTP protocol slice the original data into L packets of 1100 bytes. Sources maintain *in-window*, and *in-transit* packet lists, while the receiver maintains a *received* packet list. The receiver sends cumulative feedback based on its *received* list. Based on the feedback and the *in-transit* list, sources create a *sendable* packet list that is used to choose a packet uniformly at random without replacement for transmission.

Protocol 3.8 (Coded MUTP)**.** Coded MUTP organizes the L source packets into g-sized generations and applies RLNC to them. It uses the rarest-generation-first approach for packet scheduling to send a packet from a generation that has the least received DoF and in *in-transit* packets.

I have developed a system that intercepts YouTube video downloads and forwards them through multiple servers by using a simple Parallel HTTP or my MUTP protocols. I have carried out an extensive measurement campaign and showed that using four or more sources, both MUTP protocols outperform the simple Parallel HTTP. Furthermore, I have achieved a two- and three-fold normalized goodput increase with Uncoded and Coded MUTP, respectively as Figure 3 shows. I have shown that applying RLNC in a loosely orchestrated multi-source scenario can achieve a significant goodput increase.

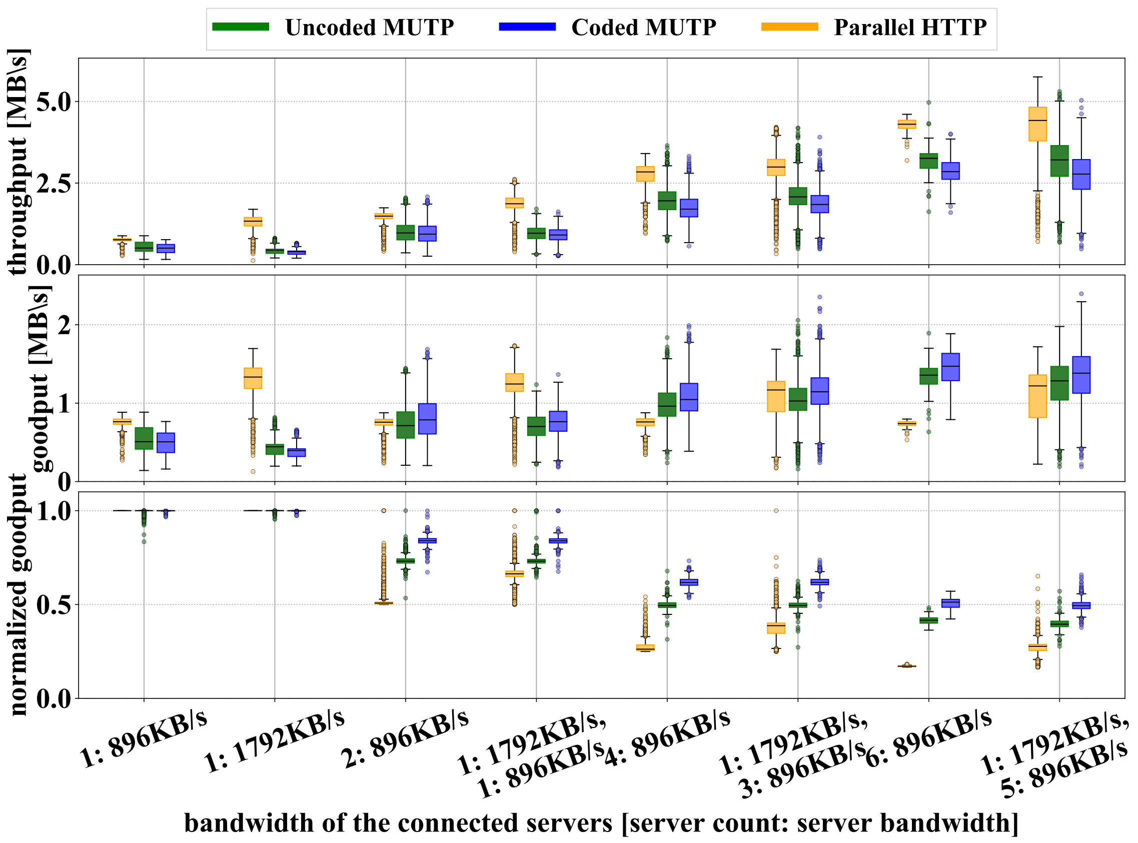

Figure 3: Downloading 1-2MB data from $N \in \{1, 2, 4, 6\}$ servers with 896 KB/s and 1,792 KB/s upload bandwidth with window size $w = 240$ and generation size $g = 24$.

Thesis II: Peer-to-Peer data distribution

I have shown that RLNC increases the network coherence and decreases the data distribution time in P2P networks during its initial phase. I have validated my results by designing and implementing a BitTorrent protocol extension that enables nodes to send RLNC encoded data but keeps compatibility with the original protocol. To compare the throughput of different P2P network implementations, I have proposed multiple network metrics. I have designed multiple algorithms to handle and avoid receiving linearly depended RLNC encoded packets in P2P networks.

Publications related to this thesis: [2], [8], [9], [10]

In this thesis, I focused on the initial phase of a P2P network, when only the content server posses all L source packets.

Subthesis II.1: Proposing metrics for P2P networks

I have identified network coherence as one of the main characteristics of a P2P network. To provide metrics that comprise the duration of the data distribution and the change of the network coherence during the download, I have proposed network health and network performance metrics. Using my metrics, I have shown that RLNC enhanced P2P protocols improve the network throughput during the initial phase of the network.

Definition 3.9 (Network coherence)**.** Network coherence is $i \leq N$, if removing any i nodes from the network, all L packets are still present in the network at least once.

To measure the network coherence, the aggregated packet vector $\mathbf{B}$ should be examined: Let $\mathbf{b_i} = \{p_{i1}, p_{i2} \dots p_{iL}\}$ represent the downloaded packets at node i, where $p_{i,l}$ is 1 if node i has packet l and 0 otherwise, then:

$$\mathbf{B} = \{B_1, B_2, \cdots, B_N\} = \sum_{i=1}^{N} \mathbf{b_i} \tag{11}$$

Using $\mathbf{B}$, network health and network performance can be expressed:

Definition 3.10 (Network Health at time k)**.**

$$\begin{aligned} p_{\mathrm{c}} &= |\{l \in \mathbf{B} \,|\, l > p_{\min}\}| \\ H_k &= p_{\min} + \frac{p_{\mathrm{c}}}{L}, \end{aligned} \tag{12}$$

where $p_{\min} = \min(\mathbf{B})$.

Definition 3.11 (Network performance)**.**

$$E = \sum_{k=0}^{k_{\mathrm{end}}} H_k, \tag{13}$$

where k_{end} is number of time slots that a network needed to distribute all L packets to all N nodes.

Network performance incorporates both the time the network takes to distribute all packets to all nodes and also the increase of network coherence throughout the process.

Subthesis II.2: Designing RLNC-based protocol extension for BitTorrent

I have shown the potential of RLNC in P2P networks by designing and implementing Network Coding Messaging Extension (NCME). To verify my results, I have run several measurements and compared a conventional BitTorrent implementation with my NCME extended implementation. I have shown that during the initial phase of the network NCME outperforms the baseline BitTorrent implementation by up to 20%.

Protocol 3.12 (Network Coding Messaging Extension (NCME))**.** NCME groups the L source packets into g-sized generations and applies RLNC to them. These encoded packets travel the network. After a node receives g linearly independent packet, it decodes the generation and shares the packets with other nodes, including nodes that use conventional BitTorrent protocol. NCME works over reliable connections. The structure of the messages follows the basic BitTorrent messages structure, but extends them with the capability of sending generation id instead of packet id.

I have extended a BitTorrent implementation with NCME and run measurements on an emulated network. I have investigated the network based on several key parameters, including node count (N) and packet count (L). I have shown NCME outperforms the baseline BitTorrent implementation by up to 20%. The gain is higher as N increases, and L decreases, as Figure 4 shows.

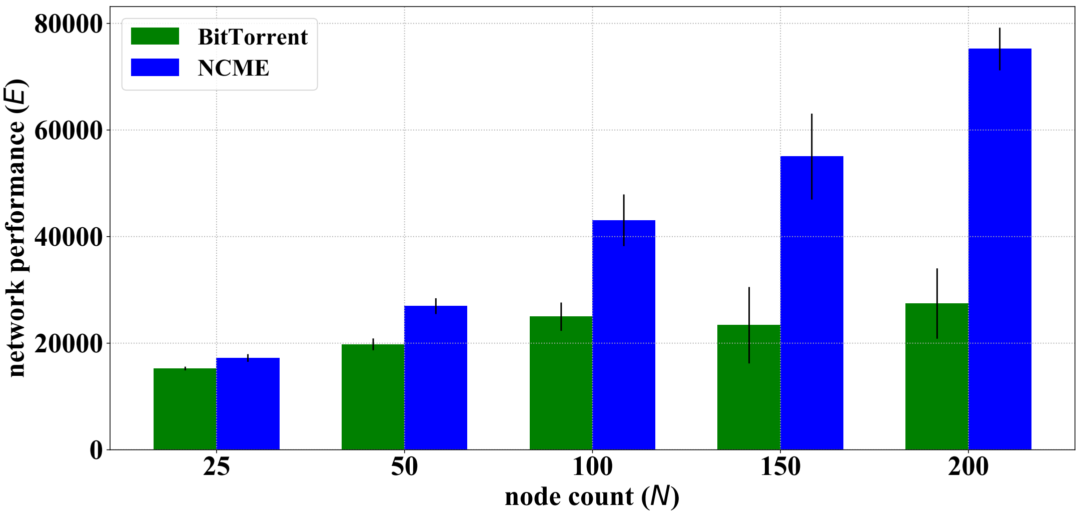

Figure 4: Nodes count impact on network performance.

Subthesis II.3: Constructing algorithms to handle and avoid linearly-depended RLNC packets

I have identified typical P2P network patterns that increase the chance of creating linearly dependent packets. I have proposed three algorithms for handling and avoiding linearly dependent packets. I have shown through measurement results that my solution can significantly reduce the number of linearly dependent packets.

I have identified network patterns when node A can requests packets from a node B that has lower rank than A (i.e.: $\nu_\text{A} \geq \nu_\text{B}$, where ν_i is the rank at node i). These patters can lead to the creation of linearly dependent packets that do not increase the DoF at the receiver node.

I have proposed three algorithms for handling linear dependent packets in RLNC enhanced P2P networks:

Algorithm 3.1 (Ignoring linearly dependent packets)**.** *If a node receives a linearly-dependent packet, it continues downloading packets from that generation.*

Algorithm 3.2 (Just-in-time filtering)**.** *If node A receives a linearly dependent packet from generation i of connection B, node A stops further request from i of B until node A receives a have message from node B indicating an increased rank of generation i at B.*

Algorithm 3.3 (Probe-based filtering)**.** *Before requesting a new encoded packet from generation i of node B, node A requests a probe packet of generation i from B. Node B creates a new encoded packet from generation i and only sends back a small coefficient vector of the encoded packet. Receiving the probe, A verifies whether the full encoded packet would increase its DoF. Node A only requests data packets from node B if there is a potential DoF increase.*

Using my NCME extended BitTorrent implementation, I have run extensive measurements to evaluate the performance of my algorithms. I have shown that ignoring linearly dependent packets sends the most packets, but it generates significant network and computational overhead as linearly dependent packets need to be processed too. Just-in-time filtering distributes the data fastest, but it uses more bandwidth than probe-based filtering. Therefore, using probe-based filtering is recommended as it offers a reasonable trade-off between network traffic overhead and data distribution duration.

Thesis III: Peer-to-Peer assisted streaming

I have minimized server load of P2P-assisted streaming in mobile environment by proposing analytical and practical solutions. I have evaluated the server load based on different key parameters by constructing a model and proposing an analytical framework. To show the impact of RLNC on P2P-assisted streaming, I have considered four caching strategies, including random, and RLNC caching. I have shown the practical applicability of P2P-assisted streaming by designing protocols that implement random and RLNC caching and proposing algorithms for download scheduling and connections management. I have shown through measurements that RLNC caching needs 50% fewer packets to achieve a close-to-zero average server load by designing a system, called PasNet.

Publications related to this thesis: [3], [11], [12], [13], [14], [15], [16], [17]
Talks and Demos related to this thesis: [23], [24], [25]

This thesis focuses on a mobile environment that is a partially distributed network environment. Nodes are dynamic with limited network bandwidth and storage. Furthermore, detailed information about the nodes is not available; only if a node is part of the network or not.

This thesis aims to minimize the server load that is:

Definition 3.13 (Server load)**.** The server load is the minimum required server upload rate that is enough to serve the nodes without stream interruptions.

Subthesis III.1: Proposing model for P2P-assisted streaming

I have formalized the problem definition of this thesis, by creating a time slotted model that incorporate the possibilities and constraints of a Mobile P2P-assisted streaming system. The model incorporates node life cycle, node connection, download scheduling, and limited node storage.

P2P-assisted streaming networks contain N_k nodes at time slot k. Nodes join the network following a Poisson process and leave the network randomly, following a skew-normal distribution.

Nodes have an up-to-date list of other network nodes. At time slot k, they are connected to a subset of the available $N_k - 1$ nodes. The active connections of node i at time slot k are denoted by vector $\mathbf{c}^{(i)}(k)$ and bounded by C:

$$\mathbf{c}^{(i)}(k) \in \{0,1\}^{N_k},\ i = 1, \ldots, N_k,\ w(\mathbf{c}^{(i)}(k)) \le C, \tag{14}$$

where $w(\mathbf{c}^{(i)}(k))$ gives the active connection count that node i has to other nodes.

Nodes download the L packets sequentially in a streaming fashion, but to enable the use of advanced download managers, I define a download window:

Definition 3.14 (Download window)**.** Download window, $\mathcal{W}^{(i)}(k)$ is a set of packets that node i aims to download at time slot k:

$$\mathcal{W}^{(i)}(k) = \{l \in \mathcal{L} \mid \pi_{\text{p}}^{(i)}(k) < l \leq \pi_{\text{p}}^{(i)}(k) + w\}, \tag{15}$$

where $\mathcal{L} = \{1, \ldots, L\}$ is the set of all packets and $\pi_{\text{p}}^{(i)}(k)$ is packet-based download progress that is defined the following way:

Definition 3.15 (Packet-based download progress)**.** Packet-based download progress, $\pi_{\text{p}}^{(i)}(k) \in 0, \ldots, L$ is highest id of all downloaded packet at node i at time k, representing the download progress of node i at time slot k:

$$\begin{aligned} \pi_{\text{p}}^{(i)}(k) &= max(\mathcal{L}_{\text{down}}^{(i)}(k)) \\ \mathcal{L}_{\text{down}}^{(i)}(k) &= \{l \in \mathcal{L} \mid \text{packet } l \text{ was downloaded by time slot } k\} \end{aligned} \tag{16}$$

A node has a A-sized storage to cache downloaded packet and to share them later with other nodes. Nodes use singe-try caching mechanism that I define as:

Definition 3.16 (Single-try cache)**.** A cache is single-try if $\mathcal{A}_{\text{pot}}^{(i)}(k)$ is the potential packets that can be cached and $\mathcal{A}^{(i)}(k)$ is the set of packets that are included in the cache at node i at time k:

$$\begin{aligned} &\mathcal{A}_{\text{pot}}^{(i)}(k) \subset \{p \in \mathcal{W}^{(i)}(k) \,|\, p \text{ is downloaded at time } k\} \\ &\mathcal{A}^{(i)}(k) \subset \mathcal{L}_{\text{down}}^{(i)}(k),\ w(\mathcal{A}^{(i)}(k)) \leq A, \end{aligned} \tag{17}$$

where $w(\mathcal{A}^{(i)}(k))$ is the number of elements in set $\mathcal{A}^{(i)}(k)$.

In the described system, I minimize the average server load $\overline{r}$:

$$\begin{aligned} \overline{r} &= \lim_{T\to\infty} \sum_{t=1}^{T} \frac{1}{T} r(k) \\ r(k) &= \frac{\text{downloaded packets from the server at time } k}{\text{all downloaded packets at time } k}. \end{aligned} \tag{18}$$

Subthesis III.2: Proposing analytical framework to minimize server load

I have shown the potential of the caching strategies and the cache size on server load in P2P-assisted streaming systems, by proposing an analytical framework. To make the framework versatile, I have incorporated four key parameters of P2P-assisted network to my analysis: peer joining and leaving process, packet caching strategy, connection management, and download scheduling. I used the framework to investigate the impact of different caching strategies to the server load.

To simplify my calculations in my analytical framework, I assume a most likely peer configuration that I represent with the Packet-based node density that can be expressed by using the transform function Ψ and the node age p.d.f.:

Definition 3.17 (Transform function)**.** A transformation function $\Psi : \pi_{\text{t}} \to \pi_{\delta}$ orders download progress $\pi_{\delta} \in [0, 1]$ to a node, based on its age $\pi_{\text{t}} \in [0, 1]$

Definition 3.18 (Node age Probability density function (p.d.f.) $f_{\text{alive}}(t)$)**.**

$$f_{\text{alive}}(t)\Delta t = \frac{\Delta t(1 - F_{\text{skew}}(t))}{\int_0^\infty (1 - F_{\text{skew}}(t))\mathrm{d}t}, t \geq 0, \tag{19}$$

where $F_{\text{skew}}(t)$ is the c.d.f. of the node leaving skew-normal distribution.

Definition 3.19 (Packet-based node density $\varrho = \{\varrho_1, \ldots, \varrho_L\}$)**.**

$$\varrho_l = \int_{\frac{l}{L+1}}^{\frac{l+1}{L+1}} f_{\text{prg}}(n)\mathrm{d}t,\ l = 0, \ldots, L-1, \tag{20}$$

where $f_{\text{prg}}(n) = f_{\text{alive}}(\Psi^{-1}(n))$.

Using the density, I expressed the cache miss ration with single P2P connection:

Cache miss ratio for $C = 1$

$$s(n) = \sum_{m=0, m\neq n}^{L} (1 - \frac{\mathbb{E}(\delta_{nm})}{\min\{w, L-n\}})\varrho_n w_p(n,m), \tag{21}$$

where $w_p(n,m)$ is the probability that node i with $\pi_{\text{p}}^{(i)} = n$ has a connection to a j with $\pi_{\text{p}}^{(j)} = m$. $\mathbb{E}(\delta_{nm})$ is the expected number of packets obtained by node i from node j.

Using the cache miss ration for $C = 1$, I expressed it for arbitrary number of connections:

Cache miss ratio for $C >= 1$ Considering the node's connection establishment as independent events, the cache miss ratio for node i with $\pi_{\text{p}}^{(i)} = n$ is:

$$\mathbf{S}(n) = \sum_{j=0}^{N}(1 - (1 - s(n))^j)w_c(j), \tag{22}$$

where $w_c(j)$ is the probability that node i with $\pi_{\text{p}}^{(i)} = n$ has j connections.

Using the cache miss ratio, average server load, can be calculated the following way:

Average server load Average server load $\bar{r}$ as the ratio between the required server upload rate and the average download rate per node is:

$$\bar{r} = \frac{1}{L}\sum_{n=0}^{L}(\mathbf{S}(n))\varrho_n. \tag{23}$$

Subthesis III.3: Proposing single-try caching strategies

I have shown the impact of caching strategies on P2P-assisted streaming by proposing four single-try caching strategies including infinite caching, FIFO caching, random caching, and RLNC encoded caching. I have shown that random caching outperforms FIFO caching, while RLNC further decreases the server load by expressing their $\mathbb{E}(\delta_{nm})$ *and using my analytical framework to compare them.*

I propose infinite caching as a base line:

Definition 3.20 (Infinite caching)**.** With infinite caching, nodes can store all downloaded packets: $\mathcal{A}^{(i)}(k) = \mathcal{L}^{(i)}_{\text{down}}(k)$

Infinite cache performs as optimal caching with the given connection and download management constraint, defined in my model.

The expected number of packets obtained by node i with $\pi_{\text{p}}^{(i)} = n$ from node j with $\pi_{\text{p}}^{(j)} = m$, $m \geq n$ for infinite caching is:

$\mathbb{E}(\delta_{nm})$ **for infinite caching**

$$\mathbb{E}(\delta_{nm}) = \min\{w, m-n\} \tag{24}$$

Definition 3.21 (FIFO caching)**.** FIFO caching stores the last A downloaded packets: $\mathcal{A}^{(i)}(k) \in \{p \mid \max\{\pi_{\text{p}}^{(i)}(k) - A, 0\} < p \leq \pi_{\text{p}}^{(i)}(k)\}$

$\mathbb{E}(\delta_{nm})$ **for FIFO caching**

$$\mathbb{E}(\delta_{nm}) = \max\{0, \min\{A^{(m)}_{\text{end}} - w^{(n)}_{\text{start}}, w^{(n)}_{\text{end}} - A^{(m)}_{\text{start}}\}\}, \tag{25}$$

where $A^{(m)}_{\text{start}} = \max\{0, m - A\}$ and $A^{(m)}_{\text{end}} = m$ represent the start and the end of the cache at node j and $w^{(n)}_{\text{start}} = n$ and $w^{(n)}_{\text{end}} = \min\{n + w, L\}$ represent the start and the end of the window of node i.

Definition 3.22 (Random caching)**.** Random caching caches packets uniformly at random across the downloaded data.

$$P(p_1 \in \mathcal{A}^{(i)}(k)) = P(p_2 \in \mathcal{A}^{(i)}(k)),\ p_1, p_2 \in \mathcal{L}^{(i)}_{\text{down}}(k), \tag{26}$$

where P is the probability function.

$\mathbb{E}(\delta_{nm})$ **for random caching**

$$\mathbb{E}(\delta_{nm}) = \sum_{q=0}^{\min\{A,w,m-n\}} \frac{\binom{\min\{w,m-n\}}{q}\binom{\max\{n,m-w\}}{A-q}}{\binom{m}{A}} q. \tag{27}$$

Definition 3.23 (RLNC caching)**.** RLNC caching groups the L packets into g-sized generation and applies RLNC to the generations.

As RLNC caching works with generations, I have proposed a generation-based analysis that uses "′" to mark the updated notations:

$\mathbb{E}'(\delta_{n'm'})$ **for RLNC caching**

$$\mathbb{E}'(\delta_{n'm'}) = \begin{cases} \min\left\{\frac{A}{n'}, w\right\} & \text{if } g \geq w \\ \min\{w', m'-n'\} g^{m'}_{\min} + \mathbb{E}(P'_{n'm'}) & \text{otherwise,} \end{cases} \tag{28}$$

where

$$\mathbb{E}(P'_{n'm'}) = \sum_{q=0}^{\min\{(A \bmod m'), w', m'-n'\}} \frac{\binom{\min\{w', m'-n'\}}{q}\binom{\max\{n', m'-w'\}}{(A \bmod m')-q}}{\binom{m'}{(A \bmod m')}} q. \tag{29}$$

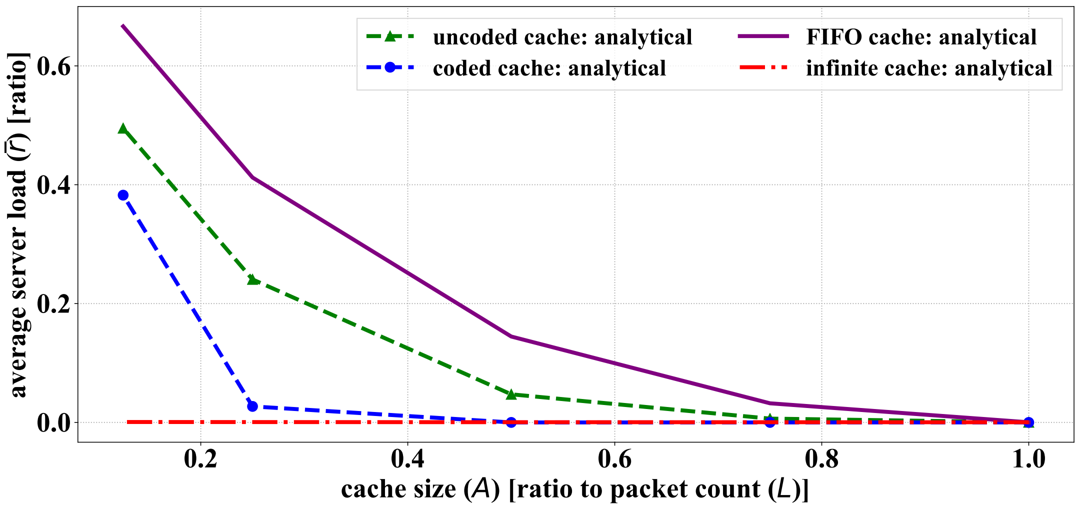

Figure 5: Server load with peers $N = 18$, generation size $g = 8$ and packets $L = 633$.

As Figure 5 shows, random caching needs to store 40% fewer packets to achieve the same performance as FIFO caching if $connections = 5$. Compared to the random caching, RLNC caching needs 50% fewer packets to achieve a close-to-zero average server load. Furthermore, caching half of the packets, RLNC reaches zero server load, when the network is capable of serving all newcoming nodes without the help of the content server.

Subthesis III.4: Designing protocols and algorithms for mobile P2P-assisted streaming

I have shown the applicability of P2P-assisted streaming systems by designing Peer-to-Peer Assisted Streaming Network (PasNet) and running it on more than 100 tablets. To construct PasNet, I have proposed multiple algorithms and two protocols for mobile P2P-assisted streaming, based on my single-try caching strategies. I have verified my analytical framework by comparing its results with the practical results of PasNet.

Protocol 3.24 (WebPeer)**.** WebPeer is a communication protocol that implements the random caching strategy. Supported messages are summarized in Table 1. All messages

Table 1: WebPeer messages

Availability vector	Similarly to BitTorrent's bitfield, it represents the packets in a node's cache.
Have	Signals if a new packet is available in the node's cache.
Lost	Signals if a packet was deleted from the node's cache.
Request	Node requests a given packet from its connection.
Cancel	Cancels requests.
Data	Contains the requested data packet.

contain a packet ID. Once two nodes connect, they exchange their *availability vector*. Later, they only use *have* and *lost* messages to indicate the changes of their *availability vectors*. The handshake procedure is handled by the underlying protocol.

Protocol 3.25 (CodedWebPeer)**.** CodedWebPeer extends WebPeer and implements the RLNC caching strategy with the following updates: *have* message contains the rank of a given generation, and *lost* message is not supported, as sending a *have* with decreased rank serves this purpose. *Request* and *cancel* messages contain an amount value beside the generation index that specifies the number of requested or canceled packets.

Algorithm 3.4 (Node connection management)**.** *Nodes accept all incoming connections, while they create new ones, if they have less than C open connections. Each connection is bidirectional, behaving in the same way, regardless of the initiator. Each connection stays open at least for T_{con} timeout. If a node has more than $C - 1$, it closes those connections that offer the lowest download rate.*

Algorithm 3.5 (Node download management)**.** *Download manager schedules a packet for download from the server only if none of the connected nodes have it. The number of packets scheduled for download from nodes is linearly proportional to their offered rate. If a packet is not received within T_{down} timeout, it reschedules the packet for download.*

Using my protocols and algorithms, I have implemented Peer-to-Peer Assisted Streaming Network (PasNet) and run it on more than 100 tablets to show the practical potential of P2P-assisted streaming systems. I used *PasNet* to validate my analytical framework at multiple stages of my calculations. The evaluation has shown that the practical and analytical results show similar trends with ≤ 0.149 mean square error (MSE).

Thesis IV: Mobile edge cloud

I have minimized the cloud migration time by studying and comparing different migration solutions and proposing Agile Cloud Migration (ACM). To present the capabilities of ACM, I have designed and implemented an ACM-based system that cable of migrating data and computation power to the edge of the network. I have shown through measurements that my solutions significantly outperforms state-of-the-art KVM- and Docker-based migration solutions.

Publications related to this thesis: [19], [18], [20]
Talks and Demos related to this thesis: [26], [27], [28], [29], [30], [31], [32], [33], [34]

I have focused on scenarios, where the content server have enough upload rate to fully server all nodes, while all nodes have enough download rate to live stream the content without Quality of Experience (QoE) degradation.

Subthesis IV.1: Proposing software architecture for ACM

To achieve fast cloud migration, I have proposed three-state software architecture for the content server that separates the network communication, computation, and application state. I have shown through measurements that a system, based on my architecture, significantly outperforms state of the art KVM- and Docker-based migration solutions in terms of migration data size.

I have proposed a software architecture for fast cloud migration for content servers that are also referred to as *engines* in this thesis.

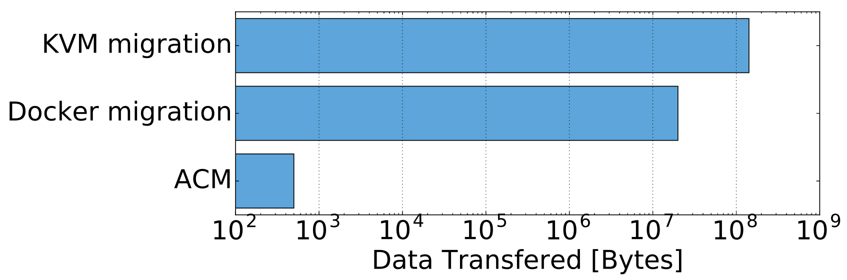

Figure 6: Comparison of migration techniques.

My architecture separates the application *State*, the *Worker* module that manipulates the *State* and the *Network* module that handles incoming messages and distributes the *State* among the connected nodes. *Network* module contains three channels: 1) the command channel that receives commands from the nodes and forwards them to the *Worker*. 2) The client-state channel is used to distribute the updated *State* to all the connected nodes. 3) The engine channel is used to send packets from the old engine to the location of the new engine.

The architecture supports three states: 1) Active, when the engine receives a command from the nodes, the *Worker* executes them and sends the updated *State* back to the nodes. 2) Migrating, when the incoming commands are forwarded to the new engine location. 3) Idle, after a successful migration.

I have shown through measurements that applying my architecture can reduce the migrated amount of data from 143 MB to a few kilobytes.

Subthesis IV.2: Designing communication protocol for ACM

I have proposed a communication protocol for fast cloud migration that keeps the migration transparent for the users while guarantees zero packet loss throughout the process. To present the applicability of my architecture and protocol, I have implemented a browser-based multiplayer gaming application that is capable of moving the computation power and the data between server locations with minimal delay. I have demonstrated the potential of my system by migrating the content server of the game between server locations at different continents and to the edge of the network without game interruptions.

I have designed a protocol for cloud migration:

Protocol 3.26 (Migration protocol for ACM)**.** The protocol has the following five steps: 1) The engine (content server) creates a connection with the engine at the new location. 2) The old engine goes to migration state (when it forwards all incoming node commands to the new engine), while transfers its application *State* to the new engine. 3) The new engine receives the *State* and goes to the active state. 4) The old engine goes to the idle state, while signals the connected nodes to connect to the new engine. 5) Nodes connect to the new engine and authenticate themselves.

To calculate the migration time for my protocol, I have proposed the following formula:

Calculating migration time t

$$t = L_{\text{ctrl,dst}} + C_{\text{dst,src}} + L_{\text{dst,src}} + T_{\text{state}} + L_{\text{src,dst}} + L_{\text{dst,src}} + L_{\text{src,cl}} + C_{\text{cl,dst}} + L_{\text{cl,dst}}, \quad (30)$$

where $L_{\alpha,\beta}$ is the latency between α and β, $C_{\gamma,\delta}$ is the connection establishment time between *Engines* γ and δ and T_{state} is the *State* transmission time.

Figure 6 shows that my solution significantly outperforms the state-of-the art migration techniques. I have shown through measurements that migration time can be reduced to a few ms, which is the aim of future 5G networks.

4 Application of the new results

All four theses, presented in this document, contained system development to show the practical applicability of the theoretical results: In Thesis I., I have presented a multi-source system application that included several content servers at Amazon Web Service (AWS) and Chrome and Firefox extensions that intercept YouTube videos and downloads them through those servers. The system has run for eleven months, with more than 75 users, and collected 1,300,000 log records. In Thesis II., I have designed and implemented a BitTorrent extension to a Java-based BitTorrent client. I have created a testbed with a controlled network environment and run several measurements to validate my hypotheses. In Thesis III., I have designed a *PasNet* that implements my uncoded and Coded MUTP protocols. I have run *PasNet* on more than 100 tablets to show its practical potential. I have also presented *PasNet* at multiple conferences and trade shows, including 5G Summit at Dresden, CES, and CCNC at Las Vegas, where it has received a demonstration award. In Thesis IV, I have designed and implemented a browser-based multiplayer gaming application that used my proposed architecture and protocol designs to run measurement and compare my solution with KVM- and Docker based solutions. The gaming application has been presented at multiple conferences and trade shows, including 5G Summit at Dresden, IFA at Berlin and CES, and CCNC at Las Vegas.

Scientific publications

Journal papers

[1] **Patrik J. Braun**, M. Médard, and P. Ekler. "Practical Evaluation of Multi-source Coded Downloads". In: *IEEE Access* 7 (2019). **IF*: 4.098**, pp. 120304–120314. DOI: 10.1109/ACCESS.2019.2936638.

[2] Márton Sipos, **Patrik J. Braun**, D. Lucani, F. H. P. Fitzek, and H. Charaf. "On the Effectiveness of Recoding-based Repair in Network Coded Distributed Storage". In: *Periodica Polytechnica Electrical Engineering and Computer Science* 61.1 (2017), pp. 12–21. DOI: https://doi.org/10.3311/PPee.9377. URL: https://pp.bme.hu/eecs/article/view/9377.

[3] **Patrik J. Braun**, Á. Budai, J. Levendovszky, M. Sipos, P. Ekler, and F. H. P. Fitzek. "Mobile Peer-to-Peer Assisted Coded Streaming". In: *IEEE Access* 7 (2019). **IF*: 4.098**, pp. 159332–159346. DOI: 10.1109/ACCESS.2019.2950800.

Prepared journal papers

[4] **Patrik J. Braun**, D. Malak, M. Médard, and P. Ekler. "Goodput Analysis for Multi-Source Coded Downloads". In: *IEEE Transactions on Wireless Communications* (2019). **IF*: 6.394**.

Conference papers

[5] Dániel Pásztor and **Patrik J. Braun**. "Improving content delivery systems with Network Coding over WebRTC". In: *2017 Automation and Applied Computer Science Workshop (AACS)*. Budapest, Hungary, June 2017.

[6] **Patrik J. Braun**, D. Malak, Muriel Médard, and P. Ekler. "Multi-Source Coded Downloads". In: *IEEE International Conference on Communications (ICC)*. Shanghai, China, May 2019, pp. 1–7.

[7] **Patrik J. Braun**, D. Malak, M. Médard, and P. Ekler. "Enabling Multi-source Coded Downloads". In: *IEEE International Conference on Edge Computing (EDGE)*. Milan, Italy, July 2019.

[8] **Patrik J. Braun**, M. Sipos, P. Ekler, and H. Charaf. "Increasing data distribution in BitTorrent networks by using network coding techniques". In: *Proceedings of 21th European Wireless Conference*. Budapest, Hungary, May 2015, pp. 1–6.

[9] **Patrik J. Braun** and M. Sipos. "Reducing linear dependencies in network coding assisted BitTorrent networks". In: *Automation and Applied Computer Science Workshop (AACS)*. Budapest, Hungary, June 2015.

[10] **Patrik J. Braun** and M. Sipos. "Analysis of power usage on android platform". In: *Automation and Applied Computer Science Workshop (AACS)*. Budapest, Hungary, June 2014.

[11] **Patrik J. Braun**. "Peer-to-peer streaming using Typescript and Network Coding". In: *Automation and Applied Computer Science Workshop (AACS)*. Budapest, Hungary, June 2016.

[12] **Patrik J. Braun** and P. Ekler. "Improving QoS in web-based distributed streaming services with applied network coding". In: *The 10th Jubilee Conference of PhD Students in Computer Science*. **Best Talk Award**. Szeged, Hugnary, 2016, p. 48.

[13] **Patrik J. Braun**, P. Ekler, and F. H. P. Fitzek. "Network coding enhanced browser based Peer-to-Peer streaming". In: *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. Budapest, Hungary, Oct. 2016.

[14] **Patrik J. Braun**, M. Sipos, P. Ekler, and F. H. P. Fitzek. "On the Performance Boost for Peer to Peer WebRTC-based Video Streaming with Network Coding". In: *IEEE International Conference on Communications (ICC)*. Paris, France, May 2017, pp. 1–6.

[15] **Patrik J. Braun**, Péter Ekler, and F. H. P. Fitzek. "Demonstration of a P2P assisted video streaming with WebRTC and Network Coding". In: *14th IEEE Annual Consumer Communications Networking Conference (CCNC)*. **Demonstration Award Runner-Up**. Las Vegas, USA, Jan. 2017.

[16] **Patrik J. Braun**. "Hot content analysis in a P2P-assisted VoD streaming system with network coding". In: *Automation and Applied Computer Science Workshop (AACS)*. **Best Paper of the Section Award**. Budapest, Hungary, June 2017.

[17] Khalid Kahloot, **Patrik J. Braun**, and Péter Ekler. "Behavior Analysis for WebRTC Peer-to-Peer Streaming with Dynamic Topology". In: *13th International Conference on Wireless and Mobile Communications (ICWMC)*. 2017, p. 43. ISBN: 978-1-61208-572-2.

[18] **Patrik J. Braun**, S. Pandi, R. Schmoll, and F. H. P. Fitzek. "On the study and deployment of mobile edge cloud for tactile Internet using a 5G gaming application". In: *14th IEEE Annual Consumer Communications Networking Conference (CCNC)*. Las Vegas, USA, Jan. 2017, pp. 154–159. DOI: `10.1109/CCNC.2017.7983098`.

[19] S. Pandi, R. S. Schmoll, **Patrik J. Braun**, and F. H. P. Fitzek. "Demonstration of mobile edge cloud for tactile Internet using a 5G gaming application". In: *14th IEEE Annual Consumer Communications Networking Conference (CCNC)*. Las Vegas, USA, Jan. 2017, pp. 607–608.

[20] R. Schmoll, S. Pandi, **Patrik J. Braun**, and F. H. P. Fitzek. "Demonstration of VR / AR offloading to Mobile Edge Cloud for low latency 5G gaming application". In: *15th IEEE Annual Consumer Communications Networking Conference (CCNC)*. Las Vegas, USA, Jan. 2018, pp. 1–3. DOI: `10.1109/CCNC.2018.8319323`.

Patent applications

[21] M. Médard, J. László, and **Patrik J. Braun**. *System and Technique for Generating, Transmitting and Receiving Network Coded (NC) QUICK UDP Internet Connections (QUIC) Packets.* PCT/US18/59340. Sept. 2018.

Selected talks and demonstrations

[22] **Patrik J. Braun**, D. Malak, M. Médard, and P. Ekler. "Keynote talk about Multi-Source Coded Downloads for Future Networks". In: *IEEE 5G Summit Dresden 2019.* **Presenter: Patrik J. Braun**. Dresden, Germany, Sept. 2019.

[23] **Patrik J. Braun**, P. Ekler, and F. H. P. Fitzek. "Demonstration of a Network Coding Enhanced Browser-based Peer-to-Peer Streaming system". In: *5G Lab Germany Industry Day 2015.* **Presenter: Patrik J. Braun**. Dresden, Germany, Sept. 2015.

[24] **Patrik J. Braun**, P. Ekler, and F. H. P. Fitzek. "Demonstration of a P2P-assisted Data Distribution with WebRTC". In: *IEEE 5G Summit Dresden 2016.* **Presenter: Patrik J. Braun**. Dresden, Germany, Sept. 2016.

[25] **Patrik J. Braun**, P. Ekler, and F. H. P. Fitzek. "Demonstration of a P2P-assisted video streaming system with WebRTC and Network Coding". In: *2017 Consumer Electronics Show (CES'17).* **Presenter: Patrik J. Braun**. Las Vegas, USA, Jan. 2017.

[26] **Patrik J. Braun**, S. Pandi, R. Schmoll, and F. H. P. Fitzek. "Demonstration of Mobile Edge Cloud for Tactile Internet using a 5G Gaming Application". In: *International Funkausstellung (IFA) 2016.* Presenter: Deutsche Telekom. Berlin, Germany, Sept. 2016.

[27] **Patrik J. Braun**, S. Pandi, R. Schmoll, and F. H. P. Fitzek. "Demonstration of Mobile Edge Cloud for Tactile Internet using a 5G Gaming Application". In: *IEEE 5G Summit Dresden 2016.* Presenter: Sreekrishna Pandi. Dresden, Germany, Sept. 2016.

[28] **Patrik J. Braun**, S. Pandi, R. Schmoll, and F. H. P. Fitzek. "Demonstration of Mobile Edge Cloud for Tactile Internet using a 5G Gaming Application". In: *Tag der Deutschen Einheit 2016.* Presenter: Sreekrishna Pandi. Dresden, Germany, Oct. 2016.

[29] **Patrik J. Braun**, S. Pandi, R. Schmoll, and F. H. P. Fitzek. "Demonstration of Mobile Edge Cloud for Tactile Internet using a 5G Gaming Application". In: *2017 Consumer Electronics Show (CES'17)*. Presenter: Sreekrishna Pandi. Las Vegas, USA, Jan. 2017.

[30] R. Schmoll, S. Pandi, **Patrik J. Braun**, and F. H. P. Fitzek. "Demonstration of VR / AR offloading to Mobile Edge Cloud for low latency 5G gaming application". In: *Centrum für Büroautomation, Informationstechnologie und Telekommunikation (CeBit) 2017*. Presenter: R. Schmoll. Hanover, Germany, Mar. 2017.

[31] R. Schmoll, S. Pandi, **Patrik J. Braun**, and F. H. P. Fitzek. "Demonstration of VR / AR offloading to Mobile Edge Cloud for low latency 5G gaming application". In: *2018 Consumer Electronics Show (CES'18)*. Presenter: R. Schmoll. Las Vegas, USA, Jan. 2018.

[32] R. Schmoll, S. Pandi, **Patrik J. Braun**, and F. H. P. Fitzek. "Demonstration of VR / AR offloading to Mobile Edge Cloud for low latency 5G gaming application". In: *International Funkausstellung (IFA) 2018*. Presenter: Deutsche Telekom. Berlin, Germany, Sept. 2018.

[33] R. Schmoll, S. Pandi, **Patrik J. Braun**, and F. H. P. Fitzek. "Demonstration of VR / AR offloading to Mobile Edge Cloud for low latency 5G gaming application". In: *IEEE 5G Summit Dresden 2018*. Dresden, Germany, Sept. 2018.

[34] R. Schmoll, S. Pandi, **Patrik J. Braun**, and F. H. P. Fitzek. "Demonstration of VR / AR offloading to Mobile Edge Cloud for low latency 5G gaming application". In: *5G Experience Hub 2018*. Warsaw, Poland, Dec. 2018.

References

[Eri18] Ericsson. *Ericsson Mobility Report*. Ericsson. Nov. 2018. URL: https://www.ericsson.com/assets/local/mobility-report/documents/2018/ericsson-mobility-report-november-2018.pdf.

[Ho+03] T. Ho, R. Koetter, M. Medard, D. R. Karger, and M. Effros. "The benefits of coding over routing in a randomized setting". In: *IEEE International Symposium on Information Theory, 2003. Proceedings.* June 2003, pp. 442–. DOI: 10.1109/ISIT.2003.1228459.

[AN07] K. Ausavapattanakun and A. Nosratinia. "Analysis of Selective-Repeat ARQ via Matrix Signal-Flow Graphs". In: *IEEE Transactions on Communications* 55.1 (Jan. 2007), pp. 198–204. ISSN: 0090-6778. DOI: 10.1109/TCOMM.2006.885092.

[GR07] I. S. Gradshteyn and I. M. Ryzhik. *Table of integrals, series, and products.* Seventh. Elsevier/Academic Press, Amsterdam, 2007, pp. xlviii+1171. ISBN: 978-0-12-373637-6.