OPTIMIZATION OF CONTENT CENTRIC NETWORKS

A Dissertation Presented

by

Milad Mahdian

to

The Department of Electrical and Computer Engineering

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in the field of

Electrical Engineering

Northeastern University

Boston, Massachusetts

September 2017
Optimization of Content Centric Networks

Milad Mahdian

Department of Electrical and Computer Engineering,
Northeastern University

September 28, 2017
ACKNOWLEDGEMENTS

As the last touch on my dissertation, I would like to show my sincere appreciation to the people who have supported me throughout this journey. First and foremost, I would like to thank my Ph.D. advisor Professor Edmund Yeh for his consistent support and patient guidance during the last five years. I have learned invaluable lessons from Professor Yeh’s intellectual wisdom, not only in the scientific arena, but also on a personal level. His generous spirit has been, and will always be, a great role model for me.

I also thank my committee members Professor Stratis Ioannidis, and Professor Milica Stojanovic for their thoughtful guidance and insightful comments. Professor Ioannidis’s keen insights and patient support were an invaluable help during the preparation of this thesis.

First chapter of this thesis represents a collaborative work with Dave Oran, Jim Gibson, Somaya Arianfar, Ilya Moiseenko, and Maziar Mirzazad Barijough. This project was mainly undertaken during my summer internship at Cisco Systems, inc., under supervision of Dave Oran. During this stay, I learned invaluable lessons from my colleagues, and I am grateful for them. In particular, I thank Dave for his consistent support and his insightful suggestions, as well as his generosity and hospitality. Moreover, I would like to give special thanks to Jim Gibson for his patient support and mentoring during my internship and his help with the writing of the MIRCC paper.

Chapter two represents a joint work with Professor Muriel Médard at
M.I.T. Professor Médard played an essential role in shaping this thesis. Her intellectual insights were an indispensable source of suggestions and ideas for me.

The last five years at Northeastern University has been a wonderful journey largely due to the friendships I made here. My special thanks go to Aybike Ulusan for her never-ending kindness, generosity, and love. I thank Amir Tadayon, Soroush Irandoust, Roozbeh Raoufi, Nima Attaran Rezaei, and Mehdi Sadeghzadeh for their joyful and lasting friendships. I also thank Ran Liu and Khashayar Kamran for being great teammates, friends, and office neighbors.

Finally, I thank my parents for their unconditional love and support. You are always there for me. Words cannot express what I wish to say to you for all you have done for me.
Contents

1 MIRCC: Multipath-aware ICN Rate-based Congestion Control 6
  1.1 Introduction ................................................. 6
  1.2 Related Work ............................................... 12
  1.3 MIRCC for Single-Path Flows ............................... 15
    1.3.1 Single-Path Overview ................................. 16
    1.3.2 Comparison to RCP ................................. 23
  1.4 MIRCC for Multi-Path Flows ............................... 27
    1.4.1 Multipath Fairness vs. Utilization ................. 27
    1.4.2 Dual-Class Best-Subflow Multipath Rate Management 31
    1.4.3 Multipath Path Management .......................... 33
    1.4.4 Multipath Simulation Results ..................... 38
  1.5 Conclusions ............................................... 44

2 MinDelay: Joint Forwarding and Caching Algorithms in Content Distribution Networks 46
3 Minimum Cost Multicast with Network Coding and Storage in Multi-commodity Settings

3.1 Introduction .............................................. 77
3.2 Network Model ........................................... 81
3.3 Congestion Control for Elastic Rate Demand ............... 93
3.4 Linear Cost Functions ..................................... 95
3.5 Convergence Rate of Dual Subgradient Method ............. 100
   3.5.1 Convergence Rate in the Dual Space ................. 102
   3.5.2 Convergence Rate in the Primal Space ............... 105
3.6 Simulation Results ........................................ 106
3.7 Conclusion ............................................... 115
List of Figures

1.1 Modeling a forwarder under MIRCC .................................. 17
1.2 Linear topology with 4 nodes. ................................. 25
1.3 Rate convergence comparison between (a) MIRCC and (b) 
    RCP’s update algorithm as adapted to ICN. ............... 26
1.4 Diamond Topology ................................................. 30
1.5 Slingshot Topology .................................................. 30
1.6 Dual-Class Best-Subflow Rate Management:  2 Consumers, 
    Slingshot Topology, RTT= 25ms .............................. 38
1.7 Diamond Topology - Scenario 1, Link Utilization vs. Simulation Time(s): each plot shows the link utilization of incoming 
    links for each node.  (Above each graph, the two boxes contain 
    a Node ID from Figure 1.4 and a random link ID.) ........ 39
1.8 Diamond Topology- Scenario 1, CDF of Flows’ Completion 
    Times vs. completion time (ms) ............................... 40
1.9 Abilene Topology: average delivery time(ms) at each node; 
(a) MIRCC vs. RCP: Single path only. (b) MIRCC effect 
of enabling multipath; S.i denotes node i for singlepath. M.i 
denotes node i when multipath is enabled. . . . . . . . . . . . . 42

1.10 Abilene topology. . . . . . . . . . . . . . . . . . . . . . . . . . 43

2.1 A simple topology. . . . . . . . . . . . . . . . . . . . . . . . . . 57
2.2 Network topologies used for the simulations. . . . . . . . . . 64
2.3 Total network delay (sec) vs. request arrival rate (requests/node/sec). 73
2.4 Average total cache hits (requests/node/sec) vs. Arrival rate 
(requests/node/sec). . . . . . . . . . . . . . . . . . . . . . . . . . 74

3.1 Proposed model to incorporate storage. . . . . . . . . . . . . . 86
3.2 Network topologies studied in our experiments. . . . . . . . . 108
3.3 Rate convergence of different commodities for “No Caching” 
and “Edge + P2P Caching” vs. iterations for the Backhaul 
topology with eight peers per peer cloud, with $W = 100$. . . . 110
3.4 Network cost ($f(x)$) vs. average throughput tradeoff for the 
Service Network topology. . . . . . . . . . . . . . . . . . . . . . 111
3.5 Network cost ($f(x)$) vs. average throughput tradeoff for the 
Backhaul topology. . . . . . . . . . . . . . . . . . . . . . . . . . 112
3.6 Network cost ($f(x)$) vs. throughput tradeoff for the Abilene 
topology for different flows. . . . . . . . . . . . . . . . . . . . . . 113
ABSTRACT

Today’s Internet has become widely a distribution network where its end users are not concerned with where the data is located, but mainly interested in obtaining the data they want. To evolve the Internet toward its current primary use, Information-Centric Networking (ICN) architectures were developed with in-network caching as one of their key features. However, they have yet to settle on specific traffic engineering and caching strategies.

In the first chapter of this thesis, we develop MIRCC, a rate-based, multipath-aware congestion control approach for ICN. We first present MIRCC’s algorithm for single-path flows and develop a non-recursive rate-calculation algorithm which achieves max-min fairness, high link utilization and short flow completion time. We then focus on multi-path flows and design a novel hybrid scheme with dual-class rate management, in which each flow has two rate levels: the primary rate is ensured a level of max-min fairness between all flows and the secondary rate is managed to consume remaining bandwidth resulting in full link utilization.

In chapter two, we model and formulate the traffic engineering (forwarding strategy) and in-network caching jointly under ICN architectures into an optimization problem, with the objective of minimizing the network delay. We then solve the problem and develop fully distributed and modular algorithms for both hop-by-hop forwarding and caching strategies. Through extensive simulations, we demonstrate that our forwarding and caching strate-
gies outperform the state-of-the-art schemes out there in terms of network delay.

In chapter three, we propose a novel framework for incorporating storage functionality into the problem of minimum-cost multicast in a multi-commodity setting with network coding. The proposed framework can model architectures such as ICN, Content Distribution Networks (CDNs), Peer-to-Peer (P2P) networks, as well as CDN-P2P hybrid architectures in a coded environment. We first formulate the problem with given multicast rates using convex cost functions and present primal-dual algorithms to solve the convex problem. We then add congestion control mechanism to accommodate elastic traffic. Next as a special case, we consider linear cost functions and adopt a simple and distributed dual subgradient algorithm to find the minimum-cost integrated multicast and caching strategy.
Introduction to Information Centric Networking

The communication network design of IP was aimed to solve point-to-point communication problems, while recent growth in e-commerce, digital media, and smartphone applications has resulted in the Internet mostly being used as a distribution network. The main issues arising from this incompatibility between these two models are [1]:

- Availability: Reliable and fast content delivery needs pre-planned and application-specific mechanisms such as CDNs and P2P networks, and/or requires excessive bandwidth capacity.
- Location-dependence: Mapping content to host locations requires complicated configuration and implementation of network services.

The direct and unified approach to solve these issues is by replacing where with what. i.e., we need to replace the traditional end-to-end communication model with one based on the identity of data. Named-Data Networking (NDN) [2], and Content-Centric Networking (CCN) [1], are two proposed network architectures for the Information-Centric Networking (ICN) that replaces the traditional host-to-host communication paradigm with one in which the main networking functionalities are directly driven by object identifiers, rather than host addresses. This change is derived from the primary use of today’s Internet. Today’s Internet end users are mostly interested
in obtaining the data they want, rather than where it is located. Aiming at redesigning the entire Internet architecture, including core routers, with named data as the central element of the communication, NDN has gained a lot of attention over the past few years. An essential element of NDN, or in general ICN, is the caching strategy. That is, how many copies of the available contents to store in the network and where. The advantages of in-network caching have been demonstrated in different networking models, such as P2P, and Publish-Subscribe networks.

In order for NDN to accomplish content delivery, two types of packets are being used [1], [3]. Nodes requesting data objects generate Interest Packets (IPs), which contain the data names and a random nonce value. Then, the IPs are forwarded along the routes determined locally by the Forwarding Information Base (FIB) at each node. The FIB determines to which neighboring node(s) one should transmit the IP. Received IPs are recorded in the Pending Interest Table (PIT) at each node, thus allowing repeated requests for the same object to be suppressed. Also, by using nonce value contained in each IP, duplicate IPs received over different paths may be discarded. Nodes can have the ability to store some of the data objects they receive in their Content Stores (CSs) according to a caching strategy. When receiving an IP, if the node has the requested content object in its CS, it creates a Data Packet (DP) containing the requested data object. The DP is, then, transmitted back along the path taken by the corresponding IP, as recorded by the PIT at each node. Requesting nodes as well as the intermediate nodes along
the path may choose to cache the content object enclosed in the received DPs based on the caching strategy, so that they can fulfill the received IPs requesting the same content in the future.
Chapter 1

MIRCC: Multipath-aware ICN
Rate-based Congestion Control

1.1 Introduction

Information-Centric Networking [1] is a new network architecture that differs radically from IP. The question of how ICN networks achieve congestion control is one of many that must be resolved if ICN is to be successful.

An initial question is whether ICN congestion control should be window-based or rate-based. Window-based congestion control, such as is used by TCP (e.g. [4]), allows a window of outstanding packets which grows and shrinks in response to the absence or presence of congestion indications. Rate-based congestion control, such as used by RCP [5], gives endpoints a sending rate that is increased or decreased based on the absence or pres-
ence of congestion. The relative characteristics of the two approaches are a standard topic in Networking textbooks.

The success of TCP/IP, the decades of deployment, and the extensive research into multiple generations of TCP congestion control schemes make a windowing approach a natural candidate for ICN congestion control (e.g. [6], [7], [8]) Nevertheless, this chapter investigates a rate-based alternative for ICN congestion control. Several observations led us to explore this approach.

First, the differences between ICN and IP mean the best approaches may differ. In particular:

- ICN’s Interest/Data exchange is receiver-driven. Thus, in an overload situation, congestion control must affect the behavior of the requester of the data, not the producer whose Data messages are the proximate cause of the overload.

- As a consequence of ICN’s stateful forwarding, the Interest’s upstream path and the corresponding Data message’s downstream path are symmetric. This allows rate feedback state to aggregate well on the downstream path.

- ICN networks may use in-network caching, which affects where Data is pulled from, even on a message-by-message basis.

- Several interlocking ICN properties (the absence of address-based sessions, more robust delivery of Nacks, tolerance for out-of-order delivery, quick suppression of looping packets due to stateful forwarding) all
support the possibility that ICN flows may be able to derive significant benefit from multipath.

Second, TCP congestion control is imperfect and still evolving in spite of considerable investment over decades, as shown by a random sample ([4], [9], [10], [11], [12]) among countless recent papers. Only a subset of this research is specific to window-based congestion control, but clearly this is not a solved problem.

Third, the overwhelming installed base of mature window-based TCP/IP flow control and the requirement to be friendly to these flows likely prevents the deployment of alternate schemes regardless of technical tradeoffs, which in turn prevents such alternate schemes from evolving and improving over time.

In summary, we believe, both because of the differences between ICN and IP and because RCP was a late entrant in IP congestion control, that rate-based congestion control for ICN is worth investigating. Given the multiple dimensions on which the two approaches differ and the complexity of algorithmic options, however, a short-term proof of superiority of either over the other is not feasible.

We identify a number of goals for an ICN congestion control scheme. It should provide some level of fairness between flows while allowing those flows to benefit from available network resources. (We target flow fairness, in spite of the arguments of [13].) Given a stable overall load, flows should receive a stable allocation of resources that maximizes network utilization.
while avoiding congestion and excess latency. In response to changes in load, the scheme should converge quickly to maintain those properties.

Allowing ICN flows to benefit fully from available network resources requires support for multipath. Coupling multipath with other objectives is of particular interest. When flows have multiple paths to their respective content, the tension between fairness and maximizing utilization is more complicated to resolve.

From a practical standpoint, we want a scheme that does not require forwarders to maintain long-lived per-flow state (in addition to the short-lived state in the Pending Interest Table).

Finally, we consider only distributed algorithms in the style of the current Internet, not centralized omniscient controllers.

In this chapter, we propose MIRCC (Multipath-aware ICN Rate-based Congestion Control), a rate-based multipath-aware congestion control scheme for ICN networks. It is inspired by but significantly divergent from IP RCP. Two main contributions of MIRCC:

- better convergence time with less overshoot and oscillation than RCP.
- a multipath strategy allowing a flow to fully utilize the network resources along all the available paths to it, while maintaining fairness among competing flows regardless of number of paths that each flow has.

The main elements of MIRCC include an algorithm in each forwarder cal-
culating Dual-Class rates for each link, protocol mechanisms to communicate rates and path identifiers to consumers in Data messages, an algorithm in each consumer to determine Interest sending rates for each class and to determine a sensible distribution of Interests across available paths, a mechanism for the consumer to hint at the desired path for an Interest, and forwarder support for interpreting the hint. Additionally, as a path discovery mechanism, the consumer can omit hints from Interests, in which case forwarders with multiple next hops will probabilistically distribute Interests across those next hops.

The remainder of this chapter is structured as follows. After describing related work (Section 1.2), we first (Section 1.3) explain how the RCP-style end-to-end feedback loop operates in an ICN network, considering only single-path flows. The feedback loop is driven by $R(t)$, the per-link stamping rate: we explain MIRCC’s per-link algorithm that iteratively updates $R(t)$ and show that, as flows start and finish, using the MIRCC algorithm (1.5) leads to better convergence time and less overshoot/oscillation than the classic RCP per-link algorithm (1.1) adapted to ICN.

We then generalize MIRCC to the multipath case (Section 1.4) to address this issue: How can we assure a flow a fair share of the network resources while maximizing network utilization? We describe MIRCC Dual-Class Best-Subflow scheme (Section 1.4.2), a novel multipath-aware rate-based congestion control scheme which allows flows to fully utilize network resources along their available paths while maintaining a fairness among competing flows, re-
gardless of number of paths each flow has.

We have an ICN forwarder under development that is tracking the various drafts under development in the IRTF ICN Research Group. In addition to its normal target environments, we have integrated our ICN forwarder with ns-3 for simulation purposes. The basic elements of this integration are an ns-3 “model” that adapts ns-3 to our ICN forwarding code, an ns-3 consumer application that implements the various endpoint behaviors given in the chapter, and a simple producer application that responds to Interests with Data messages of configurable size. Throughout the chapter, we have used this simulation environment to evaluate the various alternatives.

**Key Definitions** The concept of a flow and related terms are used throughout the chapter, with the following specific meanings.

A flow is a set of Interests generated by a single client, sharing a prefix, and handled by one or more producers.

Multipath is the situation where packets for a single flow take (or have the potential to take) more than one path.

Subflow describes a subset of the Interests and Data messages that belong to the same flow and follow the same path. (Subflow is only useful in the presence of multipath.)
1.2 Related Work

ICN Congestion Control  In [14], Lei et al. propose adopting RCP for ICN networks. They illustrate the potential benefits of RCP for the ICN architecture, including high network utilization and max-min fairness and superior results compared to NDN-AIMD, an ICN window-based congestion control scheme. The authors demonstrate that, due to the symmetric and hop-by-hop routing of the NDN architecture, RCP is an attractive candidate to tackle the congestion control problem of such networks. As compared to this research, our chapter proposes an alternate rate calculation algorithm with improved convergence time and less overshoot, and proposes an approach for managing multipath with rate-based congestion control.

In [6], Carofiglio et al. formulate the problem of joint multipath congestion control and request forwarding in ICN as a global optimization problem. By decomposing the problem into two subproblems of maximizing user throughput and minimizing overall network cost, they develop a receiver-driven window-based congestion control and a hop-by-hop dynamic request forwarding algorithm, respectively.

In [15], Wang et al. propose a simple hop-by-hop Interest shaping algorithm for NDN to avoid network congestion and achieve optimal network resource utilization. The proposed solution accounts for the interdependence between Interests and Contents in opposite directions and shares link bandwidth optimally without extra message exchange.
In [3], Yeh et al., propose VIP, a systematic framework for joint dynamic Interest request forwarding and cache placement, within the context of the NDN architecture. In VIP, a virtual control plane operates on the user demand rate for data objects in the network, and an actual plane handles Interest Packets and Data Packets. The authors develop jointly optimal and distributed dynamic forwarding and caching algorithms within the virtual plane. Later, a congestion control mechanism is added to the framework, which acts to maximize user utilities subject to network stability.

TCP/IP Congestion Control Many improvements to TCP congestion control have been proposed and adopted over the years. We do not attempt a comprehensive summary of general TCP congestion control state of the art, but highlight some areas particularly relevant to our work.

Explicit Congestion Notification (ECN) allows end-to-end notification of network congestion without dropping packets [16]. If ECN is successfully negotiated, an ECN-aware router with active queue management (AQM), when experiencing congestion, may set a mark in the IP header instead of dropping a packet in order to signal impending congestion. The receiver of the packet echoes the congestion indication to the sender, which reduces its transmission rate as if it detected a dropped packet.

MP-TCP [17] describes a window-based congestion control algorithm operating at the endpoints of a multipath TCP flow that increases (or at least does not decrease) flow throughput compared to a single path and is fair to
other traffic.

In [5], Dukkipati proposed the Rate Control Protocol (RCP), a congestion control scheme intended for IP. RCP’s main objective is to minimize flow completion times by allowing flows to rapidly determine their correct rate. Each packet carries a rate field in its header. A route maintains a flow rate, $R(t)$, on each link. On each packet, the router “stamps” the header rate with $R(t)$ unless the packet already carries a lower value. The sender thus learns the bottleneck rate of a flow’s path in one round trip. As a new rate is delivered with each packet, routers update $R(t)$ so that senders will update their rate. The equation for $R(t)$ attempts to find a rate that fills links while keeping queue sizes close to zero.

The RCP rate update equation used by the authors differs from our MIRCC rate update equation, and we compare the two in detail. The RCP equation is

$$R(t) = R(t - T) \left( 1 + \frac{T}{d} \left( \frac{\alpha(\eta C - y(t)) - \beta \frac{q(t)}{d}}{C} \right) \right), \quad (1.1)$$

where $\eta$ is the target link utilization, $d$ is a moving average of the RTT measured across all packets, $T$ is the update interval ($T \leq d$), $R(t - T)$ is the latest updated rate, $C$ is the link capacity, $y(t)$ is the measured aggregate input traffic rate during the last update interval, $q(t)$ is the instantaneous queue size, and $\alpha, \beta$ are parameters chosen for stability and performance.

Although RCP, as compared to TCP and XCP [18], improves flow comple-
tion times, promotes high network utilization and provides max-min fairness [19], it has not seen much deployment due to a number of factors: 1) In IP, the packets with rate feedback do not necessarily traverse the same path as the packets with the rate stamp (asymmetric routing). 2) RCP may exhibit oscillatory behavior and lead to significant overshoot or undershoot during abrupt changes in traffic load, as described in [20]. 3) RCP is relatively slow in converging to the equilibrium point compared to XCP and PERC[21]. 4) RCP is not TCP-friendly, imposing a challenging compatibility issue in IP. 5) RCP does not address multipath forwarding.

PERC[21] is a proactive congestion control scheme inspired by message-passing algorithms, proposed for datacenters with speeds higher than 100 Gb/s. PERC explicitly computes, in a decentralized fashion, rates independent of congestion signals such as the incoming traffic. The authors show that for high speed networks (100 Gb/s and above), PERC improves convergence times by a factor of 7 compared to reactive explicit rate control protocols such as RCP. However, for lower speed networks (10 Gb/s) PERC’s convergence time is higher than RCP’s and other reactive protocols’.

1.3 MIRCC for Single-Path Flows

This section discusses the application of MIRCC to a flow with a single path. Starting with the special case of single-path flows allows for simpler presentation of core elements of MIRCC before moving on to MIRCC’s multipath
Dual-Class Best-Subflow scheme, described in Section 1.4.

The heart of the system is the choice of control equation that the forwarders run, on each link, to calculate $R(t)$, the per-link stamping rate. This algorithm determines the overall distributed system’s tradeoff between responsiveness and stability. A feedback loop operates to communicate rate information to consumers.

1.3.1 Single-Path Overview

MIRCC assumes an ICN architecture and system such as provided by CC-CNx [22] or NDN [23]. These architectures include separate Interests (initiated by consumers) and Data messages (sent by producers in response to Interests). These two types of messages are treated quite differently. A consumer wishing to retrieve a flow of Data objects starts by sending Interests at some initial rate determined by the MIRCC consumer logic. These are routed through the ICN network, with temporary state being created in the *Pending Interest Table* (PIT) at each forwarder. Upon receiving an Interest, a producer generates a Data message in response, and sends it back toward the consumer. Each Data message carries a rate value, initialized to $MaxRate$ by the MIRCC producer logic. The forwarder at each hop forwards the Data message along the reverse path of the corresponding Interest, with the link for the reverse hop determined by the PIT entry.

The MIRCC component in each forwarder maintains a rate, $R(t)$, for each link. As the Data message traverses the path back to the consumer, MIRCC
Figure 1.1: Modeling a forwarder under MIRCC

at each forwarder compares the \( R(t) \) assigned to the local link from which it received the Data message and the rate in the received Data message. If the local \( R(t) \) is smaller than the carried rate value, the forwarder updates the outgoing Data message’s value with the local \( R(t) \). Hence, the bottleneck forwarder for the packet’s path is the one which last updated the packet’s rate value, and the bottleneck rate is the corresponding link rate.

When the Data packet arrives at the consumer, the rate in the Data packet informs the consumer of the rate for the flow’s path. The consumer uses this value to update its sending rate. The forwarder calculation of \( R(t) \) at each link is smoothed, so the consumer immediately applies the received \( R(t) \) value as its new sending rate without any further smoothing.

The per-link \( R(t) \) calculation includes a feedback mechanism: if the link is overloaded, the flow stamping rate is reduced and if the link is underloaded, the flow stamping rate is increased.

ICN (unlike TCP) is receiver-controlled via Interests. The consumer varies the rate at which it sends Interests as the mechanism to control end-to-end load of both Interests and the (generally much larger) Data messages.
As will become clear, this mechanism and others rely on both consumers and forwarders maintaining an estimate of the ratio of Data message size to Interest size.

In addition to MIRCC’s per-link $R(t)$ calculator/stamper, each forwarder is assumed to have a per-link Interest Shaper, e.g. as described in [15] (see figure 1.1). This resource control mechanism monitors traffic per-link and paces Interests per-link. This shaper, given its per-link functions, is a natural place for a software implementation to calculate MIRCC’s $R(t)$.

The shapers at either end of the link, by pacing Interests as necessary, control returning Data message load and avoid Data message overflow on the reverse path. Note that the operation of this Interest shaper mechanism also acts to control bandwidth of the outgoing Interests and leaves enough headroom on the reverse path for the other end of the link to send its Interests.

As a consequence of the Interest shaper mechanism, Data messages should rarely be dropped: the Interest shaper books the reverse bandwidth for the Data message via the shaping. During an overload, Interests back up in the shaper until some are eventually not forwarded. In general, these overload Interests are not silently dropped, but are Nacked back towards the consumer, using the bandwidth implicitly “booked” in anticipation of a returning Data message. Receipt of a Nack informs the consumer that an Interest was not delivered due to congestion. Assuming all the data is required, the consumer will know to re-transmit a new Interest with the same name, without needing to wait for a timeout.
Under MIRCC, Nacks also carry $R(t)$. As the rate information is only over a partial path, the consumer only updates its flow rate if the received $R(t)$ is smaller than the current rate. Unlike with windowing schemes, the consumer in our rate-based scheme does not take any additional steps on receipt of Nacks, as the network can set $R(t)$ as low as it chooses.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R(t)$</td>
<td>The stamping rate computed at time $t$</td>
</tr>
<tr>
<td>$R_p(t)$</td>
<td>The stamping rate for the Dual-Class primary traffic class</td>
</tr>
<tr>
<td>$R_s(t)$</td>
<td>The stamping rate for the Dual-Class secondary traffic class</td>
</tr>
<tr>
<td>$C$</td>
<td>Capacity of incoming link</td>
</tr>
<tr>
<td>$C_s(t)$</td>
<td>Secondary class capacity of incoming link</td>
</tr>
<tr>
<td>$\hat{N}$</td>
<td>Equivalent number of flows with full rate</td>
</tr>
<tr>
<td>$T$</td>
<td>Period of rate calculation iteration</td>
</tr>
<tr>
<td>$q(t)$</td>
<td>Inflated instantaneous queue size</td>
</tr>
<tr>
<td>$y(t)$</td>
<td>Inflated incoming Interest rate during $[t - T, t)$</td>
</tr>
<tr>
<td>$d(t)$</td>
<td>Smoothed average RTT</td>
</tr>
<tr>
<td>$\beta(t)$</td>
<td>Self-tuned parameter chosen for stability</td>
</tr>
<tr>
<td>$\eta$</td>
<td>Target link utilization</td>
</tr>
</tbody>
</table>

Table 1.1: List of Parameters.

**Equation for calculating $R(t)$**

MIRCC at each forwarder maintains $R(t)$ for each outgoing link to which it sends Interests, and offers this rate to each passing flow. The classic RCP calculation, proposed in [5], is shown here as equation (1.1). However, as explained earlier, this calculation overshoots significantly when new flows
arrive, leading to packet drops. In addition, its convergence time is relatively slow. To address this issue, we instead developed the following algorithm. In this equation, each forwarder computes $R(t)$ for each of its outgoing links as the aggregate of $base\_rate(t)$, calculated by (non-recursive) equation (1.3) to split the allowed link bandwidth among the passing flows, and $excess\_rate(t)$, calculated by (recursive) equation (1.4) to fill the extra available bandwidth with traffic equally. As with RCP [5], MIRCC does not need to count the actual number of flows.\(^1\) RCP uses an equivalent number of flows with full rate, given by $C/R(t - T)$. Our simulations show that while this estimate sometimes works well, it often reacts to new flow arrivals relatively late, resulting in queue build-up and ultimately oscillation and Data packet drops. Hence, we have chosen a different estimate given by

$$\hat{N} = \max(C, y(t))/R(t - T),$$

(1.2)

which takes into account $y(t)$ and, as we show in simulation, does not suffer from those problems.

Each forwarder then computes $base\_rate(t)$ as follows:

$$base\_rate(t) = \frac{\eta C - \beta(t) \frac{\eta(t)}{\delta(t)}}{\hat{N}},$$

(1.3)

\(^1\)Although $N$ is described as the number of flows, $\hat{N}$ is a derived value based on the observed change in link traffic that follows a change in $R(t)$. Flows that are not elastic or are bottlenecked on links do not respond to changes in $R(t)$. The total count of flows using the link in a given period is thus neither needed nor wanted. Hence we use the description “full-rate-equivalent flows”.

20
where $\eta$ is the fraction of physical link bandwidth that $R(t)$ will attempt to fill, $C$ is the capacity of the incoming link as shown in Figure 1.1, $q(t)$ is the instantaneous queue size at the Interest Shaper inflated to the expected size of the corresponding reverse Data message, $d(t)$ is the average RTT, and $\hat{N}$ is the estimated number of flows. Further, $\beta(t)$ is a self-tuned positive parameter given by

$$
\beta(t) = \max \left( \beta', \frac{y(t) - y(t - T)}{y(t)} \right)
$$

where $\beta'$ is a constant parameter set to 0.1 by default, $y(t)$ is based on the measured incoming Interest rate arriving during the time interval $[t - T, t)$ to be transmitted on the outgoing link, but is inflated to the expected size of the corresponding reverse Data messages.\(^2\) Note that both $y(t)$ and $q(t)$, although based on Interest rate, are intended to estimate and manage the incoming rate of the much larger Data messages from the upstream neighbor and prevent buildup in the upstream neighbor’s Data message queue, which is why they are scaled up by the Data/Interest inflation factor. As $y(t)$ and $q(t)$ include Nacked Interests, $y(t)$ and $q(t)$ are both overestimates of the incoming rate and queue buildup of the Data messages at the upstream neighbor.

\(^2\)The “incoming Interests” making up $y(t)$ are incoming to the link’s output Interest shaper, not Interests arriving on the link. We cannot accurately describe $y(t)$ as “outgoing Interests” because $y(t)$ includes Interests that are not transmitted in $[t - T, t)$, either because the shaper Nacks or queues them, and excludes Interests transmitted in $[t - T, t)$ that were queued in previous periods.
To estimate the inflation factor, each forwarder maintains, on each link, exponentially weighted moving averages of the size of output Interests and input Data messages. The ratio of these is the inflation factor.

The next step in the scheme is to allow the flows not passing any saturated links to fill the extra available bandwidth with traffic equally. On each link, the incremental excess rate is given by

\[
excess\_rate(t) = R(t - T) - y(t)/\dot{N}.
\] (1.4)

To avoid high-frequency oscillation, an exponential weighted moving average (EWMA) is applied to both \(base\_rate(t)\) in (1.3) and \(excess\_rate(t)\) in (1.4). In our simulations, the EWMA \(\alpha\) is 0.5. Combining these elements gives us the stamping rate, i.e.,

\[
R(t) = base\_rate\_ewma(t) + excess\_rate\_ewma(t). \tag{1.5}
\]

**Forwarder Determination of \(T\) and \(d(t)\)**

\(T\) is the period for an iteration of the \(R(t)\) calculation. For instance, \(R(t - T)\) is the flow stamping value used in the interval \([t - T, t)\) and \(R(t)\) is the stamping value in the interval \([t, t + T)\). Traffic statistics are collected in the interval \((t - T, t)\) for use when calculating \(R(t)\).

In general, \(T\) should be as short as possible, and no greater than RTT.

\footnote{i.e. \(\mu(t) = (1 - \alpha)\mu(t - T) + \alpha x(t)\)}
The limitation on reducing $T$ is that traffic measurements become lumpy and misleading as $T \Rightarrow 0$, tending towards either 0 or $C$ (link rate).

$d(t)$ is the EWMA of the per-packet RTT.

For single-path flows, the consumer’s main job is to send Interests at a rate consistent with the $R(t)$ received for the flow.

Recall, however, that $R(t)$ is the rate in bits of the Data messages that the consumer should be receiving, not the bit rate of Interests the consumer is sending. The consumer must therefore deflate the path’s $R(t)$ by the Data/Interest inflation factor to determine the Interest rate. The consumer using $R(t)$ (like the forwarder calculating $R(t)$) must therefore have some estimate of Interest-to-Data inflation. For some applications (e.g. constant bit rate video flows), this is known to the consumer a priori. For other applications, the data sizes may be known by the producer ahead of time and communicated from producer to consumer using a Manifest.

1.3.2 Comparison to RCP

Comparing RCP’s algorithm to MIRCC’s algorithm As shown in our simulations and also reported in [20] and [21], RCP’s algorithm has a relatively slow convergence time and a tendency to overshoot or undershoot during abrupt changes in traffic load. This is due to the way that rate is calculated in RCP, given in equation (1.1). The rate calculation in RCP is done in a recursive manner, thereby taking some time to converge to the equilibrium point. Therefore its reaction to abrupt changes in traffic load
becomes slow, resulting in overshoot and undershoot. By using the rate calculation given in equations (1.3) - (1.5), MIRCC removes this recursive element. Further, equations (1.4) and (1.5) reduce the dependency of the algorithm outcome on sudden changes in traffic and act as a safeguard against oversooting and undershooting.

To compare the performance of MIRCC’s and RCP’s algorithms, we have adapted the IP RCP algorithm to ICN and compared the two algorithms in the simple ICN scenario depicted in Figure 1.2. In this linear network topology with 4 nodes, node 0 is the requesting node, and Node 3 is the producer. Each link has a capacity of 1Mbps. There are 4 consumer applications installed on Node 0. Consumer 2 is a long flow working as the background traffic. The other three consumers start sending packages at time 83 seconds and compete for bandwidth. As can be seen by comparing figures 1.3a and 1.3b, MIRCC converges to the new rate much faster than RCP. Also, the traffic rate in MIRCC shows less oscillation. It is worth noting that in both of these simulations we have used the Interest Shaper at each forwarder. The presence of the Interest Shaper has reduced the oscillatory behavior of the RCP scheme greatly.

Also note that MIRCC and RCP have similar network utilization and flow completion time properties. This is easy to see, as equations (1.3)-(1.5) reach the same equilibrium point as equation (1.1).
Comparing RCP-like schemes under IP and ICN. Another RCP comparison of interest is between the control loops for IP RCP and for ICN MIRCC. In IP RCP, the rate is stamped in sent traffic, received by the destination, and echoed back to the sender in the resultant ACK. In the event of a link becoming overloaded, the feedback path is at least half a round trip (if the overloaded link is very close to the receiver). Regardless of which link is overloaded, the sender will originate an RTT’s worth of traffic between the forwarder detecting the link overload and the sender’s receipt of a rate that has been updated based on the overload.

In an ICN network, due to symmetric paths of Interest and Data, the updated rate can be stamped directly in returning Data messages rather than needing to be stamped in Interests and echoed back by the receiver. The feedback time to a given sender is thus the wait for the next Data message plus at most half a round trip (if the overloaded link is very close to the receiver). The sender thus originates at most a half round trip’s worth of traffic (and very little if the overloaded link is close to the sender) between detection of the overload and the sender’s receipt of a rate reflecting the overload.
Figure 1.3: Rate convergence comparison between (a) MIRCC and (b) RCP’s update algorithm as adapted to ICN.
1.4 MIRCC for Multi-Path Flows

An RCP-style rate-based congestion control algorithm such as MIRCC is relatively straightforward for single-path flows. The calculations at each link are designed to be simple. The sending endpoint simply reacts to the returned rate. The producer endpoint in ICN has no responsibilities.

The existence of multiple paths with different rates introduces several new challenges.

- Given multiple paths for a flow, each with their own rate, at what rate should the consumer send the Interests for a flow? What per-flow fairness does this imply? What portion of the Interests should follow each path?

- By what mechanisms are the multiple paths and their rates discovered? By what mechanisms are Interests directed onto the appropriate paths?

For this chapter, the producer at the upstream end of each path is assumed capable of supplying the entire flow. The effects of opportunistic caching of some subset of the Data messages in a flow are not considered at this time.

1.4.1 Multipath Fairness vs. Utilization

Two multipath flow rate concepts are used throughout the chapter:
• The best-subflow rate is the largest rate of any of the flow’s subflows. Having all flows sending at their best subflow rate achieves max-min fairness between flows, by an extension of the argument that single-flow RCP achieves max-min fairness between flows. Network utilization is not maximized by this approach.

• The total per-subflow rate is the sum of the rates of all the flow’s subflows. Sending at this rate (distributing Interests onto each subflow at its rate) maximizes network utilization, by an extension of the argument that single-flow RCP maximizes network utilization. (Each subflow is simply treated as a separate single-path flow.) Max-min fairness between flows is not achieved.

Some examples motivate why sending at the best-subflow rate does not maximize network utilization, and why sending at the total per-subflow rate does not achieve max-min fairness.

In figure 1.4, consumer A has two 4-hop paths to producer A, where the two paths share $l_a, l_g$. Suppose further that many other consumers, not shown, are using these links such that $R(t)$ is only minimally affected by the presence of an additional consumer.

In both scenarios shown, the bottleneck links for the two subflows are $l_c, l_d$ and the rates returned to the consumers in Data messages traversing the two subflows are thus 10Mbps. A key difference between the scenarios is that, in scenario 2, the minimum cut is less than the sum of the subflows,
i.e. $l_g < l_c + l_d$.

Consumer A might request at the best-subflow rate, i.e. 10Mbps. In this case however, the consumer is not taking advantage of multipath and network utilization is not maximized.

Alternatively, consumer A might request data at the total per-subflow rate, i.e. $\sum_{\text{subflow}} R = 20$ Mbps, to take advantage of multipath. In scenario 2, sending at 20Mbps means taking more than the 15Mbps single flow rate on $R_g$. Not only might that seem unfair (e.g. to consumer E, which will send at 15Mbps on its single path), but consumer A has no way of distinguishing the two scenarios to treat them differently, since the returned per-subflow stamping rates are identical.

Alternatively, in scenario 2, an omniscient consumer A, if it had a way to understand the entire topology, might send at 15Mbps, taking a fair share of the bottleneck link $l_g$. This would be fair to consumer E and maximize utilization of the bottleneck link. But MIRCC is a distributed mechanism rather than an omniscient SDN-style mechanism.

In figure 1.5, assume that only the consumers shown are present, so an additional consumer has a large effect on $R(t)$. If only Consumer 2 is running, its single path is bottlenecked on $l_c$ and its rate is 9Mbps. If only Consumer 1 is running, it has 2 paths but $l_a$ is the bottleneck link, with the 15Mbps of traffic split between $l_b$ and $l_c$. (Multiple distribution between those two links are possible.)

With Consumers 1 and 2 both running, $l_a$ has 2 flows/3 subflows, $l_c$
Figure 1.4: Diamond Topology

Figure 1.5: Slingshot Topology

has 2 flows/2 subflows and \( l_b \) has 1 flow/1 subflow. With only two flows, the decision of how to distribute Consumer 1’s traffic between \( l_b \) and \( l_c \) has a big effect on the stamping rates on those links, the link responsible for the lowest path stamping rate can change between \( l_a \) and \( l_b/l_c \). Absent the MIRCC mechanisms, oscillations in bottleneck link can lead to instability in total traffic, per-link traffic, and per-flow traffic.
1.4.2 Dual-Class Best-Subflow Multipath Rate Management

MIRCC’s Dual-Class Best-Subflow scheme has the following main elements.

The Dual-Class scheme includes primary and secondary traffic classes. Each Interest is marked with a class. For each link, the owning ICN forwarder maintains per-class rates: $R_p(t)$ and $R_s(t)$, a generalization of $R(t)$ as described in section 1.3. Primary and secondary rates are both stamped in each Data message, which thus carries its path’s lowest value for each rate.

The forwarder’s maintenance of $R_p(t)$ for each link is very similar to the maintenance of single-path $R(t)$ in section 1.3. (In the absence of secondary class traffic, the two are identical.) However $y_p(t)$ reflects only primary class Interests, with consequent effect on dependent parameters such as $\hat{N}$. ($q(t)$ includes all the queued Interests, regardless of class, for both primary and secondary rate calculation.)

\[ \hat{N}_p = \max(C, y_p(t))/R_p(t - T), \quad (1.6) \]
\[ \text{base} \_ \text{rate}_p(t) = \frac{\eta C - \beta_p(t)q(t)}{N_p}, \quad (1.7) \]
\[ \beta_p(t) = \max\left(\beta', \frac{y_p(t) - y_p(t - T)}{y_p(t)}\right), \quad (1.8) \]
\[ \text{excess} \_ \text{rate}_p(t) = R_p(t - T) - y_p(t)/\hat{N}_p, \quad (1.9) \]
\[ R_p(t) = \text{base} \_ \text{rate} \_ \text{ewma}_p(t) + \text{excess} \_ \text{rate} \_ \text{ewma}_p(t). \quad (1.10) \]
A MIRCC consumer sends primary class Interests at the flow’s best-subflow rate. Thus, primary class traffic is max-min fair between flows, but does not maximize network utilization.

The purpose of the secondary class is to “soak up” bandwidth not used by primary class traffic. This is achieved by three key differences between the primary and secondary classes.

First, the capacity term in the algorithm for $R_s(t)$ is $C_s(t)$ rather than $C$. The link’s secondary capacity consists of what is not used by primary traffic.

$$C_s(t) = \max(C - y_p(t), 0). \quad (1.11)$$

Also, $y(t)$ is replaced by $y_s(t)$, which reflects only secondary class traffic.

Second, the per-link shaper under load, drops queued secondary class Interests to make room for new primary class Interests.

Third, a MIRCC consumer sends secondary class Interests at the flow’s total-subflow secondary rate. Thus, secondary class traffic maximizes network utilization without having any particular per-flow fairness property.\(^4\)

The combined fairness of the primary and secondary class traffic is therefore a mixture, but with a core of per-flow max-min fair traffic sent in the (preferred) primary class.

$$\hat{N}_s = \max\{\max(C_s(t), y_s(t))/R(t - T), 1\}, \quad (1.12)$$

\(^4\)Note that secondary class traffic has per-subflow fairness.
\[ base\_rate_s(t) = \frac{\eta C_s(t) - \beta_s(t) \frac{\eta(t)}{d(t)}}{\hat{N}_s}, \quad (1.13) \]

\[ \beta_s(t) = \max \left( \beta'_s, \frac{y_s(t) - y_s(t - T)}{y_s(t)} \right), \quad (1.14) \]

\[ excess\_rate_s(t) = R_s(t - T) - y_s(t)/\hat{N}_s. \quad (1.15) \]

\[ R_s(t) = base\_rate\_ewma_s(t) + excess\_rate\_ewma_s(t). \quad (1.16) \]

### 1.4.3 Multipath Path Management

MIRCC management of multiple paths includes a significant role for the consumer endpoint. Recall that MIRCC is a distributed algorithm that does not rely on an omniscient SDN-style controller. Even with long-lived per-flow state (which MIRCC specifically tries to avoid), no node other than the endpoint has the full view of the available paths that is needed to achieve fairness between flows. Thus, the end-user is assigned this enhanced role in MIRCC. The end-user also has the job of distributing Interests onto subflow paths.

For secondary rate Interests, the consumer endpoint sending at the total per-subflow secondary rate simply distributes the Interests onto each subflow based on its secondary rate.

For the primary class traffic, although sent at the rate of the best subflow, MIRCC distributes the traffic across the available subflow paths rather than sending all the traffic on the best-subflow path. This balancing achieves smoother evolution of traffic on each path and link by avoiding situations
where small changes in a subflow’s bandwidth can cause all the primary class traffic to switch onto and off of that subflow.

We choose a distribution proportional to the primary “stamping” rate on each path, subject to the constraint that no path and no link carry more than their stamping rate.

Having specified the desired behavior and identified that the consumer endpoint’s transmission strategy achieves those behaviors, the next question is what mechanisms MIRCC uses to produce those results. In particular,

- How is it determined that multiple paths exist?
- How is traffic distributed onto paths?
- Each subflow must be identifiable by the consumer, so that the consumer knows the rates of each path and the changes in the best path and best path rate over time.
- To avoid overloading any lower-rate path, the consumer must be able to steer Interests onto paths.
- To discover paths, initially and over time, the consumer must send some subset of Interests without a steering hint, and the network must do some probabilistic multipath forwarding that leads to discovery of useful paths. (Recall that only a single initial message for any given path is needed to discover the path’s rate.) For Interests without steering
hints, the forwarders must apply some probabilistic multipath forwarding strategy to allow paths to be discovered.

Path Identification

By definition each subflow has a separate path, so identifying subflows is equivalent to distinguishing their paths. The pathId is signaled in a Data message header TLV and is calculated hop-by-hop, by hashing node+link identifier at each hop with the partial pathId received in the incoming Data message header. The Data message is sent downstream with an updated TLV.

Directing Traffic onto a Specific Subflow Path

The consumer, when distributing Interests to subflows, includes subflow-specific hints in the Interest that the network uses to steer the Interest along the subflow’s path. The following mechanisms have been considered. The network of course can act on these hints as it chooses.

Bloom Filter Steering Hints: One approach is to use “Bloom filter steering” as in [24]. The Data message carries a header TLV in which is constructed a Bloom filter of the node+link identifier chosen at each split point. The consumer includes this Bloom filter in future Interests for the corresponding subflow. At split points, the forwarder looks up, in the received Bloom Filter, the node+link for each nexthop in the FIB entry. A hit indicates which nexthop should be used.
(For our simulations, steering is currently done with a modification of this scheme that was easier to implement, but is not appropriate for a real network. Each forwarder tracks, in its FIB, the upstream (received) partial pathhash, upstream face, and downstream (sent) partial pathhash. For Interests with a hint, the hash is looked up in the FIB entry’s list of downstream partial hashes and the Interest is sent to the corresponding upstream interface, with the corresponding upstream hash. This simulation shortcut preserves the key properties of the Bloom Filter approach, though never misses.)

Random Value Steering Hints: An alternate approach is for the consumer to include a “random value steering” header TLV in Interests. The forwarder at each split point hashes this to select the next hop. The Data message, in addition to carrying a separate pathId, reflects back the steering hint received in the message. The consumer can probe for different paths by using new random numbers, and can detect path changes if old random numbers return new paths.

Path Discovery

Consumers sending Interests for a new flow do not yet have a list of paths for that flow. Therefore, initial Interests must be sent without a steering hint. A forwarder receiving a hintless Interest that matches a FIB entry with multiple next hops makes a probabilistic decision among the next hops. The MIRCC forwarder weights the next hops by the bandwidth of the outgoing links,
though other schemes are possible.

To allow for the possibility of newly available paths, MIRCC consumers send a subset of Interests without steering hints even after discovering an initial set of paths. Currently, consumers send 3% of Interests without a steering hint.

**Sample Cases**

Three scenarios give a flavor of the operation of the Dual-Class Best-Subflow scheme.

First, consider a scenario with many elastic flows. The secondary rates on the various paths will tend to be 0. Consumers will thus send at the best-subflow rate with a per-flow max-min fair outcome.

A second scenario is where the links are used by a single flow. Figure 1.4, scenario 2 is a useful example topology. The best-subflow rate will be 10Mbps and traffic will be split equally with 5Mbps of primary class traffic on each path. The secondary rate on each path will initially be approximately 5Mbps for a total of 10Mbps. The flow converges to sending 5Mbps of secondary rate traffic, 2.5Mbps on each path, filling $l_g$, the flow’s bottleneck link.

Third, with a small number of flows sharing the links, each flow will get its best-subflow rate of primary traffic, and the secondary rate capacity will be shared between the flows. Flows with more paths will get more of the secondary rate traffic.
1.4.4 Multipath Simulation Results

Figure 1.6: Dual-Class Best-Subflow Rate Management: 2 Consumers, Sling-shot Topology, RTT=\~25ms

Using the simulator, we ran many simulations of these multipath schemes and present representative results. In summary, MIRCC uses two rate classes to achieve fairness and good network utilization. Additionally, MIRCC converges to sensible behavior even with a small number of flows.

For the first scenario, we use the slingshot topology of figure 1.5, with a long-lived Consumer talking to Producer 1 and a shorter-lived Consumer talking to Producer 2. In figure 1.6, we show the rates for both Consumers over time for MIRCC’s dual-class best-subflow schemes.
Initially, Consumer 1’s multipath flow is the only flow in the topology. Consumer 1 gets the full bandwidth of $l_a$ (15Mbps derated by $\eta$). As shown by the dashed graphs, the primary class component is 12Mbps (corresponding to the best subflow through $l_b$) and the secondary rate component is 3Mbps ($l_a$’s remaining available bandwidth).

Once Consumer 2 starts up, MIRCC’s rate calculations and consumer responses evolve to a new convergence point that fills $l_a$, with no excess bandwidth available. Consumer 1 bottlenecks on $l_a$ and Consumer 2 bottlenecks on $l_c$, the shared 9Mbps second hop. It happens that the Consumers’
use of $l_a$ converge to a ratio of just under 2:1. (If Producer 2 is moved to E3 so that the 12Mbps $l_b$ is the shared second hop, the Consumers both bottleneck on $l_a$, which is shared evenly at 1:1.) When Consumer 2 finishes, the situation reverts to the starting use.

For the second scenario, we use the Diamond topology - Scenario 1 as depicted in Figure 1.4 except that only consumers A-D are present. Flow arrival time follows a Poisson process with mean 0.5 seconds. We have two types of consumers chosen randomly upon arrival to be either a short flow (1000 messages) with probability 0.99, or a long one (50,000 messages) with probability 0.01. Long flows are considered to be the background traffic as the short flows come and go. Figures 1.7 and 1.8 are plotted to demonstrate the network link utilization and the Cumulative Distribution Function (CDF)
(a) MIRCC vs. RCP.
Figure 1.9: Abilene Topology: average delivery time (ms) at each node; (a) MIRCC vs. RCP: Single path only. (b) MIRCC effect of enabling multipath; S.\textit{i} denotes node \textit{i} for singlepath. M.\textit{i} denotes node \textit{i} when multipath is enabled.
of the flows’ completion time, respectively. Figure 1.7 shows high and stable link utilization even though messages are arriving and leaving quite often. (On average, a new flow joins in every 0.5 seconds and leaves after receiving all of its messages.) Further, simulations do not exhibit any dropped Interest or Data messages.

In Figure 1.8, the CDF of the flows’ completion time has been plotted, showing that MIRCC, unlike TCP-like schemes, does not discriminate against short flows.

In the third scenario, we consider the performance of MIRCC in the Abilene topology in figure 1.10 as used in [6]. This scenario features a more realistic set of nodes, dynamic consumer arrival time and varying flow size. Three repositories, at nodes 12, 16, 19, each store content under a given name prefix (/Amazon, /Google, /Warner, respectively). Each content is chunked in 5000 Data messages of size 1.8 kbytes each. The size of each Interest mes-
sage is 300 bytes. Request arrival time follows a Poisson distribution with mean 3 seconds. The consumer is randomly assigned to one of nodes 11, 13, 14, 15, 17, 18, 20, 21 with a prefix among the aforementioned prefixes. We are mainly interested in 1) comparing MIRCC to RCP (with multipath forwarding disabled for fair comparison) and 2) demonstrating the performance of MIRCC when multipath mechanism is enabled. As shown in figure 1.10, we can modify the topology to enable or disable multipath forwarding. In Figure 1.9a, we show that MIRCC beats RCP in terms of average delivery time at every node. This further verifies that our rate calculation scheme does a better job than RCP’s algorithm of converging to the adjusted rate. In figure 1.9b, M.i denotes node i when multipath is enabled. S.i denotes node i when multipath is disabled. As can be seen, the multipath solution significantly reduces delivery times, except at nodes 14 and 15, which experience a slightly higher latency due to an increase in the number of flows on links (3, 4), (4, 5) sharing available bandwidth fairly. Still, the overall advantage of using multipath is significant.

1.5 Conclusions

In this chapter, we proposed MIRCC, a rate-based multipath-aware congestion control scheme for ICN networks, which is inspired by but significantly divergent from RCP. We first considered single-path flows and developed a non-recursive rate-calculation algorithm which achieves max-min fairness,
high link utilization and short flow completion time. The proposed algorithm, as shown through extensive simulation, has much better convergence time and less overshoot and oscillation than classic RCP. We then focused on multipath routing/forwarding and tackled the problem of coupling multipath routing/forwarding with objectives such as fairness, congestion control and network utilization. We showed that assuring full network utilization while maintaining fairness among competing flows regardless of number of paths each one has is a challenging problem. We designed a hybrid scheme with dual-class rate management, in which each flow has two rate levels: the primary rate, which is managed to ensure a level of max-min fairness between all flows, and the secondary rate, which is managed to consume remaining bandwidth to achieve full link utilization.

Possible further topics, among many, include the handling of UDP-tunneled links between devices, studying the effect of wider RTT spreads among competing consumers, and accommodating the effects of opportunistic caching by forwarders, particularly, following the resource pooling principle [25].
Chapter 2

MinDelay: Joint Forwarding and Caching Algorithms in Content Distribution Networks

2.1 Introduction

Research on information-centric networking (ICN) architectures over the past few years has brought focus on a number of central network design issues. One prominent issue is how to jointly design traffic engineering and caching strategies to maximally exploit the bandwidth and storage resources of the network for optimal performance. While traffic engineering and caching have been investigated separately for many years, their joint optimization within an ICN setting is still an under-explored area.
There have been many interesting papers on caching strategies within ICN architectures [26], [27], [28], [29], [30], [31], [32], [33], [34], [35]. When designing and evaluating the effectiveness of a cache management scheme for such networks, the primary performance metrics have been cache hit probability [30], the reduction of the number of hops to retrieve the requested content [31], age-based caching [32], [33] or content download delay [34].

Recently, in [35], Ioannidis and Yeh formulate the problem of cost minimization for caching networks with fixed routing and linear link cost functions, and propose an adaptive, distributed caching algorithm which converges to a solution within a (1-1/e) approximation from the optimal.

Similarly, there have been a number of attempts to enhance the traffic engineering in ICN [6], [36], [37], [38], [39]. In [6], Carofiglio et al., formulate the problem of joint multipath congestion control and request forwarding in ICN as an optimization problem. By decomposing the problem into two subproblems of maximizing user throughput and minimizing overall network cost, they develop a receiver-driven window-based Additive-Increase Multiplicative-Decrease (AIMD) congestion control algorithm and a hop-by-hop dynamic request forwarding algorithm which aim to balance the number of pending Interest Packets of each content object (flow) across the outgoing interfaces at each node. Unfortunately, the work in [6] does not consider caching policies.

Posch et al. [36] propose a stochastic adaptive forwarding strategy which maximizes the Interest Packet satisfaction ratio in the network. The strat-
egy imitates a self-adjusting water pipe system, where network nodes act as crossings for an incoming flow of water. Each node then intelligently guides Interest Packets along their available paths while circumventing congestion in the system.

In [3], Yeh et al., present one of the first unified frameworks for joint forwarding and caching for ICN networks with general topology, in which a virtual control plane operates on the user demand rate for data objects in the network, and an actual plane handles Interest Packets and Data Packets. They develop VIP, a set of distributed and dynamic forwarding and caching algorithms which adaptively maximizes the user demand rate the ICN can satisfy.

In this work, we present a new unified framework for minimizing congestion-dependent network cost by jointly choosing node-based forwarding and caching variables, within a quasi-static network scenarios where user request statistics vary slowly. We consider the network cost to be the sum of link costs, expressed as increasing and convex functions of the traffic rate over the links. When link cost functions are chosen according to an M/M/1 approximation, minimizing the network cost corresponds to minimizing the average request fulfillment delay in the network. As caching variables are integer-constrained, the resulting joint forwarding and caching (JFC) optimization problem is NP-hard. To make progress toward an approximate solution, we focus on a relaxed version of the JFC problem (RJFC), where caching variables are allowed to be real-valued. Using techniques first introduced in [40],
we develop necessary optimality conditions for the RJFC problem. We then leverage this result to design MinDelay, an adaptive and distributed joint forwarding and caching algorithm for the original JFC problem, based on a version of the conditional gradient, or Frank-Wolfe algorithm. The MinDelay algorithm elegantly yields feasible routing variables and integer caching variables at each iteration, and can be implemented in a distributed manner with low complexity and overhead.

Finally, we implement the MinDelay algorithm using our Java-based network simulator, and present extensive experimental results. We consider three competing schemes, including the VIP algorithm [3], which directly competes against MinDelay as a jointly optimized distributed and adaptive forwarding and caching scheme. Over a wide range of network topologies, simulation results show that while the VIP algorithm performs well in high request arrival rate regions, MinDelay typically has significantly better delay performance in the low to moderate request rate regions. Thus, the MinDelay and VIP algorithms complement each other in delivering superior delay performance across the entire range of request arrival rates.

2.2 Network Model

Consider a general multi-hop network modeled by a directed and (strongly) connected graph \( \mathcal{G} = (\mathcal{N}, \mathcal{E}) \), where \( \mathcal{N} \) and \( \mathcal{E} \) are the node and link sets, respectively. A link \( (i, j) \in \mathcal{E} \) corresponds to a unidirectional link, with ca-
capacity $C_{ij} > 0$ (bits/sec). We assume a content-centric setting, e.g. [1], where each node can request any data object from a set of objects $\mathcal{K}$. A request for a data object consists of a sequence of Interest Packets which request all the data chunks of the object, where the sequence starts with the Interest Packet requesting the starting chunk, and ends with the Interest Packet requesting the ending chunk. We consider algorithms where the sequence of Interest Packets corresponds to a given object request are forwarded along the same path.

Assume that loop-free routing (topology discovery and data reachability) has already been accomplished in the network, so that the Forwarding Interest Base (FIB) tables have been populated for the various data objects. Further, we assume symmetric routing, where Data Packets containing the requested data chunks take the same path as their corresponding Interest Packets, in the reverse direction. Thus, the sequence of Data Packets for a given object request also follow the same path (in reverse). For simplicity, we do not consider interest suppression, whereby multiple Interest Packets requesting the same named data chunk are collapsed into one forwarded Interest Packet. The algorithm we develop can be extended to include Interest suppression, by introducing a virtual plane in the manner of [3].

For $k \in \mathcal{K}$, let $src(k)$ be the source node of content object $k$. Each node in the network has a local cache of capacity $c_i$ (object units), and can

---

1We assume there is one source for each content object for simplicity. The results generalize easily to the case of multiple source nodes per content object.
optionally cache Data Packs passing through on the reverse path. Note that since Data Packs for a given object request follow the same path, all chunks of a data object can be stored together at a caching location. Interest Packets requesting chunks of a given data object can enter the network at any node, and exit the network upon being satisfied by a matching Data Packet at the content source for the object, or at the nodes which decide to cache the object. For simplicity, we assume all data objects have the same size $L$ (bits). The results of the paper can be extended to the more general case where object sizes differ.

We focus on quasi-static network scenarios where user request statistics vary slowly. Let $r_i(k) \geq 0$ be the average exogenous rate (in requests/sec) at which requests for data object $k$ arrive (from outside the network) to node $i$. Let $t_i(k)$ be the total average arrival rate of object $k$ requests to node $i$. Thus, $t_i(k)$ includes both the exogenous arrival rate $r_i(k)$ and the endogenous arrival traffic which is forwarded from other nodes to node $i$.

Let $x_i(k) \in \{0, 1\}$ be the (integer) caching decision variable for object $k$ at node $i$, where $x_i(k) = 1$ if object $k$ is cached at node $i$ and $x_i(k) = 0$ otherwise. Thus, $t_i(k)x_i(k)$ is the portion of the total incoming request rate for object $k$ which is satisfied from the local cache at node $i$ and $t_i(k)(1 - x_i(k))$ is the portion forwarded to neighboring nodes based on the forwarding strategy. Furthermore, let $\phi_{ij}(k) \in [0, 1]$ be the (real-valued) fraction of the traffic $t_i(k)(1 - x_i(k))$ forwarded over link $(i, j)$ by node $i \neq src(k)$. Thus, $\sum_{j \in O(i, k)} \phi_{ij}(k) = 1$, where $O(i, k)$ is the set of neighboring nodes for which
node $i$ has a FIB entry for object $k$. Therefore, total average incoming request rate for object $k$ to node $i$ is

$$t_i(k) = r_i(k) + \sum_{l \in \mathcal{I}(i,k)} t_l(k)(1 - x_l(k))\phi_{il}(k), \quad (2.1)$$

where $\mathcal{I}(i,k)$ is the set of neighboring nodes of $i$ which have FIB entries for node $i$ for object $k$.

Next, let $F_{ij}$ be the Data Packet traffic rate (in bits/sec) corresponding to the total request rate (summed over all data objects) forwarded on link $(i, j) \in \mathcal{E}$:

$$F_{ij} = \sum_{k \in \mathcal{K}} L \cdot t_i(k)(1 - x_i(k))\phi_{ij}(k). \quad (2.2)$$

Note that by routing symmetry and per-hop flow balance, the Data Packet traffic of rate $F_{ij}$ actually travels on the reverse link $(j, i)$.

As in [40] and [41], we assume the total network cost is the sum of traffic-dependent link costs. The cost on link $(j, i) \in \mathcal{E}$ is due to the Data Packet traffic of rate $F_{ij}$ generated by the total request rate forwarded on link $(i, j)$, as in (2.2). We therefore denote the cost on link $(j, i)$ as $D_{ij}(F_{ij})$ to reflect this relationship.\footnote{Since Interest Packets are small compared to Data Packets, we do not account for costs associated with the Interest Packet traffic on link $(j, i)$.} We assume $D_{ij}(F_{ij})$ is increasing and convex in $F_{ij}$. To implicitly impose the link capacity constraint, we assume $D_{ij}(F_{ij}) \to \infty$ as
\( F_{ij} \to C_{ji} \) and \( D_{ij}(F_{ij}) = \infty \) for \( F_{ij} \geq C_{ji} \). As an example,
\[
D_{ij}(F_{ij}) = \frac{F_{ij}}{C_{ji} - F_{ij}}, \quad \text{for } 0 \leq F_{ij} < C_{ji}.
\tag{2.3}
\]
gives the average number of packets waiting for or under transmission at link \((j, i)\) under an \( M/M/1 \) queuing model [42], [43]. Summing over all links, the network cost \( \sum_{i,j} D_{ij}(F_{ij}) \) gives the average total number of packets in the network, which, by Little’s Law, is proportional to the average system delay of packets in the network.

### 2.3 Optimization Problem

We now pose the Joint Forwarding and Caching (JFC) optimization problem in terms of the forwarding variables \((\phi_{ij}(k))_{(i,j)\in\mathcal{E},k\in\mathcal{K}}\) and the caching variables \((x_i(k))_{i\in\mathcal{N},k\in\mathcal{K}}\) as follows:

\[
\begin{align*}
\min \ & \sum_{(i,j)\in\mathcal{E}} D_{ij}(F_{ij}) \\
\text{subject to:} & \\
\sum_{j\in\mathcal{O}(i,k)} \phi_{ij}(k) & = 1, \quad \text{for all } i \in \mathcal{N}, k \in \mathcal{K} \\
\phi_{ij}(k) & \geq 0, \quad \text{for all } (i, j) \in \mathcal{E}, k \in \mathcal{K} \\
\sum_{k\in\mathcal{K}} x_i(k) & \leq c_i, \quad \text{for all } i \in \mathcal{N} \\
x_i(k) & \in \{0, 1\}, \quad \text{for all } i \in \mathcal{N}, k \in \mathcal{K}.
\end{align*}
\tag{2.4}
\]
The above mixed-integer optimization problem can be shown to be NP-hard [44]. To make progress toward an approximate solution, we relax the problem by removing the integrality constraint in (2.4). We formulate the Relaxed JFC (RJFC) problem by replacing the integer caching decision variables $x_i(k) \in \{0, 1\}$ by the real-valued variables $\rho_i(k) \in [0, 1]$:

$$
\begin{align*}
\min D & \triangleq \sum_{(i,j) \in E} D_{ij}(F_{ij}) \\
\text{subject to:} & \\
\sum_{j \in O(i,k)} \phi_{ij}(k) &= 1, \quad \text{for all } i \in N, k \in K \\
\phi_{ij}(k) &\geq 0, \quad \text{for all } (i,j) \in E, k \in K \\
\sum_{k \in K} \rho_i(k) &\leq c_i, \quad \text{for all } i \in N \\
0 &\leq \rho_i(k) \leq 1, \quad \text{for all } i \in N, k \in K.
\end{align*}
$$

(2.5)

It can be shown that $D$ in (2.5) is non-convex with respect to (w.r.t.) $(\phi, \rho)$, where $\phi \equiv (\phi_{ij}(k))_{(i,j) \in E, k \in K}$ and the caching variables $\rho \equiv (\rho_i(k))_{i \in N, k \in K}$.

In this work, we use the RJFC formulation to develop an adaptive and distributed forwarding and caching algorithm for the JFC problem.

We proceed by computing the derivatives of $D$ with respect to the forwarding and caching variables, using the technique of [40]. For the forwarding variables, the partial derivatives can be computed as

$$
\frac{\partial D}{\partial \phi_{ij}(k)} = (1 - \rho_i(k))Lt_i(k)\delta_{ij}(k),
$$

(2.6)
where the marginal forwarding cost is

\[ \delta_{ij}(k) = D'_{ij}(F_{ij}) + \frac{\partial D}{\partial r_j(k)}. \]  (2.7)

Note that \( \frac{\partial D}{\partial r_j(k)} \) in (2.7) stands for the marginal cost due to a unit increment of object \( k \) request traffic at node \( j \). This can be computed recursively by

\[ \frac{\partial D}{\partial r_j(k)} = 0, \text{ if } j = \text{src}(k), \]

\[ \frac{\partial D}{\partial r_i(k)} = (1 - \rho_i(k))L \sum_{j = O(i,k)} \phi_{ij}(k)\delta_{ij}(k), \text{ if } i \neq \text{src}(k). \]  (2.8)

Finally, we can compute the partial derivatives w.r.t. the (relaxed) caching variables as follows:

\[ \frac{\partial D}{\partial \rho_i(k)} = -Lt_i(k) \sum_{j = O(i,k)} \phi_{ij}(k)\delta_{ij}(k). \]  (2.9)

The minimization in (2.5) is equivalent to minimizing the Lagrangian function

\[ L(F, \lambda, \mu) = \sum_{(i,j) \in \mathcal{E}} D_{ij}(F_{ij}) - \sum_{i,k} \lambda_{ik} \left( \sum_{j} \phi_{ij}(k) - 1 \right) + \sum_{i} \mu_i \left( \sum_{k \in \mathcal{K}} \rho_i(k) - c_i \right). \]  (2.10)
subject to the following constraints:

\[ 0 \leq \rho_i(k) \leq 1, \quad \text{for all } i \in \mathcal{N}, k \in \mathcal{K}, \]
\[ \phi_{ij}(k) \geq 0, \quad \text{for all } (i, j) \in \mathcal{E}, k \in \mathcal{K}, \]
\[ \mu_i \geq 0, \quad \text{for all } i \in \mathcal{N}. \]

A set of necessary conditions for a local minimum to the RJFC problem can now be derived as

\[
\frac{\partial D}{\partial \phi_{ij}(k)} \begin{cases} 
= \lambda_{ik}, & \text{if } \phi_{ij}(k) > 0 \\
\geq \lambda_{ik}, & \text{if } \phi_{ij}(k) = 0
\end{cases}
\] (2.11)

\[
\frac{\partial D}{\partial \rho_i(k)} \begin{cases} 
= -\mu_i, & \text{if } 0 < \rho_i(k) < 1 \\
\geq -\mu_i, & \text{if } \rho_i(k) = 0 \\
\leq -\mu_i, & \text{if } \rho_i(k) = 1
\end{cases}
\] (2.12)

with the complementary slackness condition

\[ \mu_i \left( \sum_{k \in \mathcal{K}} \rho_i(k) - c_i \right) = 0, \text{for all } i \in \mathcal{N}. \] (2.13)

The conditions (2.11)-(2.13) are necessary for a local minimum to the RJFC problem, but upon closer examination, it can be seen that they are not sufficient for optimality. An example from [40] shows a forwarding configuration (without caching) where (2.11) is satisfied at every node, and yet...
the operating point is not optimal. In that example, $t_i(k) = 0$ at some node $i$, which leads to (2.11) being automatically satisfied for node $i$. This degenerate example applies as well to the joint forwarding and caching setting considered here.

A further issue arises for the joint forwarding and caching setting where $\rho_i(k) = 1$ for some $i$ and $k$. In this case, the condition in (2.11) at node $i$ is automatically satisfied for every $j \in O(i,k)$, and yet the operating point need not be optimal. To illustrate this, consider the simple network shown in Figure 2.1 with two objects 1 and 2, where $r_1(1) = 1$, $r_1(2) = 1.5$, $c_1 = 1$, $c_2 = 0$ and $src(1) = src(2) = 3$. At a given operating point, assume $\rho_1(1) = 1$, $\phi_{12}(1) = 1$ and $\phi_{13}(2) = 1$. Thus, $\rho_1(2) = 0$, $\phi_{13}(1) = 0$ and $\phi_{12}(2) = 0$. It is easy to see that all the conditions in (2.11) and (2.12) are satisfied. However, the current operating point is not optimal. An optimal point is in fact reached when object 2 is cached at node 1, instead. That is, $\rho_1(2) = 1$, $\phi_{13}(1) = \phi_{13}(2) = 1$.

This example, along with the example in [40], show that when $\rho_i(k) = 1$ or
$t_i(k) = 0$, node $i$ still needs to assign forwarding variables for object $k$ in the optimal way. In the degenerate cases where $\rho_i(k) = 1$ or $t_i(k) = 0$, removing the term $t_i(k)(1 - \rho_i(k))$ from (2.11) prevents non-optimal forwarding choices. Furthermore, since the term $t_i(k)(1 - \rho_i(k))$ is not a function of $j \in O(i, k)$, it can also be removed from condition (2.11) when $t_i(k)(1 - \rho_i(k)) > 0$. We therefore focus on the following modified conditions

\[
\delta_{ij}(k) \left\{ \begin{array}{ll}
= \delta_i(k), & \text{if } \phi_{ij}(k) > 0 \\
\geq \delta_i(k), & \text{if } \phi_{ij}(k) = 0.
\end{array} \right.
\] (2.14)

\[
t_i(k)\delta_i(k) \left\{ \begin{array}{ll}
= \mu_i, & \text{if } 0 < \rho_i(k) < 1 \\
\leq \mu_i, & \text{if } \rho_i(k) = 0 \\
\geq \mu_i, & \text{if } \rho_i(k) = 1.
\end{array} \right.
\] (2.15)

where

\[
\delta_i(k) = \min_{m \in O(i, k)} \delta_{im}(k).
\] (2.16)

In (2.15), we used the fact that \(\sum_{j=O(i,k)} \phi_{ij}(k)\delta_{ij}(k) = \delta_i(k)\) if condition (2.14) is satisfied. Condition (2.15) suggests a structured caching policy. If we sort the data objects in decreasing order with respect to the “cache scores” \(\{t_i(k)\delta_i(k)\}\), and cache the top $c_i$ objects, i.e. set $\rho_i(k) = 1$ for the top $c_i$ objects, then condition (2.15) is satisfied. This is indeed the main idea underlying our proposed caching algorithm described in the next section.
2.4 Distributed Algorithm: MinDelay

The conditions in (2.14)-(2.15) give the general structure for a joint forwarding and caching algorithm for solving the RJFC problem. For forwarding, each node $i$ must decrease those forwarding variables $\phi_{ij}(k)$ for which the marginal forwarding cost $\delta_{ij}(k)$ is large, and increase those for which it is small. For caching, node $i$ must increase the caching variables $\rho_{i}(k)$ for which the cache score $t_{i}(k)\delta_{i}(k)$ is large and decrease those for which it is small.

To describe the joint forwarding and caching algorithm, we first describe a protocol for calculating the marginal costs, and then describe an algorithm for updating the forwarding and caching variables.

Note that each node $i$ can estimate, as a time average, the link traffic rate $F_{ij}$ for each outgoing link $(i, j)$. This can be done by either measuring the rate of received Data Packets on each of the corresponding incoming links $(j, i)$, or by measuring the request rate of Interest Packets forwarded on the outgoing links $(i, j)$. Thus, given a functional form for $D_{ij}(.),$ node $i$ can compute $D'_{ij}(F_{ij}).$

Assuming a loop-free routing graph on the network, one has a well-defined partial ordering where a node $m$ is downstream from node $i$ with respect to object $k$ if there exists a routing path from $m$ to $src(k)$ through $i$. A node $i$ is upstream from node $m$ with respect to $k$ if $m$ is downstream from $i$ with respect to $k$.

To update the marginal forwarding costs, the nodes use the following
protocol. Each node $i$ waits until it has received the value $\partial D/\partial r_j(k)$ from each of its upstream neighbors with respect to object $k$ (with the convention $\partial D/\partial r_{\text{src}}(k) = 0$). Node $i$ then calculates $\partial D/\partial r_i(k)$ according to (2.8) and broadcasts this to all of its downstream neighbors with respect to $k$. The information propagation can be done by either piggybacking on the Data Packets of the corresponding object, or by broadcasting a single message regularly to update the marginal forwarding costs of all the content objects at once.

Having described the protocol for calculating marginal costs, we now specify the algorithm for updating the forwarding and caching variables. Our algorithm is based on the conditional gradient or Frank-Wolfe algorithm [45]. Let

$$\Phi^n = \left[ \begin{array}{c}
(\phi^n_{ij}(k))_{i \in \mathcal{N}, k \in \mathcal{K}, j \in \mathcal{O}(i, k)} \\
(\rho^n_i(k))_{i \in \mathcal{N}, k \in \mathcal{K}}
\end{array} \right]$$

be the vector of forwarding and caching variables at iteration $n$. Then, the conditional gradient method is given by

$$\Phi^{n+1} = \Phi^n + a^n(\bar{\Phi}^n - \Phi^n), \quad (2.17)$$

where $a^n \in (0, 1]$ is a positive stepsize, and $\bar{\Phi}^n$ is the solution of the direction finding subproblem

$$\bar{\Phi}^n \in \arg \min_{\Phi \in \mathcal{F}} \nabla D(\Phi^n)'(\Phi - \Phi^n). \quad (2.18)$$
Here, $\nabla D(\Phi^n)$ is the gradient of the objective function with respect to the forwarding and caching variables, evaluated at $\Phi^n$. The set $F$ is the set of forwarding and caching variables $\Phi$ satisfying the constraints in (2.5), seen to be a bounded polyhedron.

The idea behind the conditional gradient algorithm is to iteratively find a descent direction by finding the feasible direction $\Phi^n - \Phi^n$ at a point $\Phi^n$, where $\Phi^n$ is a point of $F$ that lies furthest along the negative gradient direction $-\nabla D(\Phi^n)$ [45].

In applying the conditional gradient algorithm, we encounter the same problem regarding degenerate cases as seen in Section 2.3 with respect to optimality conditions. Note that when $t_i(k)(1 - \rho_i(k)) = 0$, the $\frac{\partial D}{\partial \phi_{ij}(k)}$ component of $\nabla D(\Phi^n)$ is zero, and thus provides no useful information for the optimization in (2.18) regarding the choice of $\Phi^n$. On the other hand, when $t_i(k)(1 - \rho_i(k)) > 0$, eliminating this term from $\frac{\partial D}{\partial \phi_{ij}(k)}$ in (2.18) does not change the choice of $\Phi^n$, since $t_i(k)(1 - \rho_i(k)) > 0$ is not a function of $j \in \mathcal{O}(i,k)$. Motivated by this observation, we define

$$G(\Phi^n) \triangleq \begin{bmatrix} \left( \delta^n_{ij}(k) \right)_{i \in \mathcal{N}, k \in \mathcal{K}, j \in \mathcal{O}(i,k)} \\ -t^n_i(k) \sum_{j=\mathcal{O}(i,k)} \phi^n_{ij}(k) \delta^n_{ij}(k) \end{bmatrix}_{i \in \mathcal{N}, k \in \mathcal{K}},$$

where $\delta^n_{ij}(k)$ and $t^n_i(k)$ are the marginal forwarding costs and total request arrival rates, respectively, evaluated at $\Phi^n$.

We consider a modified conditional gradient algorithm where the direction
finding subproblem is given by

$$\tilde{\Phi}^n \in \arg \min_{\Phi \in F} G(\Phi^n)'(\Phi - \Phi^n). \quad (2.20)$$

It can easily be seen that (2.20) is separable into two subproblems.

The subproblem for \((\phi_{ij}(k))\) is given by

$$\begin{cases}
\min \sum_{(i,k)} \sum_{j \in \mathcal{O}(i,k)} \delta^n_{ij}(k)(\phi_{ij}(k) - \phi^n_{ij}(k)) \\
\text{subject to:} \\
\sum_{j \in \mathcal{O}(i,k)} \phi_{ij}(k) = 1, \quad \text{for all } i \in \mathcal{N}, k \in \mathcal{K} \\
\phi_{ij}(k) \geq 0, \quad \text{for all } i \in \mathcal{N}, k \in \mathcal{K}, j \in \mathcal{O}(i,k).
\end{cases} \quad (2.21)$$

where

$$\delta^n_{ij}(k) = D'_{ij}(F^n_{ij}) + \frac{\partial D}{\partial r^n_j(k)}. \quad (2.22)$$

It is straightforward to verify that a solution \(\tilde{\phi}^n_{ij}(k) = (\tilde{\phi}^n_{ij}(k))_{j \in \mathcal{O}(i,k)}\) to (2.21) has all coordinates equal to zero except for one coordinate, say \(\tilde{\phi}^n_{im}(k)\), which is equal to 1, where

$$m \in \arg \min_{j \in \mathcal{O}(i,k)} \delta^n_{ij}(k). \quad (2.23)$$

corresponds to an outgoing interface with the minimal marginal forwarding cost. Thus, the update equation for the forwarding variables is: for all \(i \in \mathcal{N}\),

$$\phi^{n+1}_{ij}(k) = (1 - a^n)\phi^n_{ij}(k) + a^n \tilde{\phi}^n_{ij}(k), \forall k \in \mathcal{K}, j \in \mathcal{O}(i,k). \quad (2.24)$$
The subproblem for \((\rho_i(k))\) is equivalent to

\[
\begin{align*}
\text{max} & \sum_{(i,k)} \omega_n^{n}(k)(\rho_i(k) - \rho^{n}_i(k)) \\
\text{subject to:} & \\
\sum_{k \in \mathcal{K}} \rho_i(k) & \leq c_i, \quad \text{for all } i \in \mathcal{N} \\
0 & \leq \rho_i(k) \leq 1, \quad \text{for all } i \in \mathcal{N}, k \in \mathcal{K}.
\end{align*}
\]

(2.25)

where \(\omega_n^{n}(k) = t^{n}_i(k) \left( \sum_{j \in \mathcal{O}(i,k)} \phi_{ij}^{n}(k) \delta_{ij}^{n}(k) \right)\). The subproblem (2.25) is a max-weighted matching problem which has an integer solution. For node \(i\), let \(\omega_i^{n}(k_1) \geq \omega_i^{n}(k_2) \geq \ldots \geq \omega_i^{n}(k_{|\mathcal{K}|})\) be a re-ordering of the \(\omega_i^{n}(k)\)'s in decreasing order. A solution \(\bar{\rho}_i^{n}\) to (2.25) has \(\bar{\rho}_i^{n}(k) = 1\) for \(k \in \{k_1, k_2, \ldots, k_{c_i}\}\), and \(\bar{\rho}_i^{n}(k) = 0\) otherwise. That is, \(\bar{\rho}_i^{n}(k) = 1\) for the \(c_i\) objects with the largest \(\omega_i^{n}(k)\) values, and \(\bar{\rho}_i^{n}(k) = 0\) otherwise. The update equation for the caching variables is: for all \(i \in \mathcal{N}\),

\[
\rho_i^{n+1}(k) = (1 - a^n)\rho_i^{n}(k) + a^n\bar{\rho}_i^{n}(k), \text{ for all } k \in \mathcal{K}.
\]

(2.26)

As mentioned above, the solutions \(\bar{\rho}_i^{n}\) to (2.25) are integer-valued at each iteration. However, for a general stepsize \(a^n \in (0, 1]\), the (relaxed) caching variables corresponding to the update in (2.17) may not be integer-valued at each iteration. In particular, this would be true if the stepsize follows a diminishing stepsize rule. Although one can explore rounding techniques and probabilistic caching techniques to obtain feasible integer-valued caching
We now show that for any control parameter $\phi$, the transport layer VIP count is updated according to

$$
A \left( \frac{X}{2} + \frac{D}{2} \right) = \begin{cases} 
A & \text{if } X = 0, \\
0 & \text{otherwise}
\end{cases}
$$

Similarly, in (26) Algorithm 3 adaptively stabilizes all VIP queues in the network.

Figure 2.2: Network topologies used for the simulations.
variables $x_i^a(k)$ from continuous-valued relaxed caching variables $\rho_i^a(k)$ [35], this would entail additional computational and communication complexity.

Since we are focused on distributed, low-complexity forwarding and caching algorithms, we require $\rho_i^a(k)$ to be either 0 or 1 at each iteration $n$. This is realized by choosing the stepsize $a^n = 1$ for all $n$. In this case, the update equation (2.17) is reduced to:

$$\Phi^{n+1} = \bar{\Phi}^n.$$  \hfill (2.27)

where $\bar{\Phi}^n$ is the solution to (2.21) and (2.25). That is, the solutions to the direction finding subproblems provide us with forwarding and caching decisions at each iteration. We now summarize the remarkably elegant MinDelay forwarding and caching algorithm.

**MinDelay Forwarding Algorithm**: At each iteration $n$, each node $i$ and for each object $k$, the forwarding algorithm chooses the outgoing link $(i, m)$ to forward requests for object $k$, where $m$ is chosen according to

$$m \in \arg \min_{j \in \mathcal{O}(i,k)} \delta^n_{ij}(k).$$  \hfill (2.28)

That is, requests for object $k$ are forwarded on an outgoing link with the minimum marginal forwarding cost.

**MinDelay Caching Algorithm**: At each iteration $n$, each node $i$ cal-
culates a cache score $CS^n(i,k)$ for each object $k$ according to

$$CS^n(i,k) \triangleq t^n_i(k)\delta^n_i(k). \quad (2.29)$$

where $\delta^n_i(k) \equiv \min_{j \in O(i,k)} \delta^n_{ij}(k)$. Upon reception of data object $k_{new}$ not currently in the cache of node $i$, if the cache is not full, then $k_{new}$ is cached. If the cache is full, then $CS^n(i,k_{new})$ is computed, and compared to the lowest cache score among the currently cached objects, denoted by $CS^n(i,k_{min})$. If $CS^n(i,k_{new}) > CS^n(i,k_{min})$, then replace $k_{min}$ with $k_{new}$. Otherwise, the cache contents stay the same.

The cache score given in (2.29) for a given content $k$ at node $i$ is the minimum marginal forwarding cost for object $k$ at $i$, multiplied by the total request rate for $k$ at $i$. By caching the data objects with the highest cache scores, each node maximally reduces the total cost of forwarding request traffic.

One drawback of using stepsize $a^n = 1$ in the MinDelay algorithm is that it makes studying the asymptotic behavior of the algorithm difficult. Nevertheless, in extensive simulations shown in the next section, we observe that the algorithm behaves in a stable manner asymptotically. Moreover, the MinDelay significantly outperform several state-of-the-art caching and forwarding algorithms in important operating regimes.
2.5 Simulation Experiments

In this section we present the results of extensive simulations performed using our own Java-based ICN Simulator. We have considered three competing schemes for comparison with MinDelay. First, we consider the VIP joint caching and forwarding algorithm introduced in [3]. This algorithm uses a backpressure (BP)-based scheme for forwarding and a stable caching algorithm, both based on VIP (Virtual Interest Packet) queue states [3]. In the VIP algorithm discussed in [3], multiple Interest Packets requesting the same Data Packet are aggregated. Since we do not consider Interest Packet aggregation in this paper, we compare MinDelay with a version of VIP without Interest aggregation, labeled BP. We consider the VIP algorithm (or BP) to be the direct competitor with MinDelay, since to the best of our knowledge, it is the only other scheme that explicitly jointly optimizes forwarding and caching for general ICN networks.

The other two approaches implemented here are based on the LFU cache eviction policy. We note that for stationary input request processes, the performance of LFU is typically much better than those of LRU and FIFO.\footnote{Initially we included LRU-based approaches. However, since their performance was much worse than the competitors, we omitted them in the final figures.} In the first approach, denoted by LFUM-PI, multipath request forwarding is based on the scheme proposed in [6]. Here, the forwarding decision is made as follows: an Interest Packet requesting a given object is forwarded on an outgoing interface with a probability inversely proportional to the number
of Pending Interest (PI) Packets for that object on that outgoing interface. The second LFU-based approach implemented here, denoted by LFUM-RTT, has a RTT-based forwarding strategy. In this strategy, described in [37], the multipath forwarding decision is based on the exponentially weighted moving average of the RTT of each outgoing interface per object name. An Interest Packet requesting an object is forwarded on an outgoing interface with a probability inversely proportional to the average RTT recorded for that object on that outgoing interface.

We tested the MinDelay forwarding and caching algorithm against the described approaches on several well-known topologies depicted in Fig. 2.2. In the following, we explain the simulation scenarios and results in detail.

2.5.1 Simulation Details

Each simulation generates requests for 1000 seconds and terminates when all the requested packets are fulfilled. During the simulation, a requesting node requests a content object by generating an Interest Packet containing the content name and a random nonce value, and then submits it to the local forwarder. Upon reception of an Interest Packet, the forwarder first checks if the requested content name contained in the Interest Packet is cached in its local storage. If there is a copy of the content object in the local storage, it generates a Data Packet containing the requested object, along with the content name and the nonce value, and puts the Data Packet in the queue of the interface on which the Interest Packet was received. If the local
cache does not have a copy of the requested object, the forwarder uses the FIB table to retrieve the available outgoing interfaces.\textsuperscript{4} Then, the forwarder selects an interface among the available interfaces based on the implemented forwarding strategy. In particular, for MinDelay, we update the marginal forwarding costs given in (2.22) at the beginning of each update interval (with a length between 2-5 seconds), and cache the results in a sorted array for future use. Hence, the forwarding decision given in (2.28) takes $O(1)$ operations.

After selecting the interface based on the considered forwarding strategy, the forwarder creates a Pending Interest Table (PIT) entry with the key being the content name concatenated with the nonce value, and the PIT entry value being the incoming interface ID. Note that we concatenate the nonce value to the content name since we do not assume Interest Packet suppression at the forwarder. Hence, we need to have distinguishable keys for each Interest Packet. Next, the forwarder assigns the Interest Packet to the queue of the selected interface, to be transmitted in a FIFO manner.

Upon reception of a Data Packet, the forwarder first checks if the local storage is full. If the storage is not full, it will cache the contained data object\textsuperscript{5} in local storage. If the storage is at capacity, it uses the considered cache eviction policy to decide whether to evict an old object and replace it.

\textsuperscript{4}In the simulations, we ensured that loop-free routing was done prior to the forwarding and caching experiments. The results of the routing algorithm are saved in FIB tables at each node.

\textsuperscript{5}In the experiments, all data objects contain one chunk, or one Data Packet.
with the new one. In the case of MinDelay, the forwarder regularly updates the cache score of the currently-cached contents using (2.29) at the beginning of the update intervals and keeps a sorted list of the cached content objects using a hash table and a priority queue. When a new Data Packet arrives, the forwarder computes its cache score, and compares the score with the lowest cache score among the currently-cached content objects. If the score of the incoming Data Packet is higher than the current lowest cache score, the forwarder replaces the corresponding cached object with the incoming one. Otherwise, the cached contents remain the same.

Finally, the forwarder proceeds by retrieving and removing the PIT entry corresponding to the Data Packet and assigning the Data Packet to the queue of the interface recorded in the PIT entry.

In all topologies, the number of content objects is 5000. Each requester requests a content object according to a Zipf distribution with power exponent $\alpha = 0.75$, by generating an Interest Packet each of size 1.25 KBytes. All content objects are assumed to have the same size and can be packaged into a single Data Packet of size 500 KBytes. The link capacity of all the topologies, except in Abilene topology illustrated in Fig. 2.2a, is 50 Mbps.

We first consider the Abilene topology [6] depicted in Figure 2.2a. There are three servers, at nodes 1, 5, and 8, each serving $1/3$ of the content objects. That is, object $k$ is served by server $k \mod 3 + 1$ for $k = 1, 2, \ldots, 5000$. The other eight nodes of the topology request objects according to Zipf distribution with $\alpha = 0.75$. Also, each requester has a content store of size 250
MBytes, or equivalently 500 content objects.

In the GEANT topology, illustrated in Figure 2.2b, there are 22 nodes in the network. All nodes request content objects. Each content object is randomly assigned to one of the 22 nodes as its source node. Each node has a content store of size 250 MBytes, or equivalently 500 content objects.

In the DTelekom topology, illustrated in Figure 2.2c, there are 68 nodes in the network. All nodes request content objects. Each content object is randomly assigned to one of the 68 nodes as its source node. Each node has a content store of size 250 MBytes, or equivalently 500 content objects.

In the Tree topology, depicted in Figure 2.2d, there are four requesting nodes at the leaves, C1, C2, C3 and C4. There are three edge nodes, E1, E2, and E3. Each content object is randomly assigned to one of the two source nodes, S1 and S2. Each requesting and edge node has a content store of size 125 MBytes, or equivalently 250 content objects.

In the Ladder topology [6], depicted in Figure 2.2e, there are three requesting nodes, A1, A2 and A3. The source of all the content objects are at node D3. Each node in the network, except node D3, has a content store of size 125 MBytes, or equivalently 250 content objects.

Finally, in the Fat Tree topology, depicted in Figure 2.2f, requesters are at the roots, i.e., nodes C1, C2, C3 and C4. There are 16 servers at the leaves. In this topology, each content object is randomly assigned to two servers, one chosen from the first 8 servers, and the other from the second 8 servers. All the requesting nodes as well as Aggregation and Edge nodes
have a content store, each of size 125 MBytes, or equivalently 250 content objects.

2.5.2 Simulation Results

In Figures 2.3 and 2.4, the results of the simulations are plotted. The figures illustrate the performance of the implemented schemes in terms of total network delay for satisfying all generated requests (in seconds) and the average cache hits in requests/node/second, versus the arrival rate in requests/node/second, respectively. We define the delay for a request as the difference between the creation time of the Interest Packet and the arrival time of its corresponding Data Packet at the requesting node. A cache hit for a data object is recorded when an Interest Packet reaches a node which is not a content source but which has the data object in its cache. When a cache hit occurs, the corresponding metric is incremented one.

To reduce randomness in our results, we ran each simulation 10 times, each with a different seed number, and plotted the average performance of each scheme in Figures 2.3 and 2.4.

Figure 2.3 shows the total network delay in seconds versus the per-node arrival rate in request/seconds, for the above-mentioned topologies. As can be seen, in all the considered topologies, MinDelay has lower delay in the low to moderate arrival rate regions. In the higher arrival rate regions, BP’s outperforms MinDelay in 3 of the tested topologies (Abilene, GEANT, and Tree),
Figure 2.3: Total network delay (sec) vs. request arrival rate (requests/node/sec).
Figure 2.4: Average total cache hits (requests/node/sec) vs. Arrival rate (requests/node/sec).
As shown in [3], the BP performs well in high arrival rate regions since the VIP algorithm adaptively maximizes the throughput of Interest Packets, thereby maximizing the stability region of user demand rates satisfied by the network. When the network is operating well within the stability region, however, MinDelay typically has superior performance. Thus, the MinDelay and VIP algorithms complement each other in delivering superior delay performance across the entire range of request arrival rates.

Finally, Figure 2.4 depicts the average total cache hits of the network (in requests/node/second) versus the per-node arrival rate (in request/seconds) for the Abilene, GEANT, Tree, and Ladder topologies, respectively. It can be seen that the cache hit performance of MinDelay is competitive but not necessarily superior to those of the other algorithms. This follows from the fact that MinDelay is designed with the objective of decreasing total network delay, and not explicitly with the objective of increasing cache hits.

**Conclusion**

In this work, we established a new unified framework for minimizing congestion-dependent network cost by jointly choosing node-based forwarding and caching variables. Relaxing integer constraints on caching variables, we used a version of the conditional gradient algorithm to develop MinDelay, an adaptive and distributed joint forwarding and caching algorithm for the original mixed-integer optimization problem. The MinDelay algorithm elegantly yields fea-
sible routing variables and integer caching variables at each iteration, and can be implemented in a distributed manner with low complexity and overhead.

Simulation results show that while the VIP algorithm performs well in high request arrival rate regions, MinDelay typically has significantly better delay performance in the low to moderate request rate regions. Thus, the MinDelay and VIP algorithms complement each other in delivering superior delay performance across the entire range of request arrival rates.

The elegant simplicity and superior performance of the MinDelay algorithm raise many interesting questions for future work. Specifically, we are interested in analytically characterizing the time-asymptotic behavior of MinDelay, as well as providing guarantees on the gap between the MinDelay performance and the theoretically optimal performance for the joint forwarding and caching problem.
Chapter 3

Minimum Cost Multicast with Network Coding and Storage in Multi-commodity Settings

3.1 Introduction

The ever-increasing demand for multimedia content has driven the growth of content delivery technology. In particular, live and on-demand video streaming have become greatly popular. How to efficiently handle this demand has become the focus of many research efforts. Among the proposed solutions, two different technologies, Content Distribution Network (CDN) and Peer-to-Peer (P2P) networking, have been widely implemented. However, both technologies have their own limitations [46]. The deployment and mainte-
nance of CDN servers are expensive, leading to a jump in operating cost for the content providers. On the other hand, P2P-based networks require a sufficient number of seeds on the supply side to jumpstart the distribution process, and often have security issues and low quality of service (QoS). These issues have led to a CDN-P2P hybrid architecture which integrates CDN- and P2P-based architectures (e.g. in [46], [47], [48], [49]). This hybrid architecture is shown to be highly cost-effective [48], scalable and with reliable QoS [50].

Adopting network coding within this context could improve manageability, security, and throughput. In [51], and [52], the use of network coding for P2P networks is explored. In [53], an uplink-sharing model for P2P networks is introduced and analyzed where each peer is in a fully connected graph and is constrained only by its uplink capacity. In [52], Chang et al., analyze download finish times in a P2P network by applying a linear programming approach based on network coding.

In [54], Tomozei et al., consider a rate control scheme for live streaming peer-to-peer systems with network coding. They propose a cost minimization problem subject to min-cut based feasibility constraints, and then develop an approximate distributed rate control scheme in which backlogged information is used as an approximation of the Lagrange multipliers for the cut constraints.

In [55], Mahini et al., propose a model for peer-assisted video streaming based on game theory and network coding. They model the interactions be-
between peers by a signaling game called Beer-Quiche and by analyzing the Nash equilibrium of the proposed game, they provide a reward and punishment mechanism which guarantees free-riders failure and participation motivation.

In [56], Lun et al., consider the problem of establishing minimum-cost multicast over coded packet networks. They show that the this problem can be decomposed into two parts: determining the minimum-cost coding subgraphs and designing the code applied over the optimal subgraphs. For the second part of the problem, a distributed algorithm was provided in [57], [58]. In [59], Lun et al. show that for a static multicast, the first part of the problem is reduced to a distributed polynomial-time solvable optimization problem. In [60], Zhao et al., propose a linear programming problem to establish the minimum-cost multicast, for which they show the convergence rate of the proposed distributed algorithm. In addition, Xi et al., [61] propose a distributed algorithm based on the flow model to solve the first part of the problem in a multi-commodity setting.

In this work, we consider the problem of finding minimum-cost multicast connections in a multi-commodity setting with the presence of in-network storage and network coding. We consider static multicasts in which the multicast groups remain constant for the duration of the connection. For each commodity, the source node multicasts coded packets to the set of requesting nodes (terminals). This model can be applied to study the effects and performance of edge caching and P2P networking separately and/or together.
Thus, P2P networks as well as peer-assisted CDN networks fit naturally into this model. We note that in this work, we do not model general caching networks, where each node can opportunistically cache packets and dynamically evict packets from the cache. In our model, each node can only send packets if it has already received the commodity, either because it is a requesting terminal itself and has requested the packets of the commodity beforehand, or alternatively, it is a server to which packets are being pushed proactively, such as CDN servers.

In the first part of the chapter, we use convex cost functions to formulate minimum-cost multicast for this general setting as a convex problem. We first consider inelastic traffic scenarios where commodities are being downloaded at a fixed rate. We introduce a set of primal-dual algorithms to solve the problem in a decentralized manner, and prove that they converge to a globally optimal solution. We then extend the results to the elastic traffic scenario and develop a congestion control mechanism to maximize the aggregate network utility.

In the second part of the chapter, we consider linear and separable cost functions for link and storage utilization, and cast minimum-cost multicast connections as a linear program. We introduce a dual subgradient method to solve the problem in a simple and distributed manner, and show the convergence rate for both the dual and recovered primal variables.

Through simulations, we further study the performance of the proposed algorithms and present the results on several topologies in the presence of
storage at the edge vs. at the peers.

Compared to prior work, our multi-commodity network and storage model is shown to be a natural extension of the single-commodity model introduced in [59]. Further, this model does not lead to a Mixed Integer Program due to binary storage variables. Thus, our storage decisions/algorithms are not approximations. In addition, we develop a set of distributed algorithms to find the minimum-cost multicast connections for every commodity. Since packets can be coded using random linear coding in a distributed manner [58], [57], this results in a fully decentralized approach for achieving minimum-cost multicast connections.

3.2 Network Model

Let the multi-hop network be modeled by a directed and (strongly) connected graph $G = (\mathcal{N}, \mathcal{A})$, where $\mathcal{N}$ and $\mathcal{A}$ are the node and link sets, respectively. A link $(i, j) \in \mathcal{A}$ corresponds to a lossless unidirectional link from node $i$ to node $j$. Suppose we have $M$ (coded) commodities, denoted by set $\mathcal{M}$. Each has a source node $s_m, m \in \mathcal{M}$ wishing to transmit packets at a positive rate $R_m$ to a nonempty set of terminals (sink nodes) $T_m$. Also, each node has a local storage to and from which it can store and serve coded packets.

We consider a flow model for the transmission and storage of the multicast sessions’ data in the network. Let $z_{ij}$ and $z_i$ be the (flow) rate at which coded packets are transmitted over link $(i, j)$ and the (flow) rate at which coded
packets are retrieved from the local storage of node $i$, respectively. In addition, $x_{ij}^{(t,m)}$ and $x_i^{(t,m)}$ correspond to the virtual flow rate on link $(i,j)$ and the virtual flow rate from the local storage of node $i$ which are relevant for terminal $t$, respectively. Hence, the rate vector $\mathbf{x}^{(t,m)} \triangleq (x_{ij}^{(t,m)}, x_i^{(t,m)})_{(i,j) \in \mathcal{A}, i \in \mathcal{N}}$ forms a coding subgraph [56], [59] for pair $(s_m, t)$. In addition, the flow rate of commodity $m$ and its virtual rates are related as: $z_{ij}^{(m)} = \max_{t \in \mathcal{T}_m} x_{ij}^{(t,m)}$ and $z_i^{(m)} = \max_{t \in \mathcal{T}_m} x_i^{(t,m)}$. Moreover, the total flow rate on a link $(i,j) \in \mathcal{A}$ or from a storage interface $i \in \mathcal{N}$ are, respectively, $z_{ij} = \sum_{m \in \mathcal{M}} z_{ij}^{(m)}$, and $z_i = \sum_{m \in \mathcal{M}} z_i^{(m)}$.

We assume the network cost is the sum of costs over all the links and local storage units. Let $f_{ij}$ and $f_i$ be strictly convex, monotonically increasing cost functions for link $(i,j) \in \mathcal{A}$ and node $i \in \mathcal{N}$, respectively. Such link cost functions reflect latency, congestion or contract-specified costs corresponding to link utilization. In the case of storage cost functions, $f_i$ can be a simple barrier function to enforce the storage or CPU processing capacity. The
optimization problem can now be formulated as

\[
\begin{align*}
\min & \sum_{(i,j) \in A} f_{ij}(z_{ij}) + \sum_{i \in N} f_i(z_i) \\
\text{subject to:} & \\
& z_{ij} = \sum_{m \in M} z_{ij}^{(m)}, \forall (i, j) \in A \\
& z_i = \sum_{m \in M} z_i^{(m)}, \forall i \in N \\
& z_{ij}^{(m)} = \left( \sum_{t \in T_m} \left( x_{ij}^{(t,m)} \right)^n \right)^{\frac{1}{n}}, \forall (i, j) \in A, m \in M \\
& z_i^{(m)} = \left( \sum_{t \in T_m} \left( x_i^{(t,m)} \right)^n \right)^{\frac{1}{n}}, \forall i \in N, m \in M \\
& \sum_{\{j| (j,i) \in A\}} x_{ij}^{(t,m)} - \sum_{\{j| (j,i) \in A\}} x_{ji}^{(t,m)} - x_i^{(t,m)} = \delta_i^{(t,m)}, \forall i \in N, m \in M, t \in T_m \\
& x_{ij}^{(t,m)} \geq 0, x_i^{(t,m)} \geq 0, \forall (i, j) \in A, i \in N, m \in M, t \in T_m
\end{align*}
\]

(3.1)

where

\[
\delta_i^{(t,m)} = \begin{cases} 
\max\{0, R_m - \sum_{j \in N} x_j^{(t,m)}\}, & \text{if } i = s_m, \\
-R_m, & \text{if } i = t, \\
0, & \text{otherwise.}
\end{cases}
\]

(3.2)

For all \( m \in M, t \in T_m \), we set \( x_{ij}^{(t,m)} = x_i^{(t,m)} = x_{ij}^{(t,m)} = 0 \), for all \( j \) such that \( (t, j) \in A \), and \( x_{is_m}^{(t,m)} = 0 \), for all \( i \) such that \( (i, s_m) \in A \). We note that \( z_{ij}^{(m)} \) and \( z_i^{(m)} \) defined in problem (3.1) are relaxed versions of \( z_{ij}^{(m)} = \max_{t \in T_m} x_{ij}^{(t,m)} \) and \( z_i^{(m)} = \max_{t \in T_m} x_i^{(t,m)} \), respectively. However, a code with
rate $z_{ij}^{(m)}$ on each link $(i, j)$ and $z_i^{(m)}$ for each node $i$ exists for any feasible solution $z_{ij}^{(m)}$ and $z_i^{(m)}$, respectively, since $z_{ij}^{(m)} \geq z_{ij}^{(n(m))}$ and $z_i^{(m)} \geq z_i^{(n(m))}$ for all $n > 0$. Also, $z_{ij}^{(m)}$ and $z_i^{(m)}$ approach $z_{ij}^{(n(m))}$ and $z_i^{(n(m))}$, respectively, as $n \to \infty$. Hence, we assume that $n$ is large in (3.1).

In addition, note that $\sum_{i \in N} x_i^{(t,m)}$ can be referred to as the “storage gain” for commodity $m$ and terminal $t$, i.e., the amount of traffic that is being downloaded from in-network storage.

To simplify the problem (3.1), we present a new model in which the local storage at node $i$ is represented by a virtual node $i'$, and the local storage unit interface of node $i$ is represented by arc $(i', i)$. As shown in Figure 3.1a, $i'$ is assumed to be connected to source nodes $s_m$ for $m \in \mathcal{M}$, via virtual links $(s_m, i')$. In addition, Figure 3.1b shows the effects of in-network storage on the source node. In this figure, source node $s_m$ transmits packets with rate $R_m - \sum_{j \in N} x_{s_m j'}^{(t,m)}$, relevant for terminal $t$, while each terminal receives packets with rate $R_m$.

The virtual node $i'$ on the link $(i', i)$ is assumed to have only forwarding functionalities (placing an incoming packet onto an outgoing link), rather than replicating (copying an incoming packet onto several outgoing links), or en/de-coding. These latter functionalities are assumed by node $i$. Also, no costs are associated with virtual links $\{(s_m, i')\}$. We define the extended graph $\mathcal{G}' = (\mathcal{N}', \mathcal{A}')$ where $\mathcal{N}' = \mathcal{N} \cup \{i'\}_{i \in \mathcal{N}}$, $\mathcal{A}' = \mathcal{A}^{+} \cup \{(s_m, i')\}_{m \in \mathcal{M}, i \in \mathcal{N}}$, and $\mathcal{A}^{+} = \mathcal{A} \cup \{(i', i)\}_{i \in \mathcal{N}}$.

For simplicity, we define $z_{i'i}^{(m)} \equiv z_i^{(m)}$, $z_{i'i}^{(m)} \equiv z_i$, $x_i^{(t,m)} \equiv x_i^{(t,m)}$ and $f_{i'i} \equiv f_i$.
In addition, since \( f_{s_m, i'}(.) = 0 \) for all \( m, i \) and \( x_{s_m, i'}^{(t, m)} = x_{i', i}^{(t, m)} \), we may eliminate the \( x_{s_m, i'}^{(t, m)} \)'s and obtain the equivalent optimization problem:

\[
\begin{align*}
\min & \sum_{(i,j) \in A^+} f_{ij}(z_{ij}) \\
\text{subject to:} & \\
& z_{ij} = \sum_{m \in M} z_{ij}^{(m)}, \forall (i, j) \in A^+ \\
& z_{ij}^{(m)} = \left( \sum_{t \in T_m} \left( x_{ij}^{(t, m)} \right)^n \right)^{\frac{1}{n}}, \forall (i, j) \in A^+, m \in M \\
& \sum_{\{j | (i, j) \in A^+\}} x_{ij}^{(t, m)} - \sum_{\{j | (j, i) \in A^+\}} x_{ji}^{(t, m)} = \sigma_i^{(t, m)}, \forall i \in \mathcal{N}, m \in \mathcal{M}, t \in T_m \\
& x_{ij}^{(t, m)} \geq 0, \forall (i, j) \in A^+, m \in \mathcal{M}, t \in T_m
\end{align*}
\]  

(3.3)

where

\[
\sigma_i^{(t, m)} = \begin{cases} 
R_m, & \text{if } i = s_m, \\
-R_m, & \text{if } i = t, \\
0, & \text{otherwise.}
\end{cases}
\]  

(3.4)

In problem (3.3), the flow conservation constraint for node \( s_m \) (as shown in Figure 3.1b) takes the form

\[
\sum_{\{j | (s_m, j) \in A^+\}} x_{s_m, j}^{(t, m)} = \sum_{\{j | (s_m, j) \in A\}} x_{s_m, j}^{(t, m)} + \sum_{j \in \mathcal{N}} x_{j, j}^{(t, m)} = R_m.
\]  

(3.5)

Optimization problem (3.3) is a convex multi-commodity flow problem.
(a) Local storage interface at node \( i \).

(b) A simple view of the source side of the extended graph. Dotted links are the virtual links to each node (its local storage), which represent the local contribution of a node for a given commodity \( m \) relevant for terminal \( t \). The solid arc shows the aggregate traffic downloaded from \( s_m \) via all actual links.

Figure 3.1: Proposed model to incorporate storage.
As in [59, 62], we use a primal-dual algorithm for its solution. Since \( f_{ij} \)'s are strictly convex, and \( z_{ij} \) is a strictly convex function of \( x_{ij}^{(t,m)} \), the objective function of problem (3.3) is strictly convex. Let

\[
U(x) \triangleq - \sum_{(i,j) \in A^+} f_{ij} \left( \sum_{m \in M} \left[ \sum_{t \in T_m} \left( x_{ij}^{(t,m)} \right)^n \right]^\frac{1}{n} \right),
\]

which is strictly concave in \( x_{ij}^{(t,m)} \). Then, the Lagrangian function corresponding to (3.3) follows

\[
L(x, \lambda, p) = U(x) + \sum_{(i,j) \in A^+} x_{ij}^{(t,m)} \lambda_{ij}^{(t,m)} - \sum_{i \in N} \sum_{t \in T_m} m \in M \left[ y_i^{(t,m)} - \sigma_i^{(t,m)} \right]
\]

where \( y_i^{(t,m)} \triangleq \sum_{\{j | (i,j) \in A^+ \}} x_{ij}^{(t,m)} - \sum_{\{j | (j,i) \in A^+ \}} x_{ji}^{(t,m)} \).

Let \((\hat{x}, \hat{\lambda}, \hat{p})\) be a solution for problem (3.3), then the following Karush-Kuhn-Tucker conditions can be verified to hold:

\[
\frac{\partial L(\hat{x}, \hat{\lambda}, \hat{p})}{\partial x_{ij}^{(t,m)}} = \frac{\partial U(\hat{x})}{\partial x_{ij}^{(t,m)}} - (\hat{p}_i^{(t,m)} - \hat{p}_j^{(t,m)}) + \hat{\lambda}_{ij}^{(t,m)} = 0, \forall (i, j) \in A^+, m \in M, t \in T_m,
\]

\[
\sum_{\{j | (i,j) \in A^+ \}} \hat{x}_{ij}^{(t,m)} - \sum_{\{j | (j,i) \in A^+ \}} \hat{x}_{ji}^{(t,m)} = \sigma_i^{(t,m)},
\]

\[
\forall i \in N, m \in M, t \in T_m,
\]

\[
\hat{x}_{ij}^{(t,m)} \geq 0, \forall (i, j) \in A^+, m \in M, t \in T_m,
\]

\[
\hat{\lambda}_{ij}^{(t,m)} \geq 0, \forall (i, j) \in A^+, m \in M, t \in T_m,
\]
\[
\dot{x}_{ij}^{(t,m)} \lambda_{ij}^{(t,m)} = 0, \forall (i,j) \in A^+, m \in M, t \in T_m. \tag{3.12}
\]

We now specify a continuous-time primal-dual algorithm for solving (3.3):

\[
\begin{align*}
\dot{x}_{ij}^{(t,m)} &= k_{ij}^{(t,m)}(x_{ij}^{(t,m)}) \left( \frac{\partial U(x)}{\partial x_{ij}^{(t,m)}} - q_{ij}^{(t,m)} + \lambda_{ij}^{(t,m)} \right), \tag{3.13} \\
p_i^{(t,m)} &= h_i^{(t,m)}(p_i^{(t,m)}) \left( y_i^{(t,m)} - \sigma_i^{(t,m)} \right), \tag{3.14} \\
\dot{\lambda}_{ij}^{(t,m)} &= m_{ij}^{(t,m)}(\lambda_{ij}^{(t,m)}) \left( -x_{ij}^{(t,m)} \right)^+, \tag{3.15}
\end{align*}
\]

where \( k_{ij}^{(t,m)}(x_{ij}^{(t,m)}), h_i^{(t,m)}(p_i^{(t,m)}) \) and \( m_{ij}^{(t,m)}(\lambda_{ij}^{(t,m)}) \) are positive, non-decreasing continuous functions of \( x_{ij}^{(t,m)}, p_i^{(t,m)} \) and \( \lambda_{ij}^{(t,m)} \), respectively, and

\[
q_{ij}^{(t,m)} \triangleq p_i^{(t,m)} - p_j^{(t,m)}, \tag{3.16}
\]

\[
y_i^{(t,m)} \triangleq \sum_{\{j|(i,j) \in A^+\}} x_{ij}^{(t,m)} - \sum_{\{j|(j,i) \in A^+\}} x_{ji}^{(t,m)}, \tag{3.17}
\]

\[
(y)^+_x \triangleq \begin{cases} y, & \text{if } x > 0, \\ \max\{y, 0\}, & \text{if } x \leq 0. \end{cases} \tag{3.18}
\]

The next proposition shows that the primal-dual algorithm converges to a globally optimal solution of problem (3.3), for any initial choice of \((x, p)\), provided that we initialize \( \lambda \) with non-negative entries: \( \lambda(0) \geq 0 \).

**Proposition 1.** The primal-dual algorithm described by eq.s (3.13)-(3.15) converges to a globally optimal solution of problem (3.3), for any initial choice
of \((x, p)\) and non-negative choice of \(\lambda(0)\).

**Proof.** To prove the convergence of primal-dual algorithm to a globally optimal solution of problem (3.3), we use Lyapunov stability theory, and show that the proposed algorithm is globally, asymptotically stable (see 3.10 in [62]). This proof is based on the proof of Theorem 3.7 of [62], and Proposition 1 of [59].

Following the equilibrium point \((\hat{x}, \hat{\lambda}, \hat{p})\) satisfying KKT conditions in (3.8)-(3.12), we consider the following function as a candidate for the Lyapunov function.

\[
V(x, \lambda, p) = \sum_{t \in T} \sum_{m \in M} \left\{ \sum_{(i,j) \in A^+} \left( \int_{\hat{x}_{ij}}^{x_{ij}} (\frac{1}{K_{ij}^{(t,m)}(\delta)} (\delta - \hat{x}_{ij}^{(t,m)}) d\delta + \int_{\lambda_{ij}}^{\lambda_{ij}} (\gamma - \hat{\lambda}_{ij}^{(t,m)}) d\gamma \right) + \sum_{i \in N} \int_{\hat{p}_i}^{p_i} (\frac{1}{K_i^{(t,m)}(\beta)} (\beta - \hat{p}_i^{(t,m)}) d\beta \right) \right\}.
\]

Note that \(V(\hat{x}, \hat{\lambda}, \hat{p}) = 0\). Since \(k_{ij}^{(t,m)}(\delta) > 0\), if \(x_{ij}^{(t,m)} \neq \hat{x}_{ij}^{(t,m)}\), we have

\[
\int_{\hat{x}_{ij}}^{x_{ij}} (\frac{1}{k_{ij}^{(t,m)}(\delta)} (\delta - \hat{x}_{ij}^{(t,m)}) d\delta > 0.
\]

Similarly, we can extend this argument to the other terms in \(V\), hence, we have \(V(x, p, \lambda) > 0\) whenever \((x, p, \lambda) \neq (\hat{x}, \hat{\lambda}, \hat{p})\).
Then, \( \dot{V} \) follows

\[
\dot{V} = \sum_{t \in T} \left\{ \sum_{m \in M} \left[ \sum_{(i,j) \in A^+} \left( \frac{\partial U(x)}{\partial x_{ij}^{(t,m)}} - d_{ij}^{(t,m)} + \lambda_{ij}^{(t,m)} \right) \cdot (x_{ij}^{(t,m)} - \hat{x}_{ij}^{(t,m)}) + 
\right.ight.

\[
\left. (-x_{ij}^{(t,m)})^+ (\lambda_{ij}^{(t,m)} - \hat{\lambda}_{ij}^{(t,m)}) \right) + \sum_{i \in N} \left( y_i^{(t,m)} - \sigma_i^{(t,m)} \right) (p_i^{(t,m)} - \hat{p}_i^{(t,m)}) \right\}.
\]

Let us first prove the following.

\[
(-x_{ij}^{(t,m)})^+ (\lambda_{ij}^{(t,m)} - \hat{\lambda}_{ij}^{(t,m)}) \leq -x_{ij}^{(t,m)} (\lambda_{ij}^{(t,m)} - \hat{\lambda}_{ij}^{(t,m)}).
\]

The above inequality is an equality if either \( x_{ij}^{(t,m)} \leq 0 \) or \( \lambda_{ij}^{(t,m)} > 0 \). On the other hand, when \( x_{ij}^{(t,m)} > 0 \) and \( \lambda_{ij}^{(t,m)} \leq 0 \), we have \( (-x_{ij}^{(t,m)})^+ = 0 \), and since \( \hat{\lambda}_{ij}^{(t,m)} \geq 0 \), it follows \( (-x_{ij}^{(t,m)}) (\lambda_{ij}^{(t,m)} - \hat{\lambda}_{ij}^{(t,m)}) \geq 0 \). Hence, (3.19) holds, and thereby, we get inequality (a) below.

Equation (b) is obtained by re-arranging the terms. By applying KKT conditions (3.8)-(3.12) and noting that

\[
p' y = \sum_{t \in T, m \in M} \sum_{i \in N} p_i^{(t,m)} \left( \sum_{\{j \mid (i,j) \in A^+\}} x_{ij}^{(t,m)} - \sum_{\{j \mid (j,i) \in A^+\}} x_{ji}^{(t,m)} \right) = \sum_{t \in T} \sum_{m \in M} x_{ij}^{(t,m)} (p_i^{(t,m)} - p_j^{(t,m)}) = q' x,
\]

equation (c) follows. Since \( U(x) \) is strictly concave in \( x \),

\[
(\nabla U(x) - \nabla U(\hat{x}))'(x - \hat{x}) \leq 0,
\]
and hence $\dot{V} \leq -\lambda \dot{x}$ with equality if and only if $x = \hat{x}$.

$$
\dot{V} \leq \sum_{t \in T_w} \sum_{m \in M} \left\{ \sum_{(i,j) \in A^+} \left( \frac{\partial U(x)}{\partial x_{ij}^{(t,m)}} - q_{ij}^{(t,m)} + \lambda_{ij}^{(t,m)} \right) \cdot \left( x_{ij}^{(t,m)} - \hat{x}_{ij}^{(t,m)} \right) + (-x_{ij}^{(t,m)})(\lambda_{ij}^{(t,m)} - \hat{\lambda}_{ij}^{(t,m)}) \right\} + \sum_{i \in N} \left( y_i^{(t,m)} - \sigma_i^{(t,m)} \right) \left( p_i^{(t,m)} - \hat{p}_i^{(t,m)} \right) \\
\overset{(b)}{=} (\hat{q} - q)'(x - \dot{x}) - (\hat{p} - p)'(y - \dot{y}) + \sum_{t \in T_w} \sum_{m \in M} \left\{ \sum_{(i,j) \in A^+} \left( \frac{\partial U(x)}{\partial x_{ij}^{(t,m)}} - q_{ij}^{(t,m)} + \hat{\lambda}_{ij}^{(t,m)} \right) \cdot \left( x_{ij}^{(t,m)} - \hat{x}_{ij}^{(t,m)} \right) + (-\hat{x}_{ij}^{(t,m)})(\lambda_{ij}^{(t,m)} - \hat{\lambda}_{ij}^{(t,m)}) \right\} + \sum_{i \in N} \left( \hat{y}_i^{(t,m)} - \sigma_i^{(t,m)} \right) \left( p_i^{(t,m)} - \hat{p}_i^{(t,m)} \right) \\
\overset{(c)}{=} (\nabla U(x) - \nabla U(\hat{x}))(x - \dot{x}) - \lambda' \dot{x} \leq -\lambda' \dot{x}.
$$

Note that, if the initial choice of $\lambda$ is non-negative, that is $\lambda(0) \geq 0$, it can be verified from the primal-dual algorithm, in particular eq. (3.15), that $\lambda(\tau) \geq 0$ where $\tau = 0, 1, 2, \ldots$ represents the algorithm iteration. Thus, we have $\dot{V} \leq -\lambda' \dot{x}$. Assuming $\lambda \geq 0$, it follows that $\dot{V} \leq 0$ since $\dot{x} \geq 0$. Therefore, the primal-dual algorithm is globally, asymptotically stable, and hence, it converges to a globally optimal solution of problem (3.3).

\[\square\]

The proposed primal-dual algorithm finds the minimum-cost integrated multicast and storage strategy. The algorithms in (3.13)-(3.15) can be re-
written for the storage strategy and a given node $i \neq s_m$:

$$\dot{x}_{i}^{(t,m)} = k_{i}^{(t,m)} \left( x_{i}^{(t,m)} \left( \frac{\partial U(x)}{\partial x_{i}^{(t,m)}} - q_{i}^{(t,m)} + \lambda_{i}^{(t,m)} \right) \right), \quad (3.21)$$

$$\dot{p}_{i}^{(t,m)} = h_{i}^{(t,m)} \left( p_{i}^{(t,m)} \left( y_{1i}^{(t,m)} - \sigma_{i}^{(t,m)} - x_{i}^{(t,m)} \right) \right), \quad (3.22)$$

$$\dot{p}_{s_m}^{(t,m)} = h_{s_m}^{(t,m)} \left( p_{s_m}^{(t,m)} \left( y_{1s_m}^{(t,m)} - R_{s_m} \right) \right), \quad (3.23)$$

$$\dot{\lambda}_{i}^{(t,m)} = m_{i}^{(t,m)} \left( \lambda_{i}^{(t,m)} \left( -x_{i}^{(t,m)} \right) \right) + \lambda_{i}^{(t,m)}, \quad (3.24)$$

where $q_{i}^{(t,m)} = p_{s_m}^{(t,m)} - p_{i}^{(t,m)}$, and

$$y_{1i}^{(t,m)} \triangleq \sum_{\{j| (i,j) \in \mathcal{A}\}} x_{ij}^{(t,m)} - \sum_{\{j| (j,i) \in \mathcal{A}\}} x_{ji}^{(t,m)},$$

is the net flow on the original graph $\mathcal{G}$. Also,

$$R_{s_m} \triangleq R_{m} - \sum_{i \in \mathcal{N} \setminus s_m} x_{i}^{(t,m)},$$

is the rate at which commodity $m$ is being downloaded directly from the source node.

To implement this algorithm, each node $i$ needs to keep track of variables $\{p_{i}^{(t,m)}\}_{m \in \mathcal{M}, t \in \mathcal{T}_{m}}$ and $\{\lambda_{i}^{(t,m)}\}_{m \in \mathcal{M}, t \in \mathcal{T}_{m}}$, as well as $\{x_{ij}^{(t,m)}\}_{m \in \mathcal{M}, t \in \mathcal{T}_{m}}$ and $\{\lambda_{ij}^{(t,m)}\}_{m \in \mathcal{M}, t \in \mathcal{T}_{m}}$ for every outgoing arc $(i, j)$.

As shown in (3.21), $x_{i}^{(t,m)}$ is a function of $p_{s_m}^{(t,m)}$, which represents a cost at the source node $s_m$. This dependency requires the source node to dissem-
inate its cost to the storage-enabled nodes in the network regularly, in order for these nodes to calculate the storage variables \( x^{(t,m)}_i \)'s. For instance, the \( p^{(t,m)}_k \)'s can be piggybacked on the coded packets being directly downloaded from the source node. In addition, node \( i \) needs to exchange the value of \( \{p^{(t,m)}_i\}_{m \in \mathcal{M}, t \in \mathcal{T}_m} \) with its neighbors regularly.

### 3.3 Congestion Control for Elastic Rate Demand

So far we have focused on the case of an inelastic traffic rate demand and provided the algorithms for finding the minimum-cost subgraph and storage strategy. We now extend these results to the case of elastic rate demand and incorporate a congestion control mechanism into the proposed framework. Assume that the rate demand is elastic, represented by a strictly concave utility function given in the same units as the cost functions \( f_{ij} \)'s. We seek to maximize the aggregate utility minus the network cost, denoted by \( U(x, R) \).

\[
U(x, R) \triangleq W \cdot \sum_{m \in \mathcal{M}} U_m(R_m) + U(x),
\]

where \( W \) is a positive constant controlling the tradeoff between network cost and the aggregate utility, \( U_m(R_m) \) is the utility derived by the source when commodity \( m \) is being downloaded.
at rate $R_m$. Then the minimum-cost problem is formulated as:

\[
\begin{align*}
\max U(x, R) \\
\text{subject to:} \\
\sum_{\{j | (i, j) \in A^+\}} x_{ij}^{(t,m)} - \sum_{\{j | (j, i) \in A^+\}} x_{ji}^{(t,m)} = \sigma_i^{(t,m)}, \ \forall i \in \mathcal{N}, m \in \mathcal{M}, t \in T_m \\
R_m \geq 0, \forall m \in \mathcal{M} \\
x_{ij}^{(t,m)} \geq 0, \forall (i, j) \in \mathcal{A}^+, m \in \mathcal{M}, t \in T_m
\end{align*}
\]

(3.25)

The solution of problem (3.25) will include the optimal admitted rates as well as the flow rates. This natural extension of the framework allows us to apply a similar primal-dual algorithm.

\[
x_{ij}^{(t,m)} = k_{ij}^{(t,m)} (x_{ij}^{(t,m)}) \left( \frac{\partial U(x, R)}{\partial x_{ij}^{(t,m)}} - q_{ij}^{(t,m)} + \lambda_{ij}^{(t,m)} \right).
\]

(3.26)

\[
\dot{R}_m = k_{Rm}(R_m) \left( \frac{\partial U(x, R)}{\partial R_m} - q_{Rm} + \lambda_{Rm} \right).
\]

(3.27)

\[
\dot{p}_i^{(t,m)} = \dot{h}_i^{(t,m)} (p_i^{(t,m)}) \left( y_i^{(t,m)} - \sigma_i^{(t,m)} \right).
\]

(3.28)

\[
\dot{\lambda}_{ij}^{(t,m)} = m_{ij}^{(t,m)} (\lambda_{ij}^{(t,m)}) \left( -x_{ij}^{(t,m)} \right)^+ \lambda_{ij}^{(t,m)}.
\]

(3.29)

\[
\dot{\lambda}_{Rm} = m_{Rm}(\lambda_{Rm}) (-R_m)^+ \lambda_{Rm}.
\]

(3.30)

where $\lambda_{Rm}$ is the Lagrange multiplier for the non-negativity constraint of $R_m$, $k_{Rm}(R_m)$, and $m_{Rm}(\lambda_{Rm})$ are positive, non-decreasing continuous functions.
of $R_m$ and $\lambda_{R_m}$, respectively. Furthermore, $q_{R_m} = -\sum_{t \in T_m} p_{k_m}^{(t,m)}$.

It is easy to verify that the Prop. 1 can be extended for the algorithm given in (3.26)-(3.30), as follows:

**Proposition 2.** The primal-dual algorithm described by eq.s (3.26)-(3.30) converges to a globally optimal solution of problem (3.25), for any initial choice of $(x, R, p)$, and non-negative choice of $\lambda(0)$ and $\lambda_R(0)$.

It is worth noting that, in order for terminal $t$ to compute (3.28), $s_m$ needs to disseminate the value of $R_m$ regularly to $t$. In addition, we can derive the algorithm for storage variables similar to (3.21)-(3.24) where the source node is required to disseminate its cost $p_{k_m}^{(t,m)}$ to the storage-enabled nodes in the network regularly, in order for these nodes to calculate the storage variables $x_{i,i'}^{(t,m)}$.

### 3.4 Linear Cost Functions

In the previous section, we developed a distributed primal-dual algorithm for convex cost functions and proved its global convergence. In this section, we focus on the case of linear and separable cost functions, for which we propose a simpler algorithm based on the subgradient method, and provide an upper bound on its convergence rate in both the primal and dual spaces. For convenience, we formulate the problem in this section in discrete-time format.
The case of linear, separable cost functions with separable constraints arises when a fixed cost (monetary, energy, or a multi-factor weight cost) is paid per unit rate allocated on an arc, and each arc is subject to a separate capacity constraint with a non-negative constant value [59]. Let \( a_{ij} \) and \( c_{ij} \) be the cost per unit rate and the capacity of the link, respectively. As in the previous section, the local storage interface at node \( i \) is represented by \((i', i)\) arc and is included in the extended graph \( G' = (N', A') \) where \( N' = N \cup \{i'\}_{i \in N}, A' = A^+ \cup \{(s_m, i')\}_{m \in M, i \in N} \) and \( A^+ = A \cup \{(i', i)\}_{i \in N} \). As before, we do not associate any costs with the links \((s_m, i')\), while \( a_{i'i} \) and \( c_{i'i} \) are, respectively, the utilization cost per unit rate and the capacity of the storage at node \( i \). The optimization problem can be formulated as:

\[
\begin{aligned}
\min f(z) = & \sum_{(i,j) \in A^+} \sum_{m \in M} a_{ij} z_{ij}^{(m)} \\
\text{subject to:} & \quad c_{ij} \geq \sum_{m \in M} z_{ij}^{(m)}, \forall (i, j) \in A^+ \\
& \quad z_{ij}^{(m)} \geq x_{ij}^{(t,m)}, \forall (i, j) \in A^+, m \in M, t \in T_m \\
& \quad \sum_{\{j|(i, j) \in A^+\}} x_{ij}^{(t,m)} - \sum_{\{j|(j, i) \in A^+\}} x_{ji}^{(t,m)} = \sigma_{i}^{(t,m)}, \forall i \in N, m \in M, t \in T_m \\
& \quad x_{ij}^{(t,m)} \geq 0, \forall (i, j) \in A^+, m \in M, t \in T_m
\end{aligned}
\]

(3.31)

where as in the convex problem, we set \( x_{i't}^{(t,m)} = x_{s_m s_m}^{(t,m)} = x_{il}^{(t,m)} = 0 \), for all \( j \) such that \((t, j) \in A\) and for all \( m \in M, t \in T_m \). Also, \( x_{is_m}^{(t,m)} = 0, i \in N \) such
that $(i, s_m) \in \mathcal{A}$.

We consider solving (3.31) in a decentralized manner by developing a distributed subgradient method on the Lagrangian dual problem. Let $p_{ij}^{(t,m)}$, be the dual variables associated with $z_{ij}^{(m)} \geq x_{ij}^{(t,m)}$ constraints. Then, the dual function and the dual problem are

$$q(p) = \sum_{m \in \mathcal{M}, t \in T_m} q^{(t,m)}(p^{(t,m)}) + \sum_{(i,j) \in \mathcal{A}^+} r_{ij}(p_{ij}),$$

\[(3.32)\]

$$\begin{align*}
\max q(p) \\
\text{subject to:} \\
\quad p_{ij}^{(t,m)} \geq 0, \quad \forall (i, j) \in \mathcal{A}^+, m \in \mathcal{M}, t \in T_m 
\end{align*} \quad (3.33)$$

where

$$q^{(t,m)}(p^{(t,m)}) = \min_{x^{(t,m)} \in F_{x}^{(t,m)}} \sum_{(i,j) \in \mathcal{A}^+} p_{ij}^{(t,m)} x_{ij}^{(t,m)},$$

\[(3.34)\]

and $F_{x}^{(t,m)}$ is the bounded polyhedron of points $x^{(t,m)}$ satisfying

$$\sum_{\{j|(i,j) \in \mathcal{A}^+\}} x_{ij}^{(t,m)} - \sum_{\{j|(j,i) \in \mathcal{A}^+\}} x_{ji}^{(t,m)} = \sigma_{i}^{(t,m)}, \quad \forall i \in \mathcal{N},$$

\[(3.35)\]

Also,

$$r_{ij}(p_{ij}) = \min_{z_{ij} \in F_{z_{ij}}^{(t,m)}} \sum_{m \in \mathcal{M}} \left( a_{ij} - \sum_{t \in T_m} p_{ij}^{(t,m)} \right) z_{ij}^{(m)},$$

\[(3.36)\]
and $F_{zij}$ is the bounded region of points $z_{ij}$ given by

$$z_{ij}^{(m)} \geq 0, \forall m \in \mathcal{M}, \sum_{m \in \mathcal{M}} z_{ij}^{(m)} \leq c_{ij}. \quad (3.37)$$

Subproblem (3.34) is a shortest path problem for a given commodity $m$ and terminal $t$ with arc costs $p_{ij}^{(t,m)}$. The only difference between this subproblem and a standard shortest path algorithm is that each storage-enabled node needs to include its local storage interface among its upstream neighbors. Thus, the subproblem can be solved with a number of distributed routing algorithms, e.g., Bellman-Ford. Furthermore, subproblem (3.36) is a localized linear program (LP) problem per $(i,j) \in \mathcal{A}^+$ which can be solved easily as explained in the following. The proposed algorithm is a generalized version of Algorithm B in [60], where a single commodity problem has been analyzed. Unlike Algorithm B in [60], however, we do not use the incremental subgradient method, but rather the standard subgradient method. One benefit of this choice, as explained in Section 3.5.1, is that the size of the convergence neighborhood of the optimal solution for the dual problem will be reduced by a factor of $T_m$, for $m \in \mathcal{M}$.

To solve the dual problem (3.33), we employ subgradient method. The algorithm is initialized with $p[0] \geq 0$. In the $n$th iteration, we compute $x[n]$ by running a distributed shortest path algorithm, using $p[n]$ as arc costs. To
compute $z[n]$, let $b_{ij}^{(m)}[n] = a_{ij} - \sum_{t \in T_m} p_{ij}^{(t,m)}[n]$, for all $m \in M$, and let

$$m^* \in \arg\min_{m \in M} b_{ij}^{(m)}[n].$$

(3.38)

Then, a solution to (3.36) for a fixed arc $(i, j)$ is given by

$$z_{ij}^{(m)}[n] = \begin{cases} c_{ij}, & \text{if } b_{ij}^{(m)}[n] \leq 0 \text{ and } m = m^*, \\ 0, & \text{otherwise.} \end{cases}$$

(3.39)

We then update $p[n + 1]$ using subgradient $g[n]$:

$$g_{ij}^{(t,m)}[n] = x_{ij}^{(t,m)}[n] - z_{ij}^{(m)}[n],$$

(3.40)

$$p_{ij}^{(t,m)}[n + 1] = \max \left( 0, p_{ij}^{(t,m)}[n] + \theta[n] g_{ij}^{(t,m)}[n] \right),$$

(3.41)

where $\theta[n]$ is the positive stepsize at the $n$th iteration. As explained in [63], while the subgradient method constructs a (near) optimal solution to the Lagrangian dual problem (3.33), it may not necessarily yield a primal optimal solution. To recover an optimal primal solution in the subgradient method, we can apply the technique proposed in [63], which was employed in [60] for a similar problem.

For brevity, we omit the description of the general schemes which recover an optimal primal solution, and choose a well-known example, where at the
end of iteration $n$, each node recovers a primal solution, $\tilde{x}[n]$, according to

$$\tilde{x}[n] = \frac{x[n]}{n} + \frac{n-1}{n} \tilde{x}[n-1],$$  \hspace{1cm} (3.42)

and hence, node $i$ can compute $(\tilde{x}, \tilde{z})$ iteratively using the above relation without recording all the previous $(x, z)$'s.

To implement this algorithm, each node needs to know only the capacities and costs associated with its incoming and outgoing links as well its local storage. Also, node $i$ needs to keep track of $p$, $(x, z)$ and $(\tilde{x}, \tilde{z})$ for each incoming and outgoing links as well as the local storage. For the initialization, we can set $p_{ij}^{(t,m)}[0] = a_{ij}/T_m$ at both nodes $i$ and $j$.

One drawback of the proposed algorithm is that the $\tilde{z}[n]$ values are not necessarily feasible in each iteration, and hence, a startup time is required before we can start the multicasts.

### 3.5 Convergence Rate of Dual Subgradient Method

In this section, we analyze the convergence rates of the proposed dual subgradient method in both the primal and dual spaces. In particular, we first prove two propositions which are used for the convergence analysis: Prop. 3 shows that the subgradient of problem (3.33) is bounded, and Prop. 4 proves that the Slater Condition for strong duality holds. We show that in
the dual space, the convergence rate for the subgradient method is linear for a sufficiently small constant stepsize.\(^1\) Similarly, we study the convergence rate in the primal space and show that the recovered primal solutions \((\tilde{z}, \tilde{x})\) converge to a neighborhood of the optimal solution with rate \(O(1/n)\).

**Proposition 3.** The subgradient sequence for problem (3.33) \(\{g[n]\}\) is bounded. That is, there exists a scalar \(L > 0\) such that

\[
\|g[n]\| \leq L, \quad \text{for all } n \geq 0.
\] (3.43)

**Proof.** This is true since the constraint sets \(F_x\) and \(F_z\) defined in (3.35) and (3.37), respectively, are compact and \(g^{(t,m)}_{ij}\) is an affine function of \(x\) and \(z\). Thus, there exists a scalar \(L > 0\), such that

\[
L = \max_{m \in M} \max_{t \in T_m} \max_{(i,j) \in A^+} \max_{z_{ij} \in F_{z_{ij}}} \|g^{(t,m)}_{ij}\|.
\] (3.44)

\(\square\)

**Proposition 4** (The Slater Condition). Let \(c_{ij} > \max_{m \in M} R_m\) for all \((i,j) \in A^+\). There exists a vector \(\{\tilde{z}, \tilde{x}\}\) where \(\tilde{z} \in F_z\) and \(\tilde{x} \in F_x\) such that

\[
\tilde{z}^{(m)}_{ij} > \tilde{x}^{(t,m)}_{ij}, \quad \forall (i,j) \in A^+, m \in M, t \in T_m.
\] (3.45)

**Proof.** Based on the flow conservation constraints, \(x^{(t,m)}_{ij} \leq R_m\) for all \((i,j) \in A^+\).
\[ A^+, m \in \mathcal{M}, t \in T_m. \] Since \( z^{(m)}_{ij} \) is upper bounded by \( c_{ij} \), a sufficient condition for problem (3.31) to satisfy the Slater condition is \( c_{ij} > \max_{m \in \mathcal{M}} R_m \) for all \((i, j) \in A^+\). If this condition holds, one can always find a set of \( \{ \bar{z}^{(m)}_{ij} \} \) that is strictly greater than the corresponding \( \{ \bar{x}^{(t,m)}_{ij} \} \).

Note that the above sufficient condition assumed in Prop. 4 is quite restrictive, and we can always find a vector satisfying (3.45) without having this sufficient condition. In general, if the flow problem has a solution, \( \bar{x} \geq 0 \), for a given network graph \( G' \) and the set of capacity constraints for each arc of the graph, such that it satisfies the flow conservation constraint and \( \sum_{m \in \mathcal{M}} \bar{x}^{(t,m)}_{ij} < c_{ij} \), for all \((i, j) \in A^+, t \in T_m\), then we can choose \( \bar{z}^{(m)}_{ij} = \epsilon + \max_{t \in T_m} \bar{x}^{(t,m)}_{ij} \) for some appropriate \( \epsilon > 0 \) such that \( \sum_{m \in \mathcal{M}} \bar{z}^{(m)}_{ij} < c_{ij} \), for all \((i, j) \in A^+\).

### 3.5.1 Convergence Rate in the Dual Space

The dual subgradient algorithm, given in (3.41), is shown to converge to the dual optimal solution with an appropriately chosen stepsize in each iteration [45]. Since choosing the appropriate stepsize is complex, and requires prior knowledge of the optimal solution, we show in the next section that with sufficiently small constant stepsize, the subgradient method converges linearly to a neighborhood of the optimal solution. This neighborhood size represents the convergence error of the subgradient algorithm.

**Proposition 5.** Let \( \{ p[n] \} \) be the sequence generated by the subgradient
method with constant stepsize $\theta$, where the initial point $p[0]$ is assumed to be bounded, and let $P^*$ be the set of optimal solutions. There exists a positive scalar $\mu$ such that

$$q^* - q(p[n]) \geq \mu (\text{dist}(p[n], P^*))^2, \quad n = 0, 1, \cdots,$$  \hspace{1cm} (3.46)

where $q^*$ is an optimal solution to the dual problem (3.33) and

$$\text{dist}(p[n], P^*) = \min_{p' \in P^*} ||p[n] - p'||.$$

**Proof.** Since the set of solutions for an LP problem is a set of weak sharp minima [64], there exists a positive scalar $\alpha$ such that

$$q^* - q(p) \geq \alpha \text{dist}(p, P^*), \forall p \in P.$$  \hspace{1cm} (3.47)

In particular, the above relation holds for $p \in \{p[n]\}$. Further, Lemma 3 in [65] shows that the sequence $\{p[n]\}$ as well as $||p^*||$ for an arbitrary point $p^* \in P^*$ are bounded, assuming that the initialized point $p[0]$ is bounded. In particular, it is shown that

$$||p[n] - p^*|| \leq \max\{||p[0] - p^*||, \frac{2}{\gamma^2} (f(\bar{z}) - q^*) + \frac{\theta L^2}{2\gamma} + \theta L\}$$

where $\{\bar{z}, \bar{x}\}$ is a Slater vector satisfying (3.45), $L$ is the bound on the sub-
gradient norm obtained in Proposition 3 and

$$\gamma = \min_{m \in M, t \in T_m, (i,j) \in A^+} \{ \bar{z}_{ij}^{(m)} - \bar{x}_{ij}^{(t,m)} \}$$  \quad (3.48)$$

Using the above relations, we can compute an upper bound, denoted by $B$ on $||p[n] - p^*||$, and hence we have $\text{dist}(p[n], P^*) \leq B$. Combining with (3.47) we have

$$q^* - q(p[n]) \geq \frac{\alpha}{B} \left( \text{dist}(p[n], P^*) \right)^2, \quad n = 0, 1, \cdots$$  \quad (3.49)$$

By letting $\mu = \alpha/B$ the proposition is proved. \hfill \Box

**Proposition 6.** Consider the proposed subgradient algorithm with the constant stepsize $\theta$, where $\theta \leq \frac{1}{2\mu}$, and the sequence $\{p[n]\}$ generated by the algorithm. We have for all $n$,

$$(\text{dist}(p[n + 1], P^*))^2 \leq (1 - 2\theta \mu)^{n+1} (\text{dist}(p[0], P^*))^2 + \frac{\theta L^2}{2\mu}. \quad (3.50)$$

**Proof.** From Lemma 2 in [65], we have for any $r \geq 0, n \geq 0$

$$||p[n + 1] - r||^2 \leq ||p[n] - r||^2 - 2\theta (q(r) - q(p[n])) + \theta^2 L^2.$$  

Let $r = p^*$ in the above relation for an arbitrary $p^* \in P^*$. We have

$$||p[n + 1] - p^*||^2 \leq ||p[n] - p^*||^2 - 2\theta (q^* - q(p[n])) + \theta^2 L^2,$$  

104
and by taking the minimum over all \( p^* \in P^* \), and combining it with (3.46), it follows

\[
\text{dist}(p[n+1], P^*)^2 \leq (1 - 2\theta \mu) \text{dist}(p[n], P^*)^2 + \theta^2 L^2. \tag{3.51}
\]

Using induction, it follows from (3.51) that \( \forall n \)

\[
\text{dist}(p[n+1], P^*)^2 \leq (1 - 2\theta \mu)^{n+1} \text{dist}(p[0], P^*)^2 + \theta^2 L^2 \sum_{i=0}^{n} (1 - 2\theta \mu)^i, \tag{3.52}
\]

and since \( \sum_{i=0}^{n} (1 - 2\theta \mu)^i \leq \frac{1}{2\theta \mu} \), we obtain the relation in (3.50).

Proposition 6 demonstrates a tradeoff between the error and the convergence rate of the algorithm, depending on the size of the stepsize. That is, a smaller stepsize leads to slower convergence of the algorithm but also a smaller convergence neighborhood.

Compared with the incremental subgradient method described in Algorithm B in [60], the error of our proposed algorithm is reduced by a factor of \( T_m \) for \( m \in \mathcal{M} \). Also, since we are not using sub-iterations for each terminal within one iteration, the convergence speed is increased by a factor of \( T_m \) for \( m \in \mathcal{M} \).

### 3.5.2 Convergence Rate in the Primal Space

In this section we provide a bound on the convergence rate of the recovered primal variables \( \{\tilde{z}, \tilde{x}\} \). In the following Proposition, we show that when
using a constant stepsize, the primal solutions converge to a neighborhood of the optimal solution with rate $O(1/n)$. This Proposition is similar to Proposition 5 in [60] and its proof is omitted here.

**Proposition 7.** Let $\{\bar{z}, \bar{x}\}$ be a Slater vector satisfying (3.45), and let

$$B^* = \frac{2}{\gamma} \left( f(\bar{z}) - q^* \right) + \max\{ ||p[0]||, \frac{1}{\gamma} (f(\bar{z}) - q^*) + \frac{\theta L^2}{2\gamma} + \theta L \},$$

where $\gamma = \min_{m \in \mathcal{M}, t \in T, (i,j) \in \mathcal{A}^+} \{ \bar{z}_{ij}^{(m)} - \bar{x}_{ij}^{(t,m)} \}$, $L$ is the subgradient norm bound given in Proposition 3, and $\theta$ is the constant stepsize used in the dual iterations. Then, the primal cost $f(\tilde{z})$ in (3.31), after the $n$-th iteration is bounded by

$$f^* - \frac{1}{\gamma} (f(\bar{z}) - q^*) \frac{B^*}{n\theta} \leq f(\tilde{z}[n]) \leq f^* + \frac{||p[0]||^2}{2n\theta} + \frac{\theta L^2}{2},$$

where $f^*$ is an optimal solution to (3.31).

As in the convergence rate bound in the dual space, here we have a similar tradeoff between the size of the neighborhood to which the primal solution is converging and the convergence rate.

### 3.6 Simulation Results

In this section, we demonstrate the results of our simulations of the proposed algorithms for various network scenarios. Since we consider static
multicast in which the multicast groups remain constant for the duration of the connection, the network can run these algorithms to find the minimum-cost multicast for given commodities and then packets can be coded using random linear coding in a distributed manner [58], [57], resulting in a fully distributed approach for achieving minimum-cost multicast. We focus only on the Primal-Dual algorithms with congestion control mechanism given in (3.26)-(3.30). For the simulations, we consider the cost function

$$f_{ij}(z_{ij}) = \frac{z_{ij}}{c_{ij} - z_{ij}}, \text{ for } z_{ij} < c_{ij}, \forall (i,j) \in \mathcal{A}^+,$$

which gives the expected number of packets waiting for or under transmission at arc $(i,j) \in \mathcal{A}^+$ under an $M/M/1$ queuing model. The cost function $\sum_{(i,j) \in \mathcal{A}} f_{ij}(z_{ij})$ is equal to the total number of packets in the network, which is is proportional to average network delay by Little’s Law. We adopt a similar form for the barrier function associated with storage capacity.

We consider an $\alpha$-fair utility function given by

$$U_m(R_m) = \begin{cases} \frac{R_m^{1-\alpha}}{1-\alpha}, & \text{if } \alpha > 0, \alpha \neq 1, \\ \log(R_m), & \text{if } \alpha = 1, \end{cases}$$

where $\alpha$ is a positive parameter of the utility function. In particular, we consider $\alpha = 2$ for all our simulations, as it is widely used as the parameter for TCP.

In the first part of the simulations, we focus on two sets of topologies, as
Figure 3.2: Network topologies studied in our experiments.
depicted in Figures 3.2a and 3.2b. In each of these figures, the capacity of each link and storage capacity of each node is indicated. The storage capacity of each node is written next to the node in a box. The absence of a box next to a node indicates a lack of storage at the node. In each topology set, the network (core + edge) nodes and their topology are the same, but there are a different number of peers, collectively called a peer cloud, connected to the edge nodes. In particular, we allow each of the peer clouds to have a set of terminals of size, 2, 4 and 8, respectively. Figure 3.2c illustrates an example of a peer cloud with four peers. Each peer cloud is modeling a P2P cluster, in which all the peers in the cloud are interested in the same commodity. In addition, each peer is connected to every other peer within the same cloud as well as the connecting edge node. These connections are, generally, overlays, rather than a physical connections. In the Service Network and Backhaul topologies, we do not differentiate between these connections in order to gain insight into the effectiveness of storage at peers vs. the edge. We verify these insights in the simulation on the Abilene topology which does not include these peer clouds.

We consider storage only at the edge and/or at the peers, and compare the results in the case of 1) no storage (No Caching), 2) only at the edge (Edge Caching), 3) storage at the peers (P2P Caching), 4) storage at both peers and edge nodes (Edge+P2P Caching). In particular in the Service Network topology depicted in Figure 3.2a, nodes 3, 4, 5, 6 as well as every peer in the peer clouds have a storage unit with capacity 4. The links among
peers and between peers and their connecting edge nodes all have capacity 6. The peers in each peer cloud are interested in the same commodity. Since in this topology, there are three peer clouds, we consider three types of commodities, namely \( m = 1, 2, 3 \), respectively. Further, the source node for all three commodities is node 1.

In the Backhaul topology depicted in Figure 3.2b, the storage unit at nodes 6, 7, 8, 9 and the peers have capacity 4. In addition, the link capacities among peers and between peers and their edge nodes are all equal to 7.5.
<table>
<thead>
<tr>
<th>Network Cost</th>
<th>Average Throughput (Unit Rate/Terminal)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge+P2P Caching</td>
<td>0 2 4 6 8 10</td>
</tr>
<tr>
<td>P2P Caching</td>
<td>100 200 300</td>
</tr>
<tr>
<td>Edge Caching</td>
<td>400 500 600</td>
</tr>
<tr>
<td>No Caching</td>
<td>700 800 900</td>
</tr>
</tbody>
</table>

(a) Service Network with two peers.  
(b) Service Network with four peers.  
(c) Service Network with eight peers.

Figure 3.4: Network cost ($f(x)$) vs. average throughput tradeoff for the Service Network topology.
Figure 3.5: Network cost \( f(x) \) vs. average throughput tradeoff for the Backhaul topology.
Figure 3.6: Network cost \((f(x))\) vs. throughput tradeoff for the Abilene topology for different flows.
There are 4 types of commodity in this topology, each requested by the peers of a peer cloud. The source node for commodities 1 and 3 is node 1 and the source node for commodities 2 and 4 is node 2.

We further examine the algorithm on the Abilene topology, depicted in Fig. 3.2d. Nodes 1, 3 and 5 are server nodes for flows 1, 2, and 3, respectively. Nodes 0, 2, 3, 4, 6, 7, 8, 9, and 10 (referred to as “edge nodes”) each have a storage unit of capacity 10. In addition, terminals (peers) 11, 12, ..., 22 each have a storage unit with capacity 5. Finally, terminals 13, 15, 17, and 19 are requesting flow 1; terminals 11, 18, 20, and 21 are interested in flow 2, and terminals 12, 14, 16, and 22 are requesting flow 3.

We first examine the convergence performance of the algorithm for the Backhaul topology with eight peers per peer cloud. In Figure 3.3, the convergence of the algorithm for commodities 1, 2, 3 and 4 for the two cases of No Caching and Edge + P2P Caching is illustrated. As can be seen, the converged rates of all the commodities in the case of Edge+P2P Caching are much larger than those for the No Caching case.

We proceed to examine the tradeoff between the network cost and the average throughput for different storage situations. This is achieved by changing the parameter $W$. The result is shown in Figures 3.4, 3.5 and 3.6 for all the three topologies. In all cases, the best tradeoff belongs to the Edge+P2P Caching case and the worst tradeoff belongs to the No Caching case. In addition, the difference is increasingly pronounced for larger rates. This behavior is especially interesting for Flow 3 in Abilene topology, where the No
Caching scenario can barely support throughput over 5 unit rate, while the other storage scenarios can support much higher throughput.

Several other interesting observations can be inferred from these figures. For the Service Network and Backhaul topologies, depending on the number of peers in each peer cloud, P2P Caching works better or worse than Edge Caching. One general trend is that by increasing the number of peers, the performance of P2P Caching surpasses that of Edge Caching, and becomes closer to that of P2P+Edge Caching.

3.7 Conclusion

We introduce a novel framework to study peer-assisted content delivery networking with network coding in a multi-commodity setting. We first introduce a formulation of the problem using general convex cost functions and present distributed primal-dual algorithms for its solution. Then, we consider a more specific problem with linear cost functions for the network resources (links and storage). By adopting a dual subgradient method, we propose a fast and distributed algorithm to find the minimum-cost integrated multicast and storage strategy.

Finally, we implement the proposed algorithms and test them within a number of interesting network scenarios. In particular, we present our results on network behavior in the presence of storage at the edge vs. at the peers. These results can provide us with an understanding of hybrid content delivery
architectures with peer assistance.
Bibliography


