ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS
School of Information Sciences and Technology
Department of Computer Science

Multipath Internet Transport

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Computer Science

by

Yannis Thomas

Athens
July 2018

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to express my deep gratitude to my advisor Prof. George C. Polyzos and to Associate Professor George Xylomenos for their patient guidance and enthusiastic support. It has been a great honor working with them. I am also particularly grateful for the assistance given by Associate Professor Vasilios A. Siris. His useful and valuable comments helped me progress.

I had the privilege of working at the Mobile Multimedia Lab, with great colleagues and friends. Christoforos Ververidis, Nikos Fotiou, Pantelis Fragoudis, Christos Tsilopoulos, Haris Stais, Vaggelis Douros and Xenofon Vasilakos were the first persons I met there. They were very supportive and working with them was always educating for me. I was also quite lucky to work with Merkourios Karaliopoulos, whose comments and suggestions were particularly helpful for fulfilling this dissertation. Livia Chatzieleftheriou and Kostas Tsioutas belong to the new generation of the lab and I wish them the best. I would also like to thank Nikos Zacheilas and George Darzanos for sharing the T.A. burden with me, they are good friends and I wish them the best. Furthermore, I would like to thank Nausika Kokkini, Lina Kanellopoulou, Kostas Makedos and Joanna Lellou for sharing everyday life in the lab, making it bearable on bad days and exciting on good ones.

Finally, I would like to deeply thank my parents, Kostas and Evangelia, and my sister, Theodosia, for their invaluable support throughout the years. Last, but surely not least, I have to thank my friends, who are too many to mention here but in general fall into three categories: beer-buddies, music-mates and surf-brothers. You know who you are!

VITA

2009	Diploma in Informatics, Athens University of Economics and Business, Greece
2012	M.Sc. in Information Systems, Athens University of Economics and Business, Greece
2018	Ph.D. in Computer Science, Athens University of Economics and Business, Greece

PUBLICATIONS

Journal Publications

Y. Thomas, G. Xylomenos, and G. C. Polyzos, "Network-Assisted Multipath Congestion Control," *Open Transactions on Communication Systems (OTCS)*, IFIP, *accepted for publication*, 2018.

T. de Cola, A. Ginesi, G. Giambene, G. C. Polyzos, V. A. Siris, N. Fotiou, and Y. Thomas, "Network and Protocol Architectures for Future Satellite Systems," *Foundations and Trends in Networking*, Vol. 12, No. 1-2, 2017.

Refereed International Conference and Workshop Papers

V. A. Siris, Y. Thomas, and G. C. Polyzos, "Supporting the IoT over Integrated Satellite-Terrestrial Networks Using Information-Centric Networking," in *Proc. IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, Larnaca, Cyprus, November 2016.

Y. Thomas, G. Xylomenos, and G. C. Polyzos, "Multi-flow Congestion Control with Network Assistance," in *Proc. IFIP Networking*, Vienna, Austria, May 2016.

Y. Thomas, G. Xylomenos, C. Tsilopoulos, and G. C. Polyzos, "Object-Oriented Packet Caching for ICN," in *Proc. ACM Conference on Information-Centric Networking (ICN)*, San Francisco, CA, USA, September 2015

Y. Thomas, P. A. Frnafoudis, and G. C. Polyzos, "QoS-driven Multipath Routing for On-demand Video Streaming in a Publish-Subscribe Internet," in *Proc. IEEE International Conference on Multimedia & Expo (ICME) Workshop on Multimedia Streaming in information-Centric networks (MuSiC)*, Turin, Italy, June 2015.

Y. Thomas, G. Xylomenos, and G. C. Polyzos, "Exploiting Path Diversity for Networked Music Performance in the Publish Subscribe Internet," in *Proc. International Conference on Information, Intelligence, Systems and Applications (IISA)*, Corfu, Greece, July 2015.

N. Fotiou, Y. Thomas, V. A. Siris, and G. C. Polyzos, "Security Requirements and Solutions for Integrated Satellite-Terrestrial Information-Centric Networks," in *Proc. IEEE Advanced Satellite Multimedia Systems (ASMS) Workshop on Signal Processing for Space Communications (SPSC)*, San Diego, CA, USA, June 2014.

Y. Thomas, and G. Xylomenos, "Towards Improving the Efficiency of ICN Packet-Caches," in *Proc. IEEE International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness (QShine)*, Rhodes, Greece, August 2014.

Y. Thomas, C. Tsilopoulos, G. Xylomenos, and G. C. Polyzos, "Accelerating File Downloads in Publish Subscribe Internetworking with Multisource and Multipath Transfers," in *Proc. World Telecommunications Congress (WTC)*, Berlin, Germany, June 2014.

C. Tsilopoulos, G. Xylomenos, and Y. Thomas, "Reducing Forwarding State in Content-Centric Networks with Semi-stateless Forwarding," in *Proc. IEEE INFOCOM*, Toronto, ON, Canada, May 2014.

Y. Thomas, C. Tsilopoulos, G. Xylomenos, and G. C. Polyzos, "Multisource and Multipath File Transfers through Publish-Subscribe Internetworking," in *Proc. ACM SIGCOMM Workshop on Information-Centric Networking (extended abstract)*, Hong Kong, China, August 2013.

C. N. Ververidis et al. "Experimenting with Services over an Information-Centric Integrated Satellite-Terrestrial Network," in *Proc. of the IEEE Future Network and Mobile Summit (FutureNetworkSummit)*, Lisbon, Portugal, July 2013.

C. Stais, Y. Thomas, and G. Xylomenos, "Networked Music Performance over Information-Centric Networks," in *Proc. of the IEEE International Conference on Communications (ICC) Workshops*, Budapest, Hungary, June 2013.

G. Parisis et al. "Demonstrating Usage Diversity over an Information-Centric Network," in *Proc. of the IEEE INFOCOM Workshops*, Turin, Italy, April 2013.

G. Xylomenos, C. Tsilopoulos, Y. Thomas, and G. C. Polyzos, "Reduced Switching Delay for Networked Music Performance," in *International Packet Video Workshop (Poster Session)*, San Jose, CA, USA, December 2013.

ABSTRACT OF THE DISSERTATION

Multipath Internet Transport

by

Yannis Thomas

Doctor of Philosophy in Computer Science

Athens University of Economics and Business, Athens, 2018

Professor George C. Polyzos, Chair

The proliferation of smartphones, with their multiple interfaces, and data servers, with their high-performance interconnection networks, has revived interest in multipath transport protocols. *Multipath-TCP* (MPTCP), the multipath extension of TCP, is currently available in the Apple iOS and Linux operating systems, enabling bandwidth aggregation, load balancing, and resilience to failures and disconnections due to mobility. However, the deployment of multipath transport is challenged by the address-based TCP/IP communication, which does not facilitate the seamless establishment of multiple paths among two end-points, and by the distributed hop-by-hop TCP/IP routing, which does not ensure the disjointness of the paths. Even when multiple paths are deployed, the use of many subflows is both a blessing and a curse for multipath TCP/IP protocols, as they tend to grasp

an unfair share of bandwidth, thus becoming unfriendly to single-path TCP flows. The latest congestion control algorithms for MPTCP attempt to equalize the cumulative subflow throughput with the throughput of the fastest single-path flow in the same link, thus exchanging performance for TCP-friendliness. While TCP-friendly in the long run, these approaches exhibit high throughput convergence latency, thus being effective only for long-lived flows.

Our contribution to multipath transport is two fold. First, we introduce the *multipath multisource Transport Protocol* (mmTP), a transport-layer protocol that offers reliable multipath and multisource content delivery in the *Publish-Subscribe Internet* (PSI) architecture. mmTP increases the utilization of network processing resources, exploits on-path and off-path caching and does not require additional state at routers, or complex signaling during connection establishment. Second, we propose a novel hybrid multipath congestion control algorithm that enhances resource utilization through *greedy friendliness*, a design that meets the TCP-friendliness constraint only when is needed. The hybrid congestion control scheme consists of the novel end-to-end *Normalized Multiflow Congestion Control* (NMCC) algorithm, which offers accurate and instant convergence to TCP-Friendliness, and an in-network topology management module, that provides disjoint paths when possible and notifies end-users about shared bottlenecks otherwise. We finally discuss the integration of the proposed designs with the TCP/IP architecture: mmTP through *Software Defined Networking* (SDN) and NMCC through MPTCP
.

Chapter 1

Introduction

Experience with content distribution applications and protocols indicates that multisource and multipath [4], i.e. the use of multiple sources and multiple paths to each source, respectively, can benefit both network operators and end-users. The exploitation of multiple *paths* offers bandwidth aggregation, network load balancing and resilience to link failures, while the use of multiple *sources* can further enhance throughput and offer resiliency to source failures. As a result, multisource and multipath, collectively referred to as *multiflow*, provide load balancing, higher resource utilization and fault tolerance.

Multiflow is flourishing in the constantly shifting Internet where end-users evolve to content providers and their devices increase in numbers, as well as processing, storage and networking capabilities. By acknowledging that popular content resides at numerous locations, a case emphasized by the *Content Delivery Networks* (CDNs) [5], multisource transport constitutes a timely solution that suits better to current networking needs and opportunities. In addition, multipath lately has gained much attention since multihomed devices, such as servers in data centers and smartphones with Wi-Fi, Bluetooth and Cellular connectivity, are responsible for a large portion of Internet traffic [6, 7].

Nevertheless, multiflow is smothered by the singlepath sender-driven TCP/IP architecture for two reasons: first, the networking model does not endorse the establishment of multiple paths among two communication endpoints and, second, the distributed hop-by-hop routing does not assure the disjointness of the routes.

Therefore, multiflow solutions in the TCP/IP architecture require complicated techniques with apparent infrastructural and operation overhead in order to, first, allow the deployment of multiple dissemination flows (i.e. multihoming, CDN and SOCKS servers [8]) and, second, enhance paths' disjointness (i.e. *Multi-Protocol Label Switching* (MPLS) and *Software Defined Networking* (SDN) [9, 10]). These solutions present sensible concerns with regard to performance efficiency, cost or scalability. For example, the *BitTorrent* [11] multisource application-level protocol requires an external mechanism to discover the multiple sources and does not allow fine-grained congestion control, thus compromising network fairness. The *Multi-path TCP* (MPTCP) [12] protocol, which is the flagship of multipath in TCP/IP, being available in iOS (since iOS7) and the Linux Kernel, typically requires multiple network addresses in order to establish multiple subflows, thus reducing the range of supported devices. Finally, MPLS and SDN enhance path diversity, but present serious scalability problems, due to the induced forwarding state at the on-path routers.

Multiflow congestion control is also a challenging topic for the TCP/IP architecture that primarily endorses end-to-end singlepath connectivity. Congestion control for multiflow protocols is more challenging than for single-path ones, needing to address multipath-specific issues, such as bandwidth aggregation, TCP-friendliness, stability and responsiveness [13]. A significant body of research has focused on the TCP-friendliness issue [14], where a multiflow connection of N flows grasps N times more bandwidth than a single-path flow that competes in the same bottleneck. To avoid this issue, the IP-based TCP-friendly multiflow protocols typically reduce their overall aggressiveness so as to grasp about the same bandwidth with the singlepath TCP connections. However, blindly restricting multipath flows leads to poor responsiveness and degraded resource utilization when friendliness to single flows is not necessary, for example, when subflows perform over disjoint paths. Unfortunately, in the TCP/IP architecture each node is only aware of its own routing decisions, hence it always assumes that shared bottlenecks exist in order to prevent unfair resource sharing. Although the friendliness constraint is vital in theory, the "always-friendly" policy is inefficient in practice, leading to

underutilization of network resources in realistic AS-scale scenarios.

A second weakness of existing IP-based multiflow congestion control algorithms is their slow convergence to TCP-friendliness. The majority of proposed solutions are based on the fluid model analysis of Kelly et al. [15], a well established method for deriving congestion control algorithms. The *Linked Increase Algorithm* (LIA) [3], the *Opportunistic Linked Increase Algorithm* (OLIA) [16] and the *Balanced Link Adaptation* (Balia or BALIA) algorithm [17], three of the most popular congestion control options for MPTCP, use the model to deduce a sufficient (equilibrium) condition that dynamically defines the amount of window increase, upon receipt of an ACK, and the amount of window decrease, upon receipt of a congestion event, so as to achieve high resource utilization, stability, responsiveness and TCP-friendliness. While accurate in the long run, as they manage to equalize the transfer rates in "steady state", they can exhibit poor convergence, requiring long time periods until connections grasp their friendly share of bandwidth, thus offering TCP-friendliness only to long-lived flows.

1.1 Motivation for this work

Information-Centric Networking (ICN) [18] is a novel networking approach, that evolves the Internet architecture away from the host-centric end-to-end model, towards a content-centric or information-centric model. One of the driving forces in ICN research is the design of architectures and protocols that efficiently utilize network resources. In addition to communication, ICN brings data storage and computation into the spotlight of available network resources. Therefore, most ICN-inspired data delivery mechanisms exploit caching, either on-path for packets cached in routers (short-term memory) or off-path when entire content-objects reside in dedicated caching servers (long-term memory). On-path caching is an instance of the multisource delivery, since the content is delivered by the original content source and by the caches that are located on the dissemination paths. Additionally, off-path caching is a form of multisource and multipath, since the requests for content can be redirected by the network to remote caches, that are not

located on the dissemination path. Both transfer techniques are offered natively in ICN networks.

The *Publish Subscribe Internet* (PSI) architecture [18] is an instantiation of the ICN concept based on the publish/subscribe paradigm. Besides in-network caching, PSI offers name-based content resolution via a dedicated network functionality, the *Rendezvous*, which allows seamlessly many-to-many communication among content sources, thus supporting multisource. Additionally, PSI supports centralized path selection via a special network entity, the *Topology Manager*, and source routing via LIPSIN forwarding [19], which allow natively the establishment of multiple dissemination paths among two communication endpoints, thus delivering multipath.

We exploit these features to present the *Multisource and Multipath Transfer Protocol* (mmTP), the first multiflow transport protocol for PSI. mmTP is designed to utilize all types of network resources by combining well-known content-distribution techniques into a single protocol. mmTP is receiver-driven and supports on-path caching, thus utilizing the network's short-term memory. It supports downloading files from multiple sources, thus utilizing the network's long-term memory. Furthermore, mmTP supports multiflow, i.e. an mmTP transfer can fetch content from multiple sources and use multiple paths to transfer content from each source. Last, but not least, mmTP supports all these features without complicating network operation as it does not require extending PSI with complex signaling or router operation.

A significant contribution of mmTP is the introduction of a novel multiflow congestion control design that can exploit the topological knowledge of the network in order to better balance performance and TCP-friendliness. The hybrid congestion control scheme consists of two independent modules: (i) *Normalized Multiflow Congestion Control* (NMCC), an end-to-end multiflow-aware algorithm, and (ii) an in-network topological information mechanism that assists NMCC. NMCC is a simple yet effective algorithm that manages bandwidth aggregation under the friendliness constraint, offering accurate and instant TCP-friendliness, even in the face of difficult conditions, such as mismatched paths and sudden

changes in congestion levels. The in-network mechanism participates directly to congestion control by selecting paths, e.g., only disjoint paths, or indirectly by providing information about shared bottlenecks, thus allowing NMCC to practice *greedy friendliness*, i.e, ignoring the friendliness constraint when paths are disjoint. Although our design is inspired by the PSI architecture, IP networks that operate over technologies that utilize centralized path computation components, including MPLS and SDN, are in principle capable of providing path overlap information to the end-to-end congestion controllers, thus being compatible with our solution.

Finally, the NMCC algorithm presents a novel approach to end-to-end multipath congestion control that combines instant TCP-friendliness and high resource utilization. NMCC converges instantly by exploiting a deterministic scheme that equalizes the throughput growth rate of the subflows, rather than the throughput rate itself. While this approach constitutes a minor modification to the existing models of multiflow congestion control, it results in significant gains, such as, instant convergence to friendliness. In addition, NMCC is compatible with the MPTCP congestion control handlers of the Linux Kernel, hence it is seamlessly integrated with the Linux implementation of MPTCP which is used in several of our experiments in order to verify the gains of NMCC.

1.2 Contributions

Our contributions, which we describe in this dissertation are:

- We review the PSI ICN architecture and present its main design choices with respect to multipath transport.
- We review the challenges of multipath transport in the TCP/IP architecture, as well as related solutions. Our discussion focuses on existing protocols and algorithms for establishing multiple dissemination paths and performing congestion control.
- We present the *multisource and multipath Transfer Protocol* (mmTP), a multiflow transport protocol for the PSI architecture. mmTP offers multipath

and multisource content delivery without requiring complex signaling during path establishment or additional routing state in the routers.

- We present a hybrid multipath congestion control that combines end-to-end and in-network operation. Our technique introduces the discovery and exploitation of topological information so as to enhance resource utilization, thus constituting the first solution to offer a deterministic and complete implementation of the *greedy friendliness* concept. Through network domain-scale simulations, we estimate that greedy friendliness can enhance the average network resource utilization by up to 160% compared to single-path and by up to 15% compared to standard multipath.

- We present and evaluate the *Normalized Multiflow Congestion Control* (NMCC) algorithm, a novel end-to-end multipath congestion control algorithm that offers high resource utilization and instant convergence to TCP-friendliness. Through experiments with the Linux MPTCP implementation, we verify that NMCC offers friendliness to single-path connections regradless of transfer duration, while LIA, OLIA and BALIA can require hundreds of seconds before they are effective.

- We discuss the application of mmTP's hybrid congestion control in the TCP/IP architecture through the MPLS or SDN techniques, as well as the integration of NMCC with the Linux implementation of MPTCP.

1.3 Dissertation outline

The remainder of the dissertation is organized as follows. Chapter 2 provides a review of the Publish-Subscribe Internetworking (PSI) architecture, as well as the ICN fundamental concepts. Chapter 3 discusses multipath requirements and presents existing multipath solutions specifically designed to cope with these requirements. Chapter 4 details our multipath solution, which basically consists of the mmTP multiflow protocol and the NMCC end-to-end congestion control algorithm. Chapter 5 discusses the application of our solutions to the current

Internet through the SDN technology and the MPTCP protocol. Chapter 6 discusses topics that are related to multipath transport in ICN, such as delay-based end-to-end congestion detection, and our future work. Finally, the conclusions of this dissertation are presented in Chapter 7.

Chapter 2

Information-centric Networking (ICN)

The Internet was created at a time when the main goal was to support military and research purposes. Since then, it has evolved significantly, making it an important component of mainstream modern society and increasing the number of stakeholders. As a result we have moved from the point where the common goal was to provide interconnection between existing networks, to another where access to content is the major use case.

Inevitably, through the years, the question has changed from where to what. This transition revealed that host-to-host communication, as a networking abstraction chosen to fit a problem of the '60s, is insufficient for the purposes of the modern Internet which include content availability, security and network performance. Thereupon, a new network technology was progressively developed, attempting to place content in the center of all networking functions. The result of such efforts is the birth of a fundamental shift in communications and networking presented by *Information-Centric Networking* (ICN).

ICN, also known as content-aware, content-centric or data-oriented networking, focuses on finding and transmitting information to end-users instead of connecting end hosts for communication. In contrast to current networks' lack of awareness about the type of content they are transporting and their consequent inability to adapt and offer the appropriate *Quality of Experience* (QoE) to the

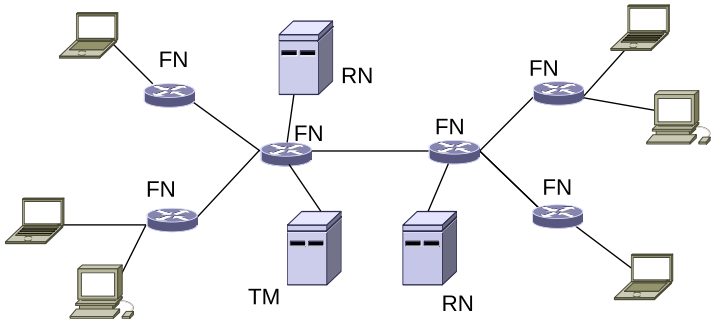


Figure 2.1: A PSI network consisting of Forwarding Nodes (FN), Rendezvous Nodes (RN), a Topology Manager Node (TM) and hosts.

end-users, ICN treats information as a primitive. The main objective of ICN is to make information consumption a core service of the network besides facilitating dissemination of content between providers and consumers.

2.1 Publish Subscribe Internetworking (PSI)

The *Publish Subscribe Internetworking* (PSI) [18] architecture follows the ICN paradigm and completely replaces the TCP/IP protocol stack with a publish-subscribe protocol stack. PSI treats information or content objects as publications, content sources as *publishers* and content consumers as *subscribers*. Publishers and subscribers are provided with a publish/subscribe API for advertising and requesting information, respectively. A fundamental design tenet in PSI is the clear separation of its core functions: (a) the Rendezvous function tracks available publications and resolves subscriptions to publishers, (b) the Topology Management function monitors the network topology and forms forwarding paths and (c) the Forwarding function undertakes packet forwarding. Thereafter, network nodes in a PSI network are classified into *Rendezvous Nodes* (RNs), *Topology Managers* (TMs) and *Forwarding Nodes* (FNs), as shown in Fig. 2.1.

The information objects are named with a (statistically) unique pair of IDs, the *scope ID* and the *rendezvous ID*. The rendezvous ID is the actual identity for a particular object that needs to belong to at least one information scope, while the scope ID groups related content items. Content names in PSI are flat but scopes

can be organized in scope graphs of various forms, including hierarchies, therefore a complete name consists of a sequence of scope IDs and a single rendezvous ID. Scopes serve as a means of defining the relevance of content items within a given context and enforcing "boundaries" based on some dissemination strategy for the scope. For example, a publisher may place content under a "single-path" scope or a "multipath" scope, with each scope having different content delivery patterns.

To deliver information in PSI, a publisher first advertises a content object by publishing it to the Rendezvous function. Subscriptions are also handled by the Rendezvous function, which locates available publishers for the requested content. The Rendezvous function is jointly performed by several RNs possible organized as a *Distributed Hash Table* (DHT), thus distributing the load of tracking publications and serving subscriptions. When publishers and subscribers of the same content are found, the Rendezvous function asks the Topology Management function to compute suitable paths in order to deliver the requested information. The Topology Management function is carried by one or more TM nodes that maintain an up-to-date view of the network topology by gathering link-state information directly from the FNs. Typically, the TM shapes the forwarding path between the publisher(s) and the subscriber(s) and creates the corresponding *Forwarding Identifier* (FID). For packet forwarding, PSI employs LIPSIN, an efficient explicit-routing scheme based on Bloom filters [19], that allows source routing without inducing state overhead at the in-network routers. Specifically, each network node assigns a tag, i.e., a long bit string produced by a set of hash functions, to each of its outgoing links, and advertises these tags via the routing protocol. A path through the network is then encoded by ORing the tags of its constituent links and the resulting Bloom filter is included in each data packet. When a data packet arrives at a FN, the FN simply ANDs the tags of its outgoing links with the Bloom filter in the packet; if any tag matches, then the packet is forwarded over the corresponding link. In this manner, the only state maintained at the FNs is the list of link tags. Multicast transmission can be achieved by simply encoding the entire multicast tree into a single Bloom filter, but multisource and multipath are also favored by this stateless source-routing technique. After the LIPSIN FID is formed, the TM

hands it to the publisher, which uses it to transmit the requested publication over the encoded path.

Chapter 3

Multipath transport

3.1 Introduction

In computer networks, the term *multipath* describes an efficient content delivery pattern that is associated with the establishment of multiple dissemination routes, also known as *paths*, for delivering an individual content item to a receiving host. Multipath is a generic term that may refer to different path exploitation patterns with distinct gains, thus constituting an agile solution. In detail, the path exploitation can be concurrent or sequential, redundant or complementary, thus exhibiting different advantages and satisfying a wide variety of requirements. For instance, a transfer scheme that delivers content concurrently via multiple paths is more likely to avoid performance bottlenecks, thus offering higher throughput. Oppositely, a transfer scheme that exploits paths in sequence exhibits enhanced resilience to network failures as connectivity is strengthened by the exploitation of alternative paths [20]. A connection that proceeds to redundant content transmissions, where information is replicated over multiple paths, exhibits timely resilience to network errors since corrupted or lost packets can be recovered over the backup path without temporal cost. On the other hand, a connection that distributes the content delivery over multiple paths increases the resource utilization of the network, thus realizing load balancing and congestion avoidance. All in all, multipath constitutes an efficient communication pattern that network operators and end-users exploit "a la cart" in order to attain valuable performance advantages, such

as increased transfer rate, better utilization of network resources and improved resilience to network failures.

A second criterion for specializing multipath, besides path exploitation schemes, is the number of content sources participating in the transmission. While the traditional IP networking assumes a single source per communication, experience from CDNs and peer-to-peer networks suggests that popular information pieces are stored in multiple locations [4, 5], thus emphasizing the potential of *multisource*. Multisource is a transfer pattern where a receiver fetches an individual content item from multiple content sources. Being a specialization of multipath, multisource preserves all the aforementioned performance advantages, but also introduces two substantial differences: first, it is supported by the address-based TCP/IP networking model and, second, it is harder to be managed by the sender-driven TCP/IP model. The differences between multipath and multisource, which are emphasized by the TCP/IP architecture, are diminished by the ICN architectures, where internetworking is information-based and the receiver-driven paradigm is dominant. Consequently, our discussion, analysis and evaluation of multipath transport in the context of ICN handles multisource and multipath indiscriminately; collectively mentioned as *multiflow*, it indicates the establishment of multiple paths to multiple content sources.

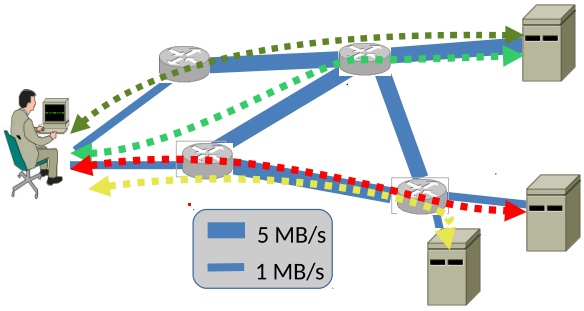


Figure 3.1: An example of multiflow transfer where two paths (multipath) are established to two content sources (multisource), thus deploying four paths in total.

3.2 Multipath solutions

3.2.1 Multipath in TCP/IP

Multipath is challenged by the singlepath sender-driven TCP/IP architecture mainly for two reasons: first, the address-based networking model does not facilitate the establishment of multiple paths among two communication endpoints and, second, the distributed hop-by-hop routing does not ensure the disjointness of the routes, thus reducing the potential gains. We hereby present four categories of multipath solutions that offer multipath in the TCP/IP architecture with apparent costs and gains. Although the list of protocols that are discussed in each category is not exhaustive, we select some indicative solutions that highlight the strengths and weaknesses of each category. Readers are referred to [1] for a thorough survey on the proposed network-level solutions for building multiple routes in IP networks.

Multisource

Multisource constitutes a special case of multipath where a receiver fetches an individual content item from multiple content sources, thus shaping a *many-to-one* transfer pattern. Each content source is identified by a different IP address that can be used for establishing an individual connection, thus allowing the creation of multiple transmission flows from the many sources to the single receiver. Despite many-to-one connections not being supported by the transport-layer protocols of the TCP/IP architecture, multisource solutions can be seamlessly deployed at the application layer. However, the many-to-one communication model is incompatible with TCP-like sender-driven transfer control, since the cooperated congestion control or path scheduling of multiple sources is rather complex and process intensive, e.g., using fountain coding [21]. Consequently, multisource transfers are usually realized by receiver-driven application-level protocols.

BitTorrent The *BitTorrent* [11] protocol is a peer-to-peer multisource application-level protocol for TCP/IP networks with increased penetration in file sharing ser-

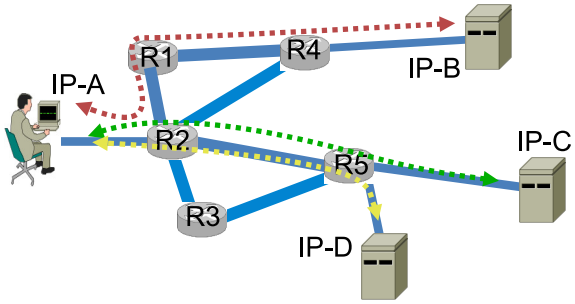


Figure 3.2: An example of multipath transfer in the TCP/IP architecture based on multisource. The single-homed receiver associates its address with the addresses of many single-homed content sources, in order to deploy multiple end-to-end connections.

vices. BitTorrent exploits the fact that popular items reside at multiple locations and establishes multiple connections to many locations in a receiver-driven manner, thus offering enhanced throughput, loose coupling of end-users and, in turn, resilience to sender failures. In order to discover multiple sources, BitTorrent exploits a meta-file that lists the IP addresses of locations delivering that content. The meta-file is necessary for realizing multisource and is acquired through out-of-band mechanisms, such as e-mail or Web Servers. The detailed description of BitTorrent is beyond the scope of this Dissertation, however it is worth noting that the content is fragmented into data chunks and the receiver utilizes an indexing data structure that maps chunks to senders. Thereupon, the receiver starts fetching simultaneously multiple chunks from multiple stateless senders, thus avoiding potential bottlenecks at the senders' side and enhancing transfer rate. In case a sender is unreachable or irregularly slow, the receiver can choose to download the chunk from different source, achieving resiliency against failures. The transfer finishes when all data chunks are downloaded and the fragmented information item is reconstructed.

BitTorrent is a rather effective multipath tool that offers seamless establishment of multiple physically-divergent paths. The deployment of multiple single-path connections that independently fetch complementary data chunks does not require any modifications to the network or the end-hosts. Furthermore, the physical distance of the sources combined with IP/TCP topology-based routing is most

likely to enhance the divergence of paths, thus enhancing the probability of avoiding performance bottlenecks. On the other hand, BitTorrent requires the out-of-bound mechanism for acquiring the meta-file information, which is not computationally intense for the network, but reduces the application range of the protocol to mainly file-sharing services. Moreover, BitTorrent uses multiple parallel TCP connections for downloading simultaneously many data chunks, thus presenting rather coarse-grained traffic management.[1] The exploitation of TCP at chunk-level penalizes the efficiency of BitTorrent in terms of congestion and flow control, since it presents over-aggressiveness and harms network fairness, as discussed in detail in Section 3.3.

DASH The *Dynamic Adaptive Streaming over HTTP* (DASH) [22] protocol is a streaming technique that dynamically adapts video quality to the end-user's capabilities and network conditions, e.g., shift from high to low resolution video when a device moves from a fast to a slower connection, thus avoiding playback interruptions and using network resources more efficiently. Although DASH is not famous for its multisource capabilities, it can download content from many locations-sources, e.g., exploiting that replication of information is spread physically through *Content Delivery Networks* (CDNs) [5], a network of cache servers that are located closer to the users so as to offload origin servers and reduce the download latency. Acknowledging that each user-device has different processing, downloading and preview capabilities and that the video content can be represented in different quality levels that can be stored in multiple servers, DASH proposes a receiver-driven chunk-based multisource-compatible streaming technique where the end-devices dynamically shift between different quality levels and, possibly, sources in order to adapt the quality of the service to their current requirements. Similarly to BitTorrent, DASH requires content fragmentation into data chunks and a meta-file that maps "chunk, quality of representation" tuples to URLs. Differently to BitTorrent, each tuple is mapped to only one URL and data chunks are fetched in sequence thus disallowing simultaneous source exploitation. Con-

[1] Although TCP is used at chunk-level, we consider BitTorrent receiver-driven due to the small granularity of data chunks.

sequently, the multipath gains presented by DASH include bottleneck avoidance, resilience to network failures and seamless user mobility, but exclude bandwidth aggregation. Finally, in video streaming services, some form of source-probing is required to detect the quality of the available paths, in order to shift timely between different sources and enhance resilience. Nevertheless, such a feature is not provided by DASH for minimizing the bandwidth requirements of the service, therefore a single source usually delivers the different representations of the video.

Multihoming

Multihoming describes the simultaneous connection of a host to multiple networks. Each network assigns a different IP address that can be used to establish a different transmission flow, hence the creation of the individual connections in the TCP/IP architecture is offered to multihomed peers seamlessly. Although, the requirement of an additional physical network interface has discouraged the wide exploitation of multihoming in the past, the current proliferation of smartphones and datacenters, where devices are usually equipped with multiple network interfaces, revived the attention to multipath via multihoming.

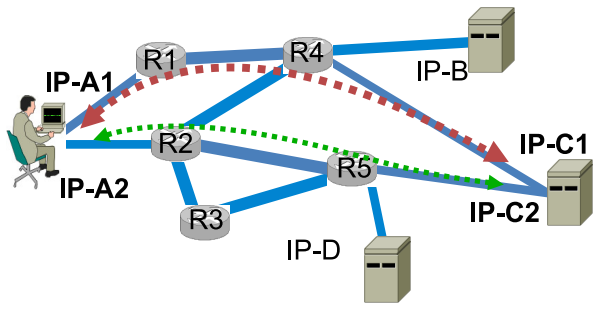


Figure 3.3: An example of multipath transfer in the TCP/IP architecture based on multihoming. The multihomed receiver uses two IP addresses to deploy two end-to-end subflows with the multihomed content source.

MPTCP The *Multipath-TCP* (MPTCP) protocol [12, 23], the multipath extension of TCP, is proposed for transmitting content over multiple paths among hosts with multiple addresses. The protocol offers the same type of service to applica-

tions as the single-path TCP protocol, but also provides the necessary components to establish and manage multiple TCP flows across potentially disjoint paths. An MPTCP connection begins similarly to a regular TCP connection, thus delivering backwards compatibility, but enriches the semantics of the TCP handshake in order to establish additional flows. We detect two types of extended TCP handshakes that take place sequentially: the *advertising* and the *joining*. The design of the two extended handshakes is rather sophisticated and complex allowing two methods (explicit and implicit) for adding and removing a flow [23]; we hereby present a simplified overview of the procedure. At first, users proceed to the advertising handshake that includes two important information pieces: (i) the *MP_CAPABLE* option, that marks the initial MPTCP flow, and (ii) some identifiers/keys for authenticating the subsequent flows of the same MPTCP session. Both hosts are required to include the MP_CAPABLE option during the joining handshake, otherwise MPTCP falls back to single-path TCP. Second, users proceed to the joining MPTCP handshake that includes three essential information pieces: (i) the *MP_JOIN* option, that marks a subsequent MPTCP flow, (ii) the authentication keys of the advertising handshake in order to associate the subsequent with the initial flow, and (iii) the addresses that the subsequent flow will be build upon. Interestingly, the identifier of an MPTCP flow is akin to the TCP 5-tuple (protocol, local address, local port, remote address, remote port), thus allowing multiple MPTCP flows per IP address via port multiplexing. This feature enables multipath transfers for single-homed devices, albeit the different flows will be typically routed over the same path thus nullifying the multipath gains.[2] After the MP_JOIN handshake takes place, the subsequent flow is combined with the existing session and, thereon, is called *subflow* because the session continues to appear as a single connection to the applications at both ends.

The main advantage of MPTPC is not requiring any modification to the network operation, thus allowing instant and partial protocol adoption. MPTCP is currently available in the Linux kernel[3], as well as in Apple's iOS7 and later ver-

[2]In this case *Equal Cost MultiPath* (ECMP) can be considered at the network routers in order to split the MPTCP stream over different paths based on the hash of the MPTCP header [24].

[3]https://www.multipath-tcp.org/

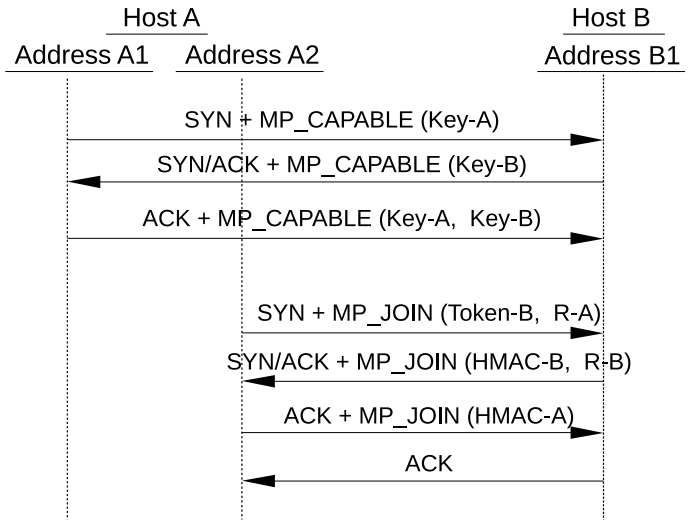


Figure 3.4: The two handshakes required for establishing two MPTCP subflows. "R-A/B" are random numbers selected by the hosts to avoid replay attackes. "Token-B", which is generated from Key-B, is used for authenticating the new subflow. "HMAC-A/B" are the *Hash-based Message Authentication Codes* (HMACs) of the hosts.

sions. On the other hand, we detect three important weaknesses of MPTCP in the TCP/IP architecture. First, the requirement for multihoming reduces significantly the scope of support, offering multipath mainly to smartphone users and data centers; in [2] the authors enumerate the network setups that currently allow MPTCP, unveiling the deployment overhead of multipath in the TCP/IP architecture. Second, (sub)flows that carry topologically proximate IP addresses are most likely to be forwarded over the same paths due to the topology-based TCP/IP routing. Third, studies have shown that four and six paths are required to effectively utilize network resources in Internet and data center topologies respectively [1, 25], hence the "2-homed" smartphone devices can not deliver the promised gains. Later in this section we discuss the integration of MPTCP with *Software Defined Networking* (SDN) in order to address these weaknesses, providing broader device support and enhanced performance due to increased path diversity.

SCTP The *Stream Control Transmission Protocol* (SCTP) [26] is a transport-layer protocol for the TCP/IP architecture that exploits multihoming so as to

enhance the connectivity of the session and, in turn, the availability of the information. Similarly to MPTCP, SCTP is connection-oriented, providing a list of IP addresses to each endpoint in order to establish multiple flows. Oppositely to MPTCP, SCTP is chunk-based and receiver-driven, thus allowing the receiver to explicitly pull pieces of content from the multihomed sender, and exploits paths sequentially, thus enhancing resilience to network failures. A *Concurrent Multipath Transfer* (CMT) extension for SCTP [27] has also been proposed, but is not included in the corresponding RFC [26].

Typically a special function, called *path management*, chooses the destination address for each outgoing SCTP packet based on the perceived reachability status of the subflows. In case other packet traffic is inadequate, heartbeats are used to monitor the reachability. The path management function is also responsible for reporting the set of local addresses during the association/startup of the SCTP end-users. An SCTP association between two endpoints is initiated with an *INIT* message, which is sent by the SCTP receiver to the sender, and is followed by an *INIT_ACK* message, that is sent by the sender to the receiver in order to verify the connection establishment. Both messages can carry a set of local addresses that are used for establishing multiple subflows.

SCTP is an effective multipath protocol that was among the first to open up the exploitation of multihoming. Nevertheless, the requirement of additional network interfaces lessened its market penetration, since the rise of multihomed devices is quite recent. In addition, the concurrent exploitation of paths has not been explored sufficiently yet, since no studies elaborate on the CMT-SCTP realization over the Internet (e.g., traversing middle-boxes [28]), and experience from MPTCP's deployment in the Internet shows that multipath transmission in the TCP/IP architecture is rather complicated [29]. Therefore, SCTP is a multipath solution that effectively enhances information availability, however CMT-SCTP is not studied in depth and SCTP is generally smothered by Internet's ossification [30].

Source routing

According to the source routing paradigm, the source of a network packet specifies the complete route until the destination [31]. In this way, path formation can meet distinct requirements, such as guaranteed paths' disjointness. Typically a source routing mechanism includes two steps: first, discovering routes and encoding the forwarding information and, second, embedding the discovered forwarding information in the packet's header. Then, the on-path routers use the forwarding information to select the next destination of the packet, being usually unaware of the entire path. The main advantage of source routing is supporting the implementation of effective traffic engineering mechanisms that map explicitly flows to dissemination paths, such as *Differentiated Services* (DiffServ) [32]. Oppositely, the main weakness of most source routing techniques compared to hop-by-hop routing is the induced forwarding state, that is either placed at the packets or the in-network routers, thus lessening the scalability of the mechanisms.

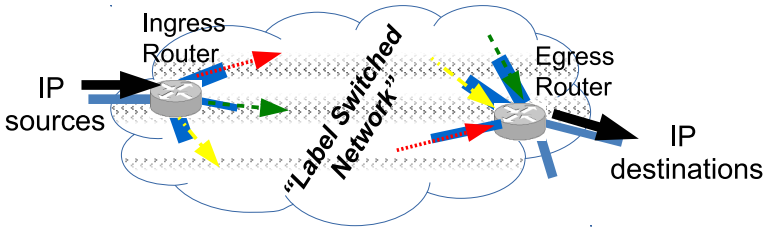


Figure 3.5: An example of multipath transfer in the TCP/IP architecture based on the MPLS source routing technique. IP traffic enters the source-routed network via the ingress router that shares the traffic volume among different explicitly defined paths until the egress router. The same approach is followed by most source routing techniques [1, Sect.II.A].

MPLS *Multiprotocol Label Switching* (MPLS) [9] is a network-layer routing technique that offers source routing in IP networks via label-stacking. MPLS is ubiquitously implemented in backbone-networks of ISPs in order to apply QoS-based traffic control based on the DiffServ approach; it classifies the incoming IP flows in

different classes and routes them via predefined unidirectional paths, called *Label-Switched Paths* (LSPs) or *MPLS tunnels*. The LSPs are materialized by the three different types of MPLS routers: one *ingress* MPLS router, that interfaces the MPLS network with the IP senders, multiple *transit* label switched routers, that forward packets along the LSP, and one *egress* MPLS router, which interfaces the MPLS network with the IP receivers. The ingress router receives IP traffic and prepares it for traversing the MPLS tunnel which includes mapping packets to a class, inserting the stack of labels that encodes the LSP of that class and forwarding them to the next transit router. When a labeled packet is received by a transit label-switched router, then the top label is examined and, typically, popped. If the label is found in the local lookup table that maps labels onto outgoing ports, then the packet is forwarded accordingly. Finally, when the packet reaches the egress router then the last label has been removed, hence the initial IP packet is forwarded to its IP destination. Consequently, besides MPLS-specific functionality, the egress router requires IP routing capabilities.

MPLS comes with apparent weaknesses such as additional infrastructure requirements and coarse-grained classification. The infrastructural requirements limit the application of MPLS to backbone networks, and, in turn, reduce the exploitation of topological richness. Furthermore, MPLS is designed for traffic engineering in Backbone networks, hence source routing relies on *traffic trunks*, which constitute aggregates of traffic flows belonging to the same class. This approach is too coarse-grained to apply multipath delivery of a single information item.

SDN *Software Defined Networking* (SDN) constitutes a new architectural paradigm for realizing network functions, such as routing and load balancing [10]. SDN decouples the control and user planes of the networking equipment similarly to MPLS, but offers more flexible and fine-grained source routing. It logically centralizes the network intelligence (i.e. the control plane) by introducing a special network actor, named *SDN controller*, which discovers on-the-fly the dissemination paths and "encodes" them as rules to the forwarding nodes, named *SDN switches*. The controller sends to the switches explicit rules that bind certain flows to their

next-hops, thus creating virtual paths and allowing source routing. When a packet arrives at a switch, the rules indicate the outgoing port that the packet must be forwarded on. If such rule does not exist then the packet is redirected to the controller that determines the dissemination flow and installs the corresponding rules to the on-path switches. Subsequent packets of the same flow traverse the same path and avoid redirection to the controller. The "flow-to-rule" matching follows a rather flexible approach where any field on the packet header can be used as routing identifier, hence the IP address or the TCP port of a (sub)flow can be used to route individual connections over specific paths.[4]

SDN can be used to increase or even create divergence of the MPTCP paths, albeit connection-level traffic engineering is not its prime objective. In [33] the authors exploit SDN and MPLS in oder to build WAN-level testbeds that route the MPTCP traffic of multihomed end-users over different paths. Similarly, in [34] the authors present an MPTCP-*aware* SDN control plane module that detects MPTCP subflows and allocates deterministically paths to them. The module offers increased flexibility in path selection allowing shortest, k-shortest and k-disjoint paths routing, thus delivering measurable performance gains compared to the stochastic *Equal Cost Multipath* (ECMP) approach [24]. Similarly, in [35] the authors present a mechanism that enhances path diversity of MPTCP subflows in order to avoid performance bottlenecks, also supporting the creation of multiple paths for single-homed MPTCP users. MPTCP allows the establishment of multiple subflows between the same IP source-destination pairs via port multiplexing. The novelty lays in constructing different SDN routes by "sniffing" the special MPTCP message for establishing a new path (MP_JOIN), thus supporting multipath communication to single-homed users that multiplex subflows based on port numbers.

Overall, these studies provide interesting results that promote the integration of SDN and MPTCP in LANs, WANs and data center networks. Nevertheless, the number of rules on the SDN switches in large-scale networks is considered too large to perform at subflow level, hence stateful forwarding of SDN switches and

[4]The header fields that can be used as forwarding identifiers are specified by the individual SDN implementation.

MPTCP raises substantial scalability concerns. On the contrary, the processing overhead of the control plane, which includes detecting and encoding paths, is found to be acceptable, since multiple co-existing SDN controllers can be installed in large scale networks.

Overlay routing

An overlay network is a "virtual" network created on top of a physical network. The overlay network offers services that are not supported by the underlaying infrastructure, such as source routing over a hop-by-hop forwarding network. In order to force a specific end-to-end packet route, the overlay nodes act as routers that are logically connected with symbolic overlay links, thus forming a self-managed overlay topology. The symbolic links, which are carried by the underlying network, can include multiple physical links, thus increasing the scalability (global-scale overlay networks can be build upon few overlay links) but, also, reducing the effectiveness of the solution (overlay links that consist of many physical links are less managed). For instance, a low-cost wide-area overlay network can be build by few overlay nodes with symbolic links that conceal long and, most likely, overlapping paths.

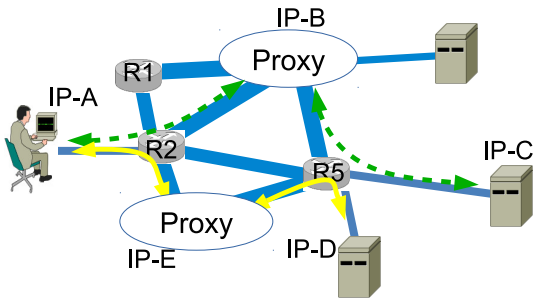


Figure 3.6: An example of multipath transfer in the TCP/IP architecture based on overlay routing. The single-homed endpoints communicate through two intermediate relay points, thus forcing multipath delivery with enhanced path diversity. Often overlay networks are combined with other techniques, such as multihoming with SOCKS servers [2] and multisource, as presented in this figure.

In application-level overlay networks, a *relay* node is used for receiving the user requests instead of the original destination node, either implicitly (on-path

relay node similar to the MPLS ingress node) or explicitly (potentially off-path relay node, similar to a Web proxy). The relay node applies the multipath routing policy of the overlay network by distributing the incoming requests over multiple overlay paths that correspond to potentially different physical dissemination paths. The number and placement of the overlay nodes highly affect the performance of multipath.

MONET The *Multi-homed Overlay Network* (MONET) [36] introduces a cooperative overlay network of peer proxies that improves the availability of Web sites across the Internet. MONET builds multiple overlay paths among the clients and the sites so as to mask failures of network routers or Web servers. A custom protocol is designed to query peers about content reachability, thus creating a list of different dissemination paths per content. This list is exploited by the *way point selection* algorithm that is used to dynamically determine the current "best" path per content based on statistics, such as path success rate and propagation delay through different peers. By pruning the large space of possible paths to a handful of the most promising ones, the complexity and, in turn, the overhead of MONET is reduced to tolerable levels.

MONET exploits application-level overlay networks to discover and realize backup paths, thus greatly enhancing service availability. The most impressive part of MONET is offering fine-grained multipath control, allowing the establishment of multiple paths based on the requirements of each specific transfer. On the other hand, the fine-grained multipath control introduces substantial scalability concerns thus limiting the application range of MONET to few Web services where availability is critical.

RON A *Resilient Overlay Network* (RON) [37] is an application-layer overlay on top of the existing Internet routing substrate that allows end-to-end communication in wide-area networks. RON acknowledges the vulnerability of the Internet to link failures and router faults and exploits multipath routing to accelerate the detection and recovery from path outages and performance degradation [37]. In a RON, the overlay nodes-routers regularly monitor the liveness and quality of

the Internet paths among themselves and use this information to decide whether to route packets directly over the Internet or over alternative overlay paths, thus optimizing application-specific routing metrics.

Although RON enhances resilience to network failures, availability and throughput to end-to-end wide-area connections, apparent weaknesses are also present. Besides the known issues of application-overlay networks, such as infrastructural overhead, reduced scalability and increased connection latency, multipath concurrent transfers are not supported in a RON, hence throughput increase is solely a result of the improved load balancing in the network. While RON can deliver multipath transport at domain-level, as singlepath transfers with the same overlay source and destination nodes can be routed over different overlay paths, concurrent multipath transport at the connection-level is not supported, thus offering only a coarse-grained multipath transport solution.

Discussion

We presented four distinct categories of solutions that address the multipath challenges posed by the TCP/IP architecture. Individually the discussed solutions either solve the establishment of multiple subflows in the singlepath TCP/IP Internet or enhance the topological diversity of the dissemination routes that is penalized by the distributed hop-by-hop IP routing. Therefore, a combination of techniques is required in order to produce a complete solution with wide scope of support and strong performance advantages, such as the integration of MPTCP and SDN. MPTCP over SDN delivers multipath connectivity to both multihomed devices, that are increasingly popular and represent a significant fraction of IP traffic [6], and singlehomed, that are the legacy IP devices, via port multiplexing, thus offering multipath to everyone. In addition, an MPTCP-aware SDN control plane amplifies the gains of multipath transport by deterministically assigning disjoint paths to MPTCP subflows, thus exploiting path richness. The only concerning weakness of this design is its questionable scalability due to the overwhelming forwarding state at the SDN switches in large-scale networks. This realization constitutes the foundation of our proposed solution that is introduced

in Section 4.

3.2.2 Multipath in ICN

Most ICN architectures come with native multipath support, thus acquiring a great advantage compared to IP networking [18]. We hereby discuss routing and forwarding mechanisms of the PSI and the *Named Data Networking* (NDN) architectures that offer inherently efficient multipath solutions [38].

In the NDN architecture the dissemination routes are instantiated by the in-network routers through the *Pending Interest Table* (PIT) and the *Forwarding Interest Base* (FIB) data structures. Following the receiver-driven paradigm, receivers emit requests for self-identified data chunks, the *Interests*, and senders respond with *Data* packets. The routing information of an NDN packet is the content name itself and forwarding is applied distributively by the on-path routers in a hop-by-hop manner, leaving the end-hosts oblivious of the formation and the number of the deployed paths. Specifically, when an NDN router receives an Interest packet, then the Interest's name is compared against a list of entries in the FIB structure that maps content names to outgoing ports, similarly to SDN yet using names as forwarding identifiers. The router will forward the Interest to the appropriate port and will insert an entry in the PIT structure that maps the content's name with the received port of the Interest, thus forming distributively the reverse dissemination path. Then, when the same router receives the corresponding Data packet, it will be able to forward it back to the requesting host using the forwarding state in the PIT table. Given that the FIB may contain multiple outgoing ports for the same content name, on-path routers can select to distribute the Interests of a single data flow among different ports, thus offering natively multipath content delivery [39]. Interestingly, the multipath transmission is concealed from the end-hosts that simply insert Interests and/or Data packets to the network, hence the on-path routers handle flow and congestion control. The efficiency of content delivery is finally enhanced by the support of on-path caching by the NDN routers, therefore exploiting the network storage resources, offloading the servers and reducing the communication delay. Overall, NDN networks offer

inherently multipath and multisource to end-users, that do not participate in the formation, the establishment or the utilization process of the routes.

The PSI architecture offers centralized path selection, source routing and on-path caching of self-identified data chunks. The communication model that is primarily advertised by the PSI architecture adopts the "channel" approach where a single source sends a stream of data to multiple receivers via a stateless multicast dissemination tree [40]. Nevertheless, different communication models that offer seamless receiver-driven multiflow transport are also supported by the network [41, 42]. PSI offers modular RV and TM operation where services exploit different strategies of content resolution and path formation [43, 44]. For instance, a multiflow-aware RV can match multiple publishers with one subscriber and the TM can discover k paths to n content sources, encode them into kn FIDs and deliver them to end-hosts, thus offering them kn distinct "manners" of transmitting data. Notice, that, the FIDs conceal the actual dissemination paths, hence the end-users are not aware of the details of the encoded path, such as the intermediate routers or the end-users. Therefore, the semantics of the FIDs are defined by the service and the end-users, thus offering a variety of content delivery patterns. For instance, in case multipath is required, then the RV allows one-to-one publisher-subscriber matching, the TM discovers multiple paths among the two hosts and delivers multiple FIDs. The applications interpret the number of FIDs as the number of different paths that can be used to reach the same destination. In case of multisource, the RV allows many-to-one publisher-subscriber matching, the TM discovers one path among the receiver and each sender and delivers multiple FIDs. However, in this case the applications interpret the number of FIDs as the number of different sources for that content. Consequently, while in-network entities undertake the discovery and realization of the multiple paths, the PSI endpoints do "read" the semantics of the FIDs and exploit the different communication paths accordingly. This scheme allows the design and implementation of traffic control policies that combine the scalability of end-to-end congestion control with the efficiency of in-network mechanisms, thus supporting effective hybrid transport solutions for the Future Internet.

3.3 Multipath congestion control

Multipath congestion control is the network operation that defines the amount of traffic that a connection can insert into the network through multiple paths. Similarly to single-path congestion control, the prime objective of multipath congestion control is to maximize the utilization of network resources, which is accomplished by filling the network links without overwhelming their capacity. Nevertheless, maximizing the utilization of network resources is more complicated with multipath transmissions since several contradictory goals must be pursued, such as maximize the aggregated throughput over multiple paths while not harming competitive single-flow transfers. Currently, multipath congestion control is an active research topic for both traditional IP networks and ICN clean-slate architectures; hence numerous studies tackle the problem from diverse perspectives [13, 45]. In the following, we present the most frequent performance requirements that are met in the bibliography:

Bandwidth aggregation: The most impressive gain of multipath is the end-to-end throughput increase. The increase presupposes that paths are exploited concurrently and complementarily, thus delivering different parts of the content in parallel. Ideally, the paths are edge-disjoint, so that the utilization of network resources is maximized, or overlapping but without a shared performance bottleneck, so that the transfer rate on each path can be summed. The bandwidth aggregation requirement motivates the adoption of multipath transport protocols as defined in [25]:
The aggregated throughput provided by a multipath connection must not be less that the throughput of a single-path connection performing on the fastest available path.

Load balancing: The exploitation of many paths must balance the traffic load across the network by allocating more traffic in the less congested paths. Assuming that the congestion level across the network is equalized, then the performance bottlenecks due to link saturation are less likely to appear, thus increasing the efficient resource utilization of the network and delivering

higher throughput to end-users [3].

Stability: Stability describes the system‘s response to dynamic traffic and status information. When path conditions change, then multipath congestion control must identify the new parameters and re-allocate the appropriate amount of traffic on each path in order to maintain high resource utilization. Kelly et al. determine an equilibrium that assures stability of multipath congestion control based on the fluid model [15].

Responsiveness: Responsiveness defines the fastness of responding to dynamic changes in the network. Although timely adaptation is critical for maintaining high performance under variable conditions, over-sensitivity can penalize the stability requirement: routing needs to respond quickly to achieve the potential benefits, but not so quickly that the network is destabilized [15].

Pareto Optimality: Pareto optimality refers to the state in which a (multipath) connection can not increase its throughput without decreasing the throughput of other coexisting (single-path) connections [16].

TCP-Friendliness: TCP-Friendliness is associated with the fair sharing of network resources among multipath and single-path flows. When a multiflow connection with N independently controlled subflows competes against a single-flow connection for *the same* bottleneck link, the multiflow connection can be up to N times as aggressive as the single-flow one [14]. While we usually say that the multiflow connection is not *TCP-friendly*, we will also use the term *friendly* to imply *single-flow friendly*, defined as follows:
When a multiflow connection competes with a single-flow connection for the same network resource, the former must not acquire a larger share of that resource than the latter.

Latency of converging to TCP-Friendliness: This term refers to the time needed to enter the state of TCP-friendliness while being TCP-unfriendly. Minimizing that period of imbalance is critical for network performance, as it ampli-

fies the effectiveness of a multipath congestion control; below this temporal boundary, the multipath connections are unfriendly.

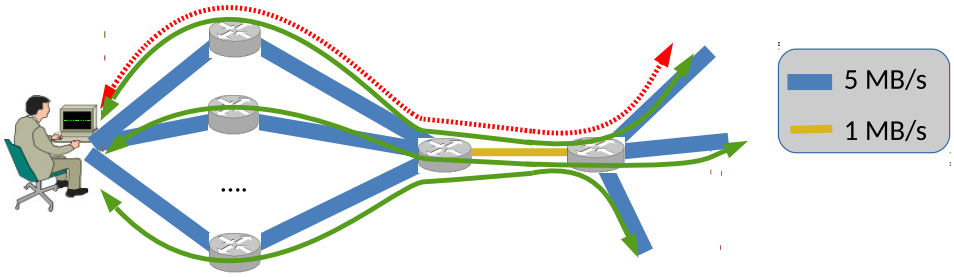


Figure 3.7: An example of the TCP-friendliness issue. The single-path connection (red dashed line) gets $1/(N+1)$ MB/s when competing in the same bottleneck (yellow link) with a multiflow connection of N subflows (green lines). Notice, that the issue is independent of the number of exploited sources, thus concerning multisource and multipath in general.

Single-path congestion control in TCP/IP

The Internet depends on transport-layer protocols such as the *Transmission Control Protocol* (TCP) [46] to provide reliable end-to-end transmission while efficiently utilizing network resources and preventing congestion collapse. Among the many transport protocols proposed since the inception of the Internet, TCP has prevailed due to it its simplicity, its low overhead and its ability to adapt to diverse network conditions.

TCP applies end-to-end acknowledgment-based congestion control that is primarily controlled by the sender. The TCP sender estimates the congestion level of the path by monitoring the evolution of the connection's *Round Trip Time* (RTT), which, in turn, is assessed by measuring the temporal difference between the transmission of a data packet and the reception of the corresponding acknowledgment (ACK). Congestion is detected through the loss of a packet, which is presumed through the lack of the reception of an ACK within the set *timeout* interval, which is dynamically estimated based on the RTT. Therefore, the transfer rate is increased to enhance resource utilization as far as ACKs are received in time and is reduced to handle congestion when an ACK is not received

within the timeout. The amount of data entering the network is also controlled by the TCP sender via the *congestion window* of the connection, which is the cornerstone of TCP's congestion control. Numerous congestion control flavors have been proposed for TCP so far [47], each one introducing novel approaches to manage the size of the congestion window. Here, we only discuss the standardized TCP flavor that is presented in [48] for it constitutes the basis of most multipath congestion control proposals. This flavor includes three distinct congestion states that differentiate the management of the congestion window, namely, the *Slow Start* (SS), the *Congestion Avoidance* (CA) and the *Fast Recovery* (FR) states.

Slow Start: The SS state is used to quickly explore the amount of available network resources. It introduces a rapid throughput increase pattern which doubles the amount of data entering the network each RTT. In order to control the aggressiveness of SS a special variable is introduced as a *Slow Start Threshold* ($ssthresh$), hence a TCP connection is in SS if and only if the congestion window is smaller than the $ssthresh$. The $ssthresh$ variable is re-estimated after each timeout when it is set to half the size of the congestion window before the congestion event, thus allowing faster recovery from congestion collapses. During SS, the growth of the congestion window (w) upon the receipt of an ACK is:

$$w_{n+1} = w_n + s \tag{3.1}$$

where s is the *Maxiumum Segment Size* (MSS) of the network.

Congestion Avoidance: The CA state aims at providing high throughput by performing longer near the link saturation point. Therefore, it introduces a slow yet diligent throughput growth scheme which increases the amount of data entering the network by one MSS per RTT. A TCP connection is in CA when the congestion window is at least $ssthresh$. During CA, the congestion window growth upon the receipt of an ACK is:

$$w_{n+1} = w_n + s/w \tag{3.2}$$

Fast Recovery: The FR state aims at generating timely information that allows faster recovery from losses. In addition to timeouts, which can be time consuming, individual packet losses are used as congestion events, assuming that packets are dropped by the on-path routers due to overflown buffers. Therefore, the transfer rate is also reduced when a packet delivery is pending but the delivery of the latter three packets is verified, an event called *Triple Dublicate* (3DUP). After a 3DUP event the congestion window is configured as follows:

$$w_{n+1} = w_n/2 + 3 * s \tag{3.3}$$

In addition to window reduction, a *Fast Retransmission* is emitted, which is a second request for the missing packet, thus allowing the TCP connection to recover timely and effectively from individual packet losses.

3.3.1 Multipath congestion control in TCP/IP

The widespread availability of path diversity on the Internet, along with the proliferation of multihomed mobile devices and datacenter servers, argue for the extension of TCP with multipath features, so as to improve throughput, resource pooling, load balancing and resilience to network failures. This issue is addressed at the transport layer by *Multipath TCP* (MPTCP) [12, 23], an extension of TCP that allows the deployment and management of multiple TCP-like subflows among two end-hosts. Being aware of the available set of subflows, MPTCP can control the cumulative transfer rate and jointly tackle performance and TCP-friendliness. Several congestion control algorithms have been proposed so far [3, 16, 49, 50, 17, 51]. Representing an evolution of single-path TCP, the majority of these approaches rely on the well-known TCP building blocks [48], but also address some multipath-specific issues, such as TCP-friendliness [52], responsiveness and stability [15], load balancing [3] and pareto optimality [16].

The simplest multipath congestion control algorithm for MPTCP, known as *Uncoupled*, introduces subflows with individual congestion windows and independent window management. As expected, this design offers enhanced throughput and fast adaptation to network conditions, but tends to be overly aggressive to-

wards singlepath connections, thus presenting serious TCP-friendliness issues. To avoid these problems, *EWTCP* [49] splits traffic "evenly" among subflows so as to cumulatively grasp the same share of resources as a regular TCP connection. As a result, EWTCP often does not fully utilize the available network resources, since the proportional management of the subflows disregards the particular properties of the dissemination paths.

The *Coupled* algorithm [50] was the first design that emphasized the importance of being TCP-friendly while shifting traffic towards the least congested path. It handles the available paths as a pool of resources where the congestion level is balanced and utilization is increased. However, pushing all traffic to the least congested path has other shortcomings, such as performance degradation with mismatched paths and poor responsiveness to network changes. The *Linked Increase Algorithm* (LIA) [3] was developed to tackle both TCP-friendliness and responsiveness. LIA pushes traffic to the least congested path so as to enhance load balancing similarly to Coupled, but also introduces an aggressiveness parameter that attempts to keep a moderate amount of traffic in the more congested paths in order to be responsive. This parameter is based on two equilibrium conditions: first, LIA balances the congestion window increases and decreases at steady state in order to be stable and, second, it equalizes the resource shares of MPTCP and TCP in the bottleneck link in order to be TCP-friendly. Presenting sufficient friendliness and resource utilization, LIA soon became the point of reference for congestion control of MPTCP.

While favoring the least congested path, LIA does not push traffic exclusively there, thus penalizing the overall network resource utilization under certain conditions. The *Opportunistic Linked Increase Algorithm* (OLIA) [16] extended LIA in order to enhance resource pooling, while maintaining high responsiveness. Specifically, OLIA increases faster the congestion window of subflows with a high transfer rate but relatively small windows. Moreover, OLIA introduces minimal probing traffic over the worst paths to achieve sufficient responsiveness. Nevertheless, recent studies [17] show that OLIA does not respond well in case of abrupt load changes. In the same paper, the authors present the *Balanced Link Adapta-*

tion (Balia) algorithm, a generalization of existing algorithms that strikes a good balance between friendliness, responsiveness and window oscillation.

Finally, *weighted Vegas* (wVegas) [51], a delay-based congestion control algorithm inspired by TCP Vegas, uses queuing packet delay to detect congestion, unlike other proposals which exploit time-out timers. The main benefit of wVegas is quick traffic shifting, as it can be more sensitive to changes in network load. However, tuning the algorithm's sensitivity is not trivial and the investigation of its behavior is not complete, for example, the handling of RTT variation in case of rerouting is questionable.

Greedy Friendliness

The TCP-friendliness constraint is roughly inversely proportional to responsiveness and, in turn, resource utilization [17], therefore generating a friendly algorithm that optimizes all constraints is out of the question. However, we can simultaneously enhance responsiveness, resource utilization and TCP-friendliness through *greedy friendliness*, a concept where a multipath considers TCP-friendliness only when fairness towards singlepath flows is jeopardized. By definition, TCP-friendliness is an issue in shared bottlenecks, where more than one subflows compete with a singlepath connection, therefore greedy friendliness can rely on the verified detection of shared links, allowing the multipath connections to disregard the TCP-friendliness constraint in disjoint paths.

MPTCP's conservative approach to friendliness is imposed by the IP routing architecture. Due to the distributed, hop-by-hop routing of IP networks, a transport protocol cannot reliably conclude whether the dissemination paths are overlapping or not, therefore its congestion control module cannot detect whether friendliness is an issue or not. There are some solutions for end-to-end detection of shared bottlenecks in the literature [53, 54], but their efficiency is debatable. In [53] the authors detect shared bottlenecks based on the temporal correlation of fast-retransmit packets, while in [54] the authors evaluate both loss-based and delay-based correlation techniques, arguing that the loss-based technique is unreliable, while the delay-based methods require considerably more time for accurate

results; even the loss-based method requires roughly 15 s to converge, which is significantly high for a general purpose multiflow protocol.

A recent proposal [55] exploits the propagation delay of subflows to detect shared bottlenecks, specifically for enhancing MPTCP performance. In this work, the authors measure the path propagation delay instead of the RTT, in order to avoid the noise of the return path and identify bottlenecks more accurately. They argue that a sampling period of 3.5 s is adequate for detecting bottlenecks with 97% mean accuracy. Interestingly, they conclude that the adaptation of aggressiveness to the path formation can deliver up to 40% higher throughput to end-users.

Modeling multipath rate control

One of the key principles in modeling MPTCP is the exploitation of the fluid model analysis by Kelly and Voice [15]. The fluid model directs the transfer rate adaptation of a TCP-sender during the CA phase to ensure system stability, protocol responsiveness and fairness. It can be adapted to the window-based congestion control design to deduce a sufficient condition (or equilibrium) for the desired amount of window increase and decrease upon the receipt of an ACK and a congestion event, respectively.

LIA tries to balance the "overall" window increase and decrease in the long-run, by modeling them as the product: *event probability* $\times$ *window modification*; for example, the total window decrease equals the product of the segment loss probability and the size of the window reduction. To solve the equation, authors assume that segment loss probability is statistically negligible - a simplification that can be responsible for LIA's poor performance in paths with error and delay mismatch. Consequently, the algorithm offers stability, friendliness and high resource utilization in steady steady as far as the error rate is low.

A second common feature adopted by the fluid model is the omission of the SS phase from the model, as it is considered a transient state with no measurable effect on the long-term performance of the protocol. Consequently, multipath connections are not TCP-friendly for a "brief" period after they are launched, gradually converging to the desired fair equilibrium. To the best of our knowl-

edge, the friendliness of MPTCP congestion control algorithms has been evaluated only in long-term performance (usually mentioned as "long-lived flows"), but the efficiency of reaching it and its correlation with network conditions has not been explored yet.

We should highlight here that responsiveness and convergence to friendliness are different concepts: the former refers to the efficiency of shifting traffic among subflows while being TCP-friendly; the latter refers to the time needed to enter the state of TCP-friendliness (while being TCP-unfriendly). Estimating that "brief" period of imbalance is critical for network stability, as it defines the range of MPTCP's TCP-friendliness; below this temporal boundary, MPTCP connections are unfriendly.

3.3.2 Multipath congestion control in ICN

In ICN networks the IP paradigm is replaced with information-based routing and forwarding mechanisms that support natively multipath, multisource and multicast. The in-network entities, such as routers and PSI's TM, are important actors in the ICN paradigm that attempts to exploit the storage and computational resources of the network too. Therefore, it is often proposed that network actors, that are aware of both transport flows (indicated by unique content names) and adjacent links, should assist in the realization of congestion control. For a complete presentation of the proposed solutions in ICN readers are referred to [38, 45]. Hereby, we discuss some indicative multiflow designs, exploring the extent of network participation in congestion control.

We detect three categories of congestion control designs in ICN architectures with regard to their placement: end-to-end, in-network and hybrid. In the first category, the end-points, usually the receivers, undertake the task of detecting the congestion level of the exploited paths and accordingly control the rate of data that enters the network. The congestion control mechanism is rather similar to the end-to-end TCP approach, introducing RTT-based congestion detection and window-based flow control. Nevertheless, this design is not compatible with the on-path ubiquitous caching of ICN, where the performance of the end-to-end

solutions is penalized by the RTT variance due to cached responses. Specifically, packets arriving from on-path caches exhibit lower delays than packets arriving from the content source, thus causing trouble to traffic control protocols that use RTTs as congestion indicators. We discuss this problem in detail in Section 6.2.

In-network operation is considered to strengthen congestion detection, thus introducing a hybrid design that consists of end-to-end flow control and in-network congestion detection. A basic instance of this design suggests that in-network routers emit explicit congestion reports, or *Explicit Congestion Notifications* (ECNs), to the end-hosts that adjust their sending rate accordingly [39]. A more developed solution in [56] exploits the enhanced role of NDN routers and proposes that in-network routers also participate in flow control, thus undertaking congestion detection and flow control duties. Although flow control and part of congestion control is still managed by the receiver, in-network congestion control is present in the form of dynamic request forwarding; intermediate routers choose on-the-fly the most appropriate interface to forward each packet, shifting flows to less congested parts of the network.

Finally, pure in-network congestion control solutions are proposed in an effort to maximize the exploitation of the network's computational resources and knowledge. For instance, the *Hop-by-hop interest shaping* approach suggests that each NDN router can detect and adjust the forwarding rate of Interest packets and, in turn, the transfer rate of returning Data, thus realizing (in-)network congestion control. In [57], traffic control is exclusively assigned to in-network nodes that have the authority to drop packets and even reject the establishment of new connections, based on link utilization and fairness constraints. In particular, each in-network router maintains a per-flow queue with the *Deficit Round Robin* (DRR) scheduling policy to determine which packets must be dropped and/or connections must be rejected. According to link conditions, the routers direct the rate of data entering the network by sending notifications to the "uncomplicated" end-users, that conform to explicit congestion signals, such as ECNs, albeit maintaining the control of the congestion window.

While introducing powerful tools that enhance the performance of the net-

work and the users, the aforementioned approaches also exhibit apparent weaknesses. First, the end-to-end congestion detection is challenged by the inherent multisource of ICN networks, where the communication "end" is not directly defined. The problem is emphasized by the NDN architecture where source selection is dynamic; typically, the routers apply dynamic routing policies, hence the number and location of the sources is not fixed throughout the transmission. Second, the end-to-end designs in NDN can be inefficient since the endpoints are unaware of the actual dissemination routes; the realization of paths is a distributed in-network operation, hence endpoints can not control the paths individually. Overall, the end-to-end congestion control model is considered inappropriate for the NDN architecture where multipath is coordinated and carried by in-network entities. In theory, the in-network operation constitutes the best option, but, in practice, poses an undeniable weakness: scalability. The state-full NDN routers face significant overheads, such as the estimation of link utilization for congestion detection in [56, 57] and the additional per packet state for fair queuing in [57], thus questioning wire speed operation [58, 59].

Experience from existing protocols and designs, such as SDN, CDNs and MPTCP, indicates that the Future Internet must consider hybrid solutions that combine the end-to-end scalability with essential yet lightweight network operation. Thereupon arises a PSI-based approach that enriches end-to-end congestion control with topological information via a scalable in-network notification service. The notification service, which is aware of network topology and participates in the path discovery process, can implement special path formation policies and deliver essential information to the endpoints, thus allowing them to practice congestion control more effectively. This design can provide accurate information (without convergence delay) to the end-users and does not stress the in-network routers, which are the weaknesses of the IP and NDN solutions, respectively. We elaborate on this perspective in the next section, where we present the first complete solution of network-assisted multiflow congestion control in the PSI architecture.

Chapter 4

Proposed multipath solution

In this section we present our proposal for multipath transport which consists of a multipath transport protocol and a multipath congestion control algorithm. The transport protocol allows the seamless establishment of multiple dissemination flows, while the congestion control algorithm offers efficient exploitation of those flows.

4.1 Multisource and Multipath Transport Protocol (mmTP)

The PSI architecture is an instantiation of the ICN paradigm that pushes the network layer higher in the stack, allowing it to understand what it transports and what the transport context is. PSI supports data multihoming, as well as path selection at the granularity of transfer sessions. We exploit these architectural features to design the *multipath and multisource Transport Protocol* (mmTP) [60, 61, 62], a sophisticated reliable transport protocol for PSI which (a) enables multisource and multipath transfers, (b) does not require complicated network operation or signaling, (b) provides modular path management (exploits paths in parallel or sequentially, redundantly or complementarily), (c) offers modular end-to-end congestion control, and (d) supports hybrid congestion control, a scheme where end-to-end congestion control exploits an in-network mechanism to

support TCP-friendliness efficiently.[1]

4.1.1 Protocol overview

Our primary goal in mmTP is to utilize all available network resources in the PSI context, i.e. communication, data storage and computation. mmTP achieves this goal via the following design choices:

Self-identified data packets: Content that goes beyond the *Maximum Transfer Unit* (MTU) of the network is fragmented into data packets, or *chunks*, that are assigned a statistically unique identifier, so as to allow packet-level caching and simplify content delivery and reconstruction. mmTP uses algorithmic identifiers, that indicate the order of the packet in the content, thus enriching the semantics of the naming scheme and, in turn, supporting simple loss detection, efficient caching designs [63], *Out-of-Order Delivery* (OODs) and more.

Receiver-driven operation: The receiver (subscriber) coordinates data transmission by sending explicit requests for self-identified data packets to the stateless sender (publisher) that simply responds to incoming requests. The receiver-driven approach of mmTP together with the self-identified packets exploits on-path caching and allows the implementation of various path scheduling policies in case of multipath and multisource.

Multisource & Multipath delivery: Content is retrieved from multiple locations simultaneously by sending requests to multiple publishers. This increases the utilization of the network's available bandwidth and enhances mmTP's resilience to node and path failures. Additionally, if there are multiple paths between the receiver and a specific source, mmTP further utilizes available resources by transmitting data via all those paths.

[1] Even though TCP is not central in such an environment (and may not even exist at all, e.g., no PSI TCP implementations exist or are planned as fas as we know), it is important to provide the behavior, which seems a very good property for networks to support.

Centralized path selection: mmTP relies on PSI's Rendezvous function for locating multiple sources and on its Topology Management function for computing available paths, thus utilizing in-network computation resources. In addition, centralized path selection allows the implementation of advanced traffic engineering techniques, such as *Differentiated Services* (DiffServ) and mmTP's hybrid congestion control approach.

Hybrid multiflow congestion control: The congestion control scheme of mmTP combines the strong features of end-to-end and in-network congestion control designs, thus constituting a hybrid solution. First, mmTP provides modular end-to-end mechanisms that monitor the level of congestion, adjust the transfer rate on each path and offer reliability. These mechanisms aim at maximizing link utilization, while avoiding congestion collapse and starvation of single-flow traffic. Placing this functionality at the network edges, similarly to the TCP/IP architecture, provides enhanced resilience and scalability, since the computational cost is distributed to the numerous end-users instead of the few in-network nodes.

Second, mmTP introduces an in-network mechanism that helps handling the TCP-Friendliness constraint. Specifically, the network service, that is embedded in the Topology Management service, can participate actively in congestion control by selecting paths, e.g., only disjoint paths, or passively by informing the mmTP users about the existence of shared bottlenecks that can impair TCP-Friendliness, hence the users can adjust their aggressiveness accordingly, a scheme that we call *greedy friendliness*. Although the support of the hybrid congestion control pattern is purely a feature of mmTP, we discuss its design and performance advantages in depth in the following section where we elaborate on our congestion control design.

Out-of-order delivery: Experience with MPTCP shows that requiring in-order delivery of data packets can penalize performance when paths are RTT-mismatched due to the head-of-line blocking issue [64, 65]. We allow mmTP to transmit out-of-order packets in order to simplify path scheduling even

though data chunks are requested in ascending order. The relaxation of the in-order requirement does not penalize the efficiency of the protocol, since mmTP exploits algorithmic packet identifiers to deduce the position of each packet in the object, thus supporting inexpensive packet reordering and object reconstruction.

4.1.2 Protocol description

mmTP operation is split in two phases, as shown in Fig. 4.1: the *slow-path rendezvous*, which deals with service establishment, and the *fast-path rendezvous*, which deals with the immediate host interaction for content delivery. In the slow-path rendezvous, content sources and receivers emit publications and subscriptions, respectively, about the desired information item. The publications are first routed to the network's RN (Fig. 4.1(a)) and then the subscriptions are similarly routed to the RN (step 1 in Fig. 4.1(b)) where they are matched with the publications. When a match occurs, the RN requests the TM to compute paths between the aforementioned publishers and the subscriber (step 2 in Fig. 4.1(b)). The TM, having a complete view of the network, computes multiple paths for each source-receiver pair and constructs the LIPSIN FIDs for both the reverse direction, i.e. subscriber-to-publisher(s), and the forward direction, i.e. publisher(s)-to-subscriber, since LIPSIN FIDs are unidirectional. Finally, the TM sends the FIDs to the subscriber (step 3 in Fig. 4.1(b)). At this point, the subscriber has obtained two sets of source-routes. One points to publishers holding the requested information item (reverse FIDs) and the other can route data from the publishers to the requesting host (forward FIDs). Note that there is no strict one-to-one mapping between source-routes and publishers; some FIDs may point to the same source if the TM decided to use the multipath capability for that particular source.

In the fast-path rendezvous, the receiver starts sending subscriptions for individual data packets to the publisher(s) using the reverse FIDs (Fig. 4.1(c)). These are fast-path subscriptions, that is, they are sent directly to the sources, bypassing the rendezvous system. Each request also carries the forward FID that encodes the path that should be taken by the response. When these subscriptions

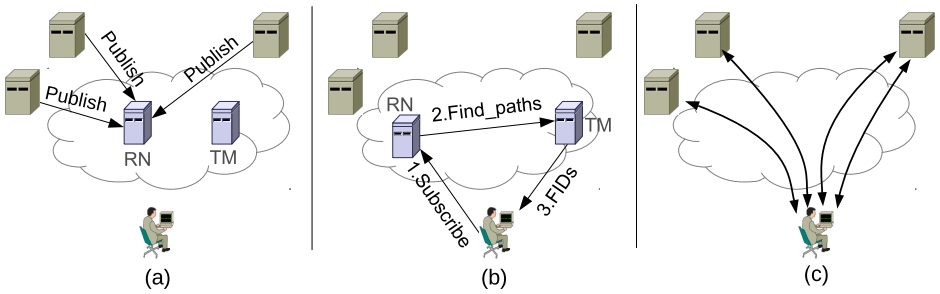


Figure 4.1: mmTP operation phases: (a-b) slow-path rendezvous, where connection is established, and (c) fast-path rendezvous, where data transfer takes place.

reach the publisher(s), the requested data packets are transmitted back to the subscriber using that forward FID.

Following ICN principles, a statistically unique identifier is assigned to each data packet. This identifier is a combination of the content's name and a counter denoting the packet's position in the content. mmTP receivers use this identifier for issuing requests that concern a specific data packet. Sources do not need to maintain transport state, since each request is self-contained: it includes an identifier for the desired data and the FID needed to return these data. Fast-path subscriptions can also be satisfied by on-path caches; an on-path router that opportunistically caches packets can inspect fast-path subscriptions and respond immediately if the requested packet is locally stored.

The first fast-path subscription in mmTP always concerns a metadata packet. This packet provides a simple description of the content, along with information such as the content's size, the number of packets in it and a hashcoded string for integrity validation. These metadata easily fit into a single Ethernet packet, being less than 1 KByte. The metadata packet is named after the content, with the reserved *mm* suffix, inherently declaring that multiflow transmissions are supported. Given that mmTP sources are stateless, the information carried by the metadata packet is utilized by the subscriber for managing the communication. Although the actual data transfer begins after the metadata packet is fetched, simultaneous transmission of the first chunk is supported to reduce the temporal overhead during connection establishment.

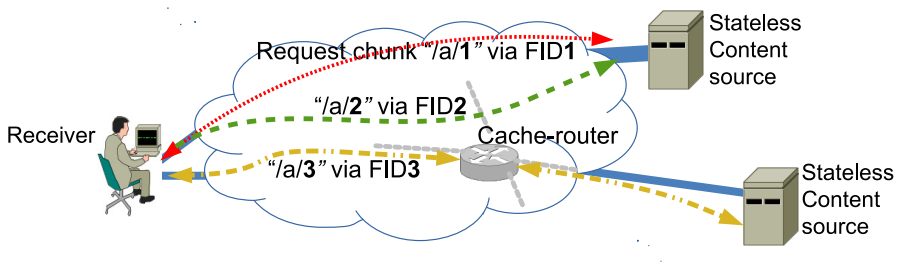


Figure 4.2: mmTP operation during fast-path rendezvous. FIDs correspond to different paths to the stateless content sources. Data chunks are self-identified through algorithmic identifiers in the form of "/*content_name*/*chunk_ranking*", i.e. "$/a/1$" is the 1^{st} chunk of content "a". The *pull* transfer model and the named data chunks allow distributed on-path chunk-level caching.

Flow, error and congestion control

In mmTP, all control mechanisms are applied by the subscriber. Upon the receipt of the LIPSIN FIDs, the subscriber creates a distinct subflow for each supplied path and initializes a congestion loop per path, including a congestion window (w) and a retransmission timer. Subflows request packets in ascending order: a subflow requests the next *idle* packet, i.e. a packet not already requested by another subflow.

Using fast-path subscriptions, the receiver requests one packet per subflow and maintains a timer from the time the subscription was sent until the time the publication arrives.[2] If the requested data packet does not arrive on time, suggesting that either the request or the data packet was lost, the subscriber re-issues the fast-path subscription. Since fast-path subscriptions are self-contained, the lost subscription can be sent over a different path if two subscriptions for the same data packet expire. Loss detection is RTT-based, estimating a *Retransmission Timeout* (RTO) per subflow, similar to that in TCP [66]. We ignore requests retransmitted over the same path when updating the RTT estimator, as in TCP, since it is unclear which packet corresponds to each request. Contrary to TCP, we do not drop the delivered packets that follow a loss thus gaining a slight performance advantage, albeit we do not increase w to maintain accurate estimation of the on-the-fly data. Finally, we do not overlook that RTO estimation can be "poi-

[2]The actual implementation of mmTP uses one timer per subflow, similarly to TCP [48].

soned" by on-path packet-level caching. The ICN end-users, who are not aware of the packet source, misinterpret the quick responses of on-path routers as congestion withdrawal, thus estimating mistakenly low RTOs. Even though studies argue that on-path packet-level caching in ICN is mostly used for error recovery [67], thus having insignificant impact on congestion control, in Section 6.2 we discuss solutions that prevent RTO-poisoning in the presence of on-path packet-level caching.

Finally, the congestion control algorithm of mmTP is modular, hence numerous end-to-end congestion control algorithms, such as the LIA, OLIA or BALIA, can be seamlessly integrated to mmTP. Currently the management of the congestion window follows the Uncoupled congestion control policy, including TCP's basic blocks, namely, the SS and CA states.

Path scheduling

When paths are concurrently exploited, the distribution of packet requests among sources and/or subflows, also known as *path scheduling*, can be critical to protocol's performance. For instance, MPTCP can loose efficiency when performing in paths with RTT-mismatch due to the head-of-line blocking issue, hence several studies propose sophisticated path schedulers with apparent gains and weaknesses [68, 65]. On the other hand, mmTP relaxes the in-order delivery requirement without apparent costs, since data chunks (and requests) are self-identified via algorithmic identifiers that determine explicitly their position in the object. The algorithmic identifiers are exploited during error recovery for detecting and retransmitting lost packets, as well as during object reconstructure. Thereafter, mmTP does not require a centralized path scheduler module, allowing distributed path and source selection as described in sequence. A similar solution is presented by the authors of MP-RDMA [69], that endorse the exploitation of packet sequence numbers and allow OODs. However, their approach also tries to eliminate OODs (by pruning slower paths) so as to minimize the memory footprint of the protocol that is stored in the tiny NIC memory, therefore solving a more complex problem and slightly compromising throughput for saving space.

When the connection starts, the receiver requests one packet per subflow,[3] since the individual path characteristics are unknown. The next packet is always requested by the first subflow to deliver a packet, provided its congestion window allows it. To achieve this, we construct an array where we project the congestion windows of *all* subflows. Each position in this array contains the state of the corresponding packet: $IDLE$ for not yet requested packets, $DOWNLDED$ for already delivered packets and $REQUESTED\_i$ for a packet requested by subflow i but not delivered yet. A subflow executes Algorithm 1, given below, upon each packet delivery to determine the next packet to request.

Algorithm 1 Path selection.

```
1:  procedure REQUEST_PACKETS
2:      i ← win_start
3:      other_reqs ← 0
4:      while i < (win_start + w + other_reqs) do
5:          if (pkt_state[i] == IDLE) then
6:              request_packet(i)
7:              pkt_state[i] ← REQUESTED_as
8:          else if (pkt_state[i] == DOWNLDED) then
9:              if (i == win_start) then
10:                 win_start ← win_start + 1
11:             else
12:                 other_reqs ← other_reqs + 1
13:             end if
14:         else if (pkt_state[i] ≠ REQUESTED_as) then
15:             other_reqs ← other_reqs + 1
16:         end if
17:         i ← i + 1
18:     end while
19: end procedure
```

[3]In the mmTP implementation the number of transmitted paths is equal to the minimum allowed size of the congestion window.

In this algorithm, the *win_start* variable marks the position of the first packet in the sliding window, the *other_reqs* indicates the number of packets in the window that have been requested by other subflows and the mark $REQUESTED\_as$ is the identifier of the current subflow. The first two variables are used to calculate the actual size of the sliding window, while the third is the instantiation of $REQUESTED\_i$ for that particular subflow. An example is presented in Fig. 4.3, which displays a fraction of the state array when subflow 1 has just received a packet and has inflated its window to 4 MSS. Initially *win_start* is 13 so, in the absence of other flows, flow 1 would request packets 13 to 16. As the array is scanned, the *other_reqs* variable is incremented due to packets downloaded or requested by other flows. The algorithm stops when $i = win\_start + w + other\_reqs = 14 + 4 + 3 = 22$, therefore flow 1 requests packets 20 and 21, in addition to the already requested 16 and 17.

The distributed path scheduling of mmTP is rather simple and effective, since each subflow explicitly acquires a packet request when it can carry an additional packet. The exploitation of a single packet state array also facilitates the retransmissions over different paths without the need for a dedicated scheduler, which is the case of the MPTCP implementation in Linux. Nevertheless, our design can become a performance bottleneck in case of paths with high bandwidth-delay product, where the congestion windows grow large and, in turn, the number of slots that need to be checked in the packet state array increases immensely. Several fixes can alleviate the impact of this performance bottleneck, the most light-weight and transparent solution is to use a global variable, namely, *next_idle* that holds the smallest id of an $IDLE$ packet, and a subflow local variable, namely, *pending* that indicates the number of on-the-fly packets of the subflow. The variables are used to directly access the packet state array, thus significantly lessening the processing overhead of path scheduling. For instance, when a subflow receives a packet it subtracts the *pending* variable from the w variable in order to infer the number of packets that can be requested. In case the result is positive, hence a new packet needs to be requested, it requests the packet at *next_idle* position, instead of parsing the cells sequentially, and then recomputes the *next_idle* variable. In

13	14	15	16	17	18	19	20	21	22
R_2	D	R_2	R_1	R_1	R_2	R_2	IDLE	IDLE	IDLE

Figure 4.3: An example of the packet state array. R_i denotes a packet requested by subflow i and D marks a downloaded packet. $IDLE$ denotes that packet remains to be requested.

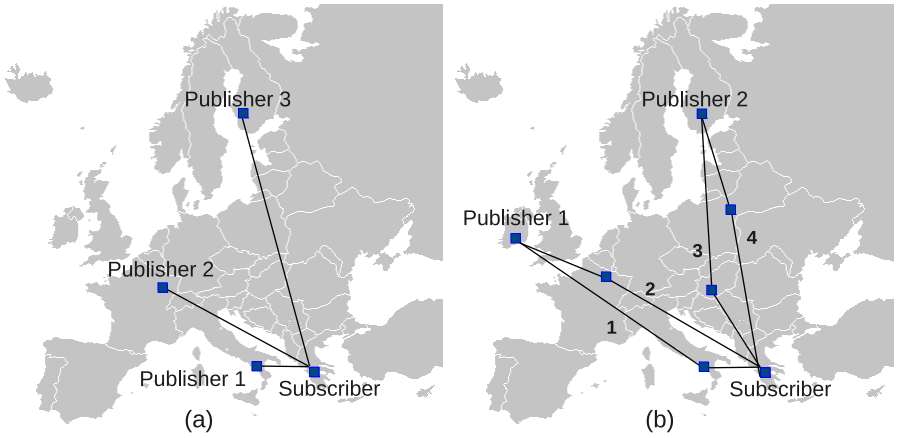


Figure 4.4: PlanetLab overlay topologies that allow (a) multisource with three content sources and (b) multiflow with two content sources and two paths to each source.

our experiments we found that this modification enables high performance under various path conditions, albeit sequential parsing of cells is not always avoided.

4.1.3 Implementation and experimentation

We implemented mmTP over Blackadder, the PSI prototype implementation [70]. Our implementation includes the mmTP sender and receiver applications, as well as a TM that can compute multiple paths between two nodes. Our TM computes the k-shortest paths from every publisher towards the subscriber, using the algorithm by Yen [71] with hop count as the metric.

For experimental purposes, we deployed Blackadder with mmTP on the PlanetLab testbed that spreads throughout Europe. We chose PlanetLab in order to evaluate our design in a realistic environment with actual propagation delays, forwarding overhead and competing traffic. The deployment is realized as an overlay network: a set of Blackadder nodes scattered across Europe (Fig. 4.4),

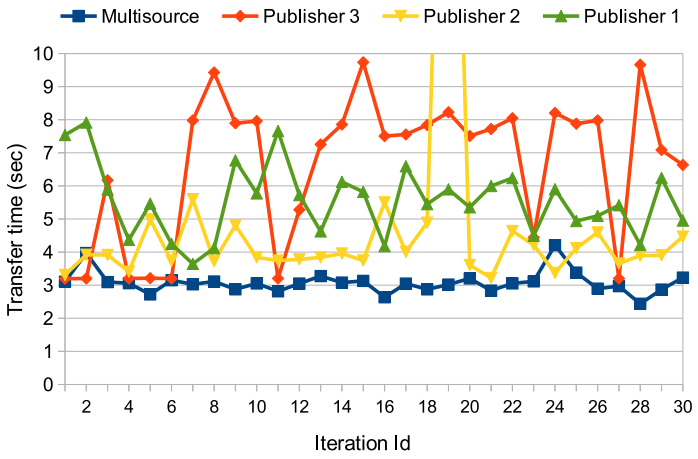


Figure 4.5: mmTP performance over PlanetLab with three single-source transfers and a multisource with three sources in 30 experiments.

communicating via UDP tunnels. We examined two network topologies and several transfer schemes (single-path, multisource and multiflow), so as to assess the gains from multipath and multisource content delivery in terms of performance, resilience, and load balancing.

Performance gains with multisource

Our first scenario examines the bandwidth gains that can be achieved when a subscriber downloads a 12 MB file from 3 publishers, using the topology in Fig. 4.4(a). The experiment consists of four phases: the subscriber (located in Greece) first downloads the file in single-source mode from each publisher (three separate downloads) and then it downloads the file from all three publishers in multisource mode. As congestion in the PlanetLab testbed is unpredictable, we performed 30 iterations of the experiment, resulting in 120 transfers. Figure 4.5 shows the transfer time for each iteration of the experiment. The best performance corresponds to the multisource case, with an average download time of 3.07 s (equivalent to 3.9 MB/s). The best single-source performance is achieved with Publisher 2 (located in France), where the average download time is 4.8 s (equivalent to 2.5 MB/s).

As evidenced by the spikes in Fig. 4.5, mmTP is much more stable in

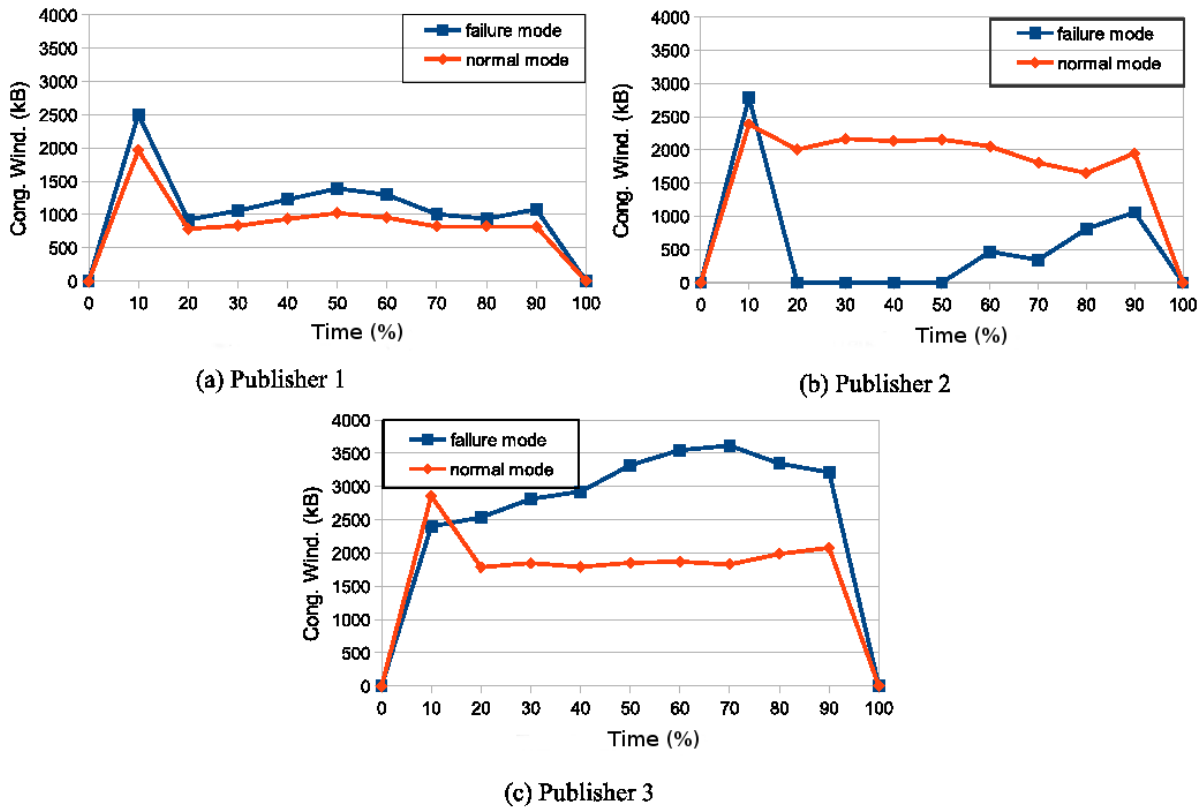


Figure 4.6: mmTP performance over PlanetLab with multisource in normal mode, where no sources fail, and failure mode, where Publisher 2 fails for 7 s. Each plot depicts the size of the congestion window (of a subflow) to a specific source.

multisource mode: the variance of the download times for our 30 iterations was only 0.1 in multisource mode, while in single-source mode the variances were 1.13, 15.75 and 4.16 for Publishers 1, 2 and 3 respectively. This is due to the adaptation of the receiver to the prevailing network conditions: mmTP dynamically avoids the paths that exhibit congestion, a situation that we often met in the PlanetLab testbed.

Resiliency to node/path failures

Our second scenario investigates mmTP's robustness to path failures. We downloaded a 50 MB file in multisource mode using the topology of Fig. 4.4(a) and emulated path failure by shutting down Publisher 2 during the file transfer. Specif-

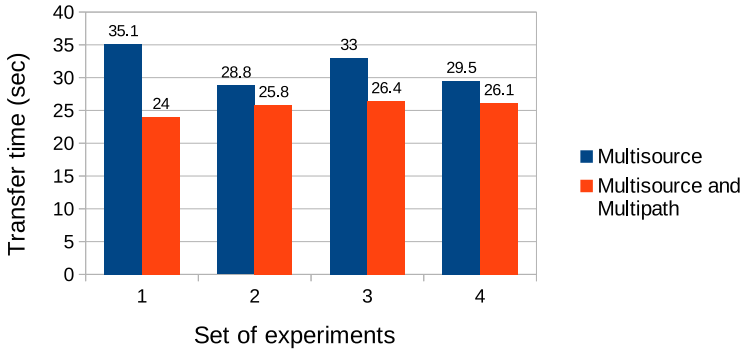


Figure 4.7: mmTP multisource performance over PlanetLab with and without multipath in 4 experiment sets of 10 runs each.

ically, Publisher 2 was programmed to stop responding to all packet requests at a certain time, remain idle for a period of 7 s and then return to normal operation. To assess the impact of path failure to mmTP, each iteration of the experiment consisted of one download with path failure and one without, and we performed 20 iterations.

Figure 4.6 shows the average size of the congestion window of each subflow over time in the *normal* and *failure* mode. In normal mode, the file is downloaded from all three sources in 12.2 s (equivalent to 4.09 MB/s). In failure mode, Publisher 2 fails at approximately 20% of the transfer duration and resumes at 60%. During that time interval, he mmTP receiver automatically switches to Publishers 1 and 3, increasing their download rates according to their path capacities: the download rate of Publisher 1 is increased by 76.5%, while the increase observed at Publisher 3 is approximately 30%. As a result, the average download time in the failure mode increases by only 2.1 s to 14.3 s (equivalent to a 0.6 MB/s drop to 3.49 MB/s), despite a failure in the highest capacity path (to Publisher 2) during half of the transfer. Even though the efficiency of exploiting the dissemination paths is primarily a property of the protocol's congestion control algorithm, we validate that the mmTP protocol provides the necessary context for enhancing resilience to node/path failures via multiflow.

Additional performance gains with multipath

Our third scenario investigates the additional performance gains due to multipath transmission. For this scenario we used the topology shown in Fig. 4.4(b), where the receiver can use two disjoint (overlay) paths towards each of the two available sources. In each experiment the receiver first downloaded a 50 MB file in multisource mode using a single path per source (paths 1 and 3 in the figure), and then it downloaded the same file in multisource and multipath mode, exploiting all four paths, repeating this pattern 40 times (80 downloads).

In this scenario we highlight that throughput and performance stability is enhanced by the additional subflows. Therefore, we split the experiment set into 4 (sub)sets based on the order of execution (1^{st} set contains runs 1-10, 2^{nd} set contains runs 11-20 and so on) and we plot 4 average scores, thus unveiling the performance deviation of each transfer mode due to the unpredictability of PlanetLab testbed. Figure 4.7 shows the average download times for each subset. The average download time when using multipath was reduced in each repetition by 31.5%, 10.4%, 19.9% and 11.6%, respectively, for an overall gain of 17%. This is due to mmTP's ability to effectively avoid bottleneck links, utilizing the least congested paths. From our experience with the PlanetLab testbed, which suffers from large bursts of congestion, the exploitation of *backup* paths allows traffic to be switched as needed, hence multipath transfers lead to better download rates. In addition, multipath transfers exhibit less variation than single-source ones, a pattern also observed in the single-source vs. multisource comparison, further increasing the stability of mmTP.

4.2 Hybrid multi-flow congestion control

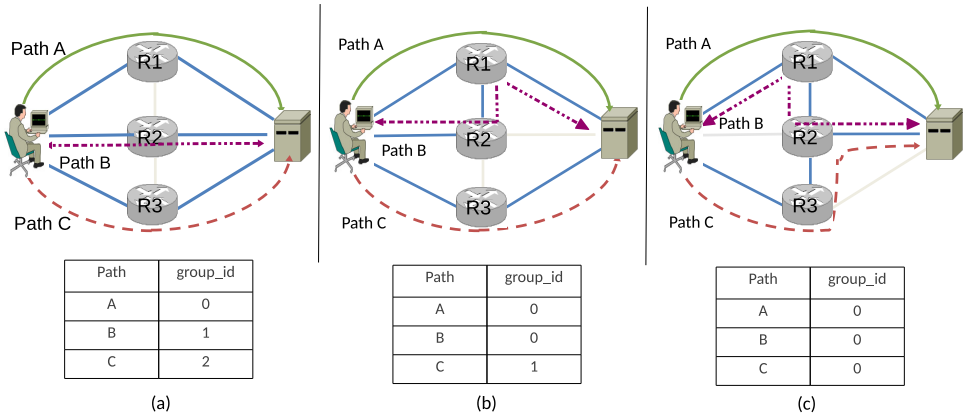
Traditional transport protocols for IP networks, such as TCP and SCTP, place the congestion control management at the communication endpoints. ICN brings to the table another possibility: in-network congestion control. There is an on-going discussion in the ICN research community on whether congestion control should be applied solely at the endpoints or whether network routers should

also play a role (see Section 3.3). We hereby introduce a novel congestion control scheme which combines the strong features of end-to-end and in-network designs, thus constituting a hybrid solution. First, we employ end-to-end congestion control, deriving its design from a well-established background that offers stability, high performance and scalability. Second, we exploit an in-network mechanism that assists congestion control, thus increasing the utilization of network's computational resources and supporting more sophisticated end-to-end congestion control designs.

The proposed scheme consists of two independent modules: (i) *Normalized Multiflow Congestion Control* (NMCC), a novel end-to-end multiflow-aware algorithm, and (ii) a network assistance module that undertakes path formation and provides topological information to the endpoints. NMCC is simple yet effective, offering instant convergence to TCP-Friendliness and high bandwidth aggregation under various conditions. The core novelties of NMCC are presented in the method to achieve friendliness, where a deterministic algorithm equalizes throughput of multipath and single-path since the beginning of the transfer (including also the SS state), and the method to implement friendliness, where we exploit an inherent issue of TCP, called *TCP fairness* [52]. In addition, the in-network mechanism supports advanced path formation strategies, such as selecting k-disjoint paths, and delivers knowledge of shared bottlenecks to the end-users so as to support *greedy friendliness*, a technique where end-users consider the TCP-Friendliness constraint only when subflows compete with unfriendly for resources.

4.2.1 Topological assistance module

The best case scenario for multiflow communication arises when all communication paths are physically *edge-disjoint*, or just *disjoint* for brevity, not sharing any links. In this case, each multiflow connection can use the same congestion control algorithm as single-flow connections without any friendliness constraints; this approach is named *Uncoupled* after the related congestion control of MPTCP (Section 3.3.1). In contrast, when some subflows use paths which are not disjoint, their aggressiveness needs to be limited in order for them to remain friendly, by consider-



Path	group_id
A	0
B	1
C	2

Path	group_id
A	0
B	0
C	1

Path	group_id
A	0
B	0
C	0

Figure 4.8: An example of TM assistance in three different cases of path composition: (a) Disjoint paths, (b) two paths sharing one link, (c) three paths sharing two links.

ing all flows together. Our congestion control scheme practices *greedy friendliness* by limiting the aggressiveness of the subflows only when needed, namely, in the second case.

Path selection in PSI is performed by the TM, whose operation details extend beyond the scope of this dissertation. An efficient yet simple solution in PSI is to direct the TM so as to discover only disjoint paths, thus eliminating the TCP-friendliness issue. Nevertheless, there are services where the k-shortest paths are preferable to the k-shortest-disjoint paths, hence our only requirement is that when the TM returns a set of paths encoded as LIPSIN identifiers, a *group_id* code should be added to each identifier so as to indicate non-disjoint paths. Specifically, all paths that share at least one link with some other (not necessarily *the same* link) are marked with the same *group_id*. In general, for any given underlying routing mechanism, the in-network assistance mechanism must be able to signal how the available paths are grouped by *group_id*.

For instance, Fig. 4.8 shows three examples of path composition along with the corresponding *group_id* codes. In Fig. 4.8(a) the three paths are disjoint, thus each path is marked with a distinct *group_id*, whereas in Fig. 4.8(b) paths A and B share a link, thus they have the same *group_id*. In 4.8(c) Paths A and B share a link and paths B and C share a different link; they still get the same *group_id*, to ensure that each path belongs to a single group. This simplifies operation, at

the cost of losing some efficiency, since a congested link may only affect some of the paths in a group.

We have also considered identifying bottlenecks with link-level granularity to further enhance the accuracy of friendliness adaptation. In this case, each group would only consist of paths sharing the same links, hence a path could belong to several groups. For example, in Fig. 4.8(c) path B would belong to a group with path A and another with path C. This complicates controlling the aggressiveness of each group, since congestion events in path B can affect path A, path C, or even both.

Operational overhead

Our notification service comes with apparent operational overhead since it requires finding the best k-paths and identifying common links between them. We use the TM operation that is presented in [72] as a performance baseline in order to asses the extra costs. Finding the k-shortest paths in a topology of n nodes and e edges has $O(e + nlogn + k)$ complexity [73], which is similar to Dijkstra's shortest path algorithm. However, finding the common links among these paths can be computationally expensive. If the k disjoint paths are up to h hops each, $h^2k(k-1)2^{-1}$ comparisons are required to compare them, hence $O(h^2k^2)$, as the h links of the k^{th} path must be compared with each of the h links of the $k-1$ previous paths. For example, in a datacenter network where multipath can provide gains with up to 6 paths ($k \leq 6$) [25] and the dissemination paths can consist of 6-8 hops, the inflicted computational overhead can be considerable.

We propose an alternative method that introduces slightly more state but reduces computational complexity to $O(kh)$. We exploit a Hash Table with $O(1)$ retrieval complexity, where the keys are link identifiers and the values are the sets of path identifiers containing the corresponding link. First, all k paths are parsed and the identifier of each path is entered in the corresponding link entries (kh writes), thus creating a collection of paths for each link. Then, all table entries are parsed (kh reads) to derive the path groups: initially, all paths associated with the same link form a group, and then we recursively merge any groups that

happen to share any entries (that is, common path identifiers). Referring again to Fig. 4.8(c), we would initially create one group for paths A and B and another for paths B and C, due to their shared links. Then, we would merge the two groups due to their shared path. Although this design requires storing the entire path information, assuming 4-byte integers for link and path identifiers, the storage cost is negligible.

4.2.2 Normalized Multiflow Congestion Control (NMCC)

When the available paths have different *group_ids* (i.e., they do not share any links), window management does not consider TCP-friendliness: our algorithm creates a distinct subflow for each path with an individual congestion window variable, RTT-based loss detection timer and retransmission mechanism, and subflows operate independently the SS and CA algorithms, similarly to MPTCP's Uncoupled congestion control scheme.

In contrast, when some paths have the same *group_id* (i.e., they share some links) the *Normalized Multiflow Congestion Control* (NMCC) algorithm is used to manage them as a group. NMCC is a novel congestion control algorithm for multipath connections that offers TCP-friendliness and high resource utilization under various path setups, including disjoint, overlapping and mis-matched paths. It differs from existing designs, such as LIA, OLIA and BALIA, in two distinct ways: first, it introduces a new approach for pursuing friendliness and, second, it introduces a new approach for implementing friendliness.

Pursuing TCP-friendliness The gains of NMCC compared to the existing algorithms arise from the way it approaches TCP-friendliness. NMCC achieves friendliness by normalizing the *growth* of the transfer rate of each flow, rather than the transfer rate itself. NMCC exploits the fact that all connections start at the same state, that is, they begin with the minimum allowed congestion window, and remain friendly as long as their throughput increase rates are equal. NMCC thus focuses on distributing the throughput increase rate of the fastest subflow among its pool of available subflows. The friendliness requirement is deterministically

met at each window increase, hence NMCC is instantly friendly and remains so throughout the entire connections lifespan.

Implementing TCP-friendliness NMCC exploits inherent properties of TCP to control over-aggressiveness. Specifically, NMCC is based on a well-known TCP-fairness characteristic, the fact that connections with higher RTTs are less aggressive [74]. Instead of restraining the growth of the congestion window per RTT like previous solutions, NMCC indirectly controls congestion window growth by inflating the RTTs used in the calculations; this simplifies friendliness in the SS phase and avoids multiflow-related issues due to RTT-mismatch, sudden load and congestion shifts.

In the following we elaborate on the mechanics of NMCC in greater depth. For clarity, we first discuss the operation of NMCC in the CA phase and then extend our discussion to the SS phase in order to provide a complete solution.

TCP-friendliness during Congestion Avoidance

In the CA phase, NMCC uses an inflated RTT, r'_i, for each subflow i to control window growth; the inflated RTT slows down the rate of increasing the congestion window since $r'_i \geq r_i$. In order to estimate the amount of inflation, we introduce the *friendliness factor in congestion avoidance*, m_{ca}, where

$$r'_i = m_{ca} r_i \tag{4.1}$$

The calculation of m_{ca} is derived based on two fairness goals: (i) the growth rate of all subflows sharing a link should be no more than that of the fastest single-flow connection and (ii) the overall growth rate should not be less than that of the fastest single-flow connection.

We assume that the growth rate of the most aggressive single-flow connection, is equal to growth rate of the subflow with the minimum RTT, r_{min}, among our pool of subflows. During this phase, subflow i increases its congestion window by one MSS, s, every RTT, so its window growth rate is s/r_i and its throughput growth rate is s/r_i^2. Therefore the throughput growth rate of subflows in this phase must satisfy the following equation:

$$\frac{s}{r_{min}^2} = \sum_{i=1}^{N} \frac{s}{r'^2_i} = \sum_{i=1}^{N} \frac{s}{m_{ca}^2 r_i^2}$$

where N is the set of jointly controlled subflows. We can therefore estimate m_{ca} using the following equation:

$$m_{ca}^2 = \frac{r_{min}^2}{s} \sum_{i=1}^{N} \frac{s}{r_i^2} = r_{min}^2 \sum_{i=1}^{N} \frac{1}{r_i^2} \tag{4.2}$$

To understand the friendliness factor m_{ca}, consider a simple example. Assume that the TM offers two paths marked with the same *group_id*, with $r_A = 5$ ms and $r_B = 10$ ms. The operation of NMCC includes three simple steps to offer instant TCP-friendliness:

Step 1. Initiliazation: We initially set m_{ca} equal to the number of jointly controlled paths,[4] hence: $m_{ca} = 2$.

Step 2. Estimation of m_{ca}: Upon every packet receipt,[5] we can calculate the friendliness factor using (4.2), hence in our example:
$m_{ca} = 5\sqrt{(1/5^2 + 1/10^2)} \simeq 1.118$.

Step 3. Estimation of r'_i: Using (4.1), we calculate the inflated RTTs, therefore $r'_A = 5.59$ ms and $r'_B = 11.18$ ms.

The inflated RTTs allow NMCC to increase its overall congestion window by $1/5.59 + 1/11.18 = 0.268$ MSS/ms, while the fastest single-flow connection will inflate its window by $1/5 = 0.2$ MSS/ms. However, the throughput increase rates are equalized: NMCC increases the overall throughput by $1/(5.59)^2 + 1/(11.18)^2 = 0.04$ MSS/ms^2 while the fastest single-flow connection by $1/5^2 = 0.04$ MSS/ms^2. Consequently, both connections extend their portion of network resources evenly, thus fairly sharing the available bandwidth.

By applying m_{ca} to the RTTs of all subflows, we adapt the growth rate of all paths, which means that, although we favor the subflow which operates over

[4] This is equivalent to assuming that all RTTs are equal in 4.2.

[5] Although the computation is not expensive, m_{ca} can be updated less frequently (e.g., every RTT) for resource constraint devices.

the fastest path, we do not neglect the other paths. Therefore, NMCC does not require probing to detect load changes on unused paths, unlike LIA that introduces a special parameter to keep a moderate amount of traffic on slow paths and OLIA that requires probing. NMCC can therefore perform efficiently in heterogeneous environments, adapting fast to path failures and congestion bursts. For instance, consider an integrated terrestrial-satellite network where the terrestrial link has 10 ms delay and the satellite one has a 250 ms delay. In this case $m_{ca} = 1.00079$, which causes a tiny adjustment to the RTT of each flow that does not constrain subflow growth, allowing NMCC to effectively grasp the available resources.

Converting RTT inflation to window reduction Rather than radically modifying the existing implementations of window-based congestion control to rely on modified RTTs, we convert the inflated RTT algorithm to an equivalent one that controls the window growth per ACK. The throughput increase rate of a subflow with NMCC is:

$$\frac{s}{r'^2} = \frac{s}{(m_{ca}r)^2} = \frac{s/m_{ca}^2}{r^2} \tag{4.3}$$

hence the increase of a friendly congestion window is s/m_{ca}^2 over the unmodified RTT. Measuring congestion window, w, in bytes, TCP increases its window by s^2/w bytes, w/s times within an RTT, for an overall growth of 1 MSS. By reducing the amount of per-ACK increase of a subflow to $s^2/(m_{ca}^2 w)$ bytes, the cumulative increase of NMCC within an RTT is s/m_{ca}^2, thus satisfying the friendliness requirement. In this case, the friendliness factor m_{ca} directly controls the growth of the congestion window upon the receipt of an ACK, thus allowing NMCC to be integrated with TCP-like transport protocols.

TCP-friendliness during Slow Start

Most work on multiflow transport deals only with the CA phase, since SS is considered a transient state with no measurable impact on the long-term performance. Nevertheless, during the evaluation of NMCC we noticed that friendliness was compromised when (i) the content was relatively small and (ii) the path was very congested. An analysis of the evolution of the congestion windows showed that

NMCC with N subflows gained bandwidth almost N-times faster than a single-flow connection during SS. Since short and very congested connections spend a measurable fraction of their lifetimes in SS, meeting the friendliness goals in CA was not enough to amortize NMCC's aggressive behavior during SS. Interestingly, in [75] the authors indicate that MPTCP suffers from the same fairness problem and show that the over-aggressiveness during SS often results in a large number of retransmissions that deteriorate flow performance.

One way to reduce aggressiveness during SS is to reduce the *ssthresh* parameter, so as to make the algorithm switch from SS to CA sooner. Unfortunately, this has two disadvantages. First, when a connection starts, the available bandwidth of the communication path is unknown, hence *ssthresh* should be set high enough to probe it. Second, reducing *ssthresh* only limits the amount of bandwidth that the protocol will re-acquire before it slows down, not the rate of acquisition. In [75] the authors present a design that reduces the growth rate but overlooks the reduction of the *ssthresh* parameter. Such a solution is incomplete because the subflows are allowed to perform longer in the SS.

NMCC controls the amount of bandwidth gained during SS, as well as its rate of growth. The NMCC friendliness approach for CA can be seamlessly adapted to the SS phase for controlling aggressiveness. Specifically, in SS, when a subflow i doubles its congestion window every r_i, its instant throughput growth rate is w_i/r_i^2, where w_i is the congestion window of subflow i. We introduce m_{ss} *the friendliness factor for SS* where $r'_i = m_{ss} r_i$. Similarly to the CA phase, m_{ss} must equalize the throughput growth rate of all multipath flows with the fastest increasing single-path, hence:

$$\frac{w_k}{r_k^2} = \sum_{i=1}^{N} \frac{w_i}{r'^2_i} = \sum_{i=1}^{N} \frac{w_i}{m_{ss}^2 r_i^2}$$

where k is the subflow with the highest growth rate. We can therefore estimate m_{ss} as follows:

$$m_{ss}^2 = \frac{r_k^2}{w_k} \sum_{i=1}^{N} \frac{w_i}{r_i^2} \tag{4.4}$$

The similarity of (4.2) and (4.4) allows the creation of a unified method for estimating the friendliness factor when subflows are in different states. We

introduce Ω_i and Ω'_i, the regular and the friendly throughput growth rate of subflow i, respectively. The estimation of Ω_i and Ω'_i depends on the apparent congestion phase of the subflow, therefore:

$$\Omega_i = \left\{ \begin{array}{ll} s/r_i^2, & \text{in cong. avoidance} \\ & \\ w_i/r_i^2, & \text{in slow start} \end{array} \right\} \tag{4.5}$$

$$\Omega'_i = \left\{ \begin{array}{ll} s/r_i'^2 = s/m_{ca}^2 r_i^2, & \text{in cong. avoidance} \\ & \\ w_i/r_i'^2 = w_i/m_{ss}^2 r_i^2, & \text{in slow start} \end{array} \right\} \tag{4.6}$$

The combination of (4.2), (4.4) and (4.6) provides a unified formula for estimating m, the *friendliness factor* of NMCC, taking into account subflows in both the CA and the SS phase:

$$m^2 = m_{ca}^2 = m_{ss}^2 = \frac{\sum_{i=1}^{N} \Omega_i}{\Omega_{max}} \tag{4.7}$$

To better understand the operation of NMCC with subflows in different congestion phases, assume that NMCC exploits three subflows, A, B and C, but only subflow C is in CA. The algorithm only requires estimating m based on (4.6) and, then, calculating the inflated RTT based on (4.1). Assume that initially the RTTs are 10, 5 and 5 ms and the windows are 100, 20 and 10 MSS for subflows A, B and C, respectively.

Step 1. Initiliazation: We initially set m_{ca} equally to the number of jointly controlled paths, hence: $m = 3$.

Step 2. Estimation of m: From (4.5), the unfriendly throughput increase rates, Ω_i, for paths A, B and C are $100/10^2 = 1$, $20/5^2 = 0.8$ and $1/5^2 = 0.04$ MSS/ms^2, respectively, hence from (4.7) we can calculate that $m = 1.356$.

Step 3. Estimation of r'_i: From (4.1), the inflated RTT's are 13.56, 6.78 and 6.78 ms.

Thereafter, the throughput increase rates, Ω'_i, are 0.543, 0.435 and 0.022 MSS/ms^2 for paths A, B and C, respectively, and the aggregate throughput growth rate is equal to that of the fastest single-path, or 1 MSS/ms^2.

A final issue that needs to be addressed during the SS state is the *stateful* window increases that can penalize the aggressiveness of NMCC. Specifically, during SS the congestion window of a NMCC subflow on the jth RTT is set to:

$$w_j = w_{j-1} + w_{j-1}/m^2 = w_{j-1}((m^2+1)/m^2) \tag{4.8}$$

while a single-path flow would set it to $w_j = 2w_{j-1}$. At the next RTT, NMCC would apply the friendliness factor to an already reduced window, thus increasing its lag behind the single-path flow. NMCC thus exhibits a "leak", l, in the growth of the congestion window, which is equal to:

$$l_j = ((m^2-1)/m^2)w_{j-1} \tag{4.9}$$

The growth leak exhibits two properties: first, a new leak is introduced each RTT due to the application of m^2, and, second, old leaks grow every RTT as m^2 is applied to w_{j-1}. The first property is important for NMCC as it assures friendliness, but the second falsely penalizes performance. The total amount of lag produced is equal to the summation of old leaks which is

$$\sum_{i=1}^{j-1} l_i m^{2(j-1-i)} \tag{4.10}$$

on the jth RTT. To avoid this problem, NMCC uses the window size of an equivalent single-path flow as the basis of increase. Specifically, the new window size is estimated as:

$$w_j = w_{j-1} + w_j^{sp}/m^2 \tag{4.11}$$

where w_j^{sp} is the window size of a single-path flow running in the same path on the j^{th} RTT. The combined SS and CA algorithm is presented in Algorithm 2.

TCP-friendliness during throughput reduction

Having modeled the friendliness rules for throughput increase, we need to specify and model the rules for throughput decrease too. We hereby demonstrate that, even if the throughput increase is TCP-friendly, the TCP-friendliness equilibrium can be unbalanced in case of a special scenario, namely, when congestion

Algorithm 2 Window adjustment and estimation of m.

1: **procedure** INCREASE_WINDOW
2: **if** $(w < ssthresh/m^2)$ **then**
3: $w \leftarrow w + w^{sp} * s/(w * m^2)$
4: $w^{sp} \leftarrow w^{sp} + w^{sp}$
5: **else**
6: $w \leftarrow w + s * s/(w * m^2)$
7: $w^{sp} \leftarrow w^{sp} + s$
8: **end if**
9: **end procedure**

1: **procedure** ESTIMATE_M
2: $max\_rate \leftarrow 0$
3: $total\_rate \leftarrow 0$
4: **for** $(i \in subflows)$ **do**
5: **if** $(w_i < ssthresh_i/m^2)$ **then**
6: $rate \leftarrow w_i/r_i^2$
7: **else**
8: $rate \leftarrow s/r_i^2$
9: **end if**
10: $total\_rate \leftarrow total\_rate + rate$
11: **if** $(rate > max\_rate)$ **then**
12: $max\_rate \leftarrow rate$
13: **end if**
14: **end for**
15: $m \leftarrow sqrt(total\_rate/max\_rate)$
16: **end procedure**

events are triggered for a *subset* of the subflows. In this section, we model the problem and, then, we elaborate on the solution.

To clarify the issue, consider a friendly multipath connection with two subflows and a single-path competing in a bottleneck. During a congestion escalation period, the single-path and *only one* subflow receive a congestion event, thus reducing their transfer rate to roughly one-half their previous rate. The single-path and the multipath commit a reduction of 50% and 25% of their (cumulative) throughput, respectively, thus throwing friendliness out of balance. This effect grows with the frequency of congestion events that affect a subset of subflows. The worst case scenario is for only one subflow to receive a congestion event and the best case scenario is for all subflows to get the same feedback; we call these *partial* and *global* congestion events, respectively.

In order to provide some performance bounds, we simulate three congestion event scenarios: the worst case, the realistic and the best case. In all cases, we assume a multipath connection (MP) with two subflows and a single-path (SP) connection competing for the same link. Both connections increase their transfer rate in a friendly manner and, when the link is full, flows receive congestion events that reduce their throughput. Figure 4.9(1.a) plots the results of the worst case scenario, where only the (sub-)flow with the largest congestion window gets a packet loss, resulting in MP grasping 67% of the resources. Figure 4.9(1.b) illustrates the results of the realistic scenario, where global congestion events can occur, but faster connections are more likely to experience loss.[6] Again, MP shows measurable over-aggressiveness by grasping 58% of the bottleneck resources. Finally, in the best case scenario, where global congestion events take place, the performance of MP and SP are identical, thus sharing the medium equally.

Modeling throughput reduction NMCC shows measurable over-aggressiveness during partial congestion events, thus motivating an algorithm extension that regulates the throughput decrease under general conditions (partial and global con-

[6]When a link is full, a random sample per (sub-)flow is drawn from a uniform distribution [0,100) and a congestion event is sent if the sample is larger than the bandwidth share of the (sub-)flow.

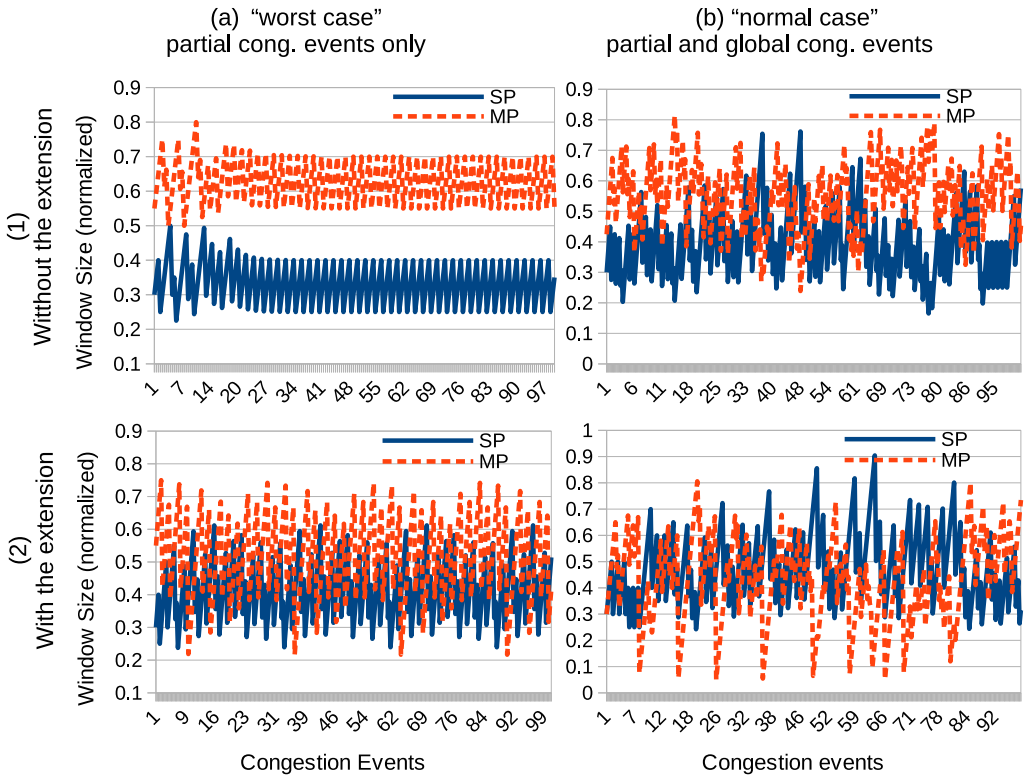


Figure 4.9: NMCC (MP) and single-path (SP) window sizes in case of only "partial" (column a) and "partial and global" congestion events (column b); row 1 depicts performance without the extension, row 2 depicts performance with the extension.

gestion events). We choose not to change the NMCC part that controls throughput growth, since it offers instant convergence to TCP-friendliness, but we propose to add the following friendliness rule that regulates the throughput reduction:

The cumulative throughput reduction of multipath subflows after any number of contemporaneous window reductions due to congestion, should be equal to the reduction of a single-path flow over the "best" path.

The path with the highest throughput increase rate, Ω_{max}, is considered "best", thus using the same benchmark path during throughput growth and throughput reduction.

Oppositely to the deterministic[7] throughput increase per ACK, a determin-

[7]NMCC algorithm does not include any random variables, such as packet loss probability; it enters TCP-friendliness "steady-state" since the first iteration.

istic throughput reduction per congestion event is not trivial. We do not a priori know the number of (sub-)flows that will be affected by the same congestion event, hence we can not estimate the cumulative throughput decrease per congestion event in order to equalize the performance of multipath and singlepath instantly. However, we can estimate the overall reduction over a period of time by assuming that the bandwidth share and the error rate of the flows is stabilized in steady state, thus experiencing an invariant number of losses in time. We hereby define f_{sp} and f_i the frequency that the single-path and the i^{th} subflow of the multipath receive a congestion event, respectively. Similarly, we define d_{sp} and d_i the throughput reduction after a congestion event for the fastest single-path and the i^{th} multipath subflow, respectively. Thereupon, our problem is expressed by the following equilibrium:

$$f_{sp} d_{sp} = \sum_{i \in S} f_i d_i \tag{4.12}$$

where S is the set of subflows.

According to RFC 5681 [48], a congestion event leads TCP into Slow Start or Fast Recovery state, thus causing approximately 50% reduction of transfer throughput; in the first case, the congestion window grows exponentially, reaching the Slow Start threshold very quickly, while, in the second case, the window is directly set to the Slow Start threshold plus 3 MSS, where MSS is the *Maximum Segment Size*.[8] Therefore, we approximate the throughput reduction of singlepath through the following equation:

$$d_{sp} = \frac{w_{sp} - w_{sp}/2}{r_{sp}} = \frac{1}{2} \frac{w_{sp}}{r_{sp}} \tag{4.13}$$

Furthermore, we can model f, the frequency of loss, as the product of throughput and packet error rate, p, hence:

$$f = p \frac{w}{r} \tag{4.14}$$

[8] In both cases we assume that the RTT remains relatively unchanged, since the congestion level of the bottleneck is affected by numerous competing flows.

Now, we can formally demonstrate the friendliness issue by assuming that d_i, the throughput decrease of subflow i, is unregulated, thus following (4.13). Then, we can rewrite (4.12) as follows:

$$p_{sp}\frac{w_{sp}}{r_{sp}}\frac{w_{sp}}{2r_{sp}} = \sum_{i\in S} p_i\frac{w_i}{r_i}\frac{w_i}{2r_i}$$

$$\Leftrightarrow \frac{p_{sp}w_{sp}^2}{r_{sp}^2} = \sum_{i\in S}\frac{p_i w_i^2}{r_i^2} \tag{4.15}$$

Now consider the simple case where two subflows, that are in congestion avoidance, perform in similar paths, with equal bandwidth, latency and error rate. Then, according to (4.7), $m^2 = 2$, thus halving the growth rate of the subflows' windows, hence:

$$r_i = r_{sp},\ p_i = p_{sp},\ w_i = \frac{w_{sp}}{2}$$

One can easily see that (4.15) is not satisfied under these conditions, thus verifying the unfriendliness issue that needs to be addressed.

TCP-Friendly Throughput reduction In order to maintain TCP-friendliness under various types of congestion events, we introduce the *threshold factor*, m_t, a positive real number ($|S| \geq m_t \geq 1$), that regulates the throughput reduction of a subflow during a congestion event, as shown in (4.16).

$$d_i = d_{sp}m_t = \frac{w_i - w_i/2}{r_i}m_t = \frac{w_i}{2r_i}m_t \tag{4.16}$$

We rewrite (4.12), using (4.16) to express d_i, and estimate the threshold factor, m_t, as follows:

$$\frac{p_{sp}w_{sp}^2}{r_{sp}^2} = \sum_{i\in S}\frac{p_i w_i^2}{r_i^2}m_t$$

$$\Leftrightarrow m_t = \frac{p_{sp}w_{sp}^2/r_{sp}^2}{\sum_{i\in S} p_i w_i^2/r_i^2}$$

The estimation of m_t by a multipath subflow, requires the knowledge of w_{sp}, r_{sp} and p_{sp}, which express the performance of a TCP-like flow on the best available path; the best path is the one with the highest throughput increase rate, Ω_{max}. By

definition, the packet drop rate of the path, being a feature of the medium, is not affected by NMCC, hence $p_{sp} = p_{max}$. The throughput of the singlepath is equal to the cumulative throughput of the multipath, as a result of meeting the friendliness constraint during window increase, hence $w_{sp}/r_{sp} = \sum_{i \in S} w_i/r_i$. Consequently, any subflow can estimate m_t via the following formula:

$$m_t = \frac{p_{max}(\sum_{i \in S} w_i/r_i)^2}{\sum_{i \in S} p_i w_i^2/r_i^2} \tag{4.17}$$

The proposed extension behaves similarly to the proportional throughput growth scheme of NMCC, thus maximizing aggressiveness reduction when paths are equally fast. Specifically, when the connection deploys one subflow, then $m_{th} = 1$, thus falling back to singlepath behavior. When $|S|$ identical subflows (in identical paths) are deployed, then $m_{th} = |S|$, thus offering the maximum throughput reduction. Finally, as paths get more diverse and a subset of subflows grasps the most resources, then $m \to 1$, thus not affecting the throughput reduction.

We repeated the simulations of Fig. 4.9, but this time including the threshold modification factor, m_t, in the congestion control scheme. In the worst case scenario, multipath (MP) gains 54% of resources (Fig. 4.9(2.a)), thus improving friendliness by 13% (Fig. 4.9(1.a)). In the more realistic scenario, MP gets 49% of resources, thus achieving friendliness and preliminary validating our solution (Fig. 4.9(2.b)).

Finally, the convergence to TCP-friendliness is not expected to be delayed by this extension. The NMCC connection is friendly instantly, since the throughput reduction mechanism is enabled only after the first packet loss. Then on, the algorithm requires that the (sub-)flows converge to their fair share of resources, in order to estimate the packet drop probability of the paths; TCP flows converge after few congestion rounds [76]. Consequently, our extension is expected to converge to TCP-friendliness by the time the last subflow converges to its share of resources. This argument is validated through experiments with the Linux implementation of MPTCP in Section 5.2.

The NMCC algorithm is presented below:

Algorithm: NMCC

- For each ACK on path i,

$$w_i \leftarrow w_i + \frac{1}{w_i} \frac{\Omega_{max}}{\sum_{i \in S} \Omega_i}$$

- For each loss on path i,

$$w_i \leftarrow w_i - \frac{w_i}{2} * \frac{p_{max} (\sum_{i \in S} w_i / r_i)^2}{\sum_{i \in S} p_i w_i^2 / r_i^2}$$

4.2.3 Implementation and experimentation

In this section, we evaluate the performance of our hybrid congestion control algorithm. We start by first examining NMCC in simple LAN topologies using the prototype implementation of mmTP in order to assess TCP-friendliness and bandwidth aggregation. Later, we examine more sophisticated benchmark scenarios in LAN topologies using the *htsim* simulator[9] to estimate load balancing, resource utilization and TCP-friendliness throughout long transmissions. Finally, we explore realistic domain-scale scenarios using the NS-3 simulator[10] in order to estimate the effect of NMCC and TM's assistance on the overall network resource utilization.

LAN emulation with mmTP

We have implemented our hybrid congestion control algorithm as part of the mmTP protocol that runs over Blackadder, the PSI prototype implementation [70]. Our implementation includes the mmTP sender and receiver applications with NMCC enabled, as well as a TM that computes the k-shortest paths from every publisher to a subscriber, using the algorithm by Yen [71] with hop count as the metric.

We deployed Blackadder with mmTP in LAN topologies in our laboratory, using 100 Mbps switches and workstations as network nodes. In this environment we have full control of the communication paths, we can avoid unwanted traffic that could influence the results and we can monitor link capacities and delays, as well as node and router status. Our experiments examine (i) TM's assistance effect

[9] http://nrg.cs.ucl.ac.uk/mptcp/implementation.html
[10] https://www.nsnam.org

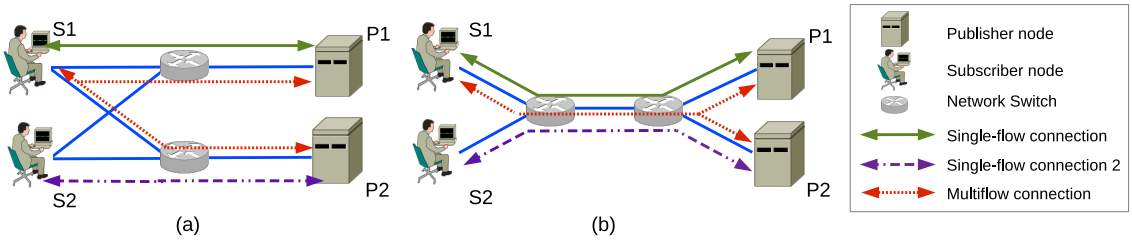


Figure 4.10: LAN testbed topologies for assessing performance in (a) disjoint paths and (b) paths with shared bottlenecks.

with disjoint paths, (ii) NMCC's behavior with overlapping paths, (iii) NMCC's behavior in short transfers and (iv) NMCC's behavior in heterogeneous networks.

In our testbed, the transmission latency among publishers and subscribers is set to 100 ms and the bandwidth of each link is 11.7 MBps, as estimated using `iperf`.[11] The duration of transfers during all experiments is 300 s, but we consider only the final 60 s where the system has been stabilized, except when mentioned otherwise. In order to enhance the reliability of our conclusions, we repeated each experiment until the margin of error was less than 2%, so as to achieve a confidence level of 95%.

Disjoint paths We first deployed mmTP in the topology of Fig. 4.10(a), in order to investigate the performance gains of our approach when paths are known to be disjoint. Figure 4.10(a) supports one multisource path from publishers P1 and P2 to subscriber S1 and two disjoint paths from publishers P1 and P2 to subscribers S1 and S2, respectively. We first executed some experiments with no contending traffic, so as to establish a performance baseline, leading to the average transfer rates shown in Table 4.1; each line depicts results from a different experiment. The first two experiments involve running mmTP in multisource mode to both publishers, with and without TM assistance, while the next three experiments involve running single-flow mmTP connections to each publisher, first independently and then together. We notice that each path offers roughly 10.6 MBps throughput and multiflow mmTP achieves 21.3 and 20.7 MBps with and without TM assistance, respectively. These preliminary results validate that mmTP fully exploits available

[11] Available at `http://iperf.sourceforge.net/`.

Transmission mode	Transfer rate (MBps)
Multisource with TM assistance	21.3
Multisource with no TM assistance	20.7
Single-flow from P1 to S1	10.6
Single-flow from P2 to S2	10.7
Single-flows on both paths	21.1

Table 4.1: Average transfer rates with disjoint paths.

capacity and imply that TM assistance slightly enhances performance, even in the absence of competing flows, since with TM assistance the window growth in each path is not throttled in any way.

We then deployed mmTP in multisource mode over the same topology (S1 requests data from both P1 and P2), with one or two single-flow connections competing over one or both disjoint paths (S1 to P1 and S2 to P2). In Fig. 4.11 we show the average share of the total bandwidth that mmTP achieved in each case, depending on whether TM assistance was turned on or off. The results validate the performance gains and the friendliness of NMCC. Ideally, with one contending single-flow connection NMCC should use half of the bandwidth over one path and the entire bandwidth over the other, or 75% of the total bandwidth. With two contending single-flow connections NMCC should use half of the bandwidth over each path, or 50% of the total bandwidth. In our experiments, mmTP with TM assistance acquires 67.5% and 49.2% of the overall bandwidth, respectively. On the other hand, without TM assistance the bandwidth shares of mmTP are significantly lower, namely 54.6% and 38.5%, respectively, reflecting a far more conservative sharing of the available bandwidth. However, these connections do not all share the same bottleneck link, hence aggressiveness mitigation is unnecessary.

Shared paths To investigate the case where paths share some links, mandating a less aggressive behavior to ensure friendliness, we used the topology shown in Fig. 4.10(b), where the endpoints are connected by overlapping paths. We deployed a multisource connection from subscriber S1 to publishers P1 and P2, in parallel with 1, 2, 4 and 9 single-flow connections from subscriber S1 to publisher P1 and

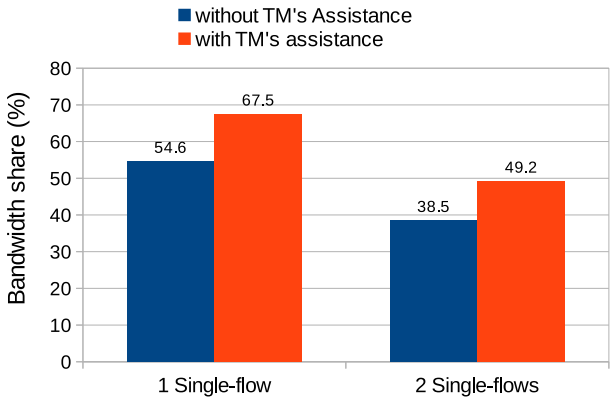


Figure 4.11: mmTP performance in LAN topology with disjoint paths, exploring friendliness with and without TM assistance.

from subscriber S2 to publisher P2; these connections are distributed uniformly between the two paths.

Figure 4.12(a) demonstrates the average bandwidth percentage acquired by NMCC and *all* single-flow connections, while Fig. 4.12(b) displays the average transfer rate achieved by NMCC and the *average* singlepath connection. NMCC acquires 53%, 37%, 22% and 12% of the bottleneck link's bandwidth when competing with 1, 2, 4 and 9 single-flow connections, respectively, marginally over the "perfect" sharing ratios of 50%, 33.3%, 20% and 10%, respectively, thus satisfying the friendliness goal. The slight performance advantage of NMCC, also evident in the transfer rates, arises from NMCC's goal to match the fastest single-path available. With multiple similar paths, the fastest available path over a prolonged period is not fixed, as congestion levels fluctuate. NMCC chooses the best path based on current RTT, hence it performs similarly to a multipath congestion control algorithm that exploits only the best path from a pool, thus gaining a slight performance advantage. The mean friendliness factor, m, of NMCC in these experiments was roughly 1.4, while the optimal would be 1.41, indicating a slight over-aggressiveness.

We also examined NMCC's response to a sudden change in the congestion level, by repeating the previous experiment, but this time starting the multiflow connection either 30 s after or 30 s before the start of the single-flow connections.

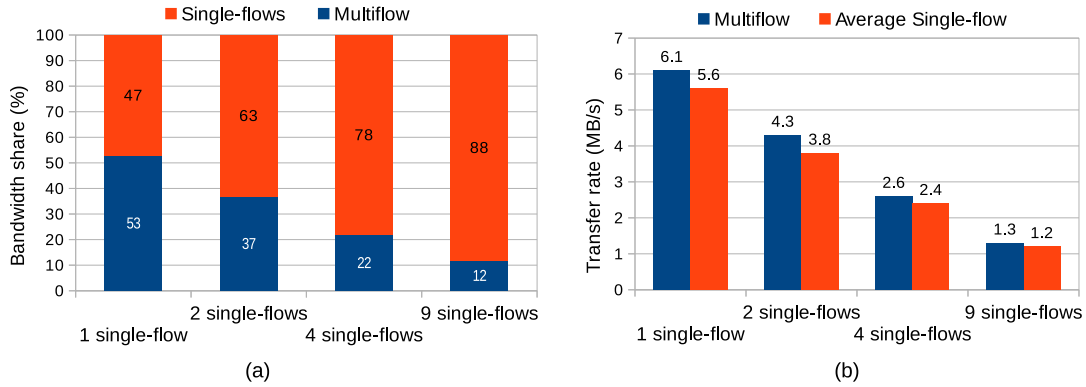


Figure 4.12: NMCC performance in LAN topology with shared bottleneck, exploring multipath friendliness against (a) all single-flow connections and (b) the average single-flow connection.

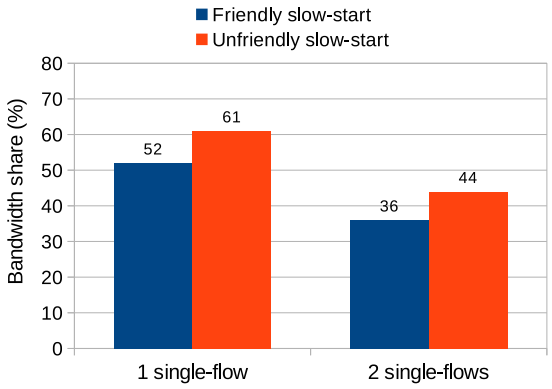


Figure 4.13: NMCC performance in LAN topology with shared bottleneck, exploring friendliness in short transfers with and without friendly SS.

The results of these experiments are nearly identical to the previous ones, as NMCC acquires 54%, 36%, 23% and 12% of the bandwidth when competing with 1, 2, 4 and 9 single-flow connections, respectively. Consequently, NMCC manages to efficiently share bandwidth with newly established connections, as well as to obtain a fair share of bandwidth when launched in an already congested path.

Short transfers NMCC is friendly during the SS phase, unlike the majority of existing multipath algorithms that are only concerned with the CA phase. This is particularly important for short transfers, where friendliness during CA cannot compensate for an unfriendly SS. To evaluate this aspect of NMCC, we reused the

shared link topology of Fig. 4.10(b), deploying one multisource NMCC connection and either 1 or 2 contending single-flow connections. Each connection transfers a 10 MB object, which would require less than 1.1 s to complete in the absence of contention. Figure 4.13 presents the percentage of overall bandwidth acquired by NMCC when friendly SS is turned on or off.

With unfriendly SS, NMCC grabs a disproportionate amount of bandwidth from the competing connections, compared to the ideal shares of 50% and 33%. In the first case, NMCC gets 61% of the bandwidth; while in the second case it gets 44%, or 11% more than the fair share in both cases. On the other hand, NMCC with friendly SS gains 52% and 36% of the total bandwidth. Consequently, NMCC is friendly even with short transfers.

Discussion of results We have preliminary evaluated NMCC and the network-assistance module in toy-topologies using the PSI prototype implementation. Our results validate that the TCP-friendliness requirement is met regardless of the number of competing flows (Fig. 4.12) or the connection duration (Fig. 4.13). We also validate that the TM assistance can effectively notify mmTP, so as to realize the greedy friendliness concept (Fig. 4.11).

LAN simulations with htsim

In this section, we explore the behavior of NMCC, LIA and Uncoupled[12] by replicating the five benchmark scenarios used in the evaluation of LIA in [3] using the same simulator, *htsim*. The scenarios investigate the performance of multipath in terms of TCP-friendliness, resource utilization and load balancing. Each scenario was repeated 500 times and each run lasted 1000 seconds, but we take into account only the last 200 seconds when the performance is stabilized. Notice that htsim has a coarse grained *Retransmission TimeOut* (RTO) estimation, therefore we consider the long run behavior of flows with non-frequent timeouts, as otherwise timeouts would be incorrectly grouped in time. Finally, we observe that the congestion events reported by htsim are 99% triple duplicates and 1% timeouts,

[12]OLIA and Balia are not implemented in htsim.

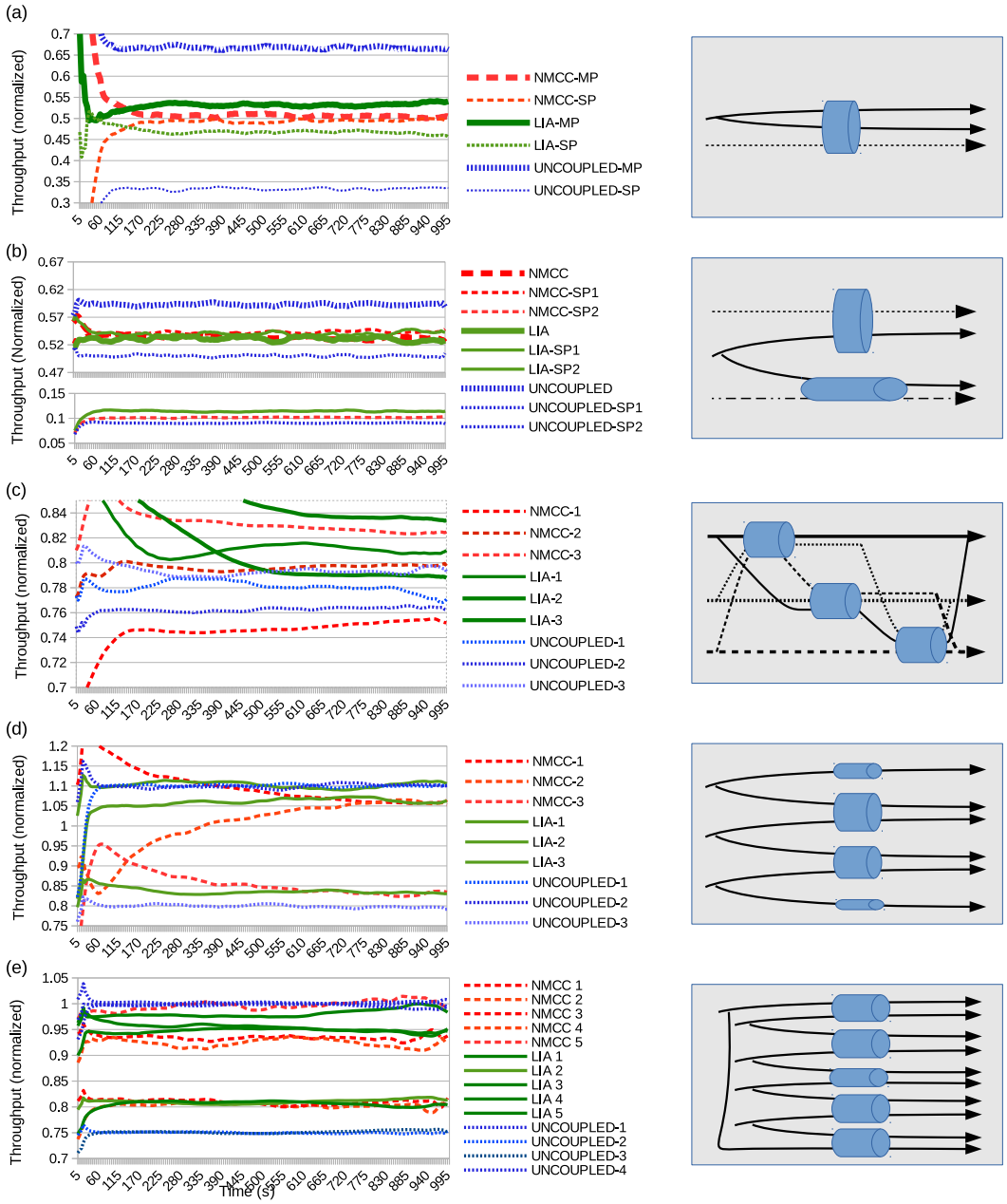


Figure 4.14: LIA and NMCC performance comparison (left column) in the benchmark topologies of [3] (right column). Figures illustrate the instant bandwidth share of multipath (MP) and single-path (SP) connections normalized to the overall network capacity unless otherwise stated. Figure (a) examines NMCC's efficiency in terms of TCP-friendliness, (b)-(c) resource utilization and (e)-(d) load balancing.

even though timeouts typically outnumber triple duplicates in the Internet [76]. This behavior is expected to emphasize on the NMCC friendliness issue, which is caused by partial congestion events, since Fast Retransmit controls congestion more timely, preventing global congestion events.

TCP-friendliness In [3, Fig. 1] a bottleneck topology is used to investigate resource sharing between a multipath (with two subflows) and a single-path connection. We replicated the experiment and found that NMCC exhibits perfect sharing, getting 50% of the available resources, while LIA is slightly more aggressive grasping 53%, as shown in Fig. 4.14(a). We repeated the same experiment for different link throughputs (200-1000 packets/s), without observing measurable differences. Therefore, we argue that NMCC is TCP-friendly when sharing a bottleneck.

Resource Utilization In [3, Fig. 4] a two-links topology with RTT and error-rate mismatch is used to explore resource utilization by the multipath connection. We replicated the experiment, where multipath competes a single path on each path. All algorithms achieve 100% resource utilization but offer different levels of friendliness: NMCC gets 53% of resources on the widest path, LIA gets 52% and Uncoupled 59%. The results are presented in Fig. 4.14(b), where flow throughput is normalized to the bandwidth of the widest path.

In [3, Fig. 2] a topology with three links of capacity C is used to evaluate resource utilization as a result of choosing the least-congested path. Specifically, three multipath sessions are deployed, each having one subflow through one link and a second one through the other two links, so that each link is used by three subflows. Each multipath session should use only the least congested path (the single link) and get a cumulative transfer rate of C, instead of using the two links shared by the other subflows, which would lead to a transfer rate of only $2C/3$. Figure 4.14(c) shows the throughput of each subflow normalized to C, the resource share of the fastest single path. Results yield that the algorithms do not maximize resource utilization, but LIA performs slightly better than NMCC, that performs sightly better than Uncoupled, scoring 81%, 79% and 77%, respectively. While this

under-utilization is considered a weakness and is the core motivation for OLIA [16], the importance of pushing traffic exclusively to the "best" path is debatable, as preferring the least congested path can exhibit poor responsiveness [17, Fig. 4] and penalize performance in datacenters [3].

Load balancing In [3, Fig. 3] a topology with four parallel links of different capacities is used to estimate the load balancing efficiency of the multipath algorithm. Three multipath connections are deployed, each establishing one subflow through a different link, so that each connection competes with a different multipath connection on a different link. Ideally, the connections will balance congestion load across all links and perform similarly getting cumulatively C capacity. Figure 4.14(d) illustrates the performance of the three connections normalized to C. NMCC utilizes 98% of network resources while other algorithms reach 100%. The slight performance degradation is measured at the connection that performs in the widest paths, while the slower connection perform similarly to LIA, thus achieving better load balancing (standard deviation of NMCC is 12%, LIA's 14% and Uncoupled's 17%) at the cost of lower resource utilization.

Finally, in [3, Fig. 7] a torus topology with five parallel links is used to assess again the efficiency of the multipath algorithms in load balancing. Each link is used by two subflows from different connections, but one link is considerably narrower. The multipath connections must balance congestion load in all links and perform similarly. The results are presented in Fig. 4.14(e), which illustrates the throughput of the five flows normalized to the fastest single-path measured. In all cases throughput is not perfectly equalized, as the score of connections sharing the narrow link differs from the average with the standard deviation being 9%, 9% and 15% for NMCC, LIA and Uncoupled, respectively. In this experiment resource utilization is 100% of network capacity.

Discussion of results The results of the simulations in the benchmark topologies yield that the TCP-friendliness of NMCC is met under various conditions, while resource utilization and load balancing is efficient. NMCC is slightly better than LIA (and Uncoupled) in sharing the bottlenecks, since it tackles friendli-

	Nodes	Edges	Access Nodes
Globalcenter.gml	12	39	3
Janetlense.gml	20	40	3
Gridnet.gml	12	23	3
Internetmci.gml	23	43	5
Goodnet.gml	17	31	4
Iij.gml	37	65	10
Geant2012.gml	40	61	8
SwitchL3.gml	42	63	12
Bics.gml	33	48	5
Uninett2011.gml	69	98	11
PionierL3.gml	38	52	9
Ans.gml	20	27	3
Aarnet.gml	21	26	4
Nsfnet.gml	13	15	3
Bren.gml	37	42	20

Table 4.2: Characteristics of the AS topologies used in the domain-scale experiments.

ness roughly perfectly (Fig. 4.14(a)), it exhibits a minor resource underutilization compared to LIA (2% less in Fig. 4.14(c)) due to not pushing all traffic in the least congested path, and it offers similar load balancing to LIA (Fig. 4.14(d,e)). The results are reasonable, since NMCC is designed to offer instant and accurate TCP-friendliness, instead of using only the "best" path in order to balance the congestion load.

Domain-scale simulations with NS-3

Having evaluated the performance of NMCC in benchmark topologies, we now turn our attention to more realistic WAN environments. Our goal is to examine the fairness of NMCC and MPTCP in real network topologies, as well as to assess whether multipath in general, and TM awareness in particular, make a difference in the real world. We therefore implemented mmTP with NMCC and the LIA congestion control algorithms, over a detailed implementation of the entire PSI architecture in the NS-3 simulator.

For this evaluation, we used the 15 Autonomous System (AS) topologies

listed in Table 4.2, taken from the Internet Topology Zoo repository.[13] Since path richness is expected to have an influence on multipath performance, we intentionally selected topologies with different density factors[14] (from 0.04 to 0.5). For each AS topology we simulated a number of connections initiated from clients outside the AS to servers inside the AS. We considered two types of clients: single-homed clients are connected to an Access Node (AN)[15] of the AS with a 100 Mbps connection; dual-homed clients are also connected to a second AN with a slower 12 Mbps connection, simulating smartphones with Wi-Fi and 4G interfaces. As dual-homed smartphone users are normally connected to different ISPs over each interface, the two ANs are selected randomly, as shown in Fig. 4.15. The access link delays are also randomly selected in the 5 to 125–500 ms range (the range depends on the scenario). The link delay and capacity inside the AS is the same for all links (10 Mbps to 1 Gbps, depending on the scenario) with a 5 ms delay.

The servers that these clients connect to are also randomly placed in the AS, but we made sure that the number of servers is 10-20% of the number of clients, as indicated in [77]. In each simulation run, all clients started requesting content simultaneously from the appropriate server. We measured the throughput and error rate of each connection for 3 s after the metrics converged to their final values. Although we only attached two users per AN, we conducted 100 experiments per topology, leading to many different server locations and client-server paths. In the following we present average results measured across all topologies with an error margin of less than 2% for a confidence level of 95%.

Friendliness of NMCC and LIA Our first set of experiments focuses on how NMCC and LIA handle friendliness. In these experiments we only used single-homed clients to guarantee that paths are overlapping, hence friendliness is always an issue. We randomly selected 50% of the clients to initiate single-path transfers, with all other clients initiating multipath transfers. We varied two parameters that can significantly affect the aggressiveness of multipath flows: path capacity

[13] http://www.topology-zoo.org/dataset.html

[14] $density = \frac{2|Edges|}{|Nodes|(|Nodes|-1)}$

[15] A node with degree equal to one.

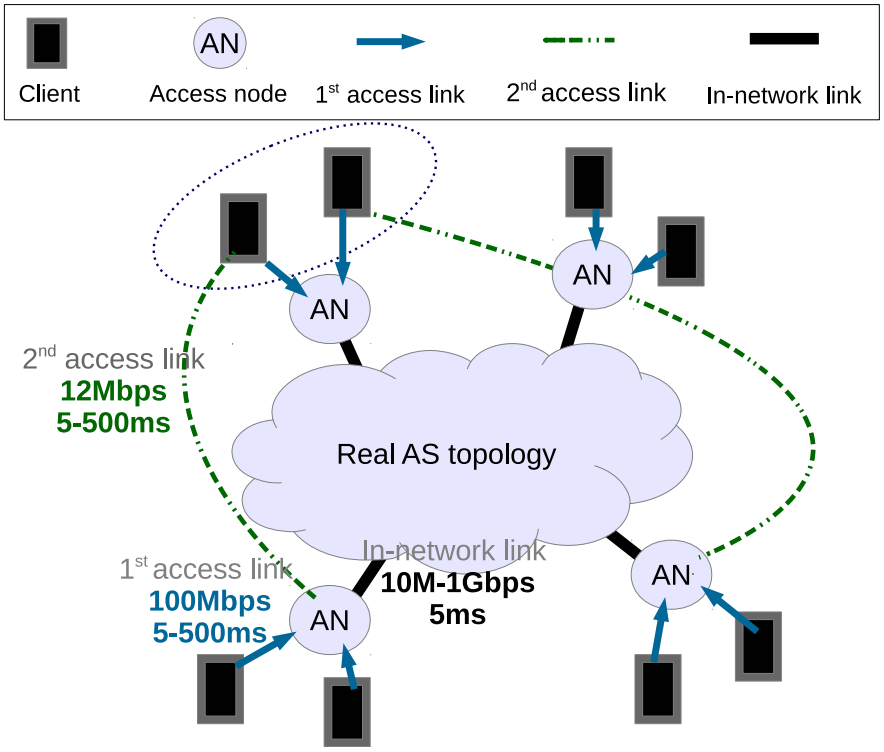


Figure 4.15: AS-scale topologies: client attachment to the Access Nodes (AN) of the AS with two access links. The testbed allows configuring different access links' latencies (per user) and in-network link capacities (per experiment).

and delay variance. Limited capacity is the reason congestion occurs, while delay variance leads to RTT mismatch among paths, which is known to challenge multi-path protocols. We simulated different network capacities by configuring different values for *all* intra-AS links, while different path delays were configured *only* for access links.

We used two metrics to assess performance at the user and network levels. *Relative Throughput* is the throughput of each connection normalized to the throughput of the fastest connection in that run, thus expressing the relative performance of all users. A more friendly algorithm will exhibit higher relative throughput values for the slowest users. For *NMCC Throughput Gains* we sort all connections in ascending throughput order for both NMCC and LIA, and then we calculate for each rank the NMCC gain over LIA, relative to the LIA throughput; positive values indicate that NMCC is faster, while negative ones that LIA is

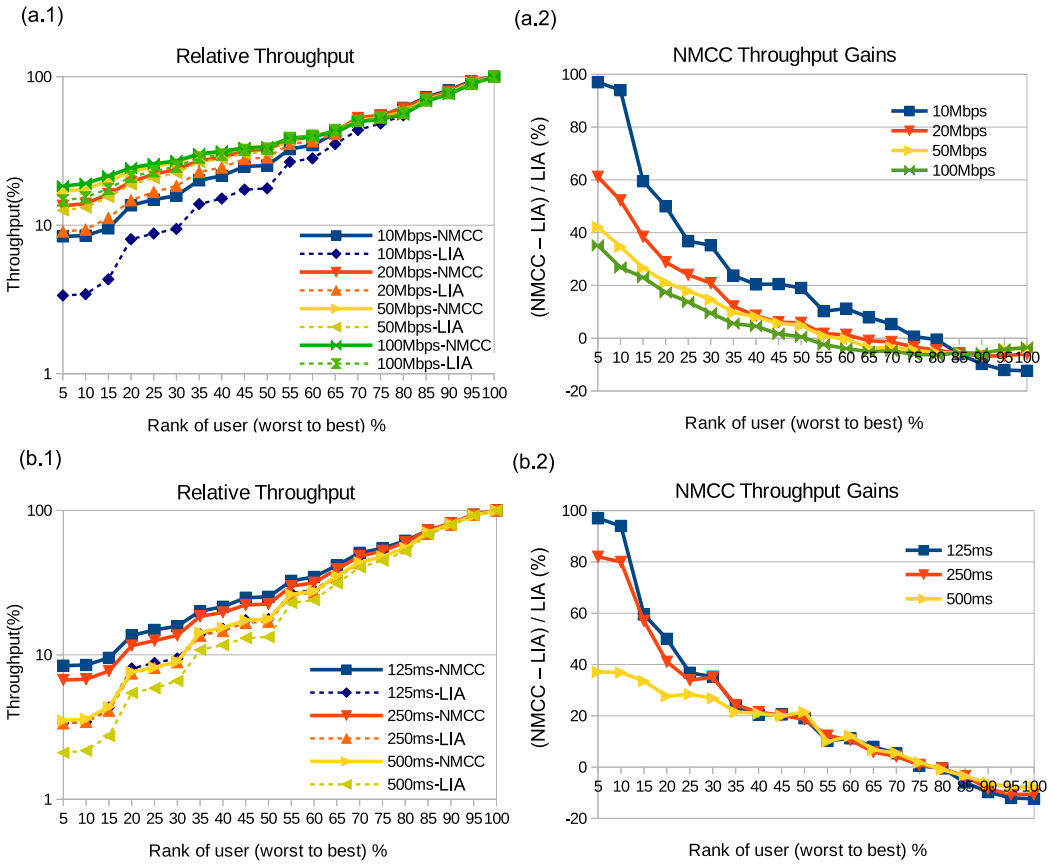


Figure 4.16: NMCC performance in the domain-scale topologies compared to LIA under different (a) in-network link capacities and (b) access link delay ranges.

faster. Since the aggregate throughput across all connections (the overall network utilization) was equal in the NMCC and LIA experiments, fairness is enhanced when the slower users gain throughput, at the expense of the faster users.

We first explored these metrics for in-network link capacities ranging from 10 Mbps to 1 Gbps, when the access link delays are randomly and uniformly drawn from the 5 to 125 ms range; we show the results in Fig. 4.16(a.1-2), omitting the results for 1 Gbps as they are identical to those for 100 Mbps. Figure 4.16(a.1) shows that the relative throughput of the slowest users in NMCC is closer to that of the fastest users compared to LIA, regardless of the link speed. It also shows that the differences are larger as links get narrower (10Mbps exhibits the largest performance gap), since in these cases the bottleneck is reached more easily, making

friendliness more of an issue. Moreover, Fig. 4.16(a.2) shows that NMCC tackles friendliness more efficiently than LIA, as it enhances the throughput of slow users up to 100% for 10 Mbps links, while slightly reducing the throughput of fast users, thus improving overall fairness. We also observe that the throughput gains of NMCC are smaller for larger link capacities, since the congestion level is lower, the variance of user throughput is less and, therefore, the margins for improving friendliness are thinner.

We then repeated these experiments, but this time we fixed the intra-AS link speed to 10 Mbps and varied the propagation delay of access links by randomly and uniformly choosing latencies in the 5 to 125 ms, 5 to 250 ms and 5 to 500 ms ranges; the results are shown in Fig. 4.16(b.1-2). Figure 4.16(b.1) indicates that fairness is more of an issue when the delay variance is high, due to the RTT-unfairness of TCP. Nonetheless, NMCC offers significant gains in terms of friendliness to the slowest users; specifically, Fig. 4.16(b.2) depicts a substantial fairness improvement for roughly 80% of the slowest NMCC transfers compared to LIA. Finally, we notice that the friendliness gains offered by NMCC are reduced as delay mismatch increases.

The effect of TM assistance Our second set of experiments investigates the throughput gains with NMCC due to the exploitation of information about disjoint paths. The experimental setup is similar to the previous section, using AS-scale simulations with 15 real AS topologies. However, since we are interested in disjoint paths, we only configured dual-homed clients. We first ran each experiment using single-flow transport, and then repeated it using NMCC, with and without TM assistance.

We varied three parameters which can impact performance when TM assistance is offered: path capacity, number of paths used and the path formation algorithm used. We expect that bandwidth availability will be proportional to the throughput gains, since the more aggressive TM-assisted connections will maximize their performance when more unused resources exist. Increasing the number of paths should enhance the benefits of TM assistance, as the probability of avoiding performance bottlenecks is increased. We examined the establishment of two and

three paths, in addition to the single-path baseline; three is the maximum number of paths expected to exhibit gains [1]. Finally, in addition to the k-shortest paths algorithm, which can return both disjoint and non-disjoint paths, we also used Bhandari's algorithm [78], which discovers pairs of disjoint paths with Dijkstra-like complexity. To assess performance, we used *Multipath Throughput Gain* which expresses the aggregate gain in network throughput offered by multipath NMCC over single-path performance. Since each topology has a different density, the per topology results help assess how the gains of NMCC are distributed under different scenarios. Notice that topologies are plotted in ascending order of density from left to right.

Figure 4.17 depicts the Multipath Throughput Gain for each topology. In all cases, the access link delays are drawn from the 5-125 ms range. In Fig. 4.17(a) we show the gains offered with two and three-shortest paths, with and without TM assistance, in a resource-constrained network with 10 Mbps intra-AS links. The average gain for two and three-shortest paths is 14% and 24%, respectively, highlighting the effectiveness of multipath even with slow links. Since we only have two access links, hence at most two disjoint paths, the additional gains when using a third path are exclusively due to the avoidance of in-network bottlenecks. On the other hand, even though 32% and 51% of users get disjoint paths in the two and three-shortest path scenarios (49% gets overlapping paths), respectively, TM assistance does not have a noticeable effect on multipath gain in this scenario.

Raising the intra-AS link speed to 100 Mbps makes TM assistance matter: on average, it increases multipath throughput by 2% compared to the case without TM assistance. As a result, with 3-shortest path connections we see an average throughput gain of 57% over single-path connections. Notice that the network links are fully saturated in these runs, therefore any performance gains are still due to bottleneck avoidance. TM assistance delivers the clearer benefits with 1 Gbps intra-AS links, where in-network congestion is negligible, hence NMCC's aggressiveness over disjoint paths actually exploits idle resources, as shown in Fig. 4.17(c). In this case, TM assistance offers a further increase in throughput of 6% and 8% on average over mmTP without TM assistance with 2 and 3-shortest paths, re-

spectively, for an average gain of 69% and 87% over single-path connections.

We then explored the performance limits of TM assistance by implementing Bhandari's path formation algorithm to discover pairs of shortest disjoint paths. In this case, 86% of the connections managed to get disjoint paths, thus benefiting from TM assistance; the remaining 14% got overlapping paths. The results shown are the averages among all connections, regardless of whether they used disjoint paths or not, with 1 Gbps intra-AS links. As shown in Fig. 4.17(d), TM assistance in this case provides 11% additional gains, for an average gain of 87% over single-path connections; this is 18% higher than with the k-shortest paths algorithm with 2-shortest paths and TM assistance (Fig. 4.17(c)), since in that case we did not get so many disjoint paths. Consequently, TM assistance does make a difference when there are resources to exploit, as it consumes more aggressively the unused bandwidth in disjoint paths.

Finally, we notice a correlation between the gains of TM assistance and the overall multipath gains. TM assistance delivers roughly 10% higher performance to multipath connections, i.e., when multipath users experience 100% more throughput than single-path ones, then the TM assistance will provide 10% additional gains. As expected, multipath gains are correlated with network density, as in all figures the performance in the (denser) topologies to the right are generally higher. This experimentally validates our intuition that TM assistance performs best in dense topologies where disjoint paths are easier to find and more resources are pooled.

Discussion of results We evaluated NMCC and TM assistance mechanism in real-life domain-scale topologies, where multiple users perform simultaneously, sharing the available network resources. First, the results highlight the fairness gains that NMCC offers to the network compared to LIA, due to achieving accurate and instant friendliness. NMCC improves by up to 100% the throughput of the slowest users by reducing the throughput of the fastest and, potentially, over-aggressive multipath users, thus democratizing the sharing of network resources (Fig. 4.16.(a.2)). Second, the results demonstrate that the greedy friendliness technique increases impressively the network resource utilization in domain-

scale scenarios under different path formation policies, when network capacity is high. In the case of 2-shortest disjoint paths, multipath with greedy friendliness increases the network resource utilization of multipath without greedy friendliness by 10% on average (Fig. 4.17.(d)).

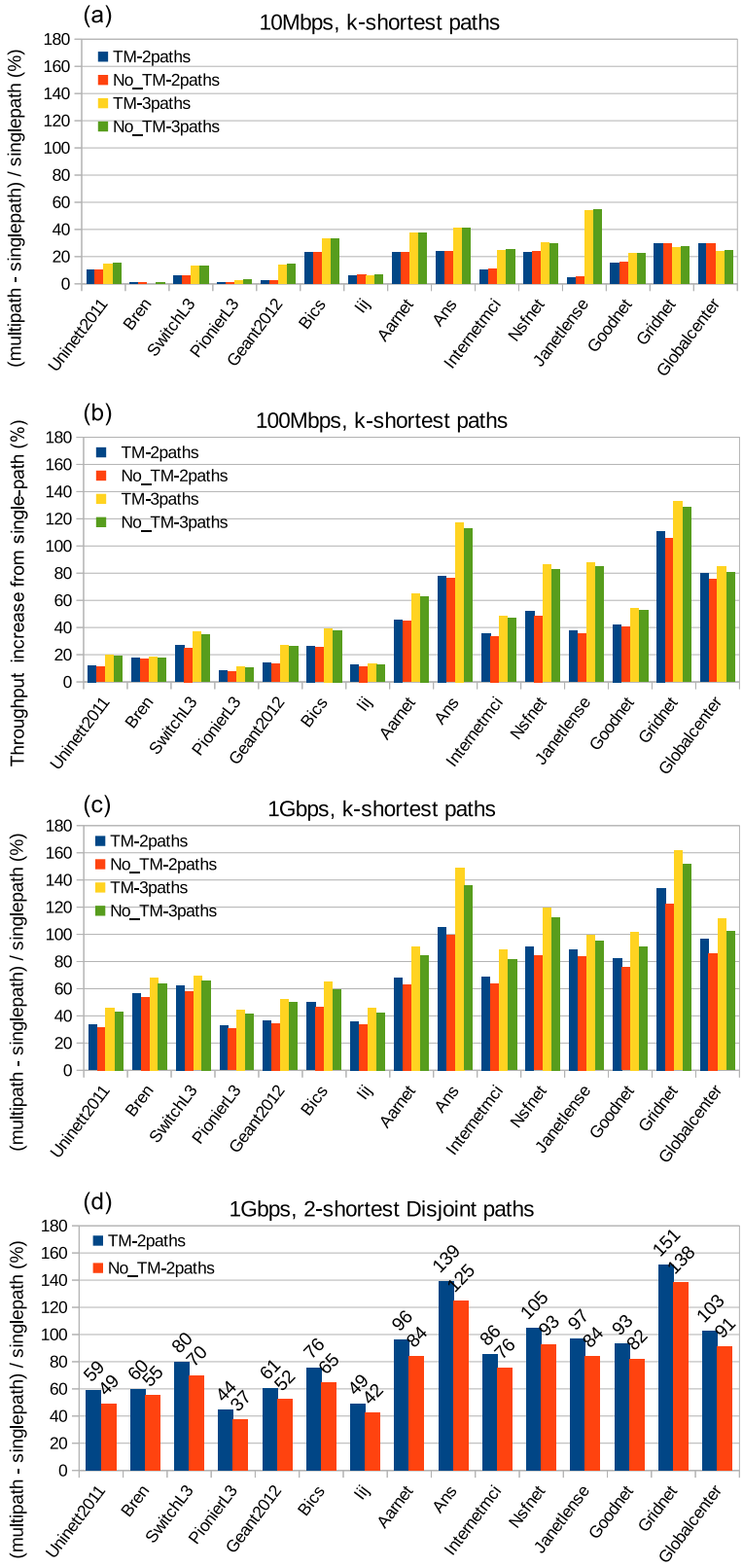


Figure 4.17: mmTP performance in the domain-scale topologies with and without TM assistance under different path formation policies and in-network link capacities: (a-c) Yen's path formation algorithm and (d) Bhandari's disjoint path formation algorithm. Topologies are plotted in ascending order by density from the left to the right.

Chapter 5

Integration of hybrid multi-flow congestion control in IP Networks

In this section we discuss the integration of our multiflow solution with the IP architecture. First, we elaborate on the installation of our topological assistance module in IP networks through the MPLS and the SDN technologies. Second, we integrate NMCC with IP networks through the Linux implementation of MPTCP and explore experimentally the performance gains that NMCC can bring to IP networks compared to the LIA, OLIA and BALIA algorithms.

5.1 Topological assistance module in TCP/IP

Our hybrid congestion control mechanism for multiflow transfers relies on an in-network module to apply path formation policies and to notify the end-hosts about shared bottlenecks. The PSI architecture is an appropriate terrain for this design, since it provides a TM function that discovers the dissemination paths and interacts with the end-hosts. We detect three essential features of PSI that facilitate our topological assistance module. First, the TM knows the physical structure of the network, so it can easily detect shared bottlenecks. Second, the path formation policy, such as finding the k-shortest disjoint paths, can be deployed at request-level, where the end-users declare their preference via meta-data or name-conventions, e.g., registered postfixes of item names. Third, when two

publish-subscribe requests are matched, the TM sends the LIPSIN identifiers directly to the applications, therefore it directly pushes the topological information to the users. Any changes to these paths, whether due to failures or load balancing decisions, require the distribution of new LIPSIN identifiers by the TM, therefore the applications are always aware of path overlaps. In order to extend our scheme to other types of networks, such as IP-based ones, we need equivalent in-network mechanisms to provide such information, as well as mechanisms to ensure that this information remains valid as routing decisions change.

5.1.1 Topological assistance in MPLS

Multi-Protocol Label Switching (MPLS) [9] is a quite popular solution for providing centralized path selection and source routing in IP networks. Currently, MPLS is primarily used to apply domain-scale traffic engineering, rather than to enhance the performance of individual connections, hence, connections are distributed to different paths according to static sharing weights for general load balancing, e.g., in case of three available paths, 33% of incoming traffic is pushed over a different path, using the Round Robin technique to map a connection to a path. Consequently, congestion control takes place at the actual end-hosts (i.e the users), while the ingress MPLS router is confined to the flow control of the available paths.

The greedy friendliness technique can be enabled by exploiting the MPLS network administrator, who discovers the paths in the MPLS cloud, assigns connections to these paths and pushes the routing information to the MPLS routers. Along with the routing information, the administrator can push to the MPLS routers information about the disjointness of the paths. Thereupon, the ingress router can act as the congestion manager of the MPLS cloud, similarly to a *Performance Enhancement Point* (PEP), like Split TCP for wireless links [79], thus becoming the end-host of a local MPLS service. When the network administrator discovers multiple paths for bulk flows and sends the corresponding labels to the ingress router, it also sends information about shared bottlenecks, as described in Sec. 4.2.1. The ingress router, that splits the connections and applies conges-

tion control in the MPLS network, can then exploit this information along with source routing to selectively engage the friendliness mechanism, thus enhancing the resource utilization of the MPLS cloud. Although this solution offers apparent gains, such as relying on the tested technology of PEPs, it also puts significant computation overhead to the ingress router, that deploys a (multipath) congestion loop per connection, thus questioning the feasibility of this design.

In order to provide a computationally feasible solution, we sketch a combination of the previous approaches that brings the efficiency of the second and the scalability of the first. We propose that the ingress MPLS router performs as a PEP, splitting the connections and practicing the congestion control within the MPLS cloud, thus offering efficient performance. We also suggest that the ingress router groups the individual incoming connections that *are sent to the same egress router*, and manages congestion control on groups instead of connections, thus reducing the computational costs. Thereupon, the congestion control of "grouped" connections is jointly managed by the ingress router that exploits a congestion control algorithm, such as NMCC and LIA, and the network-assistance of the network administrator, so as to apply domain-scale greedy friendliness (instead of connection-level). The computation overhead of this solution is caused by establishing one congestion loop per group of connections, or per egress router, and by splitting the connections, like Split-TCP, thus being acceptable. However, the technical details, such as grouping the connections and jointly performing the congestion control of multiple connections, need to be further explored.

5.1.2 Topological assistance in SDN

Software-Defined Networking (SDN) [80] is a novel networking technology that can be used to achieve similar goals to PSI, including centralized path selection. The SDN controller is equivalent to PSI's TM, being aware of the network topology and discovering the dissemination paths that are materialized by the on-path SDN switches, thus offering source routing. Nonetheless, the SDN controller does not communicate with the end-hosts, hence it cannot pass topological information to them. We can apply the same ideas as for MPLS to introduce

in-network assistance and NMCC to SDN clouds, by considering the ingress SDN switch as the congestion manager of bulk flows. When the SDN controller creates forwarding paths by sending the appropriate rules to the SDN switches, it can send information on how flows are grouped depending on path sharing to the ingress SDN router, as well as instructions on how to tag each IP header so as to implicitly select the appropriate path. The ingress SDN router will then run NMCC for each bulk flow, as in the MPLS case.

A different approach to provide topological information to the end-users is stirred by the integration schemes of SDN and MPTCP [34, 35]. In these setups, the SDN switches "sniff" the special MPTCP messages that carry the MP_JOIN option signaling the establishment of additional subflows. The switches also maintain certain state that allows the identification of subflows that belong to the same MPTCP session, in order to force path diversity deterministically. Upon this design, SDN switches can insert their unique identifiers to the MPTCP header so that the entire path can be derived at the receiving host. Then, the MPTCP sender examines the identifiers and simply deduces the existence of possible bottlenecks. However, this design may be inefficient for long wide-area paths, where path information does not fit in the MPTCP header, or network operators that do not want to disclose their internal routing decisions.

To avoid these issues, we propose an alternative design where the SDN switches insert distributively topological information to the MPTCP header, similarly to the "group_id" of the TM assistance module, so as to inform the MPTCP end-user about shared links without indicating the entire path. Specifically, the MPTCP header is extended with a field, namely, *PATH_OVERLAP*, that contains the list of the transport identifiers (e.g., destination IP addresses) of subflows competing in the same bottleneck. The field is initially empty but it is updated by any SDN switch that forwards multiple MP_JOIN messages over the same link. The SDN switch stores the destination address of each subflow during the establishment, so it can list the destination addresses of all subflows being forwarded over the same link when a new subflow is established. The topological information its delivered to the MPTCP end-user via the MP_JOIN message and then greedy

friendliness is applied.

Even though the only overhead of this extension is the special field in the MPTCP header that must be included to carry the topological information of the paths, the inherited scalability concerns described in [34, 35] remain. On the one hand, the processing overhead at the switches is insignificant since the procedure takes place only during the establishment of the subflows upon the receipt of the MP_JOIN option. In addition, there is no memory overhead compared to [34, 35] since the required state at the switches is already kept. On the other hand, the operation of the switches in large-scale networks with numerous MPTCP endpoints is questioned by the induced session state at the switches. A second weakness of the design is not being compliant with the dynamic forwarding of SDN networks. The topological information can be invalidated in case network routing is internally rearranged without MPTCP being notified (SDN rules change but MP_JOIN is not resent), thus compromising the correctness of the mechanism.

In order to avoid those issues we sketch a third integration plan where the SDN controller has a primary role in the procedure. Again, we assume MPTCP-aware SDN switches and controllers,[1] but now the controller stores the MPTCP state that associates subflows of the same multipath session, while the switches remain stateless with regard to MPTCP. In this case, the SDN switches forward the MP_JOIN messages to the controller that deduces the performance bottlenecks thus forming a centralized topological assistance module similar to PSI's TM. Upon the receipt of the MP_JOIN message the controller either stores the status of the connection when this is the first subflow to establish, or updates the connection status if an additional subflow is to be established. The status of the connection includes the (statistically) unique identifier of the connection, the IP addresses of the subflows and the paths assigned to each subflow; the latter is soft-state memory (and can be derived) for enhancing the performance during the second step. In the second step, the controller compares the new path with the established (if any) and determines the existence of shared links, following the procedure presented in Section 4.2.1. This information then is inserted in the special *PATH_OVERLAP* field

[1]Multiple SDN controllers are assumed so as to control large-scale networks.

of the MPTCP packet's header and forwarded to its original destination.

The main advantage of this approach is that topological information remains valid throughout the entire transmission, even if subflows change on-the-fly. Routing decisions and bottleneck detection are made by the SDN controller, thus offering consistency, but also rising scalability concerns. The processing overhead for the controller is not significant, since the procedure to detect shared bottlenecks is found to be relatively inexpensive (Section 4.2.1). However, the introduced memory state can be critical in case MPTCP becomes the dominant transport protocol. In order to alleviate the storage overhead, we can assume that the connection status is stored for a rather brief period of time and then is automatically dropped. The period of time needs to be long enough to aggregate the establishment of the subflows but brief enough to minimize the memory requirements. 1-2 seconds seem acceptable as MPTCP typically deploys the subflows instantly. MPTCP also allows to change the subflow address and, in turn, the established path during the lifetime of a connection, an option that is practical for mobility. Expectedly, these cases are not handled by our extended controller, therefore the endpoints will be notified to act friendly by default so as to avoid penalizing the TCP-friendliness constraint.

5.2 Normalized Multiflow Congestion Control in TCP/IP

The NMCC is compatible with MPTCP, since it is applied at the end-users and exploits information that is already available to MPTCP-end hosts: the RTTs, congestion window sizes and the congestion states of the subflows. In the following we discuss the integration details and challenges that we faced while embedding NMCC in the Linux implementation of MPTCP.[2] Then, we exploit the Linux implementation of MPTCP in order to evaluate the performance gains of NMCC compared to the LIA, OLIA and BALIA algorithms, especially in terms of latency in converging to TCP-friendliness.

[2]https://www.multipath-tcp.org/

5.2.1 Design and implementation in Linux MPTCP

NMCC can be seamlessly integrated in the Linux kernel, as TCP (and MPTCP) offers pluggable congestion control via a special handler interface [81, 82]. Therefore, our code simply overrides the handlers that tackle the estimation of the window increase upon the successful delivery of an ACK and the estimation of the SS threshold when a congestion event takes place. The information required to calculate m and m_t, such as the RTT, the congestion window and the SS threshold of the subflows, is available to each subflow, hence the implementation is direct.

Our primary challenge was to convert the normalized window growth algorithm from RTT-based to packet-based. This procedure is described in Section 4.2.2, where we elaborate on an algorithm that translates inflated RTTs, the technique that NMCC exploits to implement friendliness, to reduced window increases, the typical method to control MPTCP's aggressiveness.

A second challenge was to avoid integer overflow while applying m, which is a real number, to the congestion window of TCP, which is a positive natural number with packet-level granularity. In the CA phase, TCP exploits an increase threshold, which is the counter of the ACKed packets since the last window increase, to trigger window growth. We found in our experiments that the accuracy of this method is acceptable for windows larger that 10 packets, which is also the minimum allowed window size of Linux MPTCP. In the SS state though, the TCP increase algorithm is memoryless and does not provide an increase threshold that we can exploit, thus penalizing NMCC's performance. Therefore, we implemented a probabilistic increase pattern, where a random number in the range [0, 1) is drawn from a uniform distribution on every ACK receipt. The fractional increase of the normalized window growth is rounded up when its fractional part is higher than the random number. Although producing random numbers is relatively cheap, it can induce measurable overhead to resource constrained devices. In this case, a stateful increase algorithm akin to CA's increase threshold can be also implemented, at the expense of adding one more variable per subflow to keep the SS increase threshold.

5.2.2 Evaluation of MPTCP convergence with NMCC

To investigate the convergence time of the proposed multipath algorithms, we conducted experiments using the Linux kernel implementation of MPTCP. We provide below a detailed description of our setup where we investigate the correlation between MPTCP's convergence to friendliness and three network parameters: propagation delay, bandwidth and error rate.

To avoid being biased by the hardware and software used, we configured a testbed where hardware resources are shared by multipath and single-path connections. We cloned a virtual machine (VM) that runs MPTCP v0.91 on Linux kernel v.4.1.37 and hosted two clones, as clients, in a host machine using VirtualBox,[3] reserving the exact same resources for both VMs (2 cores at 4 GHz and 2 GB RAM). A third cloned VM, acting as the server, was placed in a different machine, ending with the topology of Fig. 5.1. The configuration and IP assignment of these VMs allows the establishment of two subflows between the client-VMs and the server-VM, but MPTCP is enabled only at one client-VM, as shown in the figure.

In each experiment we simultaneously deployed 4 iperf[4] connections from each client-VM, thus creating 2 single-path flows and 4 multipath subflows over each path. Each run lasted 30 minutes and was repeated 30 times. The netem[5] tool was used to emulate different values of propagation delay, bandwidth and packet loss rate. Different configurations of delay, bandwidth and error rate are expected to have an influence on the convergence time of the congestion algorithms, hence we deployed TCP Reno in the MP-disabled host and MPTCP with one of NMCC, LIA, OLIA and BALIA in the MP-enabled host. Notice, that any TCP-friendly TCP variant, such as CUBIC TCP, can be used instead of Reno without influencing the results. Finally, we tested MPTCP using the (unfriendly) Uncoupled algorithm, where each subflow behaves as a Reno connection, to establish a performance baseline.

[3] https://www.virtualbox.org/
[4] https://iperf.fr/
[5] http://man7.org/linux/man-pages/man8/tc-netem.8.html

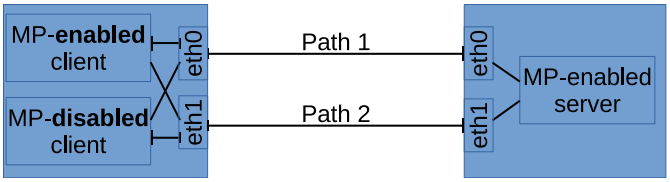


Figure 5.1: LAN testbed topology for evaluating MPTCP in Linux. Two client VMs are co-hosted in the same client node.

Impact of propagation delay

We first investigate the impact of path propagation delay on the convergence of MPTCP. We configure three setups with equal RTTs in both paths, namely, 10 ms, 100 ms and 200 ms, when congestion level is minimum. The bandwidth was set to 8 Mbps per path (2 Mbps per connection) with no additional packet drops from netem. The results are plotted in Fig. 5.2, where rows and columns depict the performance of different algorithms with different delays, respectively. Specifically, each plot illustrates the average bandwidth share of the MPTCP and TCP RENO flows normalized to the overall traffic for every second of the experiment. As expected, we observe that NMCC and MP-RENO are not affected by path latency, converging very fast to friendliness and unfriendliness, respectively. LIA, OLIA and BALIA on the other hand, exhibit significant convergence time that reaches 600 s and 1300 s with 10 ms and 200 ms latencies, respectively. As expected, the higher the network delay, the more convergence time is required, as congestion feedback is more frequent for lower latency paths allowing faster convergence. It is important to notice that LIA and OLIA are TCP-friendly in the long run (BALIA is over-aggressive), but they do not achieve fairness until some minutes into a session, thus being unfriendly for connections that last less than 600 s. Finally, the resource utilization is roughly 1% more than MP-RENO for all three friendly algorithms.

Impact of bandwidth

We next investigate the impact of bandwidth on the convergence of MPTCP. Again, we configure three setups with the same transfer rate in each path, namely,

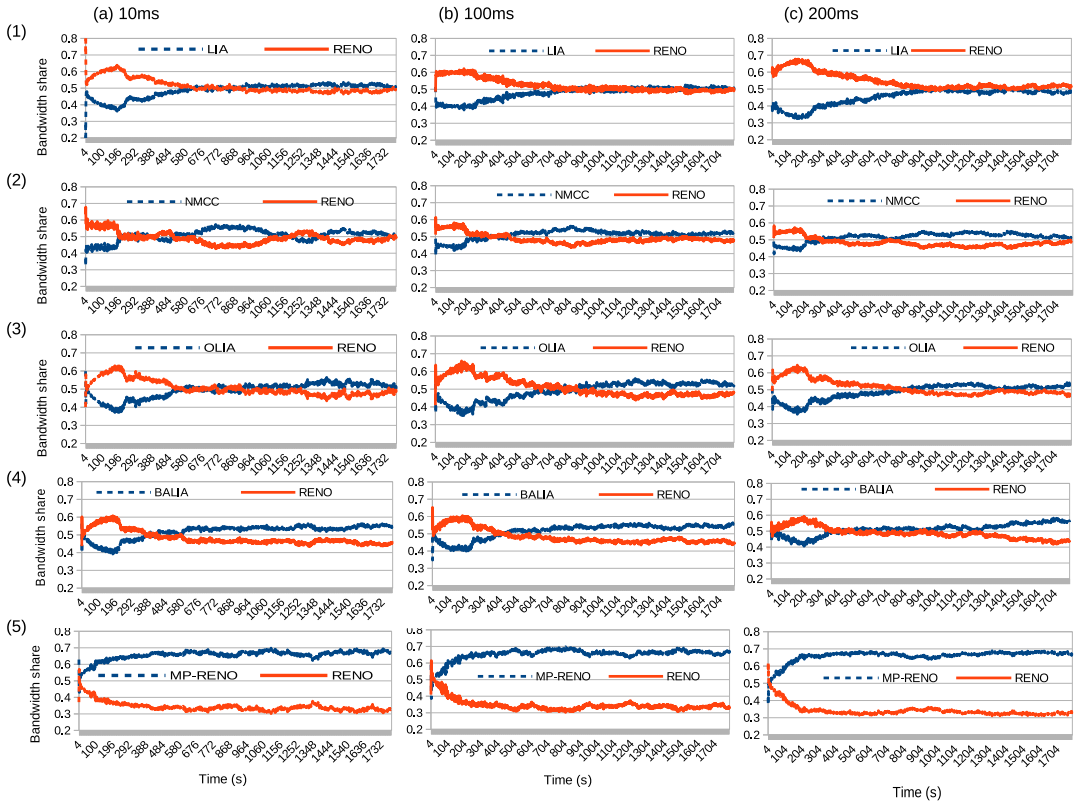


Figure 5.2: MPTCP performance in the LAN testbed with two disjoint paths, different congestion algorithms (rows 1-5) and different path latencies (columns a-c).

4, 8 and 16 Mbps, setting the RTT to 100 ms (when congestion level is minimum) and no additional packet loss by the netem tool. The results are plotted in Fig. 5.3, where rows and columns depict the performance of different algorithms with different transfer rates, respectively. For space reasons we omit the case of 8 Mbps which was already presented in Fig. 5.2. We observe that NMCC is not affected by the amount of available bandwidth, as it converges to stability as fast as MP-RENO in both narrow and wide links. However, LIA, OLIA and BALIA are struggling in narrow links, being unfriendly for roughly 1400 s, 800 s and 1100 s, respectively, until their bandwidth share is stabilized. MPTCP by design is a little more aggressive in wide links as MP-RENO gains more than the theoretical limit of 0.68%, which, in turn, explains the relative over-aggressiveness of OLIA and NMCC, however BALIA is again noticeably over-aggressive getting 56% in wide links. Finally,

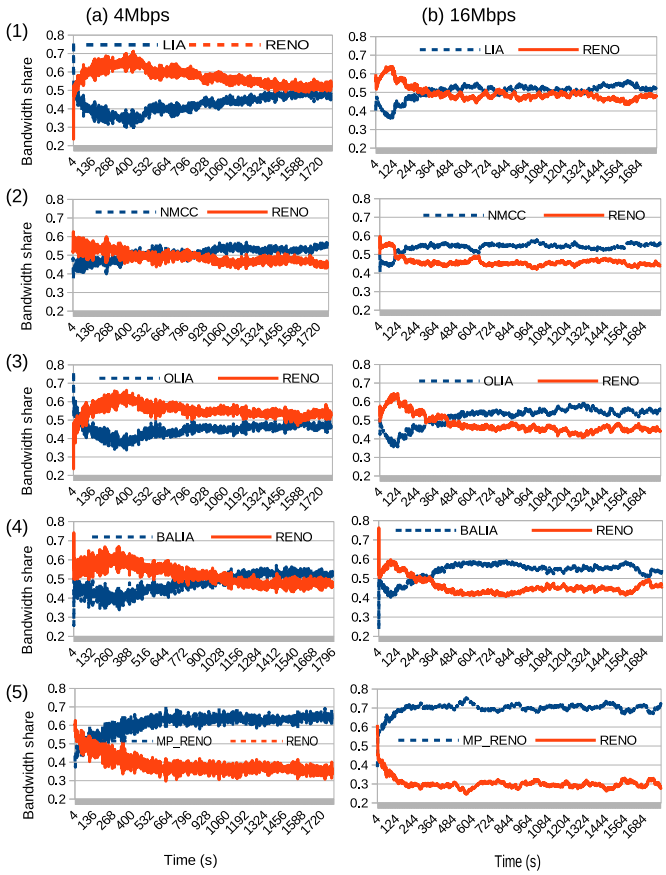


Figure 5.3: MPTCP performance in the LAN testbed with two disjoint paths, different congestion algorithms (rows 1-5) and different transfer rates (columns a,b).

although we repeated each experiment 30 times, the plotted performance lines for narrower links present more fluctuation, implying that the increased intensity of flow competition in narrow links challenges convergence. The resource utilization is 1-2% more than MP-RENO for all three friendly algorithms.

Impact of packet loss

We then investigate the impact of random packet loss on the convergence of MPTCP. We configure three setups with the same packet drops rate over each path, namely, no loss, 0.0001% and 0.1%, setting the bandwidth to 8 Mbps and the RTT to 100 ms (when congestion level is minimum). The losses follow a uniform distribution, thus engaging the Fast Recovery state of TCP in most cases. A

0.1% packet loss can produce multiple "false" congestion events within a second; however, connections still reach the maximum transfer rate, thus maintaining high resource utilization and vigorous flow competition. Figure 5.4 shows the results obtained, where each column of plots presents MPTCP performance under a different segment loss rate. Again, for space reasons we omit the case of zero loss, which is presented in Fig. 5.2. We observe that NMCC is not affected by the error rate, as it converges to stability as fast as MP-RENO in all cases. In contrast, LIA and OLIA present the highest deviation from the fair share in the beginning of the experiment when packet drops are 0.0001%, although their convergence time is not altered with regard to the zero loss scenario. On the other hand, in case of frequent losses (0.1% loss rate) all algorithms are stabilized instantly. The frequent losses probably accelerate the detection of changed network conditions and the algorithms adapt rapidly. However, note that both LIA and OLIA are consistently less aggressive than needed, getting 47% and 48% of bandwidth, respectively. Oppositely, NMCC is slightly over-aggressive (53%) and BALIA achieves perfect sharing.

Impact of path mismatch

We finally investigate the impact of path mismatch on the convergence of MPTCP. We configure three setups with mismatched paths in terms of bandwidth, error rate or propagation delay. Figure 5.5 shows the results obtained for paths with different error rates (column a) and different RTTs (column b). We do not include the plots with different bandwidths, since the results are similar to Fig. 5.2(b). Bandwidth is set to 8 Mbps, RTT to 100 ms (when congestion level is minimum) and error rate to 0%, unless stated otherwise. In both setups NMCC converges roughly instantly, thus offering friendliness from the beginning of the session. In contrast, on paths with different error rates LIA, OLIA and BALIA present a rather slow convergence, requiring roughly 400 s, 900 s and 800 s until their throughput is stabilized. In paths with different latencies, LIA, OLIA and BALIA present even slower convergence, needing approximately 1000 s until they are stabilized. In addition, when the error rates are different, LIA leads to a moderate resource

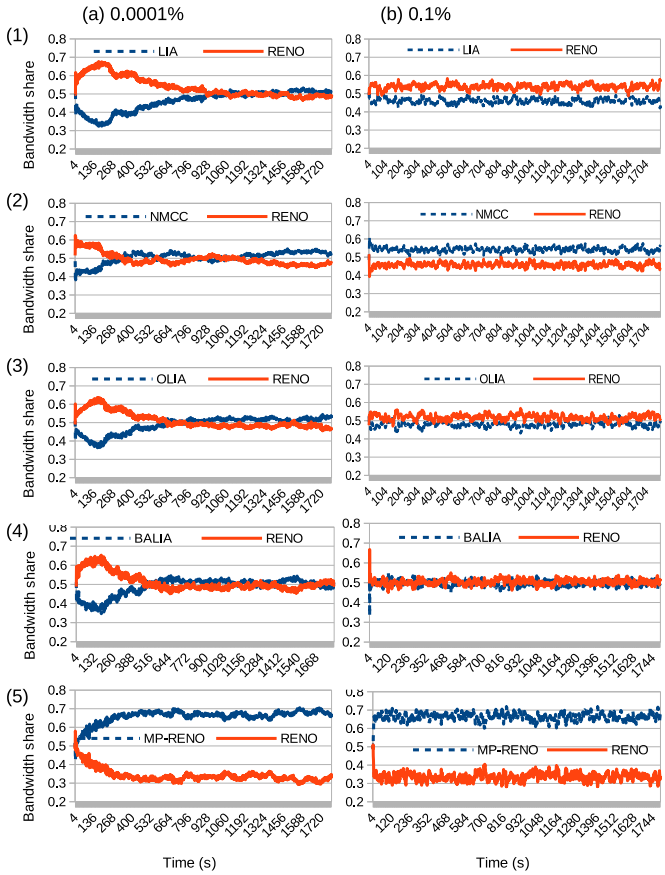


Figure 5.4: MPTCP performance in the LAN testbed with two disjoint paths, different congestion algorithms (rows 1-5) and different packet drop rates (columns a,b).

underutilization, grasping 94% of the available bandwidth, while OLIA grasps only 42%, compromising the throughput gains promised by MPTCP. Again, when error rate is high (even in one path only) BALIA offers perfect sharing, but only after 900 s.

Discussion of results

The evaluation of MPTCP unveiled rather interesting results about the accuracy and the speed with which the different algorithms achieve TCP-friendliness. In general, we validate that friendliness towards single-path connections is respected by LIA, OLIA, BALIA and NMCC, however the efficiency of reaching

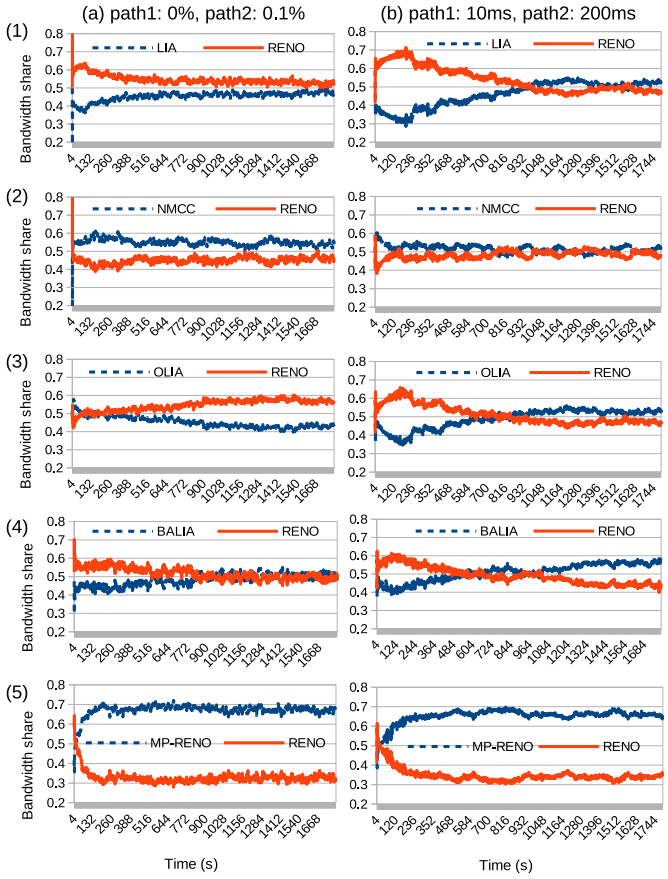


Figure 5.5: MPTCP performance in the LAN testbed with two disjoint paths, different congestion algorithms (rows 1-5) and mismatched paths in terms of error-rate (column a) and RTT (column b).

it exhibits impressive differences. First, we point out that MPTCP in Linux inherently introduces some convergence latency; even MP-RENO that does not include any friendliness mechanism requires several seconds to stabilize. Second, we infer that NMCC converges instantly as it is stabilized simultaneously with MP-RENO in all experiments, thus not inducing any temporal overhead. Third, the TCP-friendliness of LIA, OLIA and BALIA is severely questioned since their converge latency lays in the range of roughly 200 s (Fig. 5.2(a), 5.3(b)) to 1000 s after MPTCP (Fig. 5.2(c), 5.3(a)). Their performance is better in wide links with low propagation delay, deteriorating measurably as paths get narrower and longer. However, all algorithms converge instantly when both paths include high error-rate

links (Fig. 5.4(b)) probably because the frequent congestion events (timeouts and triple duplicate ACKs) feed the congestion algorithms faster than in the error-free cases.

In terms of accuracy in TCP-friendliness, NMCC exhibits less deviation in different setups compared to LIA, OLIA and BALIA. Specifically, NMCC is slightly more aggressive than single-path in all scenarios, getting roughly 52-55% of resources. This mild over-aggressiveness is an inherent characteristic of NMCC when deployed in disjoint paths since it normalizes the performance of the fastest available path, a non-fixed characteristic over prolonged periods, and gains a slight performance advantage due to path plurality. On the other hand, the performance of LIA, OLIA and BALIA varies, being faster than single-path in some scenarios (Fig. 5.3(b)) and slower in others (Fig. 5.3(a)), indicating that their friendliness is visibly affected by network conditions. When paths present significant error rate, BALIA constitutes the most accurate algorithm, splitting the network resources perfectly even.

Chapter 6

Discussion and future work

In this Chapter we discuss apparent issues that multiflow faces in ICN. At the same time, we outline our future research targets that integrate additional ICN-specific features into multflow transfers.

6.1 Multisource and multipath made easy in PSI

mmTP combines well-known content distribution techniques into a single transport protocol: multisource downloading is widely adopted by P2P applications [4] and multipath transfer is a well-established research topic. What mmTP adds is the exploitation of the PSI centralized path selection property and the LIPSIN explicit-routing scheme, so as to not only support multisource and multipath transport, but also keep network operation simple, provide a generic interface for content delivery and utilize network storage. In this section we explain how these aspects of PSI simplify multisource and multipath transport.

First, the separation of the core network functions in PSI along with the choice of explicit-routing results in simple signaling and stateless Forwarding Nodes (FNs). Routing is orthogonal to forwarding: FNs operate in a stateless manner by using in-packet LIPSIN FIDs, without any routing state, knowledge of the actual data path or the transfer state. These gains are achieved in exchange for the additional delay required at the slow-path rendezvous phase, where the initial subscription is resolved and the TM computes suitable data paths. We acknowledge that the

centralized nature of path computation in the TM raises scalability concerns. Yet, we believe that TM functionality will be placed in multiple nodes with enhanced capabilities that meet the processing requirements [72], thus utilizing in-network computation resources. In addition, this design is well-aligned with on-going developments in the area of Software Defined Networking (SDN), where the process of discovering and installing paths in a centralized manner is not found to be a performance bottleneck [34].

Second, the separation of routing and forwarding allows the transparent implementation of multisource and multipath services. The FIDs that the TM delivers to the endpoints are a set of distinct options for requesting data. These options may involve different publishers and/or different paths, but this information is concealed from the hosts. The subscriber evaluates in real-time the performance of each option (i.e. path) and adjusts the amount of data to be delivered through it accordingly. Hence, mmTP provides a generic interface, transparently supporting any combination of multisource and/or multipath services. The nature of the service will be decided by the TM, according to a network domain's policies and goals. Essentially, the TM selects the paths, the FNs realize those paths and the subscriber controls each path's utilization.

Third, explicit-routing and centralized path selection assure diversity of transmission routes and friendliness towards single-flow connections. Multipath protocols are commonly compromised by IP's hop-by-hop routing, which can force paths from multihomed hosts to converge over the same bottleneck links. In contrast, in PSI the FIDs indicate dissemination paths that are predefined, hence avoiding convergence at bottleneck links. At the same time, TCP-based multisource solutions, such as BitTorrent, are not friendly towards single-flow TCP when operating over the same link, while others, like MTCP with LIA, throttle bandwidth aggregation in order to avoid starving standard TCP flows. In PSI, path formation is rather agile [44], i.e, can be made to select only disjoint paths, thus allowing each subflow to operate over distinct links. Alternatively, exploiting greedy friendliness, the TM informs the subscriber when many subflows perform over the same network resources, so as to appropriately tune their congestion win-

dow management. All in all, there are several ways to enforce friendliness among end-to-end connections in PSI, undertaken either by the TM or by the end-hosts.

6.2 Delay-based end-to-end congestion detection in ICN

On-path packet-level caching in ICN can be detrimental to end-to-end RTT-based congestion control schemes, since packets arriving from the cache exhibit lower delays than packets arriving from the content source. Packet caches can cause serious trouble to traffic control protocols using RTTs as congestion indicators. When random content packets are stored in the cache, the RTT greatly fluctuates depending on whether a given packet arrived from the cache or from the content source, causing spurious timeouts and degrading transport layer performance. This does not occur in typical IP-based object caches, where the cache either holds the entire object or the object is fetched from the source without any RTT fluctuations. Authors in [83] find that roughly 2-3% of Web requests are aborted before they are fully downloaded, with the total volume of downloaded bytes until aborted reaching 30% of the entire traffic in some cases. Therefore, in a packet-level cache we can expect various partial objects to reside in each cache, leading to such RTT-related issues.

Several solutions have been proposed so far, most of them are designed for NDN networks where the problem is intensified due to the dynamic selection of sources; the on-path routers make forwarding decisions on-the-fly, thus exploiting different and unknown sources throughout the transmission. In the ICP protocol [84] the authors exploit the weighted average of a history of the last N (typically 20) RTT measurements in order to provide a reliable estimator. In CCTCP [85] the authors propose the installation of individual RTT estimators per source and predict the location of chunks before actually emitting the request. While these solutions perform relatively well under difficult conditions, they constitute stochastic solutions that mitigate the impact of the problem without providing a solid solution. A deterministic design can be inspired by [56], where the *Route*

labeling technique explicitly defines the exploited path per chunk, since on-path routers distributively insert path information in each packet. Nevertheless, this approach stresses the in-network operation, hence it is expected to reduce the scalability of the service. A light-weight variation of this technique is presented in the following section where we discuss the adaptation of mmTP for the NDN architecture.

In order to deliver a complete and scalable solution we propose rethinking the problem from the perspective of in-network caching. Introducing novel cache replacement policies can eliminate the problem of RTT variance, for instance, OPC [63, 86] that stores exclusively the initial part of any content item, from the first to the n-th packet with no gaps, can substantially limit the on-path multi-sourcing. The design can be further evolved by assuming that caches explicitly inform the congestion control endpoints about the origin of a packet (e.g., source identifier) and the state of the cache (e.g., the subsequent requests that can be found cached). The information will assist the end-users in adjusting their sending rate accordingly to the current source location by either establishing dedicated RTT-estimators per source or ignoring the cached responses. While this design is not sufficient for the NDN architecture, it can offer enhanced performance in PSI. Specifically, the NDN endpoints cannot associate a packet with the exploited route, therefore the detection of congestion level in the different parts of the network and, in turn, the separate flow control per path is inaccurate without in-network mechanisms. Acknowledging that end-to-end congestion control in NDN can be inefficient and that active in-network congestion control (see Section 3.3.2) penalizes the scalability of the network operation, constitutes a strong reality check for the NDN architecture.

6.3 mmTP design for NDN

The mmTP protocol is a complete transport[1] solution that exploits the many-to-one communication model of ICN networks, as well as the centralized path

[1]According to the TCP/IP stack mmTP can be classified as a cross-layer solution, spanning over network and transport layers.

selection and source routing of PSI (or SDN as discussed in Section 5.1.2) in order to provide multiple paths to many locations while practicing end-to-end congestion control with greedy friendliness. The NDN architecture also offers inherently many-to-one communications and supports multipath among two endpoints, thus satisfying one of the two requirements of mmTP. In order to use mmTP over NDN though, the distributed hop-by-hop in-network routing must allow the establishment of individual subflows that can be independently monitored and controlled by the end-hosts.

In NDN a request for a name is considered to match any piece of content whose name has the requested name as a prefix, for example, `/aueb.gr/a` can be matched by a content item named `/aueb.gr/a/_v1/_s1`, which could mean the first segment (`_s1`) of the first version (`_v1`) of the requested data (`a`). Assuming that content `/aueb.gr/a` resides at multiple locations, then at the on-path routers we have multiple entries in the FIB leading to those locations, hence the received Interests can be distributed among the different FIB entries materializing a multiflow transmission. The routing information about the individual paths can be created by the network distributively and transmitted to the endpoints through the content name itself, thus maintaining consistency with the NDN principles. In detail, the "branching" routers can append a subflow-specific postfix, that identifies the route followed by the packet, to each received Data packet, for example, `/aueb.gr/a/_v1/_s1/_R1` implies that the first segment of the first version of the requested data was transmitted over router R via the first FIB entry (`_R1`) for that content. Then, the Data packet will be pushed successfully until its the destination since the registered name prefix in the PIT will be matched with the Data packet's content name as well, albeit the later is "extended".

Our solution is quite similar to the *Route labeling* technique discussed in [56] but being placed only at the branching nodes of the dissemination paths is expected to impose significantly less overhead, thus maintaining scalability. It also serves as an in-network mechanism that delivers topological information to the endpoints, thus offering the implementation of the greedy friendliness concept in order to enhance network performance. The main weakness of the design is that it disallows

the (cache everything) on-path caching, where Data packets are stored at the on-path routers in order to respond faster to future requests of the same information piece. Contrary to routing that allows longest-prefix matching, caching requires absolute name matching, therefore disallowing on-path caching at routers between the "branching" routers and the receivers.

Chapter 7

Conclusions

ICN has emerged as a promising candidate for shaping Future Internet architectures. With regard to multiflow content delivery, ICN presents inherent advantages, such as loose coupling of peers, content-based routing and ubiquitous packet-level caching, that assist in the realization of efficient content distribution patterns. The PSI architecture, an ICN instance based on the publish-subscribe paradigm, allows natively the establishment of multiple paths among two communication endpoints, as well as ensures the disjointness of those paths through stateless source routing and centralized paths management. Oppositely to TCP/IP networks, PSI supports multipath without the need for multihoming and assures path divergence without overwhelming routing state at in-network nodes.

We exploited these features and designed and implemented mmTP, the first multipath and multisource transport protocol for PSI. mmTP combines well-known content distribution techniques in a single protocol, without requiring complicated network signaling or adding state to routers. We implemented a prototype of mmTP in the PSI architecture prototype and evaluated its performance on PlanetLab. Our results verify the effectiveness of both multisource and multipath delivery, in terms of throughput, load balancing and resilience to network failures.

mmTP also introduces a novel hybrid congestion control algorithm for multiflow transport that consists of the end-to-end NMCC algorithm and an in-network assistance mechanism. Our design offers friendliness to single-path connections using TCP-like congestion control, while increasing the utilization of network re-

sources. It achieves this by detecting shared physical bottlenecks and managing aggressiveness accordingly, a scheme the we call *greedy friendliness*. We have implemented the congestion control algorithm and evaluated its performance gains in several topological and traffic scenarios, using both direct experimentation in a LAN environment and packet-level simulations in a WAN environment. Our results verify the anticipated gains, showing that the average network throughput with mmTP yields up to 160% and 15% increase compared to single-path and multipath without the greedy friendliness mechanism, respectively.

Finally, we highlighted a weakness of the LIA, OLIA and BALIA multipath congestion control algorithms, namely, the need for long time periods until they achieve a TCP-friendly state. Contrary to previous studies that focus on long-term results, we monitored the instantaneous performance of connections throughout the transmission, finding that the convergence latency is in the order of minutes, thus questioning the effectiveness of these algorithms. The NMCC congestion control algorithm exploits a deterministic rate control design that exhibits zero convergence delay, which was shown to offer fair resource sharing instantly and consistently. Using the Linux implementation of MPTCP and the htsim simulator, we explored the performance of NMCC under a set of well-known benchmark scenarios, validating that TCP-friendliness is substantially enhanced while responsiveness and load balancing in the long run are comparable to LIA, OLIA and BALIA.

Appendix A

Acronyms

ACK Acknowledgment

AS Autonomous System

BALIA Balanced Linked Adaptation

CA Congestion Avoidance

CCN Content-Centric Networking

CDN Content Delivery Network

CMT Concurrent Multipath Transfer

DASH Dynamic Adaptive Streaming over HTTP

DiffServ Differentiated Services

DRR Deficit Round Robin

ECMP Equal Cost MultiPath

ECN Explicit Congestion Notification

FIB Forwarding Interest Base

FID Forwarding Identifier

FN Forwarding Node

HTTP/2 Hypertext Transfer Protocol Version 2

HMAC Hash-based Message Authentication Code

ICN Information-Centric Networking

IP Internet Protocol

ISP Internet Service Provider

LIA Linked Increase Algorithm

LS Label-Switched Path

LDP Label Distribution Protocol

mmTP multipath and multisource Transport Protocol

MPLS Multiprotocol Label Switching

MPTCP Multipath-TCP

MSS Maximum Segment Size

MTU Maximum Transfer Unit

NDN Named Data Networking

NMCC Normalized Multiflow Congestion Control

OLIA Opportunistic Linked Increase Algorithm

OOD Out-of-Order Delivery

PIT Pending Interest Table

PSI Publish Subscribe Internetworking

QoE Quality of Experience

QoS Quality of Service

PEP Performance Enhancement Proxy

RENE Rendezvous Network

RN Rendezvous Node

RTT Round Trip Time

RTO Retransmission Time Out

SCTP Stream Control Transmission Protocol

SS Slow Start

TM Topology Manager

TCP Transmission Control Protocol

URL Universal Resource Locator

VM Virtual Machine

Bibliography

[1] J. Qadir, A. Ali, K.-L. A. Yau, A. Sathiaseelan, and J. Crowcroft, "Exploiting the power of multiplicity: a holistic survey of network-layer multipath," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, 2015.

[2] O. Bonaventure and S. Seo, "Multipath TCP deployments," *IETF Journal*, vol. 12, no. 2, 2016.

[3] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, "Design, implementation and evaluation of congestion control for multipath TCP," in *Proc. USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, Boston, MA, USA, March 2011.

[4] S. Androutsellis-Theotokis and D. Spinellis, "A survey of peer-to-peer content distribution technologies." *ACM Computing Surveys*, vol. 36, no. 4, 2004.

[5] A.-M. K. Pathan and R. Buyya, "A taxonomy and survey of content delivery networks," vol. 4, 2007.

[6] Cisco, "Visual networking index: Global mobile data traffic forecast update, 2016-2021 white paper," https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html, accessed: 2018-2-15.

[7] ——, "Global cloud index: Forecast and methodology, 2016-2021 white paper," https://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.html, accessed: 2018-3-2.

[8] M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, and L. Jones, "SOCKS protocol version 5," RFC 1928, March 1996.

[9] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus, "Requirements for traffic engineering over MPLS," RFC 2702, September 1999.

[10] K. Kirkpatrick, "Software-defined networking," *Communications of the ACM*, vol. 56, no. 9, 2013.

[11] B. Cohen, "Incentives build robustness in BitTorrent," in *Proc. Workshop on Economics of Peer-to-Peer Systems*, Berkley, CA, USA, June 2003.

[12] A. Ford, C. Raiciu, M. Handley, S. Barre, and J. Iyengar, "Architectural guidelines for multipath TCP development," RFC 6182, March 2011.

[13] C. Xu, J. Zhao, and G.-M. Muntean, "Congestion control design for multipath transport protocols: a survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, 2016.

[14] J. Widmer, R. Denda, and M. Mauve, "A survey on TCP-friendly congestion control," *IEEE Network*, vol. 15, no. 3, May 2001.

[15] F. Kelly and T. Voice, "Stability of end-to-end algorithms for joint routing and rate control," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 2, 2005.

[16] R. Khalili *et al.*, "Opportunistic linked-increases congestion control algorithm for MPTCP," Internet-Draft, 2015. [Online]. Available: https://www.ietf.org/archive/id/draft-khalili-mptcp-congestion-control-05.txt

[17] Q. Peng, A. Walid, J. Hwang, and S. H. Low, "Multipath TCP: Analysis, design, and implementation," *IEEE/ACM Transactions on Networking*, vol. 24, no. 1, 2016.

[18] G. Xylomenos *et al.*, "A survey of information-centric networking research," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 2, 2014.

[19] P. Jokela, A. Zahemszky, S. Arianfar, P. Nikander, and C. Esteve, "LIPSIN: line speed publish/subscribe internetworking," in *Proc. ACM SIGCOMM Conference*, Barcelona, Spain, August 2009.

[20] N. Fotiou, Y. Thomas, V. A. Siris, and G. C. Polyzos, "Security requirements and solutions for integrated satellite-terrestrial information-centric networks," in *Proc. IEEE Advanced Satellite Multimedia Systems Conference (ASMS) Workshop on Signal Processing for Space Communications Workshop (SPSC)*, Livorno, Italy, September 2014.

[21] G. Parisis and D. Trossen, "Filling the gaps of unused capacity through a fountain coded dissemination of information," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 18, no. 1, 2014.

[22] T. Stockhammer, "Dynamic adaptive streaming over HTTP: standards and design principles," in *Proc. ACM Multimedia Systems Conference (MMSys)*, Scottsdale, Arizona, USA, November 2011.

[23] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, "TCP extensions for multipath operation with multiple addresses," RFC 6824, January 2013.

[24] C. Hopps, "Analysis of an equal-cost multi-path algorithm," RFC 2992, November 2000.

[25] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley, "Improving datacenter performance and robustness with multipath TCP," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, 2011.

[26] R. Stewart, "Stream control transmission protocol," RFC 4960, September 2007.

[27] J. R. Iyengar, P. D. Amer, and R. Stewart, "Concurrent multipath transfer using SCTP multihoming over independent end-to-end paths," *IEEE/ACM Transactions on networking*, vol. 14, no. 5, 2006.

[28] G. Papastergiou *et al.*, "De-ossifying the internet transport layer: A survey and future perspectives," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 1, 2017.

[29] C. Raiciu *et al.*, "How hard can it be? designing and implementing a deployable multipath TCP," in *Proc. USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, Lombard, IL, April 2012.

[30] M. Handley, "Why the internet only just works," *BT Technology Journal*, vol. 24, no. 3, 2006.

[31] C. A. Sunshine, "Source routing in computer networks," *ACM SIGCOMM Computer Communication Review*, vol. 7, no. 1, 1977.

[32] F. Le Faucheur *et al.*, "Multi-protocol label switching (MPLS) support of differentiated services," Tech. Rep.

[33] B. Sonkoly, F. Németh, L. Csikor, L. Gulyás, and A. Gulyás, "SDN based testbeds for evaluating and promoting multipath TCP," in *Proc. IEEE International Conference on Communications (ICC)*, Sydney, Australia, June 2014.

[34] S. Zannettou, M. Sirivianos, and F. Papadopoulos, "Exploiting path diversity in datacenters using MPTCP-aware SDN," in *Proc. IEEE Symposium on Computers and Communications (ISCC)*, Messina, Italy, June 2016.

[35] M. Sandri, A. Silva, L. A. Rocha, and F. L. Verdi, "On the benefits of using multipath tcp and openflow in shared bottlenecks," in *Proc. IEEE International Conference on Advanced Information Networking and Applications (AINA)*, Gwangju, Korea, March 2015.

[36] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. N. Rao, "Improving web availability for clients with MONET," in *Proc. USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, Berkley, CA, USA, May 2005.

[37] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "Resilient overlay networks," *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 1, 2002.

[38] Q. Chen, R. Xie, F. R. Yu, J. Liu, T. Huang, and Y. Liu, "Transport control strategies in named data networking: A survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, 2016.

[39] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, and L. Zhang, "A case for stateful forwarding plane," *Elsevier Computer Communications*, vol. 36, no. 7, 2013.

[40] C. Stais, Y. Thomas, G. Xylomenos, and C. Tsilopoulos, "Networked music performance over information-centric networks," in *Proc. IEEE International Conference on Communications Workshops (ICC)*, Budapest, Hungary, June 2013.

[41] V. A. Siris, Y. Thomas, and G. C. Polyzos, "Supporting the IoT over integrated satellite-terrestrial networks using information-centric networking," in *Proc. IFIP New Technologies, Mobility and Security (NTMS)*, Larnace, Cyprus, November 2016.

[42] T. De Cola *et al.*, "Network and protocol architectures for future satellite systems," *Now Publisher Inc. Foundations and Trends® in Networking*, vol. 12, no. 1-2, 2017.

[43] C. Ververidis *et al.*, "Experimenting with services over an information-centric integrated satellite-terrestrial network," in *Proc. IEEE Future Network and Mobile Summit (FutureNetworkSummit)*, Lisboa, Portugal, July 2013.

[44] Y. Thomas, P. A. Frangoudis, and G. C. Polyzos, "QoS-driven multipath routing for on-demand video streaming in a publish-subscribe internet," in *Proc. IEEE International Conference on Multimedia and Expo (ICME) Workshop on Multimedia Streaming in information-Centric networks (MuSiC)*, Turin, Italy, June 2015.

[45] Y. Ren, J. Li, S. Shi, L. Li, G. Wang, and B. Zhang, "Congestion control in named data networking - A survey," *Elsevier Computer Communications*, vol. 86, 2016.

[46] J. Postel, "Transmission control protocol," RFC 793, September 1981.

[47] A. Afanasyev, N. Tilley, P. Reiher, and L. Kleinrock, "Host-to-host congestion control for TCP," *IEEE Communications surveys & tutorials*, vol. 12, no. 3, 2010.

[48] M. Allman, V. Paxson, and E. Blanton, "TCP congestion control," RFC 5681, September 2009.

[49] M. Honda, Y. Nishida, L. Eggert, P. Sarolahti, and H. Tokuda, "Multipath congestion control for shared bottleneck," in *Proc. International Workshop on Protocols for Future, Large-Scale and Diverse Network Transports (PLFD-NeT)*, Tokyo, Japan, May 2009.

[50] H. Han, S. Shakkottai, C. V. Hollot, R. Srikant, and D. Towsley, "Multi-path TCP: a joint congestion control and routing scheme to exploit path diversity in the internet," *IEEE/ACM Transactions on networking*, vol. 14, no. 6, 2006.

[51] Y. Cao, M. Xu, and X. Fu, "Delay-based congestion control for multipath TCP," in *Proc. IEEE International Conference on Network Protocols (ICNP)*, Austin, TX, USA, October 2012.

[52] M. Becke, T. Dreibholz, H. Adhari, and E. P. Rathgeb, "On the fairness of transport protocols in a multi-path environment," in *Proc. IEEE International Conference on Communications (ICC)*, OTTAWA, CANADA, June 2012.

[53] M. Zhang, J. Lai, A. Krishnamurthy, L. L. Peterson, and R. Y. Wang, "A transport layer approach for improving end-to-end performance and robustness using redundant paths." in *Proc. USENIX Annual Technical Conference*, Boston, MA, USA, June 2004.

[54] D. Rubenstein, J. Kurose, and D. Towsley, "Detecting shared congestion of flows via end-to-end measurement," *IEEE/ACM Transactions on Networking*, vol. 10, no. 3, 2002.

[55] S. Ferlin, Ö. Alay, T. Dreibholz, D. A. Hayes, and M. Welzl, "Revisiting congestion control for multipath TCP with shared bottleneck detection," in *Proc. IEEE INFOCOM*, San Francisco, CA, USA, April 2016.

[56] G. Carofiglio, M. Gallo, L. Muscariello, and M. Papali, "Multipath congestion control in content-centric networks," in *Proc. IEEE INFOCOM Workshop on Emerging Design Choices in Name-Oriented Networking (NOMEN)*, Torino, Italy, April 2013.

[57] S. Oueslati, J. Roberts, and N. Sbihi, "Flow-aware traffic control for a content-centric network," in *Proc. IEEE INFOCOM*, 2012.

[58] C. Tsilopoulos, G. Xylomenos, and Y. Thomas, "Reducing forwarding state in content-centric networks with semi-stateless forwarding," in *Proc. IEEE INFOCOM*, Toronto, ON, Canada, May 2014.

[59] D. Perino and M. Varvello, "A reality check for content centric networking," in *Proc. ACM SIGCOMM Workshop on Information-Centric Networking (ICN)*, Toronto, ON, Canada, August 2011.

[60] Y. Thomas, C. Tsilopoulos, G. Xylomenos, and G. C. Polyzos, "Multisource and multipath file transfers through Publish-Subscribe Internetworking," in *Proc. ACM SIGCOMM Workshop on Information-Centric Networking (ICN)*, Hong Kong, China, August 2013.

[61] ——, "Accelerating file downloads in publish-subscribe internetworking with multisource and multipath transfers," in *Proc. World Telecommunications Congress (WTC)*, Berlin, Germany, June 2014.

[62] Y. Thomas, G. Xylomenos, C. Tsilopoulos, and G. C. Polyzos, "Multi-flow congestion control with network assistance," in *Proc. IFIP Networking*, Vienna, Austria, May 2016.

[63] ——, "Object-oriented packet caching for ICN," in *Proc. ACM Information-Centric Networking (ICN)*, San Francisco, CA, USA, September 2015.

[64] C. Paasch, S. Ferlin, O. Alay, and O. Bonaventure, "Experimental evaluation of multipath TCP schedulers," in *Proc. ACM SIGCOMM Capacity Sharing Workshop (CSWS)*, Chicago, USA, August 2014.

[65] S. Ferlin, Ö. Alay, O. Mehani, and R. Boreli, "BLEST: Blocking estimation-based mptcp scheduler for heterogeneous networks," in *Proc. IFIP Networking*, Vienna, Austria, May 2016.

[66] V. Jacobson, "Congestion avoidance and control," in *Proc. ACM SIGCOMM*, Stanford, CA, USA, August 1988.

[67] A. Ghodsi, S. Shenker, T. Koponen, A. Singla, B. Raghavan, and J. Wilcox, "Information-centric networking: seeing the forest for the trees," in *Proc. ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets)*, Cambridge, MA, USA, November 2011.

[68] K. Wang *et al.*, "On the path management of multi-path TCP in internet scenarios based on the NorNet Testbed," in *Proc. IEEE International Conference on Advanced Information Networking and Applications (AINA)*, Taipei, Taiwan, March 2017.

[69] Y. Lu *et al.*, "Multi-path transport for {RDMA} in datacenters," in *Proc. USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, Renton, WA, USA, April 2018.

[70] G. Parisis, D. Trossen, and D. Syrivelis, "Implementation and evaluation of an information-centric network," in *Proc. IFIP Networking*, 2013.

[71] J. Yen, "Finding the k shortest loopless paths in a network," *Science Management*, vol. 17, no. 11, 1971.

[72] B. A. Alzahrani, M. J. Reed, J. Riihijärvi, and V. G. Vassilakis, "Scalability of information centric networking using mediated topology management," *Elsevier Journal of Network and Computer Applications*, 2014.

[73] D. Eppstein, "Finding the k shortest paths," *SIAM Journal on computing*, vol. 28, no. 2, 1998.

[74] T. Henderson, E. Sahouria, S. McCanne, and R. Katz, "On improving the fairness of TCP congestion avoidance," in *Proc. IEEE Global Telecommunications Conference (GLOBECOM)*, vol. 1, Sydney, Australia, November 1998.

[75] R. Barik, M. Welzl, S. Ferlin, and O. Alay, "LISA: A linked slow-start algorithm for MPTCP," in *Proc. IEEE International Conference on Communications (ICC)*, Kuala Lumpur, Malaysia, May 2016.

[76] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose, "Modeling TCP Reno performance: a simple model and its empirical validation," *IEEE/ACM Transactions on Networking*, vol. 8, no. 2, 2000.

[77] J. Heidemann, Y. Pradkin, R. Govindan, C. Papadopoulos, G. Bartlett, and J. Bannister, "Census and survey of the visible internet," in *Proc. ACM Internet Measurement Conference (IMC)*, Vouliagmeni, Greece, October 2008.

[78] R. Bhandari, "Optimal diverse routing in telecommunication fiber networks," in *Proc. IEEE INFOCOM*, Toronto, ON, Canada, June 1994.

[79] G. Xylomenos, G. C. Polyzos, P. Mahonen, and M. Saaranen, "TCP performance issues over wireless links," *IEEE communications*, vol. 39, no. 4, 2001.

[80] H. Kim and N. Feamster, "Improving network management with software defined networking," *IEEE Communications*, vol. 51, no. 2, 2013.

[81] S. Arianfar, "TCP's congestion control implementation in Linux kernel," in *Proc. Seminar on Network Protocols in Operating Systems*, Helsinki, Finland, September 2012.

[82] P. Sarolahti and A. Kuznetsov, "Congestion control in Linux TCP." in *Proc. USENIX Annual Technical Conference*, Monterey, CA, USA, June 2002.

[83] S. Ihm and V. S. Pai, "Towards understanding modern web traffic," in *Proc. ACM Internet Measurement Conference (IMC)*, Berlin, Germany, November 2011.

[84] G. Carofiglio, M. Gallo, and L. Muscariello, "ICP: Design and evaluation of an interest control protocol for content-centric networking," in *Proc. IEEE INFOCOM Workshop on Emerging Design Choices in Name-Oriented Networking (NOMEN)*, Orlando, FL, USA, January 2012.

[85] L. Saino, C. Cocora, and G. Pavlou, "Cctcp: A scalable receiver-driven congestion control protocol for content centric networking," in *Proc. IEEE International Conference on Communications (ICC)*, Orlando, FL, USA, March 2013.

[86] Y. Thomas and G. Xylomenos, "Towards improving the efficiency of ICN packet-caches," in *Proc. Heterogeneous Networking for Quality, Reliability, Security and Robustness (QShine)*, Rhodes, Greece, August 2014.