Multipath Congestion Control with Network Assistance

Yannis Thomas, George Xylomenos and George C. Polyzos
Mobile Multimedia Laboratory, Department of Informatics
School of Information Sciences and Technology
Athens University of Economics and Business
Patision 76, Athens 10434, Greece
E-mail: {thomasi, xgeorge, polyzos}@aueb.gr

***Abstract*—The use of multiple transport flows over distinct, if possible, paths, is a well-known technique for enhancing the performance and stability of data transfer. Multipath TCP (MPTCP), the most popular multipath transport protocol in-use, allows a single receiver to exploit multiple paths from a single sender. Nevertheless, MPTCP cannot fully exploit the potential gains of multipath connectivity, as it must fairly share resources with regular, single-path TCP, without knowing whether the available paths are distinct or share bottleneck links, due to IP's design choices. We introduce a hybrid congestion control algorithm for multipath transport that enables higher bandwidth utilization compared to MPTCP, while remaining friendly to TCP-like flows. Our solution employs (i) Normalized Multiflow Congestion Control (NMCC), a novel end-to-end congestion control algorithm and (ii) an in-network module that exposes topological information to the end-users in order to support the greedy friendliness technique. The end-to-end NMCC is architecture-independent and can be seamlessly integrated with MPTCP. The in-network module has been implemented for the PSI Information-Centric Networking architecture, but it can also be integrated with Multi-Protocol Label Switching (MPLS) and Software Defined Networking (SDN). Using an actual protocol implementation deployed on our testbed, as well as on a comprehensive packet-level simulator, we obtain experimental results which demonstrate clear gains for our design in terms of throughput and friendliness to other flows.**

I. Introduction

Experience with content distribution indicates that multisource and multipath [1], 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 higher throughput via bandwidth aggregation and resilience to link failures, while the use of multiple *sources* can further enhance throughput, while adding resiliency to source failures. By spreading flows across more links and sources, multisource and multipath provide load balancing, higher resource utilization and fault tolerance.[1]

Multipath transport is the focus of considerable research activity, due to the increasing numbers of multihomed devices, such as smartphones with Wi-Fi, Bluetooth and Cellular connectivity. This work has focused nearly exclusively on multipath connectivity between two endpoints, as most traffic is carried over TCP which does not support multiple endpoints at either end. A significant body of research has focused on the side-effects of multipath, such as lack of TCP friendliness [2], [3], [4], [5], [6], [7]. This issue arises from the *uncoupled* congestion control scheme originally proposed for *Multipath TCP* (MPTCP), where each sub-flow grasps bandwidth independently, similarly to a TCP connection. This causes the multipath transfer of N flows to grasp up to N times more bandwidth than a single-path flow over the same bottleneck, thus leading to unfair resource sharing. The current *coupled* MPTCP congestion control algorithms achieve friendliness by restricting the resource share of the individual sub-flows in order to jointly deliver equal throughput with a single-path flow.

Blindly restricting multipath flows can, however, lead to degraded resource utilization when friendliness concerns are not an issue, for example, when sub-flows exploit disjoint paths. Unfortunately, in the IP architecture the information about path disjointness is not available at the endpoints that perform congestion control. Each node is only aware of its own routing decisions, therefore even when an endpoint uses different physical interfaces to select distinct paths, it is unknown if the paths overlapp a few hops away. Nevertheless, IP networks operating over technologies that utilize centralized path computation components, including *Multi-Protocol Label Switching* (MPLS) and *Software Defined Networking* (SDN), are in principle capable of providing path-specific information to suitably modified endpoints.

Efficient utilization of network resources is also the driving force of the *Publish Subscribe Internet* (PSI) architecture, an instantiation of the *Information-Centric Networking* (ICN) paradigm [8]. Following ICN principles, PSI bases communication on self-identified information items, rather than endpoints, thus allowing information retrieval from multiple sources. PSI supports centralized path selection via a special network entity, the *Topology Manager*, and source routing via LIPSIN forwarding [9], thus allowing endpoints to reliably detect whether a set of paths to one or more sources do overlap. We have exploited these features in previous studies [10], [11], where we presented the *Multisource and Multipath Transfer Protocol* (mmTP), a multipath transport protocol for PSI, that was shown to deliver enhanced throughput and resilience over the, inherently unpredictable, PlanetLab testbed [11]. Moreover, in [12], we introduced the *greedy friendliness* technique, where the friendliness constraint is met opportunistically ac-

[1] In this paper, multisource is considered a special variant of multipath, hence we use the term only when the establishment of multiple paths takes advantage of multiple sources.

cording to the topological information offered by the Topology Manager.

In this paper, we delve deeper into the exploitation of any available topological knowledge of the network in order to enhance the resource utilization and friendliness of multipath congestion control. Specifically, we present and evaluate a novel congestion control scheme for multipath transport that consists of two independent modules: (i) *Normalized Multi-flow Congestion Control* (NMCC), an end-to-end multipath-aware algorithm that manages bandwidth aggregation under the friendliness constraint, and (ii) an in-network topological information mechanism that provides information about shared bottlenecks, thus allowing the application of greedy friendliness technique, i.e.,to neglect the friendliness constraint when friendliness concerns are not an issue. Our solution is practicable in network environments that support centralized path computation components, such as PSI, SDN, and MPLS. In this paper, we extend our previous work [12] in four directions:

- We introduce a more sophisticated in-network assistance algorithm to detect when friendliness concerns are not an issue.
- We present a more sophisticated version of NMCC that addresses friendliness more accurately and we model mathematically its performance.
- We asses experimentally the performance of NMCC with and without in-network assistance in domain scale scenarios.
- We provide details on the integration with IP networks operating over technologies like MPLS and SDN.

The remainder of this paper is organized as follows. In Section II we summarize existing work on multipath transport in IP and ICN networks. In Section III we briefly describe PSI and its features that allow us to realize selective friendliness and multisource connectivity. In Section IV we introduce our hybrid congestion control scheme, which consists of NMCC and the in-network assistance mechanism. In Section V we experimentally evaluate our design, using a prototype implementation in a real LAN testbed, while in Section VI we evaluate it in a WAN environment using packet-level simulations in NS-3. In Section VII we explain how the required in-network mechanism can be supported by MPLS and SDN. In Section VIII we model mathematically the throughput of NMCC showing that it achieves friendliness. Finally, we provide our conclusions in Section IX.

II. Background Work

Multipath congestion control is an active research topic for both traditional IP networks and ICN clean-slate architectures [13]. The common goal is maximizing resource utilization, in terms of exploiting the bandwidth available in multiple paths, while not harming competitive single-path transfers, a constraint also known as *friendliness* or *TCP-friendliness*.

A. Friendliness

When a multipath connection with N independently controlled sub-flows competes against a single-path connection for *the same* bottleneck link, the multipath connection can be up to N times as aggressive as the single-path one. In this work, we acknowledge the needs and opportunities presented by Future Internet technologies, where end-points can be aware of the in-network bottleneck links, hence we provide a refined definition of friendliness that examines the problem at link-level, instead of path-level:

A multipath connection should not acquire a larger share of resources in a shared bottleneck link than a single-path connection on the best of its paths.

The price of friendliness can be performance degradation: network resources are not be fully utilized by the multipath sub-flows in order to maintain friendliness to single-path connections. By exploiting our refined definition of friendliness, we can discover multiple paths that do not share the same bottleneck link, thus needlessly penalizing the performance in order to be friendly.[2]

Friendliness is different from fairness [14]. While the first defines the equal share of resource between multipath and single-path connections, the later describes the equal share of resources among all single-path flows, e.g., regardless of the delay of the flows. Consequently, TCP Reno, the TCP flavor that is the basis of friendliness research, is friendly by definition, but it is inherently unfair, as it favors shorter RTTs.

B. Multipath Congestion Control

The most prevalent multipath schemes reside in the application layer: Web browsers based on HTTP/1.1 [15] and BitTorrent clients [16]. According to HTTP/1.1, multiple requests to the same server can not be send in parallel, but must be pipelined through a single TCP connection, thus suffering from the head-of-line blocking issue. In order to achieve concurrency and thereby reduce latency, Web browsers typically establish multiple TCP connections to a single server.[3] BitTorrent clients exploit multisource by establishing a number of independent TCP connections to several content sources, achieving enhanced transmission rates and information availability, as the client simultaneously downloads different parts of a file from each source. Nonetheless, the uncoupled management of the established TCP flows often becomes unfriendly, since the transfer rate of the deployed flows can not be managed individually.

This issue is addressed at the transport layer by Multipath TCP (MPTCP) [17], an extension of TCP that allows the deployment and management of multiple TCP-like sub-flows among two endpoints. Being aware of the available set of sub-flows, MPTCP can control their transfer rate and jointly tackle performance and friendliness. Several congestion control algorithms have been proposed so far [5], [2], [18], [19], [20], with the Linked Increase Algorithm (LIA) algorith [2] being the default choice of the Linux MPTCP implementation.[4] According to the LIA algorithm, MPTCP manages its sub-flows

[2] In this paper we will use the term *friendly* to imply *single-path TCP-friendly*.

[3] This case is also considered multipath, since a single application establishes multiple TCP flows between the same communication end-points.

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

under two constraints: (i) a multipath flow should achieve at least as much throughput as it would get with single-path TCP on the best of its paths and (ii) a multipath flow should grasp no more capacity on any path or collection of paths than a single-path TCP flow that is using the best of those paths. The first constraint ensures that the use of MPTCP is beneficial. The second constraint assures MPTCP's friendliness towards unicast connections, but also compromises performance when friendliness in not an actual issue, for example, when the available paths do not share a bottleneck link.

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 detect whether the paths used are overlapping, 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 [21], [22], but their efficiency is debatable. In [21] the authors detect shared bottlenecks based on the temporal correlation of Fast Retransmit packets, while in [22] 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 too high for a general purpose multipath protocol.

A recent proposal [23] exploits the propagation delay of sub-flows 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, thus gaining up to 40% higher throughput at the user-level when friendliness is not required. Our work further investigates the gains of greedy friendliness, by conducting domain scale experiments and using network provided information, rather than inference, to detect the existence of shared bottlenecks.

Finally, a different approach is followed by studies that exploit SDN's centralized path selection module in order to avoid the shared bottlenecks of MPTCP paths. In [24] the authors present an MPTCP-*aware* SDN control plane module that detects MPTCP sub-flows 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 [3]. Similarly, in [25] the authors present a mechanism that enhances path diversity of MPTCP sub-flows in order to avoid performance bottlenecks, also supporting the creation of multiple paths for single-homed MPTCP users. 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 sub-flows based on port numbers. Although these studies have not yet delivered solid and practical results, they give insight into the transport layer of Future Internet where end-to-end congestion control can be enhanced by novel in-network functions.

III. Multipath transport in the PSI architecture

A. The PSI architecture

In the PSI architecture, content objects are treated as publications, content sources as publishers and content consumers as subscribers. User programs exploit a publish/subscribe API for advertising and requesting information. A fundamental design tenet in PSI is the clear separation of its core functions [26]: (i) the Rendezvous function tracks available publications and resolves subscriptions to publishers, (ii) the Topology Management and Path Formation function monitors the network topology and forms forwarding paths and (iii) the Forwarding function handles packet forwarding.

Accordingly, network nodes in a PSI network are classified into *Rendezvous Nodes* (RNs), *Topology Managers* (TMs) and *Forwarding Nodes* (FNs) [27]. The RNs receive and store the pub/sub requests and match publications with subscriptions of the same content. When matching takes place, the RN asks a TM to find one or more appropriate dissemination routes. The TM, which is aware of topology, network conditions and content characteristics, discovers the "best" path(s) and encodes them into LIPSIN identifiers, a representation of the set of links comprising each path [9]. Finally, the LIPSIN identifiers are delivered to the endpoint applications that exploit them for direct communication. LIPSIN forwarding, which is realized by the FNs, offers stateless source routing, since all path information is encoded in the LIPSIN identifier. Note also that paths are pinned as long as their LIPSIN identifiers are fixed: any change to a path, will require encoding a different set of links into the LIPSIN identifier.

The centralized nature of the TMs raises concerns about PSI's feasibility, since they must compute paths for all network connections. However, research shows that a centralized intra-domain TM service is feasible: for a typical national-scale network provider in the UK, it was demonstrated that a reasonable number of TM instances with precomputed paths can efficiently cope with the resulting network load [28]. Centralized path computation is not that uncommon in IP networks either, as it is prevalent in network domains using MPLS below IP and is the most common approach in SDN domains; the processing overhead of the SDN control plane, which includes detecting and encoding paths, is found acceptable when multiple co-existing SDN controllers are deployed in large scale networks [25].

B. Multipath in PSI

We have presented a multipath transport protocol for PSI in previous publications [10], [11], the *Multisource and Multipath Transfer Protocol* (mmTP). mmTP is a reliable protocol that supports multisource and multipath data transfers by exploiting PSI's source routing and centralized path selection. mmTP relies on a TM function that can discover multiple paths between a receiver and multiple senders. These paths are encoded in LIPSIN identifiers that are later sent to the endpoints. Given that LIPSIN identifiers encode dissemination routes without unveiling the actual dissemination paths, or even the destination nodes, the endpoints acquire a set of distinct "options" for pulling data, which may involve different

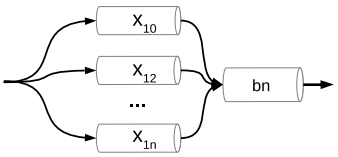


Fig. 1. Example of FB detection in case of a single shared link.

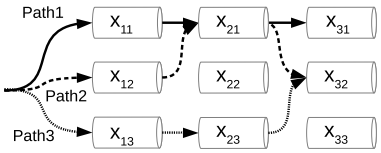


Fig. 2. Example of FB detection in case of multiple shared links.

publishers and/or different paths. Hence, mmTP provides a generic interface, transparently supporting any combination of multisource and/or multipath services.

The design of mmTP allows congestion control in two levels: (i) path selection by the TMs and (ii) path utilization by the endpoints. Specifically, the TMs, which are aware of network conditions, select appropriate routes for load balancing and bandwidth aggregation. We have previously shown the gains of centralized path formation in [29], where we used QoS routing schemes to satisfy certain throughput and error rate constraints in PSI. Based on these routes, the endpoints evaluate in real-time the performance of each path and adjust the amount of data to be delivered through it. The congestion control mechanism used at the endpoints, which is derived from TCP, pushes complexity at the network edges, thus enhancing network stability and keeping forwarding stateless. A detailed description of mmTP design and operation is provided in [11], where we also explore the performance enhancement and increased resilience offered by multipath and multisource in a PlanetLab overlay topology.

IV. Hybrid multipath congestion control

In this section we present a hybrid multipath congestion control algorithm that enhances resource utilization without violating the friendliness requirement. Our novel congestion control scheme consists of two independent modules: (i) *Normalized Multiflow Congestion Control* (NMCC), an end-to-end multipath-aware algorithm, and (ii) an in-network topological information mechanism to assist congestion control. NMCC is designed to offer high resource utilization (even in heterogeneous paths), responsiveness and friendliness since the beginning of the connection. The in-network mechanism exploits knowledge of shared bottlenecks in order to improve the friendliness control through the *greedy friendliness* technique.

A. In-network Assistance Mechanism

The best case scenario for multipath communication arises when all communication paths are physically disjoint, that is, they do not share any performance bottlenecks. In this case, each sub-flow can use the same congestion control algorithm as single-path connections without any constraints; this approach is called *uncoupled* congestion control. In contrast, when some sub-flows share links that constrain their transfer rate, their aggressiveness needs to be limited in order for them to remain friendly to single-path flows. Our congestion control scheme practices *greedy friendliness* by limiting the aggressiveness of the sub-flows only when needed, namely, in the second case.

1) Detection of friendliness bottlenecks: In single-path transfers, the *bottleneck* is the link that constrains the resource share that a connection gets in the other links of a path. When the resource share is equivalent to the transfer rate, then the bottleneck is the path's "narrowest" link that limits the (maximum) transfer rate of a connection.[5] Given a unicast path p that consists of n links, where b_{l_i} is the available transfer rate in link l_i, the transfer rate of a connection in p, b_p, is limited by the path's bottleneck, hence:

$$b_p = min\{b_{l_1}, b_{l_2}, .., b_{l_n}\} \quad (1)$$

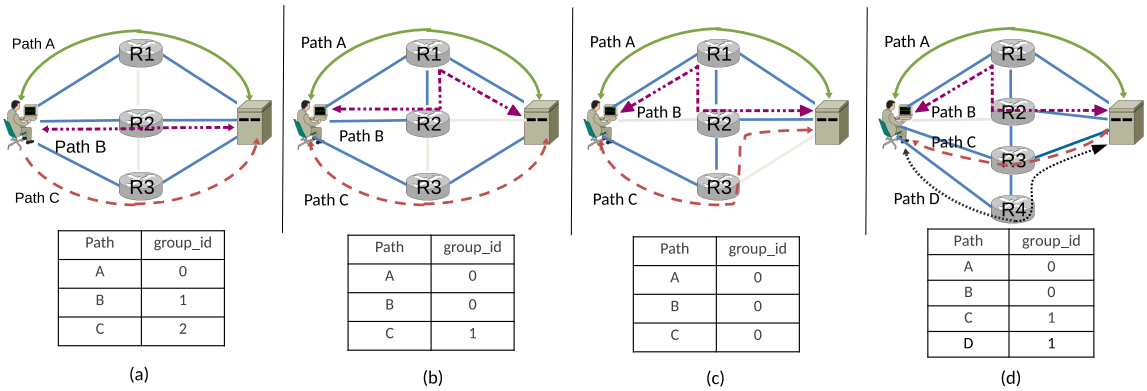
In multipath transfers, the detection of bottlenecks that penalize friendliness, namely *friendliness bottlenecks* (FBs), is more complex. A FB must satisfy two conditions: first, it must be shared by at least two sub-flows and, second, it must limit the transfer rate of these sub-flows. In this case, the link can be unevenly shared by a multipath connection and a single-path, thus requiring a friendly multipath congestion control algorithm.

The definition clarifies that not all shared links are performance bottlenecks, e.g., the shared links that are wider than the individual bottlenecks of the available paths may not FBs. Consider the simple example of Fig. 1, where n sub-flows go through n different first links but share the second link bn. Depending on the transfer rate of the links, the bottleneck may be detected at the first link, thus being a non-shared bottleneck, or at link bn, thus being a shared bottleneck. We only examine the shared links, hence the second case qualifies for a FB. We observe that the shared links are not FB as far as they do not constitute performance bottlenecks for the individual paths, namely when the best single-path connection can get more resources in the shared link than in the individual bottleneck. Thereupon, we conclude that link bn can not be a FB iff

$$b_{bn} > b_{x_{max}}(Z_{bn} + 1)/2 \quad (2)$$

where $b_{x_{max}}$ is the transfer rate of the widest first link and Z_{bn} is the number of sub-flows running in bn. In the example of Fig. 1, the best single-path connection competes with 1 and 3 sub-flows in x_{max} and bn, respectively, hence the path's performance according to (1) is $min\{b_{x_{max}}/2, b_{bn}/4\}$ and $min\{b_{x_{max}}/2, b_{bn}/2\}$ with unfriendly and friendly congestion control, respectively. If $b_{bn} > b_{x_{max}}4/2$, then the best single-path connection will be constrained in link x_{max}, thus getting $b_{x_{max}}/2$ regardless of the friendliness of multipath congestion control.

[5]We assume a link monitoring protocol that measures the available transfer rate. The details of the protocol are out of scope for this work.



Path	group_id
A	0
B	1
C	2

(a)

Path	group_id
A	0
B	0
C	1

(b)

Path	group_id
A	0
B	0
C	0

(c)

Path	group_id
A	0
B	0
C	1
D	1

(d)

Fig. 3. Three different cases of path composition and their corresponding $group\_id$ codes that mark sub-flows sharing a link: (a) Disjoint paths, (b) Paths A and B share one link, (c) Paths A and B share one link, paths B and C share another link.

In case of multiple shared links, the same approach is still valid. Consider the example of Fig. 2, where a multipath connection of three sub-flows is deployed, creating two likely FBs (shared links x_{21} and x_{32}). According to (2), there is not FB iff the share of a single-path connection in the shared link is larger than its share in the largest individual bottleneck. We have two shared links, therefore:

$$b_{x_{21}} > max\{min\{b_{x_{11}}, b_{x_{31}}\}, min\{b_{x_{12}}, b_{x_{32}}\}\}4/2$$

$$b_{x_{32}} > max\{min\{b_{x_{13}}, b_{x_{23}}\}, min\{b_{x_{12}}, b_{x_{21}}\}\}4/2$$

If any of the previous is false, then we conclude the existence of a FB and friendly multipath congestion control must be enabled.

2) Encoding of friendliness bottlenecks: Path selection in PSI is performed by the TMs, whose operation details extend beyond the scope of this paper. Our only requirement is that when the TMs return a set of paths encoded as LIPSIN identifiers, a $group\_id$ code should be added to each identifier so as to indicate paths that share a FB. Specifically, all paths that share a FB with each other (not necessarily *the same* bottleneck) 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 to NMCC how the available paths are grouped by $group\_id$.

For instance, Fig. 3 shows four examples of path composition along with the corresponding $group\_id$ codes, when all links present the same transfer rate. In Fig. 3(a) the three paths are disjoint, thus each path is marked with a distinct $group\_id$, whereas in Fig. 3(b) paths A and B share a FB according to (2), thus they have the same $group\_id$. In Fig.3(c) Paths A and B share a FB and paths B and C share a different FB; 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. Finally, in Fig. 3(d), two FBs are shared by two sets of sub-flows, each of which passes through one FB, thus leading to two different groups.

We have also considered grouping the paths in Fig.3(c) 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 FB, hence a path could belong to several groups. Consequently, 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.

3) Operational overhead: Finally, we discuss the operational overhead that our notification service induces to the standard TM operation [28], as it requires finding the best k-paths, detecting shared links and examining the existence of FBs. Finding the k-shortest paths in a topology of n nodes and e edges has $O(e+n\log n+k)$ complexity [30], 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, $O(h^2k(k-1)/2)$ comparisons are required to compare them, 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 ($k \leq 6$) [31] 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 states but reduces computational complexity to $O(2kh)$. 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. 3(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.

The examination of shared links is the last step. The calculation takes place for each group path independently,

checking the shared bottlenecks of each group until (2) is false and a FB is concluded. The best case arises when no shared links are presented, thus presenting no operation overhead. The worst case is when all groups have (multiple) shared links that are not FBs, hence all shared links must be examined. However, the examination of (2) is light-weight as the required information, such as path bottlenecks and the number of sub-flows in the shared links, can be gathered in the previous step, when parsing the paths to detect shared links.

B. End-to-end Multipath Congestion Control

When the available paths have different $group\_ids$ (i.e., they do not share any links), window management does not consider friendliness: our algorithm creates a distinct sub-flow for each path with an individual congestion window variable ($cwnd$), RTT-based loss detection timer and retransmission mechanism. In addition, each sub-flow operates independently the *Slow Start* and *Congestion Avoidance* algorithms. Therefore, window management with disjoint paths uses the TCP (uncoupled) congestion control scheme.

In contrast, when some paths have the same $group\_id$ (i.e., they share a bottleneck link) the NMCC algorithm is used to manage them as a group. NMCC is quite different from the existing congestion control algorithms of MPTCP. First, NMCC is friendly during the Slow Start phase, thus offering friendliness to unicast connections instantly. Second, NMCC addresses the RTT-mismatch issue [2] inherently, thus performing efficiently in heterogeneous paths. Third, NMCC exploits all paths equally (according to their characteristics), thus offering responsiveness.

The fundamental novelty of NMCC is detected the way it approaches friendliness. NMCC pursues friendliness by normalizing the *growth* of the transfer rate of each flow, rather than the transfer rate itself, as in other approaches. NMCC assumes 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 a connection among its pool of available sub-flows. The friendliness constraint is deterministically met at each window increase, hence NMCC remains friendly throughout the connection's lifespan.

In addition, NMCC does not focus on the best path, unlike the majority of proposed congestion control algorithms. NMCC restrains the aggressiveness of sub-flows *proportionally* thus leaving the same percentage of traffic in each path. The focus on the best path is known to deliver high throughput and friendliness easier under certain conditions [18], but it can lead to performance entrapment when the best path changes during the transfer. Thereafter, the protocols that select the best path, such as LIA, introduce probing or special mechanisms to maintain some traffic on the other paths. OLIA [18], which focuses exclusively on the best path, is found to be unresponsive in the evaluation of Balia [19], which constitutes a solution with balanced responsiveness and friendliness. NMCC natively pushes more traffic on the best path but does not neglect the other paths, thus avoiding the need for probing or additional mechanisms.

Finally, in order to achieve friendliness while maintaining a simple design, NMCC exploits inherent properties of TCP. Specifically, NMCC is based on a well-known TCP-fairness characteristic, the fact that connections with higher RTTs are less aggressive [14], in order to normalize the growth of the transfer rate of sub-flows. 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 Slow Start phase and avoids multipath-related issues due to RTT-mismatch, sudden load and congestion shifts.

1) Congestion Avoidance: NMCC uses an inflated $RTT'_i \geq RTT_i$ for each sub-flow i to control window growth; the inflated RTT'_i slows down the rate of increasing the congestion window. In order to estimate the amount of inflation, we introduce a *friendliness factor in Congestion Avoidance* $m_{ca} \geq 1$ where $RTT'_i = m_{ca} RTT_i$. The calculation of m_{ca} is derived based on two fairness goals: (i) the growth rate of all sub-flows sharing a link should be no more than that of a single-path connection and (ii) the overall growth rate should not be less than that of the "best" single-path connection.

The best path is the one with the highest throughput increase rate, since friendliness is based on equalizing the cumulative throughput increase rate of all sub-flows with the throughput increase rate of the most aggressive available single-path.

Since we do not know the RTT of the best single-path connection, we approximate it by the minimum RTT_i, RTT_{min}, among our pool of sub-flows. During Congestion Avoidance, a sub-flow i increases its congestion window by one *maximum segment size (MSS)* per RTT, so its window growth rate is MSS/RTT_i and its throughput growth rate is MSS/RTT_i^2. Therefore the throughput growth rates in this phase must satisfy the following equation:

$$\frac{MSS}{RTT_{min}^2} = \sum_{i=1}^{N} \frac{MSS}{RTT'^2_i} = \sum_{i=1}^{N} \frac{MSS}{m_{ca}^2 RTT_i^2} \tag{3}$$

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

$$m_{ca}^2 = \frac{RTT_{min}^2}{MSS} \sum_{i=1}^{N} \frac{MSS}{RTT_i^2} = RTT_{min}^2 \sum_{i=1}^{N} \frac{1}{RTT_i^2} \tag{4}$$

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 $RTT_A = 5$ ms and $RTT_B = 10$ ms. We initially set $m_{ca}^2 = 2$, the number of jointly controlled paths.[6] Upon receipt of the first packet over each path, we can calculate $m_{ca} = 5\sqrt{(1/5^2 + 1/10^2)} \simeq 1.118$. Therefore, $RTT'_A = 5.59$ ms and $RTT'_B = 11.18$ ms. The inflated RTTs allow NMCC to increase its overall congestion window by roughly $1/5.59 + 1/11.18 = 0.268$ MSS/ms, while the best single-path 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, and the best single-path connection by $1/5^2 = 0.04$ MSS/ms^2. Consequently,

[6] This is equivalent to assuming that all RTTs are equal in (4).

both connections will extend their share of network resources evenly, thus fairly sharing the available bandwidth.

Proportional path exploitation: By applying m_{ca} to the RTTs of all sub-flows, we adapt the growth rate of all paths, which means that, although we favor the sub-flow which operates over the fastest path, we do not neglect the other paths. Therefore, NMCC does not require probing to detect load changes on unused paths, performing 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 sub-flow growth, allowing NMCC to effectively grasp the available resources.

Friendliness vs. Responsiveness: NMCC also presents an interesting ability to adapt its responsiveness under different circumstances. 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. In [19] the authors analyze the trade-off between friendliness and responsiveness in multipath congestion control. They mathematically prove that the friendliness penalizes responsiveness and vice verse, since the friendly restrained throughput increase rate does not allow to aggressively grasp the resources that become free. Along these lines, NMCC is the least responsive when all sub-flows present equal throughput increase rate and the friendliness factor takes its maximum value, that is the number of sub-flows. On the contrary, NMCC is the most responsive when the sub-flows present very different throughput increase rates, hence the friendliness factor is close to 1. The poor responsiveness can penalize resource utilization in case of disjoint paths, where the subset of sub-flows performing on a path fails to grasp the available resources as quickly as a single-path connection since their cumulative aggressiveness is less. However, in our design the in-network assistance module instructs uncoupled congestion control in case of disjoint paths, thus avoiding the problem.

Compliance with (MP)TCP design: Finally, 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 sub-flow with NMCC is:

$$\frac{MSS}{RTT'^2} = \frac{MSS}{(m_{ca}RTT)^2} = \frac{MSS/m_{ca}^2}{RTT^2} \tag{5}$$

hence the increase of a friendly congestion window is MSS/m_{ca}^2 over the unmodified RTT. Measuring $cwnd$ in bytes, in Congestion Avoidance TCP increases its window by $MSS^2/cwnd$ bytes, $cwnd/MSS$ times within an RTT, for an overall growth of 1 MSS. By reducing the amount of per-ACK increase of a sub-flow to $MSS^2/(m_{ca}^2 cwnd)$ bytes, the cumulative increase of NMCC within an RTT is MSS/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.

2) Slow Start: Most work on multipath transport only deals with Congestion Avoidance, since Slow Start is considered a transient state with no measurable impact on the steady state 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 sub-flows gained bandwidth almost N-times faster than a single-path connection during Slow Start. Since short and very congested connections spend a measurable fraction of their lifetimes in Slow Start, meeting the friendliness goals in Congestion Avoidance was not enough to amortize NMCC's aggressive behavior during Slow Start.

One way to reduce aggressiveness during Slow Start is to reduce $ssthresh$, the window size which makes the algorithm switch from Slow Start to Congestion Avoidance. Unfortunately, this has two disadvantages. First, when a connection starts, the available bandwidth of the communication path is unknown, thus $ssthresh$ 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 its rate of acquisition until that happens. For this reason, NMCC must control the amount of bandwidth gained during Slow Start, as well as its rate of growth.

The NMCC friendliness approach for Congestion Avoidance can be seamlessly adapted to the Slow Start phase for controlling aggressiveness. Specifically, in Slow Start, when a sub-flow i doubles its congestion window every RTT_i, its instant throughput growth rate is $cwnd_i/RTT_i^2$. We introduce m_{ss} *the friendliness factor for Slow Start* where $RTT_i' = m_{ss}RTT_i$. Similarly to the Congestion Avoidance phase, m_{ss} must equalize the throughput growth rate of all multipath flows with the fastest increasing single-path, hence:

$$\frac{cwnd_k}{RTT_k^2} = \sum_{i=1}^{N} \frac{cwnd_i}{RTT'^2_i} = \sum_{i=1}^{N} \frac{cwnd_i}{m_{ss}^2 RTT_i^2}$$

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

$$m_{ss}^2 = \frac{RTT_k^2}{cwnd_k} \sum_{i=1}^{N} \frac{cwnd_i}{RTT_i^2} \tag{6}$$

The similarity of (4) and (6) allows the creation of a unified method for estimating the friendliness factor when sub-flows are in different states. We introduce Ω_i and Ω_i', the regular and the friendly throughput growth rate of sub-flow i, respectively. The estimation of Ω_i and Ω_i' depends on the apparent congestion phase of the sub-flow, therefore:

$$\Omega_i = \left\{ \begin{array}{ll} MSS/RTT_i^2, & \text{in Cong. Avoidance} \\ cwnd_i/RTT_i^2, & \text{in Slow Start} \end{array} \right\} \tag{7}$$

$$\Omega'_i = \left\{ \begin{array}{ll} \frac{MSS}{RTT_i'^2} = \frac{MSS}{m_{ca}^2 RTT_i^2}, & \text{in cong. avoidance} \\ \\ \frac{cwnd_i}{RTT_i'^2} = \frac{cwnd_i}{m_{ss}^2 RTT_i^2}, & \text{in Slow Start} \end{array} \right\} \quad (8)$$

The combination of (4), (6) and (8) provides a unified formula for estimating m, the *friendliness factor* of NMCC, taking into account sub-flows in both the Congestion Avoidance and Slow Start phases:

$$m^2 = m_{ca}^2 = m_{ss}^2 = \frac{\sum_{i=1}^{N} \Omega_i}{\Omega_{max}} \quad (9)$$

To better understand the operation of NMCC with sub-flows in different congestion phases, assume that NMCC exploits three sub-flows, A, B and C, but only sub-flow C is in Congestion Avoidance. The algorithm only requires estimating m based on (8) and, then, calculating the RTT's. Assume that initially the RTTs are 10, 5 and 5 ms and the windows are 100, 20 and 10 MSS for sub-flows A, B and C, respectively. From (7), 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 (9) we can calculate that $m = 1.356$. The inflated RTT's are 13.56, 6.78 and 6.78 ms, resulting in throughput increase rates, Ω'_i, of 0.543, 0.435 and 0.022 MSS/ms^2 for paths A, B and C, respectively. As expected, the aggregate throughput growth rate is equal to that of the best single-path, or 1 MSS/ms^2.

A final issue is that the window increase algorithm of NMCC can penalize its throughput during Slow Start. Specifically, during Slow Start the congestion window of a NMCC sub-flow on the jth RTT is set to $cwnd_j = cwnd_{j-1} + cwnd_{j-1}/m^2 = ((m^2+1)/m^2)cwnd_{j-1}$, while a single-path flow would set it to $cwnd_j = 2cwnd_{j-1}$. 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" in the growth of the congestion window, which is $r_j = ((m^2-1)/m^2)cwnd_{j-1}$. 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 $cwnd_{j-1}$. 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 $\sum_{i=1}^{j-1} r_i m^{2(j-1-i)}$ 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 $cwnd_j = cwnd_{j-1} + cwnd_j^{sp}/m^2$, where $cwnd_j^{sp}$ is the window size of a single-path flow running in the same path on the j^{th} RTT. The combined Slow Start and Congestion Avoidance algorithm is presented in Algorithm 1.

3) Window reduction: NMCC does not introduce a particular mechanism to maintain friendliness during window reduction, thus adopting the single-path operation. The fact that NMCC is enabled only when a FB is detected, eliminates the need of throughput regulation in case of loss, since the cumulative throughput reduction rate of multipath and single-path in the FB is equal. In particular, we estimate the throughput decrease rate, D, as the product of throughput reduction per loss and the frequency of loss, hence:

$$D = \frac{\frac{cwnd}{2}}{RTT} \frac{1}{(tRTT)}$$

where $tRTT$ is the frequency of loss with $t > 0$. The number t essentially reflects the congestion increase rounds since the last loss, thus being proportional to window size $cwnd$, which is steadily increased every RTT in Congestion Avoidance. Thereafter, we can approximate t with $cwnd/2$, which is equal to number of window increases between two successive losses, and rewrite the throughput decrease rate as the following

$$D = \frac{\frac{cwnd}{2}}{RTT} \frac{1}{\frac{cwnd}{2} RTT} = \frac{1}{RTT^2} \quad (10)$$

The equation shows that paths with larger RTTs exhibit lower throughput reduction rate, since they increase their windows slower thus achieving lower transfer rates in general. Therefore, in order to equalize the throughput decrease rate of a multipath connection and a single-path connection on the best path, the following must be true:

$$D_k = \sum_{i=1}^{N} D_i \Leftrightarrow \frac{1}{RTT_k^2} = \sum_{i=1}^{N} \frac{1}{RTT_i^2} \quad (11)$$

where k is the single-path with the fastest growth rate and, therefore, our performance point of reference. The equation is identical to the initial problem statement (Eq. (3)), which is already solved through the friendliness factor m and the inflated RTT's, thus arriving at a valid equation.

Our method relies on the sub-flows being constrained by the same FB and can not be applied otherwise, hence NMCC

Algorithm 1 Window adjustment and estimation of m.

```
procedure INCREASE_WINDOW
    if (cwnd < ssthresh/m^2) then
        cwnd ← cwnd + cwnd^sp * MSS/(cwnd * m^2)
    else
        cwnd ← cwnd + MSS * MSS/(cwnd * m^2)
    end if
end procedure
procedure ESTIMATE_M
    max_rate ← 0
    total_rate ← 0
    for (i ∈ sub − flows) do
        if (cwnd_i < ssthresh_i/m^2) then
            rate ← cwnd_i/RTT_i^2
        else
            rate ← MSS/RTT_i^2
        end if
        total_rate ← total_rate + rate
        if (rate > max_rate) then
            max_rate ← rate
        end if
    end for
    m ← sqrt(total_rate/max_rate)
end procedure
```

is unfriendly during window reduction in case of individual FBs with different error rates or delays.

RTT estimation: The computation of the friendliness factor greatly relies on the accurate and timely estimation of path's RTT. Being the cornerstone of TCP loss detection mechanism, this procedure is an established solution with well-explored gains and weaknesses. In mmTP, we avoid "poisoning" the RTT measurements by ignoring the RTT sample of expired and retransmitted packets when estimating a path's RTT [32] (unless timestamps are employed). When the RTT exhibits higher variance, usually in WANs, the *smoothed RTT* can be considered instead of the RTT [32]. In case of sender or path failure, when packets stop being delivered, thus rendering the RTT outdated, we propose the exploitation of a special flag, called *is-active*, that describes if a sub-flow is able to deliver packets. The receiver, which manages congestion control, sets this flag to false when the retransmission timer expires and changes it to true only when a packet is received. While the flag is false the RTT of the sub-flow is excluded from the friendliness factor estimation. After connectivity is restored, the RTT of the sub-flow is estimated based on the RTT sample of the first received packet, which can be a retransmission if timestamps are employed. Finally, in case the medium presents losses that are not related to the congestion level of the path, f.i., in satellite networks, our algorithm suffers from TCP's inefficiency to estimate network conditions accurately, but we argue that friendliness will not be significantly affected. Specifically, if the lossy link is not a friendliness bottleneck, then sub-flows will perform similarly to single-paths, thus not requiring the estimation of the friendliness factor. In case the lossy link is a friendliness bottleneck, then the RTT of the sub-flows will fluctuate similarly, hence the friendliness factor estimation will not be greatly affected. Although, this issue is realistically addressed through *Performance Enhancement Proxies* (PEPs) [33], which is an orthogonal issue to congestion control algorithms, we consider this an interesting topic for future work.

V. Performance Evaluation: LAN

In this section, we evaluate the performance of our hybrid congestion control algorithm, as well as its friendliness to single-path TCP Reno, using an implementation of our algorithm in a LAN environment. In the next section, we examine WAN scenarios using packet-level simulations in NS-3.

We have implemented our hybrid congestion control algorithm as part of the mmTP protocol that runs over Blackadder, the PSI prototype implementation [34]. 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 [35] with hop count as the metric. Unlike MPTCP, mmTP is a receiver-driven protocol that is based on the transfer of autonomous, self-verified chunks (or packets). Each mmTP sub-flow autonomously transfers the next data chunk that needs to be delivered, regardless if its a retransmission or not, therefore the scheduling of packets does not affect performance.[7]

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

TABLE I
AVERAGE TRANSFER RATES WITH DISJOINT PATHS (WITHOUT CONTENDING TRAFFIC).

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) the effect of TM assistance when paths are disjoint, (ii) the effectiveness of NMCC with overlapping paths and (iii) NMCC's behavior in short transfers.

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`.[8] The duration of transfers during all experiments is 300 s, but we consider only the final 60 s where the system has reached its steady state, 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%.

A. Disjoint paths

We first deployed mmTP in the topology of Fig. 4(a), in order to investigate the performance gains of our approach when paths are known to be disjoint. Figure 4(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 I; 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-path mmTP connections to each publisher, first independently and then together. We notice that each path offers roughly 10.6 MBps throughput and collectively mmTP achieves 21.3 and 20.7 MBps with and without TM assistance, respectively. These preliminary results validate that mmTP fully exploits available capacity and imply that TM assistance slightly enhances performance, even in the absence of competitive 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

[7]Our implementation will be available upon paper publication.
[8]Available at `http://iperf.sourceforge.net/`.

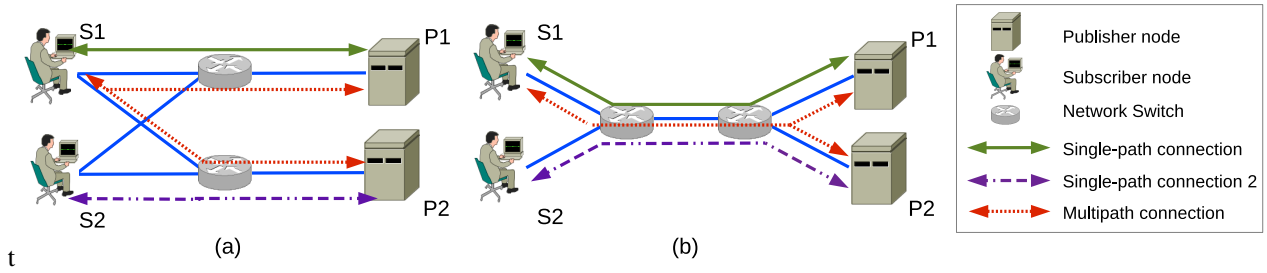


Fig. 4. Topology for performance evaluation with (a) disjoint paths and (b) shared paths.

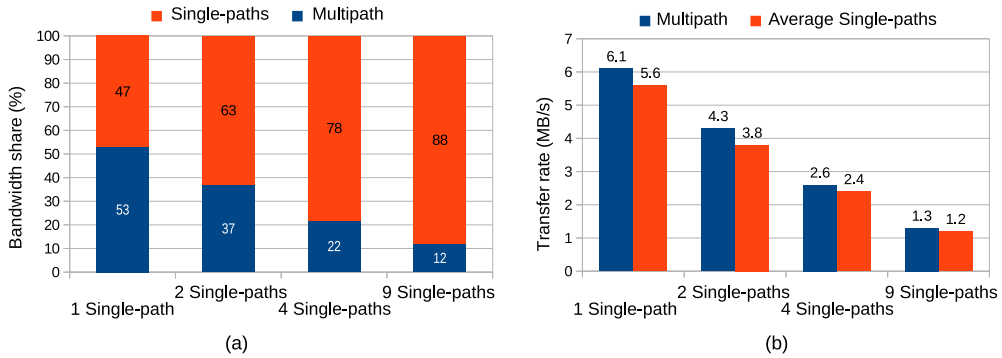


Fig. 5. (a) Bandwidth shares of NMCC and all single-path connections. (b) Transfer rate of NMCC and the average single-path connection.

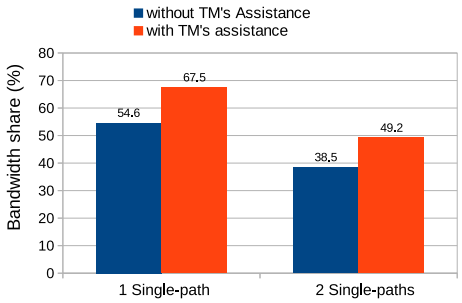


Fig. 6. Bandwidth share of mmTP with and without TM assistance.

one or two single-path connections competing over one or both disjoint paths (S1 to P1 and S2 to P2). In Fig. 6 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-path 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-path 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.

B. 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(b), where Publishers and Subscribers are connected by paths sharing a link. We deployed a multisource connection from subscriber S1 to publishers P1 and P2, in parallel with 1, 2, 4 and 9 single-path connections from subscriber S1 to publisher P1 and from subscriber S2 to publisher P2; these connections are distributed uniformly between the two paths.

Figure 5(a) demonstrates the average bandwidth percentage acquired by NMCC and *all* single-path connections, while Fig. 5(b) displays the average transfer rate achieved by NMCC and the *average* unicast connection. NMCC acquires 53%, 37%, 22% and 12% of the bottleneck link's bandwidth when competing with 1, 2, 4 and 9 single-path connections, respectively, marginally over the fair 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 best single-path available. With multiple similar paths, the best 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,

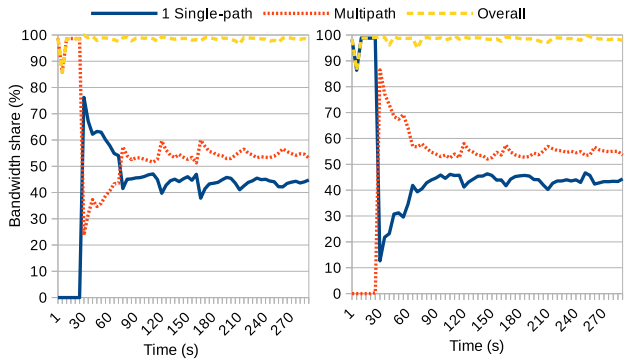


Fig. 7. Instant bandwidth shares of NMCC and 1 single-path connection when NMCC is deployed 30 s before (left) and after (right) the single-path connection. We also present the percentage of the overall instant resource utilization.

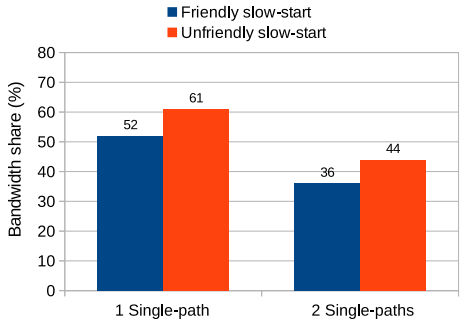


Fig. 8. Bandwidth share of mmTP with and without friendly Slow Start in short transfers.

but this time starting the multipath connection either 30 s after or 30 s before the start of the single-path connections. In our setup, with no competitive traffic, the connections converge to the steady state transfer rate within a couple of seconds, hence the initial connections are stabilized by the time the late connections are deployed. The results show that friendliness is not substantially affected by congestion bursts, as NMCC acquires 55%, 36%, 23% and 12% of the bandwidth when competing with 1, 2, 4 and 9 single-path 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. In Fig. 7 we plot the instant throughput rate of the multipath and the single-path connection (when only 1 single-path is deployed), showing that the response of the multipath connection is instant regardless of the connection deployment sequence. The convergence to friendliness presents some further delay (roughly 60 s), but the overall resource utilization remains close to 100% at all times.

C. Short Transfers

NMCC is friendly during Slow Start, unlike MPTCP's LIA, OLIA and Balia, which are only concerned with Congestion Avoidance. This is particularly important for short transfers, where friendly Congestion Avoidance cannot compensate for an unfriendly Slow Start. To evaluate this aspect of NMCC, we reused the shared link topology of Fig. 4(b), deploying one multisource NMCC connection and either 1 or 2 contending single-path connections. Each connection transfers a 10 MB object, which would require less than 1.1 s to complete in the absence of contention. Figure 8 presents the percentage of overall bandwidth acquired by NMCC when friendly Slow Start is turned on or off.

With unfriendly Slow Start, 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 Slow Start gains 52% and 36% of the total bandwidth. Consequently, NMCC is friendly even with short transfers.

For even shorter transfers, for example Web objects a few kilobytes long, the unfairness is even more pronounced, as such connections can easily complete during Slow Start. The reason for presenting results from a 10 MB transfer is to show that the initial over-aggressiveness during Slow Start cannot be compensated during Congestion Avoidance, even with longer transfers. We also note that unfriendly Slow Start would be a problem in any scenario where timeouts occur often, from lossy links to very congested paths, as they would lead to frequent invocations of the Slow Start phase.

D. Dynamic cross traffic

The estimation of the friendliness factor m is RTT-based; here we investigate the impact of RTT fluctuations, caused by dynamic cross traffic, on the computation of m and, in turn, the friendliness of NMCC. In order to simulate dynamic background traffic, we deploy Cisco's TREX[9] traffic generator in the topology of Fig. 4.(b). TREX generates Layer 4-7 traffic based on pre-processing and smart replay of real traffic templates. More specifically, TREX replays numerous connections of different types, such as HTTPS, DNS and RTP, each type exhibiting distinct frequency, size and RTT properties. By proportionally adjusting the launch frequency of the different connection types, we can vary the volume of generated traffic, while retaining the characteristics of real traffic templates. We test the performance of NMCC under four different levels of congestion, namely, without any TREX traffic and when it constitutes approximately 100%, 50% and 10% of the available bandwidth.[10] In all setups, we deploy one multisource NMCC connection and 2 contending single-path connections.

Figure 9 depicts the friendliness factor m throughout the transfer (with 5000 packets sampling frequency). Given that the topology includes a single bottleneck link and the path RTTs are equal, the optimal estimation of m^2 would be 2, thus halving the aggressiveness of the two sub-flows. We observe that without cross traffic m^2 exhibits roughly zero variance, hence the average value is 1.985, being very close

[9]https://trex-tgn.cisco.com/

[10]TREX traffic is highly dynamic, presenting frequent bursts. The indicated percentages represent the maximum ratios seen during the experiments.

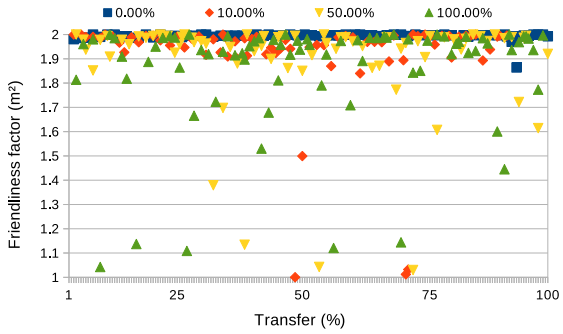


Fig. 9. Estimation of the friendliness factor when competing with dynamic cross traffic in four setups: when cross traffic consumes (during bursts) up to 0%, 10%, 50% and 100% of available bandwidth.

to the optimal value. The deviation of m^2 grows as the amount of cross traffic grows, since the bursts of traffic are more intense; the std. deviation of RTT increases by 500% when the maximum TREX traffic volume is tested. Thereafter, the average value of m^2 is equal to 1.924, 1.894 and 1.874 when TREX bursts consume up to 10%, 50% and 100% of network resources. However, the deviation of m from its optimal value is found too small to penalize friendliness, since the multipath connection consumes roughly 37% of the resources regardless of the volume of the dynamic cross traffic (we exclude the bandwidth share of the TREX traffic from our measurements). The bandwidth sharing ratio is identical to the one reported in Fig. 5, hence we conclude that the RTT-based estimation of m is robust in the face of RTT fluctuations caused by dynamic cross traffic.

VI. Performance Evaluation: WAN

Having evaluated the performance of NMCC in a controlled LAN testbed, we now turn our attention to more realistic WAN environments. Our goal is to examine the performance of NMCC 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, over a detailed implementation of the entire PSI architecture in the NS-3 simulator.[11] We also implemented and deployed LIA within mmTP in order to establish a performance baseline.

For this evaluation, we used the 15 Autonomous System (AS) topologies listed in Table II, taken from the Internet Topology Zoo repository.[12] Since path richness is expected to have an influence on multipath performance, we intentionally selected topologies with different density factors[13] (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)[14] of the AS with a high speed 100 Mbps connection; dual-homed

[11] https://www.nsnam.org
[12] http://www.topology-zoo.org/dataset.html
[13] $density = \frac{2|Edges|}{|Nodes|(|Nodes|-1)}$
[14] A node with degree equal to one.

	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 II
AS TOPOLOGIES TESTED.

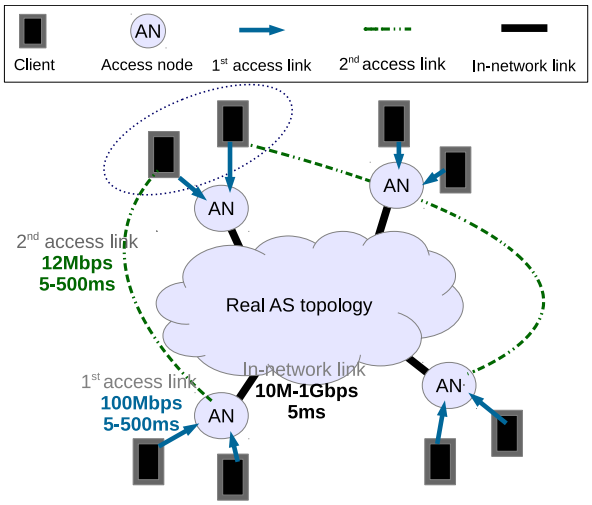


Fig. 10. Client attachment to access nodes in the simulations.

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. 10. 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, a ratio that is also observed in the Internet [36]. 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, in order to reduce the performance volatility, as the transfer rate of TCP connections in congested links can present fluctuations. 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%.

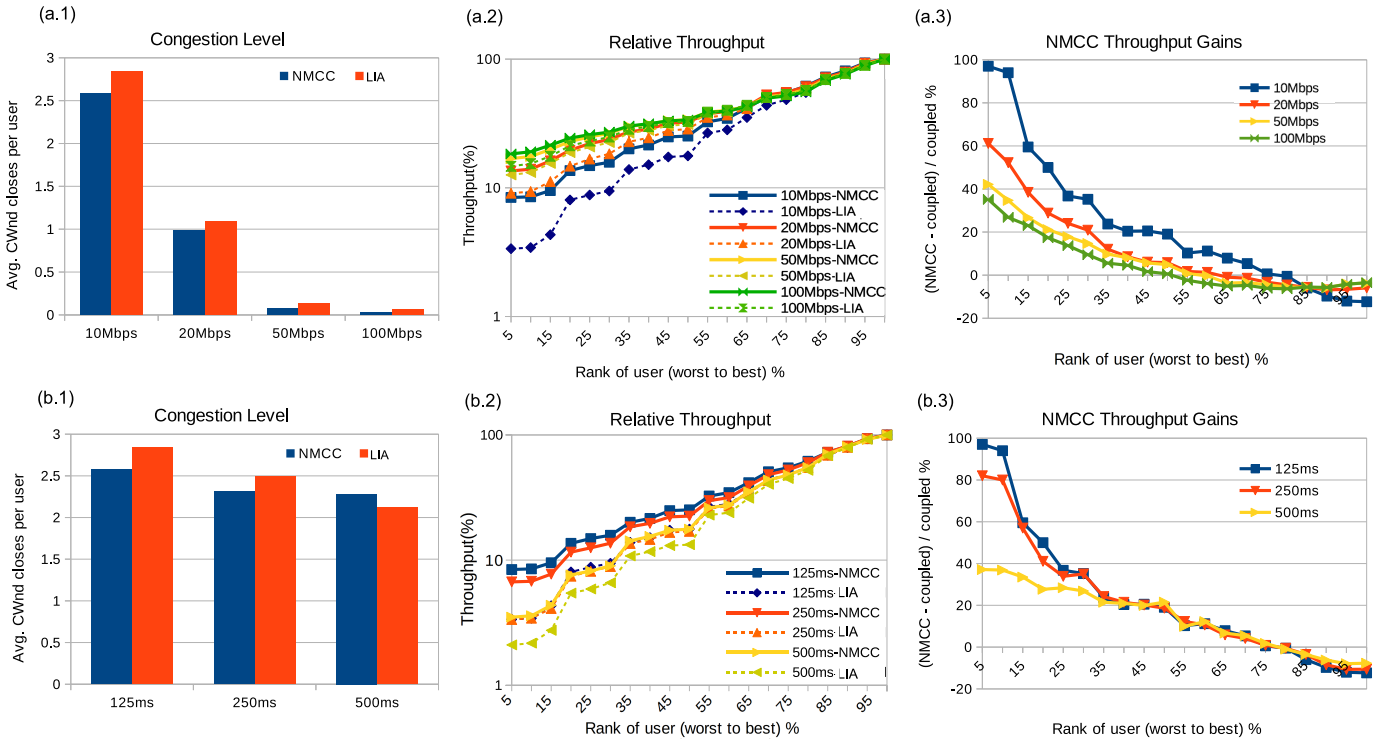


Fig. 11. Performance comparison of NMCC and LIA algorithms in overlapping paths for varying (a) link capacity and (b) delay range.

A. *Friendliness assessment of NMCC*

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 and delay variance. Limited capacity is the reason congestion occurs, while delay variance leads to RTT mismatch among paths, which is known to challenge multipath 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 three metrics to assess performance at the user and network levels. *Congestion Level* reflects the number of timeouts (which trigger window closing events) per connection during the monitoring period. *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 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 faster. Since the aggregate throughput across all connections 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. 11(a.1-3), omitting the results for 1 Gbps as they are identical to those for 100 Mbps. Figure 11(a.1) shows that the congestion level is inversely proportional to the link capacity and that the LIA algorithm produces slightly more time-outs that NMCC. Figure 11(a.2) 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 with narrower links, since in these cases the bottleneck is reached more easily, making friendliness more of an issue. Finally, Fig. 11(a.3) 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 LIA 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. 11(b.1-3). Figure 11(b.1) shows that the number of congestion events is not particularly correlated with path delay variance.Figure 11(b.2) 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. 11(b.3) depicts a substantial throughput increase 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 heterogeneity increases.

B. Assessment of greedy friendliness through TM assistance

Our second set of experiments investigates the throughput gains that greedy friendliness offers to multipath. 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 considered dual-homed clients. We first ran each experiment using single-path 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 used should enhance the benefits of TM assistance, as the probability of finding disjoint paths is increased. We examined 2 and 3 dissemination paths, in addition to the single-path baseline; 3 is the maximum number of paths expected to exhibit gains, according to [13]. Finally, in addition to the k-shortest paths algorithm, which can return both disjoint and non-disjoint paths, we also used Bhandari's algorithm [37], 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 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 12 depicts the Multipath Throughput Gain for each topology in some representative scenarios. In all cases, the access link delays are drawn from the 5-125 ms range. In Fig. 12(a) we show the gains offered with 2 and 3 paths, with and without TM assistance, in a resource-constrained network with 10 Mbps intra-AS links. The average gain for 2 and 3 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 3rd path are exclusively due to the avoidance of in-network bottlenecks. On the other hand, TM assistance does not have a noticeable effect on multipath gain, even though 32% and 51% of users get disjoint paths in the 2 and 3 path scenarios, respectively.

Raising the intra-AS link speed to 100 Mbps makes TM assistance matter: on average, it increases gains by 2% compared to the case without TM assistance. As a result, with 3-path

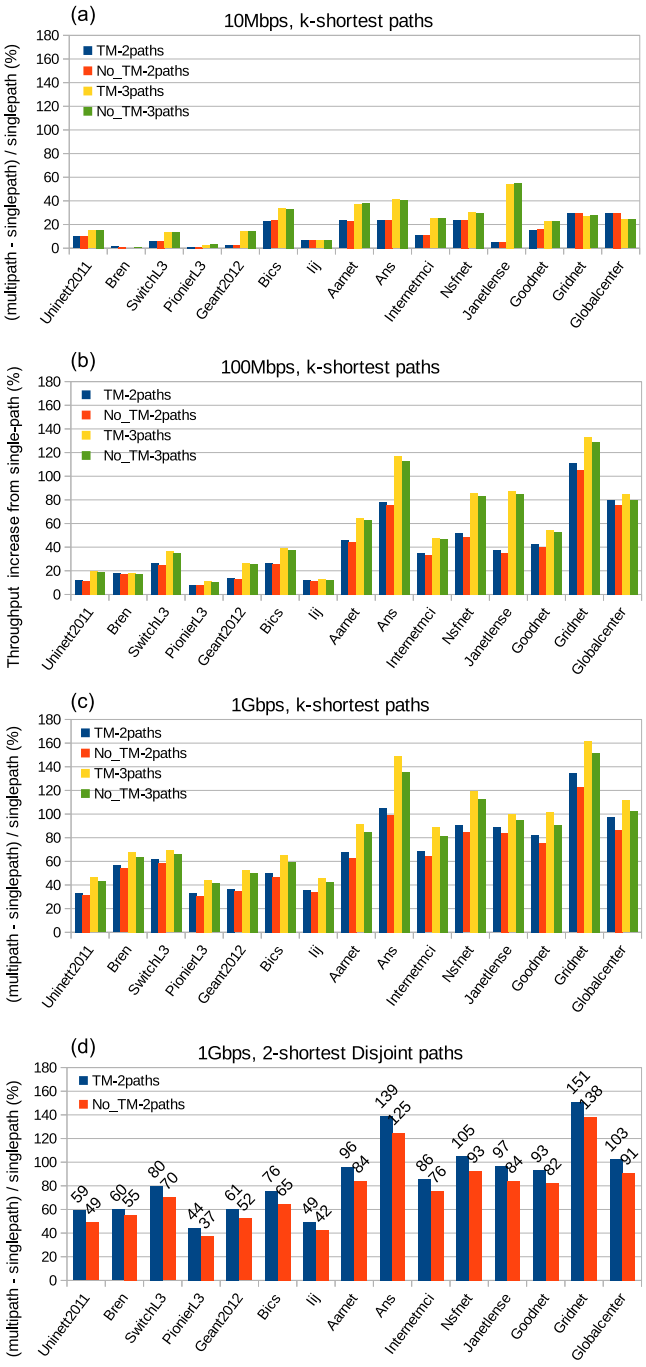


Fig. 12. Performance gains of multipath with and without TM assistance compared to single-path: (a-c) Yen's path formation algorithm and (d) Bhandari's disjoint path formation algorithm. Topologies are plotted in ascending order by density from the left to the right.

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. 12(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 paths, respectively, 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. 12(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 paths and TM assistance (see Fig. 12(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 better exploits 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.

VII. Greedy friendliness IP networks

Our hybrid congestion control mechanism for multipath transfers relies on an in-network scheme that reports shared bottlenecks to the endpoints. The PSI architecture is an appropriate terrain for this design, since it provides a TM function that (i) is aware of network topology and (ii) interacts with the endpoints. The TM knows the physical structure of the network, so it can easily detect shared bottlenecks. In addition, when two pub/sub requests are matched, the TM sends the LIPSIN identifiers of the paths directly to the applications, therefore it directly pushes the topological information to the users. Finally, any changes to these paths, whether due to failures or load balancing decisions, require the distribution of new LIPSIN identifiers, therefore 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 accurate as routing decisions change.

A. Multi-Protocol Label Switching (MPLS)

A technology that offers centralized path selection and source routing in IP networks is *Multi-Protocol Label Switching* (MPLS) [38]. MPLS is used in backbone networks, where it applies QoS-based traffic control by classifying flows and forwarding them via predefined routes. Short fixed-length labels are assigned to packets at the ingress to an MPLS cloud, and these labels are used to make forwarding decisions inside the MPLS domain. The path computation process is generic, allowing route computation by the underlying routing protocols or explicit definition by a network administrator. Multipath deliveries are also encouraged, in the form of splitting single-path connections into several sub-flows at the ingress router.

Currently, MPLS is primarily used to apply domain-scale traffic engineering, rather than to enhance the performance of individual connections, hence, connection splitting is done with static sharing weights for general load balancing. Consequently, congestion control takes place at the actual endpoints (i.e the users), while the ingress MPLS router is confined to the flow control of the available paths. However, if we consider the ingress router as the congestion manager of the MPLS cloud, as it splits the flow, assigns labels to each of its sub-flows and becomes the endpoint of a local MPLS service, then our network-assisted congestion control can be applied *inside* the MPLS network. Specifically, when the network administrator discovers multiple paths for bulk flows and sends the corresponding labels to the ingress router, it can also send information on how flows are grouped depending on path sharing, as described in Sec. IV-A. The ingress router, which runs NMCC for each bulk flow, can then exploit this information along with source routing to selectively engage the friendliness mechanism. Although, the integration of our design with MPLS is feasible and can be realistic under certain conditions, the centralized placement of congestion control at the ingress router raises serious scalability concerns.

In order to reduce the computational cost, we propose the application of congestion control on groups of connections that follow the same route in the MPLS network. Again, the ingress router splits the connections, like Split-TCP [33], and practices the congestion control within the MPLS cloud. However, the ingress router deploys one persistent multipath connection to each egress router in order to transfer the data of the individual incoming connections that *are sent to the same egress router over the same routes*. The establishment of the persistent multipath connection requires that the network manager discovers the paths and computes the associated *group_ids*, that are sent to the ingress router in order to control friendliness accordingly. Thereupon, the congestion control of grouped connections is jointly managed by the ingress router, that exploits NMCC or LIA, and the network administrator, that acts similarly to the PSI's TM. The computation overhead of this solution consists of the cost for establishing one congestion loop per egress router,[15] and the cost of splitting the connections, like Split-TCP, thus being significantly reduced. However, the technical details on grouping the connections and jointly performing the congestion control of multiple connections need to be further explored.

B. Software-Defined Networking (SDN)

Software-Defined Networking (SDN) [39] is a novel networking scheme that can be used to achieve similar goals

[15]More congestion loops are required when multiple sets of routes lead to the same egress router, f.i., when Differential Services are enabled.

to PSI, including centralized path selection. In SDN, programmable switches forward packets based on "dynamic" rules that bind flow identifiers, such as fields of the IP header, with outgoing network interfaces. These rules are defined by a centralized controller that is aware of the network topology and forms virtual circuits by explicitly sending rules to all on-path switches. Circuit creation can be reactive, where a switch ask the controller's assistance when no rule can be applied to a received packet, or proactive, where the controller forms the route a priori, for example, to achieve load balancing. In both cases, SDN operation is transparent to the endpoints that manage congestion control.

As the SDN controller does not communicate with endpoints, which means that it cannot pass any topological information to them directly, 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 or groups of 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 switch, as well as instructions on how to tag each IP header so as to implicitly select the appropriate path.

In order to enable greedy friendliness at connection-level, we propose the exploitation of MPTCP and push the topological information of every connection to the endpoints through the MPTCP packet header. In this case, the SDN controller performs as the *man in the middle* during the establishment of MPTCP sub-flows, thus forming a centralized topological assistance module similar to PSI's TM. We assume MPTCP-aware SDN switches that monitor traffic in order to detect the handshake messages of (MP)TCP; MP_CAPABLE and MP_JOIN options are enabled during the (re-)establishment of a (sub-)flow [40]. The SDN switches forward these MPTCP messages to the controller that, in addition to its standard operation, stores the routing state and the MPTCP state of the sub-flow locally. The controller can associate the sub-flows of the same multipath session through the authentication attributes carried in the MPTCP header, thus being aware of the dissemination paths and the shared performance bottlenecks.

Upon the receipt of a MPTCP handshake message the controller either stores the status of the connection (if it is the first sub-flow to establish) or updates the connection status (if an additional sub-flow is to be established). The status of the connection includes the header attributes of the MPTCP handshake messages and the physical routes assigned to each sub-flow; the latter is soft-state memory (and can be derived) for enhancing the performance during the second step. Then, the controller compares the new path with the established (if any) and determines the existence of shared links and the corresponding *group_ids*, following the procedure presented in Section IV-A. The *group_id* of each path is delivered via a special field, named *PATH_OVERLAP*, that is inserted in the MPTCP packet's header. The MPTCP packet is send back to the SDN switch, which forwards it to its original destination.

Placing the SDN controller in the middle of the connection establishment is both a gift and a curse. On the one hand, the topological information remains valid throughout the entire transmission, even if sub-flows change on-the-fly. Routing decisions and bottleneck detection are made by the SDN controller, thus offering consistency. On the other hand, the centralized nature of the controller rises scalability concerns with regard to the memory costs of the design. The processing overhead for the controller is not significant, since the procedure to detect shared bottlenecks is found to be relatively inexpensive (Section IV-A3), but the introduced memory state at the controller can become critical as the number of MPTCP connections grows. Several techniques can be used to mitigate this cost, thus enhancing the scalability of the design, however their analysis goes beyond the scope of this paper.

VIII. Friendliness through growth rate normalization

According to our definition of friendliness, a friendly multipath protocol must not get more bandwidth than any single-path connection on any available path, i.e., at any time t the bandwidth share (or throughput) of multipath ($T^{MP}(t)$) and single-path ($T^{SP}(t)$) must be equal:

$$T^{MP}(t) = T^{SP}(t) \tag{12}$$

The function of throughput in time, $T(t)$, can be studied in congestion rounds that last N RTTs. During the congestion round, we assume that the throughput increases linearly for $N-1$ RTTs, while packets are delivered successfully. Then, the throughput remains idle until the retransmission timer expires on the N^{th} RTT, when throughput is reduced multiplicatively, determining the end of the congestion round. We approximate the throughput as shown in the following:

$$T(t) = \left\{ \begin{array}{ll} ta, & \text{for } 0 \leq t \leq N-1 \\ (N-1)a, & \text{for } N-1 \leq t < N \\ (N-1)a/2, & \text{for } t = N \end{array} \right\} \tag{13}$$

where t is real number that counts RTTs and a is real number that defines the amount of throughput increase every RTT (usually 1 MSS). The function of throughput in time is continuous in the closed interval $[0, N-1]$ and differentiable in the open interval $(0, N-1)$.

In Section IV-B, we model the throughput increase rate of NMCC in order to be equal with the throughput increase rate of the best single-path available, hence in field $(0, N-1)$:

$$\frac{\partial T^{MP}}{\partial t} = \frac{\partial T^{SP}}{\partial t} \tag{14}$$

Using Lagranges Mean Value Theorem, we can prove that if two functions have equal derivatives at all points, then they differ by a constant. Assuming that $T^{MP}(0) = T^{SP}(0)$, since connections start with the minimum congestion window, we argue that the constant that describes the difference between the throughput of multipath and single-path is zero, thus delivering equal throughput and, in turn, friendliness during the time interval $(0, N-1)$. In addition, we notice that throughput does not change in the time interval $[N-1, N)$, between the last

received packet and the expiration of the retransmission timer, hence friendliness is not affected. [16] Finally, in Section IV-B, we model the throughput decrease rate in order to be equal with the throughput decrease rate of the best single-path available, thus equalizing the throughput reduction at time N, when the congestion round ends. Thereupon, we conclude that the throughput of multipath and single-path is equal during the entire congestion round.

IX. Conclusions

We presented a hybrid congestion control algorithm for multipath transport, consisting of NMCC and an in-network assistance mechanism. Being applicable to networks that support centralized path computation, such as PSI, SDN and MPLS, our design offers friendliness to single path connections using TCP-like congestion control, while increasing the utilization of network resources. It achieves this by detecting shared friendliness bottlenecks and managing aggressiveness accordingly, a scheme called greedy friendliness. We have implemented the congestion control algorithm in the PSI architecture prototype and evaluated its performance 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 effectiveness of our design, since greedy friendliness amplifies by roughly 10% the performance gains of multipath connections in WAN environment, i.e., when multipath users get 100% more throughput than single-path ones, then the greedy friendliness assistance will provide 10% additional gains. In addition, we observe that NMCC achieves friendliness more accurately than LIA congestion control algorithm in case of short transfers, as well as in a wide range of real network topologies.

Acknowledgement

This research was supported by the AUEB Research Center under project "A THOROUGH STUDY OF MPTCP'S CONVERGENCE LATENCY TO TCP-FRIENDLINESS (MPTCP-FRIEND)".

References

[1] S. Androutsellis-Theotokis and D. Spinellis, "A survey of peer-to-peer content distribution technologies." *ACM Computing Surveys*, vol. 36, no. 4, pp. 335–371, 2004.

[2] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, "Design, implementation and evaluation of congestion control for multipath TCP," in *Proc. of the USENIX NSDI*, 2011.

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

[4] J. Widmer, R. Denda, and M. Mauve, "A survey on TCP-friendly congestion control," *IEEE Network*, vol. 15, no. 3, pp. 28–37, May 2001.

[5] M. Honda, Y. Nishida, L. Eggert, P. Sarolahti, and H. Tokuda, "Multipath congestion control for shared bottleneck," in *Proc. of the PLFDNeT Workshop*, 2009.

[6] P. Key, L. Massoulie, and D. Towsley, "Path selection and multipath congestion control," in *Proc. of the IEEE INFOCOM*, 2007, pp. 143–151.

[7] M. Becke, T. Dreibholz, H. Adhari, and E. P. Rathgeb, "On the fairness of transport protocols in a multi-path environment," in *Proc. of the IEEE ICC*, 2012, pp. 2666–2672.

[16] In case of Fast Recovery, this interval is zero, hence our analysis is still valid.

[8] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos, "A survey of information-centric networking research," *IEEE Communications Surveys and Tutorials*, vol. 16, no. 2, pp. 1024–1049, 2014.

[9] P. Jokela, A. Zahemszky, S. Arianfar, P. Nikander, and C. Esteve, "LIPSIN: line speed publish/subscribe internetworking," in *Proc. of the ACM SIGCOMM*, 2009, pp. 195–206.

[10] Y. Thomas, C. Tsilopoulos, G. Xylomenos, and G. C. Polyzos, "Multisource and multipath file transfers through Publish-Subscribe Internetworking," in *Proc. of the ACM SIGCOMM ICN Workshop (poster session)*, 2013, pp. 43–44.

[11] ——, "Accelerating file downloads in publish-subscribe internetworking with multisource and multipath transfers," in *Proc. of the World Telecommunications Congress (WTC)*, 2014, pp. 1–6.

[12] Y. Thomas, G. Xylomenos, C. Tsilopoulos, and G. C. Polyzos, "Multiflow congestion control with network assistance," in *Proc. of the IFIP Networking*, 2016.

[13] 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, pp. 2176–2213, 2015.

[14] T. Henderson, E. Sahouria, S. McCanne, and R. Katz, "On improving the fairness of TCP congestion avoidance," in *Proc. of the IEEE GLOBECOM*, vol. 1, 1998, pp. 539–544 vol.1.

[15] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext transfer protocol–HTTP/1.1," Tech. Rep., 1999.

[16] B. Cohen, "Incentives build robustness in BitTorrent," in *Proc. of the Workshop on the Economics of Peer-to-Peer Systems*, Berkeley, CA, USA, Jun 2003, pp. 116–121.

[17] A. Ford, C. Raiciu, M. Handley, S. Barre, and J. Iyengar, "Architectural guidelines for multipath TCP development," Tech. Rep., 2011.

[18] R. Khalili, N. Gast, M. Popovic, U. Upadhyay, and J.-Y. Le Boudec, "MPTCP is not pareto-optimal: performance issues and a possible solution," in *Proc. of the ACM CoNEXT*. ACM, 2012, pp. 1–12.

[19] 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, pp. 596–609, 2016.

[20] Y. Cao, M. Xu, and X. Fu, "Delay-based congestion control for multipath TCP," in *Proc. of the IEEE ICNP*. IEEE, 2012, pp. 1–10.

[21] 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. of the USENIX Annual Technical Conference*, 2004, pp. 99–112.

[22] 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, pp. 381–395, 2002.

[23] S. Ferlin, Ö. Alay, T. Dreibholz, D. A. Hayes, and M. Welzl, "Revisiting congestion control for multipath TCP with shared bottleneck detection," in *Proc. of the IEEE INFOCOM*. IEEE, 2016, pp. 1–9.

[24] S. Zannettou, M. Sirivianos, and F. Papadopoulos, "Exploiting path diversity in datacenters using MPTCP-aware SDN," in *Proc. of the IEEE ISCC*, Messina, Italy, June 2016.

[25] 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. of the IEEE AINA*, Gwangju, Korea, March 2015.

[26] D. Trossen, M. Sarela, and K. Sollins, "Arguments for an information-centric internetworking architecture," *SIGCOMM Computer Communications Review*, vol. 40, no. 2, pp. 26–33, Apr. 2010.

[27] G. Xylomenos, X. Vasilakos, C. Tsilopoulos, V. Siris, and G. Polyzos, "Caching and mobility support in a publish-subscribe internet architecture," *IEEE Communications*, vol. 50, no. 7, pp. 52–58, July 2012.

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

[29] 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. of the IEEE ICME MuSIC Workshop*, 2015, pp. 1–6.

[30] D. Eppstein, "Finding the k shortest paths," *SIAM Journal on computing*, vol. 28, no. 2, pp. 652–673, 1998.

[31] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley, "Improving datacenter performance and robustness with multipath TCP," vol. 41, no. 4, pp. 266–277, 2011.

[32] C. Paxson, M. Allman, J. Chu, and M. Sargent, "Computing TCP's retransmission timer," 2000.

[33] S. Kopparty, S. V. Krishnamurthy, M. Faloutsos, and S. K. Tripathi, "Split TCP for mobile ad hoc networks," in *Global Telecommunications Conference, 2002. GLOBECOM'02. IEEE*, vol. 1. IEEE, 2002, pp. 138–142.
[34] G. Parisis, D. Trossen, and D. Syrivelis, "Implementation and evaluation of an information-centric network," in *Proc. of the IFIP Networking*, 2013, pp. 1–9.
[35] J. Yen, "Finding the k shortest loopless paths in a network," *Science Management*, vol. 17, no. 11, pp. 712–716, 1971.
[36] J. Heidemann, Y. Pradkin, R. Govindan, C. Papadopoulos, G. Bartlett, and J. Bannister, "Census and survey of the visible internet," in *Proc. of the ACM IMC*, 2008, pp. 169–182.
[37] R. Bhandari, "Optimal diverse routing in telecommunication fiber networks," in *Proc. of the IEEE INFOCOM*, 1994, pp. 1498–1508.
[38] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus, "Requirements for traffic engineering over mpls," RFC 2702, 1999.
[39] H. Kim and N. Feamster, "Improving network management with software defined networking," *IEEE Communications*, vol. 51, no. 2, pp. 114–119, 2013.
[40] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, "Tcp extensions for multipath operation with multiple addresses," 2013.