MP-ALM: Exploring Reliable Multipath Multicast Streaming with Multipath TCP

Anwaar Ali*§, Junaid Qadir*¶, Arjuna Sathiaseelan†‖, Kok-Lim Alvin Yau‡ and Jon Crowcroft†**
*Electrical Engineering Department
Information Technology University (ITU), Lahore, Pakistan
Email: §anwaar.ali@itu.edu.pk; ¶junaid.qadir@itu.edu.pk
†Computer Laboratory
University of Cambridge, UK
Email: ‖arjuna.sathiaseelan@cl.cam.ac.uk; **jac22@cl.cam.ac.uk
‡Faculty of Science and Technology
Sunway University, Malaysia
Email: koklimy@sunway.edu.my

***Abstract*—In this paper, we present a novel idea of multipath multicast, which is imperative to bandwidth intensive applications, in the context of multimedia streaming. In addition to congestion control, multipath TCP (MPTCP) has been proposed to establish multiple paths in a network to improve network reliability. Application-layer multicast (ALM) has been proposed to leverage end systems instead of dedicated routers to multicast that is important for an easy large-scale deployment as compared to IP-based multicast. This paper presents our novel idea of multipath multicast in the form of a simple experimental framework called MP-ALM in which we combine the multiplicity feature of MPTCP with the application-layer multicast (ALM). We extensively simulate MP-ALM using ns-3 and use *iPerf* to generate streaming multicast-MPTCP traffic. Simulation results show that MP-ALM can be beneficial for a better user experience and reduced overall network congestion in the perspective of multicast multimedia streaming.**

***Index Terms*—Multipath, Application-Layer Multicast, Multipath TCP, Performance Analysis, Multimedia Streaming.**

I. Introduction

A. Motivation

Multimedia streaming constitutes a significant portion of the Internet traffic with approximately 65% of the total Internet traffic in North America being multimedia streaming [1]; hence it is one of the major contributors to network congestion. Multiplicity of paths is imperative to provide improved fault tolerance, throughput and reliability [2]–[6]. In this study, we deploy multiplicity of paths to achieve larger throughput gains in multimedia streaming. By deploying multiple paths that spread the streaming traffic over a larger set of network resources, the overall congestion in the network can be reduced.

Multicast enables one-to-many connections, which is important to applications such as video streaming [7], online gaming [8] and IPTV [9]. Although there are numerous works that study and propose new methods for multipath unicast [10]–[12], little effort has been dedicated to explore the frontier of multipath multicast, which is the subject matter of our paper. Specifically, we use multicast multimedia streaming and combine it with the multiplicity feature of paths by deploying MPTCP to give a unique idea of MP-ALM, which is our experimental framework to combine and evaluate the scenario of multipath multicast in a network. Mutlipath TCP (MPTCP) [13], [14], which is a recent extension to single-path TCP (SPTCP) (henceforth we use SPTCP to mean the traditional TCP), has the capability to deploy multiple paths or subflows with distinctive IP addresses among the subflows. We deploy MPTCP as the transport-layer protocol in our MP-ALM framework to enjoy the features of multiplicity of paths and reliability, and the benefits of congestion control.

The reason for choosing *application-layer multicast (ALM)* for our study is twofold. First, the ALM gives us the freedom to introduce reliability and congestion control at lower layers. Second, if we want to deploy TCP (for reliability and congestion control) instead of UDP at the transport layer then we can not adopt conventional IP-based multicasting approach. The reason for this is that TCP is always between two points and it can not operate in a one-to-many configuration, which is required for IP-based multicasting. We have to adopt an indirect approach to multicast with TCP and ALM enables us to do so. ALM is different as compared to network-layer (IP-based) multicasting in that it makes use of the end-systems (and point-to-point connections among them) rather than dedicated routers to multicast.

The main goal of this paper is to evaluate our idea of multipath multicast by evaluating the performance of MPTCP through our proposed MP-ALM framework for multicast streaming. Networking testing tool such as *iPerf* is used at the application layer to establish parallel and distinctive point-to-point connections for ALM. Internet companies, such as YouTube and Netflix, that deploy SPTCP at the transport layer [15] can benefit from our study by replacing SPTCP with MPTCP, combined with intelligent ALM strategies, to make the streaming experience of their clients even better and to reduce the overall congestion in the network.

B. Contributions of this Paper

Following are the main contributions of this paper.

DOI 10.1109/LCN.2016.26

TABLE I: Comparison with the existing reliable multipath multimedia streaming schemes

Study	*Performance Metrics(s)*	*Support for Multicasting*	*Dedicated Path Selection*	*Dedicated Traffic Splitting*	*Transport-Layer Protocol*
MultiTCP [10]	Bandwidth	✗	Required	Required	TCP
DMP [11]	Throughput	✗	Required	Required	TCP
Kuschnig et al. [12]	Throughput	✓	Required	Required	TCP
MP-ALM	Throughput, Latency, Jitter, Packet Loss	✓	Not Required	Not Required	MPTCP

1) While a previous work [16] has examined multipath based ALM in the specific scenario of cognitive radio networks; we, however, present our discussion in a more generic fashion. To the best of our knowledge, the idea of multipath multicast using MPTCP (for reliability and the multiplicity of paths) is presented in this paper for the first time.
2) Through MP-ALM framework, we try to quantify the performance of MPTCP in terms of throughput, latency, jitter and packet loss along with the friendliness of MPTCP towards SPTCP. In this way our work can also be considered as a benchmark study for the performance evaluation of MPTCP.

C. Organization of this Paper

In Section II, we first provide a brief background on the operation of MPTCP and the concept of *friendliness* of MPTCP towards SPTCP. In Section III, we present the related work. In Section IV, we describe our network model. In Section V, we discuss in detail all the simulation results. In Section VI, we give potential future directions and real world deployment challenges to enhance the study presented in this paper. Finally, in Section VII we conclude our paper.

II. Background

A. Multipath TCP (MPTCP)

Multipath TCP (MPTCP) is a recent extension to traditional SPTCP [13], [14]. It renders a data transfer between two points to be split into multiple data streams over multiple paths. MPTCP carries out this data split over multiple IPs. Hosts, specially the ones that are multihomed, with the provision of multiple NICs (or IPs) can make use of the multiplicity of MPTCP. As a result such hosts can utilize all, or a subset, of available IP addresses for throughput gains.

Middleboxes, like firewalls and NATs, used to be the biggest hurdle against the extension/ modification of existing protocols like SPTCP. MPTCP circumvents this issue by making use of the *options* field in the SPTCP header. Each subflow an MPTCP connection adds is treated just like a SPTCP. In this way, MPTCP is able to deal with middleboxes. Next we briefly describe how MPTCP operates.

1) Connection Setup: An MPTCP connection is established between two points just like SPTCP's famous *three-way handshake*. At the start of this handshake, an MPTCP enabled host sends *Multipath Capable (`MP_CAPABLE`)* option in the SYN packet. If it receives the same `MP_CAPABLE` option in the SYN/ ACK packet then it is established that both parties of the connection understand MPTCP and are ready to create first subflow of their data transfer. The `MP_CAPABLE` option is also put into the last ACK packet of the three-way handshake and an MPTCP connection with one subflow is established between two points. In the case when there is no `MP_CAPABLE` option found in the SYN/ ACK then MPTCP automatically falls back to regular SPTCP.

2) Adding a Subflow: Subject to the availability of an extra IP, a host can establish a second subflow in its data transfer. Creating a new subflow in the existing MPTCP connection is just like making a new SPTCP connection with usual SYN, SYN/ ACK, ACK transfer. This time in each of these three packets an `MP_JOIN` option is also present. After the success of such a three-way handshake a second subflow, which is associated to the existing MPTCP connection, is created between the two end-points. Now data stream can be split over these two subflows.

3) Sequence Number Space in MPTCP: There are two types of sequence space in MPTCP. One sequence space is for each subflow and a separate sequence space for the overall connection. This is done so that middleboxes do not drop any packet stream that they perceive to be having discontinuous sequence number space.

B. Friendliness of MPTCP towards SPTCP

MPTCP is designed in such a manner so that adding multiple subflows does not starve other concurrent SPTCPs. With all the subflows of an MPTCP connection the maximum bandwidth that an MPTCP connection gets over a link is equal to the maximum bandwidth that a SPTCP connection can achieve over the same link. At first this sounds a bit confusing, as to why use MPTCP at all then? The difference arises when there are multiple links. SPTCP can only utilize one link while an MPTCP connection can create an additional subflow, subject to the availability of an extra IP address, through the other available link. MPTCP then routes most of its traffic through the subflow, over a link, that is less congested. In this manner, MPTCP remains friendly with SPTCP and also achieves throughput gains when extra links are present. In the case of a bottleneck link, where SPTCP and MPTCP connections share a same link, MPTCP behaves just like any other SPTCP no matter how many subflows it creates.

As an example, consider two links as in Fig. 1, A and B, each of 100 Mbps capacity. Let us assume that link A is being shared by two SPTCPs and three MPTCPs while the link B is being shared among four SPTCPs. In this scenario, instead of congesting link A, MPTCPs pool the capacity of all the

available links (in our case links A and B) and send a portion of their traffic, through subflows, over the link that is less congested (i.e., link B shared by four SPTCPs). Now two links are being treated as one big link of 200 Mbps being shared equally among nine connections, hence each getting a share of 22 Mbps. If, however, both the links were shared only by the SPTCPs then in link A each connection would have 20 Mbps each while in link B 25 Mbps would have been for each SPTCP connection. This traffic splitting property of MPTCP is known as its *congestion balancing* property [17].

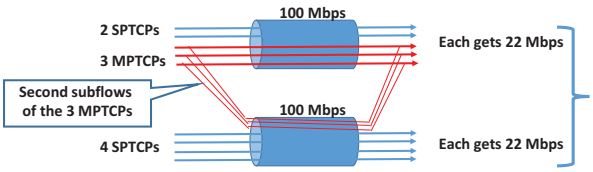

Fig. 1: MPTCP congestion balancing property and friendliness towards SPTCP

III. Related Work

Multicasting using multiple paths is relatively new as compared to multipath unicast. Ali et al. introduced and discussed this idea in [16]. However, they deployed multipath multicasting in the particular scenario of cognitive radio networks. Learning automata based reinforcement learning was used for the design of multipath multicast routing protocol. Their protocol outperforms the commonly used multicasting routing protocol (COCAST) for cognitive radio networks. Our work, however, is distinguished from [16] because of its generic nature and the fact that we deploy MPTCP at the transport layer that automatically takes care of the feature of multiplicity. Additionally, by introducing MPTCP for multicasting we make our communication inherently reliable that also takes the overall network congestion into consideration.

Now we review a few works that also make use of multiplicity of paths for multimedia streaming and compare these efforts with our idea. In [10], Tullimas et al. propose an application-layer solution, which they call MultiTCP. Their solution is a receiver-oriented multimedia streaming system that utilizes multiple SPTCP connections for the same application for bandwidth gains that would otherwise have not been achievable if only one SPTCP connection was deployed. Another scheme is called *Dynamic MPath-streaming* that is for live multimedia streaming over multiple paths using SPTCP [11]. In this scheme, path selection for sending streaming traffic is based on *implicit inferring* technique. The paths with higher achievable throughput drain the sending buffers of corresponding SPTCP connections quickly and hence these paths are *implicitly* selected first. According to the authors, this scheme outperforms single path and static streaming schemes. Another interesting work proposes a client-driven video streaming approach over SPTCP [12]. This scheme deploys multiple HTTP request-response streams. This design is particularly for lossy network infrastructure where, the fluctuations in throughput (due to SPTCP's congestion control mechanism) can deteriorate the streaming experience. Multiple HTTP streams address this fluctuation problem and at the same time it is also friendly towards the concurrent SPTCP connections.

It can be observed from the works presented above that many efforts have been made to deploy multiplicity for efficient multimedia streaming. Mostly distinct and parallel SPTCP connections are used for such purposes. Dedicated path selection, traffic splitting and packet ordering schemes have to be designed for such schemes. There is also a risk of starvation of the concurrent SPTCP connections (of applications sharing the same network resources). This happens by deploying multiple parallel SPTCPs for a single application. Dedicated efforts are made to ensure friendliness of such schemes towards other concurrent SPTCPs. With MPTCP, the benefits of multiplicity for throughput and other performance gains can be easily achieved. In the case of MPTCP, one does not have to worry about separate congestion control and traffic splitting mechanisms on the available paths. Also, MPTCP is very friendly towards SPTCP. It adjusts its own sending rates over all of its subflows so that there is negligible effect on concurrent SPTCPs. Table I provides a quick comparison between the works presented here with our MP-ALM framework. Our scheme is distinguished from these works as it deploys MPTCP at the transport layer.

IV. Network Model

We construct a simple tree topology (Fig. 2) with one multimedia streaming server (the parent node) and a varying multicast group size (the child nodes). Both the server and the clients are MPTCP enabled. The server makes distinct point-to-point MPTCP connections to each of the clients in a given multicast group to disseminate streaming content towards them. In this study we assume that server and all the clients (in a multicast group) can create an equal number of subflows for a given MPTCP connection (i.e., all the nodes are equipped with an equal number of IP interfaces).

Fig. 2 shows a *one-to-many* configuration. The source node is denoted by s. The set of client nodes, i.e., a multicast group, is represented by D where:

$$D = \{d_1, d_2, ..., d_n\} \quad where \quad n \geq 1 \tag{1}$$

R represents the set of intervening routers:

$$R = \{r_1, r_2, ..., r_m\} \quad where \quad m \geq 1 \tag{2}$$

Set of links is denoted by L where:

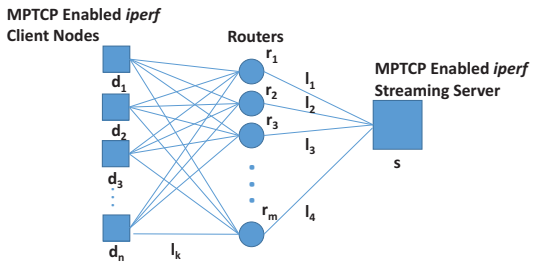

Fig. 2: Network Model: client nodes = $D = \{d_1, d_2, ..., d_n\}$; routers = $R = \{r_1, r_2, ..., r_m\}$; and links = $L = \{l_1, l_2, ..., l_k\}$

$$L = \{l_1, l_2, ..., l_k\} \quad where \quad k = m(n+1) \tag{3}$$

Different values of n and m create the following test cases:

1) Unipath unicast where we have $n = 1 \quad and \quad m = 1$
2) Unipath multicast where we have $n > 1 \quad and \quad m = 1$
3) Multipath unicast where we have $n = 1 \quad and \quad m > 1$
4) Multipath multicast where we have $n > 1 \quad and \quad m > 1$

A. Limitations of Our Model

We are well aware of the simplicity of our network model presented here. The real-world topologies are much more complex than this model. Our main purpose in this study is to evaluate (and register) the notion of multipath multicast through pushing the limits of MPTCP in terms of number of subflows and letting multiple MPTCP connections interfere with eachother as much as they can. We list the limitations and the assumptions we make as follows:

- In this study, we do not specifically construct overlay trees for the purpose of ALM. The topology considered here is a simple tree topology with one parent node connected with multiple child nodes through intervening routers.
- We assume in this study that the streaming server and the clients, in a multicast group, are all equipped with an equal number of IP interfaces and they can create an equal number of subflows at the same time.
- We use *iPerf* application in server mode, instead of an actual multimedia server, to simulate our streaming server.

V. Simulation Setup and the Results

Here we first briefly provide a background on the Direct Code Execution (DCE): a framework we deploy in ns-3 for our simulations. This discussion is followed by the complete description of simulation setup and a detailed discussion on the obtained results.

A. Direct Code Execution (DCE)

Direct code execution (DCE) is a framework that enables running the original protocols, applications and Linux kernel commands within the simulation environment of ns-3. This framework is proposed [18] for the reproducible networking research with deterministic reproducibility. In its core architecture, DCE adopts the traditional *library operating system approach (LibOS)* that has three main components: i) *core module* that provides virtualization of stacks, heaps and the global memory, ii) the *kernel layer* that provides an execution environment for Linux network stack in ns3 environment, and iii) the *POSIX layer* that provides standard socket APIs used by the emulated applications. This approach renders DCE capable of reproducible and realistic network research by integrating real Linux kernel and application code with ns-3. The DCE framework satisfies the five requirements of reproducible research: i) experimental realism, ii) topology flexibility, iii) low-cost reproducibility with ease, iv) scalability of experiments and v) debugging facility.

B. Enhancing a Use Case of Direct Code Execution (DCE)

We setup DCE with ns-3 and expand the code provided in *dce-iperf-mptcp.cc* file by Tazaki et al. [19]. The code in this file sets up two nodes equipped with *iPerf* and MPTCP. One of the nodes is configured as an *iPerf* client and the other one as the *iPerf* server. These nodes can then be connected via a variable number of routers (m) that can be given as an input argument via command line during initialization. In this code the number of routers is equal to the maximum number of subflows that a node can create. As an example, if we have $m = 2$ then there are two routers in the network: r_1 and r_2 that are connected with each of the nodes in the topology (i.e, the server node and all the client nodes). As a result, each node has two distinct IP interfaces (each connected to one of the routers) and hence each node can create maximum of two subflows. We use the same notation m to show the number of routers and also the maximum number of subflows that a node can create.

We modify the code, provided in dce-iperf-mptcp.cc, and make the number of client nodes (denoted by $|D|$ in our case) dynamic as well. $|D|$ represents the multicast group size. We also include the option to enable/ disable MPTCP on any one of the nodes, simply by passing an input argument during the initialization. This is done in order to perform the experiments related to the *friendly* behavior of MPTCP by initializing MPTCP and SPTCP connections concurrently by the same *iPerf* server.

C. Simulation Setup

We implement our topology (as described in Section IV) in ns-3 with DCE so that all the nodes in the topology are equipped with the actual implementations of MPTCP and *iPerf*. The version of MPTCP used in our study is 0.89. Each node in the topology uses *iPerf* to establish an MPTCP connection with other nodes. Each node runs the *iPerf* application for 100 seconds. This is done by setting the *time* argument of

iPerf to 100. As shown in Fig. 2, the nodes on the left hand side are configured as the *iPerf* clients and the node on the right hand side as an *iPerf* server. We use the *P* option of *iPerf* on the server side to specify the number of parallel connections that depends on the value of $|D|$ (i.e., the multicast group size). We arbitrarily set the values of data rate and delay of the links for all the clients to 5 Mbps and 10 ms respectively. At the server side the values of data rate and delay are arbitrarily set to 100 Mbps and 1 ns respectively.

For multiple iterations, we vary $|D|$ from 1 to 15 resulting in a total of 15 iterations. For each of the 15 iterations we increase the number of subflows, for both clients and the server, from $m = 1$ to $m = 8$. Overall we run 120 simulation instances (i.e., 15 iterations of multicast group size each for the 8 different number of subflows). We average the results of performance metrics for all the 15 iterations for each of the clients and the server against each given number of subflows and present the results with 95% confidence intervals (except for the results related to the server-side throughputs in Section V-D1, Fig. 3 that are better understood if results are distinctly presented for each multicast group size). In the next subsections, we describe the effect of multicast group size ($|D|$) and number of subflows (m) on the performance metrics of throughput, latency, jitter and packet loss. Second, we present the effect that the external SPTCP traffic has on the performance of MP-ALM scheme. We also analyze the friendly behavior of MPTCP towards SPTCP, when both versions coexist in the MP-ALM framework.

D. Effect of Multicast Group Size

1) Server Side Throughputs: Fig. 3 shows the trend of throughputs at sever (s) for the first six iterations (i.e., upto $|D| = 6$, the subsequent values of $|D|$ show a similar trend). It can be observed that with the increasing $|D|$, the burden on s also increases. This shows one of the downsides of ALM, in which an identical traffic stream is generated over every new MPTCP connection that is required for each newly added client in a multicast group. However, since we are using MPTCP at the transport layer, the bulk of the traffic created by s (because of ALM) can be spread over the multiple subflows of an MPTCP connection. As a result, a single link can be saved from being chocked. For each iteration, the case for $m = 1$ corresponds to one subflow. In other words, it shows that MPTCP is disabled and all the multicast traffic is sent through SPTCP. A significant increase can be observed when the transition from $m = 1$ (when the MPTCP is disabled) to $m = 2$ (when the MPTCP is enabled with two subflows) occurs. After $m = 2$ the improvement in throughput, for a given $|D|$, is not very significant and remains almost at the same level. The reason for this is the *tree-of-trees* phenomenon that we explain in the upcoming section.

2) Client Side Throughputs: As compared to s, the effect of change of $|D|$ on the throughputs of the individual clients (comprising a multicast group) is not very significant. We average the results of all the 15 iterations (i.e., 15 different multicast group sizes) for each of the clients against all the values of m and present our results, as solid bars, in Fig. 4. This figure shows the average throughput observed at client 1 (d_1 as shown in Fig. 2). This is the average of all the iterations with 95% confidence interval. Here we observe that there is generally an increasing trend of throughputs with increasing number of subflows (all the other clients share a similar trend and only the results for d_1 are shown here as a reference).

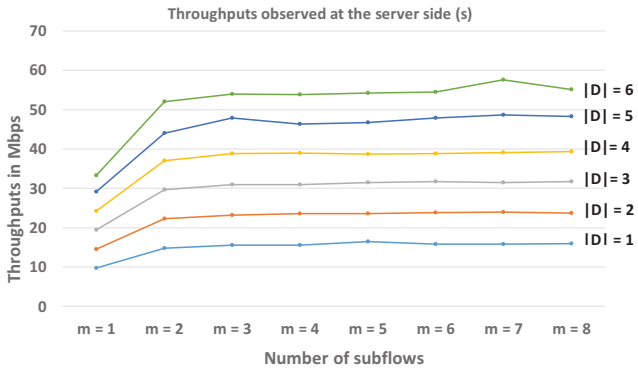

Fig. 3: Throughputs at s for changing $|D|$ and number of subflows

Here we outline one interesting aspect of our experiments that we term as the *tree of trees phenomenon*. Parallel MPTCP connections (each corresponding to an individual client) originate from s (parent node in the main ALM tree) with each connection further creating a large number of subflows (resulting in multiple trees over the same ALM tree). Due to this tree of trees phenomenon, which happens for all the values of m, results in potential collisions among different overlapping MPTCP connections and hence this leads to congestion in the network. This is the reason that we observe a slow increase in the throughputs with increasing values of m. That is why we observe (on average) a little deviant behavior for $m = 8$ (with a huge error bar in Fig. 4), which depicts the occurrence of the maximum number of collisions among different subflows created by different MPTCPs.

In Fig. 4, the most significant increase in throughput can be observed from $m = 1$ to $m = 2$, i.e., when MPTCP is enabled for the first time (this is similar to the results obtained at the server side). After $m = 2$ the increase is relatively small (because of the tree of tree phenomenon as explained in the last paragraph). It can also be observed for $m = 1$ case that the throughput remains almost at the same level (as shown by the negligible height of the error bar). This can be considered as an idealized case, which is being presented here for the comparison purposes. This shows the significant effect of switching from SPTCP ($m = 1$) to MPTCP ($m = 2$).

E. Effect of External SPTCP on MP-ALM Performance

Here we analyze the *friendly* behavior of MPTCP towards SPTCP in our MP-ALM framework. The test scenario to analyse the friendliness of MPTCP towards SPTCP is shown in Fig. 5. As shown in this figure, s now also makes a parallel

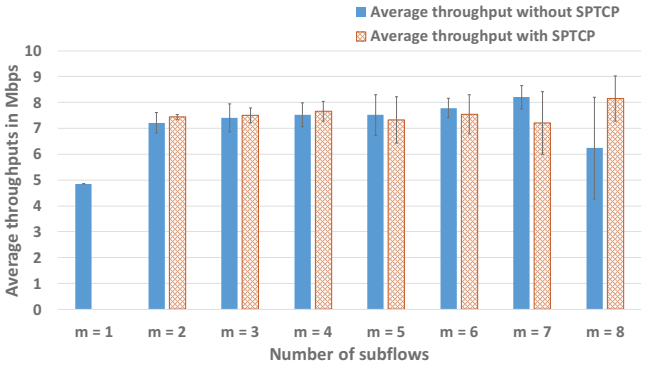

Fig. 4: Comparison of average throughputs observed at d_1, for different number of subflows, when SPTCP is absent and when it is present as external traffic

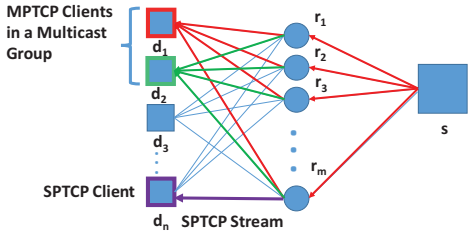

Fig. 5: Topology setup to analyze the friendly behavior of MPTCP towards SPTCP. Client d_n runs SPTCP and coexists with other MPTCP enabled clients in the same network

SPTCP connection with d_n along with its existing MPTCP connections to all the other clients in a multicast group. In this figure (as an example) there are two clients (d_1 and d_2) in a multicast group and there are m number of active subflows created between each of the clients and s. The checkered bars in Fig. 4 summarize the results of throughputs at d_1. This time, besides a multicast group we also have an additional client running SPTCP in our MP-ALM framework. The checkered bars in Fig. 4 show how the trend of throughputs of a client (d_1 in our case), in a multicast group, changes in the presence of an external SPTCP as compared to the case when there is no coexisting SPTCP. An interesting thing to note here is that now we do not observe a deviant behavior for $m = 8$, like we do for the case when there is no external SPTCP. This may be because of the congestion control algorithm design of MPTCP that behaves less aggressively (and hence more friendly [17], [20]) when a SPTCP coexists. Fig. 6 shows the variation in throughputs at d_1 with increasing multicast group size (now averaged over all the values of m against each value of $|D|$). This figure strengthens our argument about MPTCP being less aggressive in the presence of an external coexisting SPTCP. We observe in this figure that for the case when there

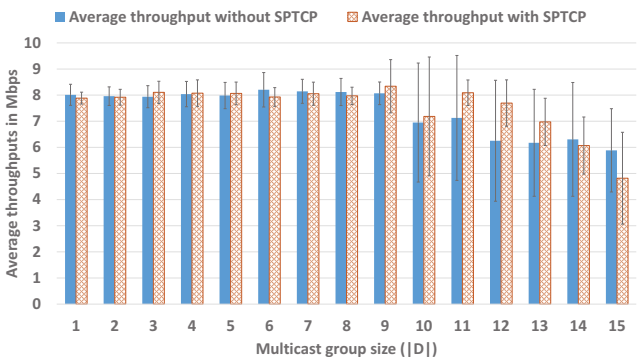

Fig. 6: Comparison of average throughputs at d_1 averaged over all the values of subflows for the case when SPTCP is absent vs. when it is present as external traffic

is no external SPTCP the average throughput starts to plummet after $|D| = 9$ as opposed to the case when SPTCP is present as an external traffic in our topology. The results obtained for latency, jitter and packet loss in the next subsection further strengthen our conjecture.

Furthermore, it is our observation that the throughputs observed at d_n remains constant at 4.848 Mbps against increasing $|D|$ and number of subflows. This figure of 4.848 Mbps is the same when observed for the case of $m = 1$. This result shows that MPTCP adjusts it own throughputs, over its multiple subflows, to ensure maximum *friendliness* towards SPTCP.

F. Latency, Jitter and Packet Loss

Here we present and discuss the latency, jitter and packet loss results from three perspectives namely: as seen from s; those observed at one of the clients (again we take d_1 as our reference); and finally we see how the results change if a client running SPTCP coexists with the multicast group formed in our ALM tree (Fig. 5). These results are important as they provide an insight regarding the quality and user experience of a streaming service. Like in the throughput experiments, we vary our $|D|$ from 1 to 15 each time for a given number of subflows, which we vary from $m = 1$ to $m = 8$. We average all the results over all the values of $|D|$ against each value of m and present them with 95% confidence interval. This results in the average of 15 simulation instances against each given value of m. An important thing to mention here, before proceeding, is that all the results corresponding to the SPTCP client (as shown by the grey checkered bars in Figs. 7-11) start from $m = 2$ instead of $m = 1$. The reason for this is that SPTCP client is only initialized to study the *friendliness* of MPTCP towards SPTCP and for $m = 1$ MPTCP is disabled so this test can not be performed.

1) Server Side: Figures from 7 to 9 present the results for average latencies, jitter and packet loss respectively as observed at s.

Fig. 7 shows that the average latencies for the case when SPTCP is present among the multicast group remain less than

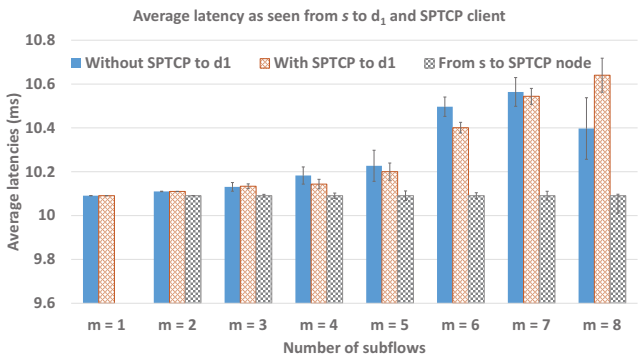

Fig. 7: Effect of change of number of subflows on average latencies as seen from s to d_1 and SPTCP client

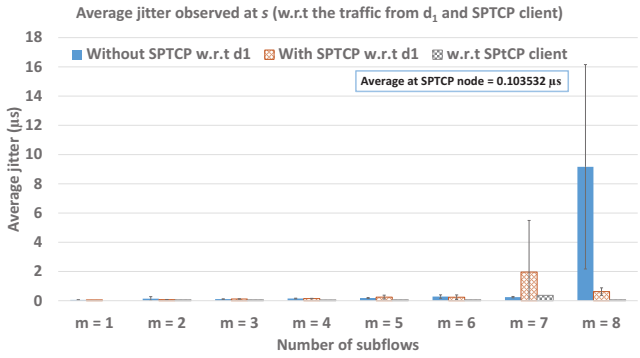

Fig. 8: Effect of change of number of subflows on average jitter as observed at s w.r.t traffic from d_1 and SPTCP client

when an external SPTCP is not present except for case when $m = 8$. This deviant behavior, as evident in throughput (Fig. 4) and packet loss results (Fig. 9), may be caused (as explained before) due to the increased number of collisions between the different subflows of different MPTCPs (the presence of a relatively large error bars on these points in all the results are also indicative of this). Fig. 7 (just like Fig. 6) also gives an insight about the fact that MPTCP is, in its congestion control, less aggressive when a SPTCP is present. This is the reason that the latencies for the increasing number of subflows remain less when SPTCP is present as compared to the case when it is not present. The fact that MPTCP remains friendly towards SPTCP is also evident from the latencies observed from s to the SPTCP client as they remain almost unperturbed (grey checkered bars in Fig. 7).

Fig. 8 presents the corresponding jitter values (i.e., variations in the latencies) observed at s with respect to the traffic received from d_1 (with and without SPTCP) and the SPTCP client. Again, in this figure we see that the response against the SPTCP client remains negligible and almost at the same level (on average at $0.103532\ \mu s$). The sharp increase in jitter when we jump from $m = 7$ to $m = 8$ for the case when SPTCP is absent is related to the sudden decrease in the corresponding value of latency (as described above for Fig. 7).

Fig. 9 summarizes the packet loss that is observed as seen from d_1 for the cases when SPTCP client is present and when it is not present among the multicast group. An increasing trend can be observed in packet loss for increasing number of subflows towards d_1. This result is indicative of an increase in the number of collisions among different MPTCP connections with increasing number of subflows. In this figure the packet loss towards SPTCP client is significantly less as compared to d_1 that shows that MPTCP connections incur minimum toll on the normal operation of a SPTCP.

2) Client Side: Fig. 10 and 11, corresponding to d_1 and SPTCP client show a similar trend for latencies and jitter results as observed at s. In Fig. 11 the sharp rise in jitter for $m = 8$ is related to a sudden decrease in latency (Fig. 10). In Fig. 11, it can be seen that jitter observed at the SPTCP

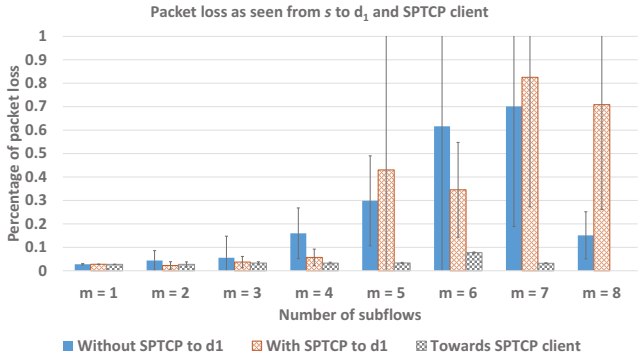

Fig. 9: Effect of change of number of subflows on the percentage of packet loss as seen from s to d_1 and SPTCP client

client with respect to the traffic from s is almost negligible (on average $0.001131\ \mu s$) that shows a good streaming experience of a SPTCP client even in the presence of a large number of clients running MPTCP with large number of subflows. An interesting thing to mention here is that we did not observe any packet loss as seen from d_1 and SPTCP clients to s. All

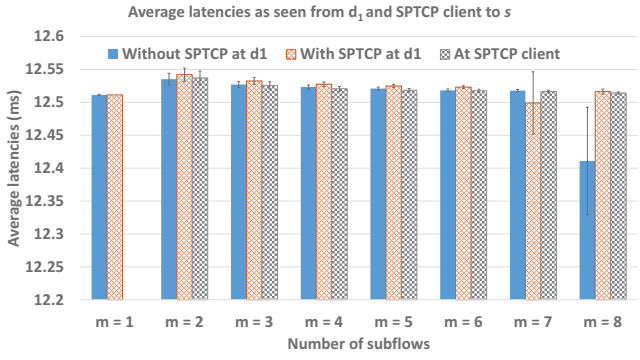

Fig. 10: Effect of change of number of subflows on average latencies as seen from d_1 and SPTCP client to s

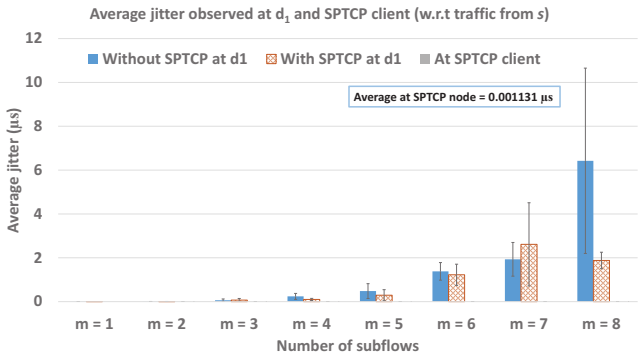

Fig. 11: Effect of change of number of subflows on average jitter observed at d_1 and SPTCP client w.r.t traffic from s

the packets transmitted by d_1 and the SPTCP clients were successfully received at s.

VI. Challenges and Future Directions

One of the main challenges in implementing MPTCP-based multipath multicast in the real world is the need to have multiple IP interfaces. With multihoming one can now make use of multiple IPs and hence can potentially create an equal number of subflows. Currently, however, many hosts only have single network interface card (NIC) with one IP. In order to reap the benefits of MPTCP, multiple NICs (or multiple IPs) are required. Now with the availability of multiport NICs, this issue can be resolved.

Another important issue is the design and implementation of an efficient ALM algorithm that can be adopted widely. To design such an efficient technique one should keep in mind the potential risks of ALM, the most important among these can be the packet duplication and the tree of trees problem (see Section V-D2) at the server side that could put huge amount of load on the network resources and cause multiple MPTCP connections to inflict great amount of interference on each other. *SplitStream* [21] is an interesting work that deals with ALM efficiently and is very close to the approach we adopt in our work. In this work high-bandwidth ALM is considered in a cooperative environment of peer-to-peer networking. The SplitStream system constructs a *forest* of multicast trees with each tree carrying a piece of original information. The interior nodes (or the peers who take part in multicast forwarding) join trees according to their (incoming and outgoing) bandwdith constraints. In this way a load-balanced approach is adopted to distribute multicast content among peers. We are exploring a similar approach in the perspective of MPTCP, which inherently takes care of traffic split at the root of a multicast tree. We will try to design a similar intelligent system, as described in [21], where all the peers will make (incoming and outgoing) MPTCP connections according to their own given network capacities (which can be done by allowing establishment of different number of subflows at each node—which are currently same in our work—for each MPTCP connection that this node creates). The design of such an intelligent ALM technique combined with streaming content coding (such as multiple description coding [22]) can create a robust and efficient multimedia streaming system.

VII. Conclusion

In this study we investigate the performance of multipath TCP (MPTCP) in multipath application-layer multicast streaming (MP-ALM). Our results show that MPTCP improves the throughput performance of individual clients in a multicast group as compared to the case when only single-path TCP (SPTCP) is deployed. All the results (throughput, latency, jitter and packet loss) show that MPTCP remains very friendly towards SPTCP. At the server side we observe that, because of ALM, individual and identical streams for each client in a multicast group are created. This duplication (or redundancy) can overburden the network resources. In our MP-ALM scheme, however, MPTCP creates multiple subflows that spread the multicast traffic of the streaming server over a larger set of network resources so a fixed set of network elements can be saved from being overburdened. We are affirmative that with the proliferation of multiport network interface cards (NICs) and efficient ALM algorithms, the MP-ALM approach can significantly improve the streaming experience of users with minimum effects on existing SPTCP streaming traffic and lesser congestion in the overall network.

References

[1] "The Global Internet Phenomena Report: North America and Latin America," Sandvine Incorporated ULC, Tech. Rep., 05 2015.

[2] B. Cohen, "Bittorrent protocol specification v1. 0," *WWW, June*, 2002.

[3] C. Raiciu, C. Pluntke, S. Barre, A. Greenhalgh, D. Wischik, and M. Handley, "Data center networking with multipath TCP," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*. ACM, 2010, p. 10.

[4] I. Van Beijnum, J. Crowcroft, F. Valera, and M. Bagnulo, "Loop-freeness in multipath BGP through propagating the longest path," in *Communications Workshops, 2009. ICC Workshops 2009. IEEE International Conference on*. IEEE, 2009, pp. 1–6.

[5] J. Qadir, A. Ali, Y. Kok-Lim, A. Sathiaseelan, and J. Crowcroft, "Exploiting the power of multiplicity: a holistic survey of network-layer multipath," *Communications Surveys Tutorials, IEEE*, vol. PP, no. 99, pp. 1–1, 2015.

[6] A. Lozano and N. Jindal, "Transmit diversity vs. spatial multiplexing in modern MIMO systems," *Wireless Communications, IEEE Transactions on*, vol. 9, no. 1, pp. 186–197, 2010.

[7] F. Wang, Y. Xiong, and J. Liu, "mtreebone: A collaborative tree-mesh overlay network for multicast video streaming," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 21, no. 3, pp. 379–392, 2010.

[8] L. Fan, P. Trinder, and H. Taylor, "Design issues for peer-to-peer massively multiplayer online games," *International Journal of Advanced Media and Communication*, vol. 4, no. 2, pp. 108–125, 2010.

[9] C.-H. Wang, Y.-H. Chu, and T.-T. Wei, "SIPTVMON: a secure multicast overlay network for load-balancing and stable IPTV service using SIP," in *Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on*. IEEE, 2011, pp. 97–102.

[10] S. Tullimas, T. Nguyen, R. Edgecomb, and S.-c. Cheung, "Multimedia streaming using multiple tcp connections," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 4, no. 2, p. 12, 2008.

[11] B. Wang, W. Wei, Z. Guo, and D. Towsley, "Multipath live streaming via tcp: scheme, performance and benefits," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 5, no. 3, p. 25, 2009.

[12] R. Kuschnig, I. Kofler, and H. Hellwagner, "Improving internet video streaming performance by parallel TCP-based request-response streams," in *Consumer Communications and Networking Conference (CCNC), 2010 7th IEEE*. IEEE, 2010, pp. 1–5.

[13] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, "Tcp extensions for multipath operation with multiple addresses," Tech. Rep., 2013.

[14] C. Raiciu, C. Paasch, S. Barre, A. Ford, M. Honda, F. Duchene, O. Bonaventure, M. Handley *et al.*, "How hard can it be? designing and implementing a deployable multipath TCP." in *NSDI*, vol. 12, 2012, pp. 29–29.

[15] A. Rao, A. Legout, Y.-s. Lim, D. Towsley, C. Barakat, and W. Dabbous, "Network characteristics of video streaming traffic," in *Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies*. ACM, 2011, p. 25.

[16] A. Ali, J. Qadir, and A. Baig, "Learning automata based multipath multicasting in cognitive radio networks," *Communications and Networks, Journal of*, vol. 17, no. 4, pp. 406–418, 2015.

[17] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, "Design, implementation and evaluation of congestion control for multipath TCP," in *Proceedings of the 8th USENIX conference on Networked systems design and implementation*. USENIX Association, 2011, pp. 8–8.

[18] H. Tazaki, F. Uarbani, E. Mancini, M. Lacage, D. Câmara, T. Turletti, and W. Dabbous, "Direct code execution: Revisiting library os architecture for reproducible network experiments," in *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*. ACM, 2013, pp. 217–228.

[19] "Code: dce-iperf-mptcp.cc," {https://www.nsnam.org/~thehajime/tmp/bake/source/ns-3-dce/example/dce-iperf-mptcp.cc}, [Online; accessed 22-October-2016].

[20] Q. Peng, A. Walid, and S. H. Low, "Multipath TCP algorithms: theory and design," in *Proceedings of the ACM SIGMETRICS/international conference on Measurement and modeling of computer systems*. ACM, 2013, pp. 305–316.

[21] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Splitstream: high-bandwidth multicast in cooperative environments," in *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5. ACM, 2003, pp. 298–313.

[22] V. K. Goyal, "Multiple description coding: Compression meets the network," *Signal Processing Magazine, IEEE*, vol. 18, no. 5, pp. 74–93, 2001.