Demo: DEMS: DEcoupled Multipath Scheduler for Accelerating Multipath Transport

Yihua Ethan Guo, Ashkan Nikravesh, Z. Morley Mao, Feng Qian[†], Subhabrata Sen[‡]

University of Michigan [†]Indiana University [‡]AT&T Labs – Research
{yhguo, ashnik, zmao}@umich.edu fengqian@indiana.edu sen@research.att.com

ABSTRACT

We present the demonstration of DEMS, a new multipath scheduler aiming at reducing the data chunk download time. DEMS consists of three key design decisions: (1) being aware of the chunk boundary and strategically decoupling the paths for chunk delivery, (2) ensuring simultaneous subflow completion at the receiver side, and (3) allowing a path to trade a small amount of redundant data for performance. We integrate the DEMS components into a holistic system and implement it on commodity mobile devices, where unmodified mobile applications can use DEMS to transmit data over multipath. We demonstrate the simple configuration of using DEMS over multipath, visualization of multipath scheduling, download time reduction of data chunks with DEMS over both emulated and real cellular/WiFi networks compared to default MinRTT scheduler, and application QoE improvement on mobile phones from DEMS.

1 INTRODUCTION

Simultaneously using multiple network paths such as cellular and WiFi to accelerate data transfer is an attractive feature on mobile devices. It is supported by many commercial products such as Apple Siri [3], Gigapath by Korean Telecom [1], and Samsung Download Booster [2]. Currently the most widely used multipath solution is MPTCP [4], which enables unmodified applications to leverage multipath by adding a shim layer to the TCP interface. MPTCP establishes a *subflow* over each network path. The MPTCP sender distributes the data onto the subflows; the receiver reassembles the data into the original byte stream and delivers it to the app transparently.

Multipath *scheduler* is an important component in multipath transport, which determines how the data is distributed onto the subflows. MPTCP supports different types of schedulers. For example, the *MinRTT* scheduler attempts to deliver the data as soon as possible by choosing a subflow with the smallest RTT unless its congestion window is full. Despite existing efforts on improving multipath scheduler [5, 7–11], we found that the multipath scheduler design is far from being optimal. In a pilot experiment, we observe that surprisingly, under representative WiFi/LTE network conditions, the MinRTT scheduler inflates the download time for a medium-sized file by up to 33% compared to the optimal scheduling decision derived offline. In real-world networks with fluctuating bandwidth or latency, MinRTT may perform even worse (up to 7.5x download time increase, 49% median increase compared to optimal scheduling). Regarding the root cause, our key insight is that for such a file download, oftentimes the subflows do *not* complete at the same time at the *receiver* side, leading to suboptimal performance.

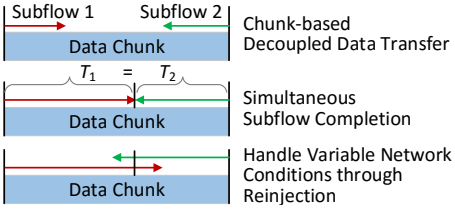

Figure 1: Key design decisions of DEMS.

This demonstration shows DEMS (**DE**coupled **M**ultipath **S**cheduler[1]) [6], a new multipath packet scheduler aiming at reducing the data chunk download time over multiple paths. A *data chunk* is simply a block of application-defined bytes, which is a very common data transfer workload in mobile applications, *e.g.,* fetching an image, JavaScript, MP3 file, or video chunk. The key idea behind DEMS is to *achieve simultaneous subflow completion at the receiver side through strategic packet scheduling over decoupled subflows* in order to minimize the chunk download time. To accomplish this, DEMS incorporates three design components: (1) chunk-based data transfer, (2) simultaneous subflow completion, and (3) dynamic reinjection. We demonstrate that DEMS is easy to use for unmodified applications, beneficial to both chunk downloads and real applications including web browsing.

2 DESIGN OVERVIEW

As shown in Figure 1, the key design decisions of DEMS include the following: (1) DEMS leverages a heuristic that treats all data in the meta buffer as a chunk, and it strategically decouples the paths for chunk delivery (§2.1). (2) DEMS ensures simultaneous subflow completion at the receiver side (§2.2). (3) DEMS allows a path to trade a small amount of redundant data for performance (§2.3). We further elaborate on how DEMS is integrated into a mobile multipath system (§2.4).

[1]Note here "decoupled scheduling" is different from the decoupled congestion control in MPTCP.

MobiCom '17, October 16–20, 2017, Snowbird, UT, USA

ACM ISBN 978-1-4503-4916-1/17/10.
https://doi.org/10.1145/3117811.3119869

2.1 Chunk-based Data Transfer

In DEMS, by default, data is delivered to the application on a *per-chunk* basis. A (data) chunk consists of a block of bytes defined by the application, which can be, for example, an image, a Javascript, an audio snippet, or a video chunk. As long as a chunk can be correctly reassembled at the transport layer, bytes within the chunk can be delivered in any order. The data chunk is thus split into different parts that are distributed onto different paths for delivery. For the common scenario involving two paths, we design a "two-way" splitting approach: the two paths transfer the data in opposite directions, one from the beginning and the other from the end; when they "meet" each other, the chunk is fully downloaded. This approach is intuitive and parameterless. Furthermore, it helps improve the multipath performance by *decoupling* the two subflows. Each subflow freely and independently transfers the data until the very end when subflows meet and merge.

2.2 Simultaneous Subflow Completion

Having all subflows complete at the same time at the receiver side is a necessary condition for achieving the optimal performance. The reason can be easily shown through *proof by contradiction*: suppose in an optimal scheme, Subflow A finishes earlier than Subflow B; in that case Subflow B can further "offload" some bytes to Subflow A, leading to an even shorter download time. To achieve simultaneous subflow completion, the high-level idea is to introduce a timing offset at the sender to compensate the heterogeneous delay across both subflows. DEMS uses bandwidth and delay prediction to calculate the timing offset and dynamically decides the network path to transmit each packet.

2.3 Handling Variable Network Conditions

DEMS tolerates unbalanced completion of subflows under variable network condition by performing *reinjection*: instead of having a full stop when all bytes of a chunk are transmitted, a subflow may further "overshoot" its portion by sending a small number of bytes that are beyond the meeting point of the two subflows. These bytes are redundant because they have already been transmitted over the other subflow. The purpose of reinjection is to trade redundant data for better performance: under uncertain network conditions, if the reinjected (redundant) data arrives earlier than its original copy, the overall download time is reduced. In DEMS, reinjection only occurs near subflows' meeting point. We develop a method that adaptively determines the amount of the redundant data to strike a sweet spot between the performance and the additional data transmission due to reinjection.

2.4 System Design

We now elaborate on how to integrate the DEMS algorithm into a real system. Figure 2 plots the system diagram. At the sender side (right), the chunk data coming from the application is stored in the meta buffer, and is then split, scheduled, and transmitted by the packet scheduler. Working with the packet scheduler, the reinjection manager keeps track of packets' transmission states and makes decisions on adaptive reinjection. We also design a module for measuring and predicting network conditions (One-way delay difference between two subflows, i.e. ΔOWD, and bandwidth).

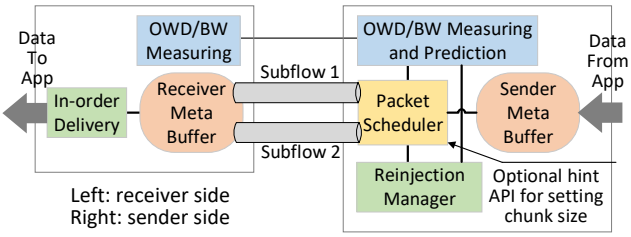

Figure 2: System diagram of DEMS.

The application can also optionally specify the chunk size through an API. The receiver side logic is much simpler. It passively receives/acknowledges the data, reassembles it in the receiver-side meta buffer, and delivers the in-order data to the application. Note that data over the (decoupled) subflows are acknowledged separately using the per-subflow ACK numbers. Meanwhile, similar to MPTCP, at the receiver meta buffer, the global sequence number carried by each packet is used to mark which portion within the chunk has been received. While Figure 2 illustrates one-way data transfer, our system supports full-duplex data transmission.

3 DEMONSTRATION

The demonstration consists of two Android phones and two Linux laptops. We use both the Android phones and Linux laptops to demonstrate the capability of running DEMS scheduler over multipath. The phones require access to both the cellular network (3G or 4G/LTE) and WiFi network. Both laptops require access to a WiFi network and power. One of the laptop serves as a multipath proxy for mobile devices and visualizes the multipath scheduling of both MinRTT and DEMS. We need one table to setup all equipments with an estimated time of 1 hour. We use these devices to demonstrate the following:

Simple configuration of using DEMS. We show how DEMS can be easily used by unmodified applications on commodity mobile devices. We leverage a customized multipath TCP proxy infrastructure. Between the proxy and the mobile device, multipath is realized as multiple conventional TCP connections each corresponding to a subflow established over a network path. For download traffic, the proxy makes the corresponding multipath scheduling decision for each data packet. A policy file on the mobile device configures the scheduler to be used for each application. We demonstrate that by using this proxy infrastructure and easily modifying the configuration file, DEMS can be enabled over multipath for unmodified mobile applications.

Visualization of multipath scheduling. This shows how different schedulers including MinRTT and DEMS makes scheduling decision for each packet over time, to better understand the design of multipath schedulers. We show that oftentimes the subflows in MinRTT do not complete at the same time at the receiver side, achieving sub-optimal download time. Instead, DEMS minimizes the data transfer time by achieving simultaneous subflow completion.

Improvement of data chunk download. This shows the performance improvement of DEMS on downloading data chunks over

both emulated and real multipath environments of WiFi and cellular networks. We take a "record and replay" approach to realistically emulate the varying bandwidth. We collect multiple 5-minute bandwidth traces of both paths from different locations. In the demonstration, we compare the performance of DEMS and MinRTT under the same network conditions, by replaying the bandwidth traces or using real networks.

Application QoE benefits. This shows how DEMS helps improve the QoE of web browsing, one of the most popular applications on mobile devices. We use web page loading experiments using off-the-shelf Chrome browser (version 53.0.2785.124) on our Android phones. We picked 10 popular websites and use their mobile-version landing pages as the target web pages. We use the page load time (PLT), which is programmatically measured by the Chrome debugging interface, as the QoE metric. We demonstrate that PLT is improved with DEMS over multipath compared to default MinRTT scheduler.

REFERENCES

[1] 2015. In Korean, Multipath TCP is pronounced GIGA Path. http://blog.multipath-tcp.org/blog/html/2015/07/24/korea.html. (2015).

[2] 2016. Samsung Download Booster. http://www.samsung.com/uk/support/skp/faq/1061358. (2016).

[3] 2017. iOS: Multipath TCP Support in iOS 7. https://support.apple.com/en-us/HT201373. (2017).

[4] Alan Ford, Costin Raiciu, Mark Handley, and Olivier Bonaventure. 2013. TCP Extensions for Multipath Operation with Multiple Addresses. RFC 6824. (2013).

[5] Alexander Frömmgen, Tobias Erbshauser, Alejandro P. Buchmann, Torsten Zimmermann, and Klaus Wehrle. 2016. ReMP TCP: Low Latency Multipath TCP. In *IEEE ICC 2016*.

[6] Yihua Ethan Guo, Ashkan Nikravesh, Z. Morley Mao, Feng Qian, and Subhabrata Sen. 2017. Accelerating Multipath Transport Through Balanced Subflow Completion. In *ACM MobiCom 2017*.

[7] Bo Han, Feng Qian, Lusheng Ji, and Vijay Gopalakrishnan. 2016. MP-DASH: Adaptive Video Streaming Over Preference-Aware Multipath. In *ACM CoNEXT 2016*.

[8] Nicolas Kuhn, Emmanuel Lochin, Ahlem Mifdaoui, Golam Sarwar, Olivier Mehani, and Roksana Boreli. 2014. DAPS: Intelligent Delay-aware Packet Scheduling for Multipath Transport. In *IEEE ICC 2014*.

[9] Yeon-sup Lim, Erich M Nahum, Don Towsley, and Richard J Gibbens. 2017. ECF: An MPTCP Path Scheduler to Manage Heterogeneous Paths. In *ACM SIGMETRICS 2017 Abstracts*.

[10] Christoph Paasch, Simone Ferlin, Ozgu Alay, and Olivier Bonaventure. 2014. Experimental Evaluation of Multipath TCP Schedulers. In *ACM SIGCOMM Capacity Sharing Workshop (CSWS) 2014*.

[11] Yeon sup Lim, Yung-Chih Chen, Erich M. Nahum, Don Towsley, Richard J. Gibbens, and Emmanuel Cecchet. 2015. Design, Implementation and Evaluation of Energy-Aware Multi-Path TCP. In *ACM CoNEXT 2015*.