An Efficient Scheduling Scheme of Multipath TCP for MPI

Popat Khushi J., Jigar A. Raval, Samuel Johnson, Bhavesh Patel

Abstract— **Multipath TCP is adopted by IETF to support use of multiple available paths for data transfer over single connection that was not supported by TCP. It is extension of TCP with multi-homing feature which tries to solve problem of utilizing multiple available NICs for providing higher resiliency and/or higher bandwidth. As multipath TCP uses more than one path for data transfer, an efficient multipath scheduler is required at sender side to distribute data efficiently over different paths in order to utilize network resources efficiently and to gain high performance. The main objective of scheduler is to utilize all available paths and select the best path over which data packet should be sent.**

In this paper we observe existing schedulers and its functionality. We focus on default scheduler provided with Linux kernel implementation and cover problems associated with it. Here we propose a scheduler to obtain higher performance in MPI applications on HPC environment. We have designed this algorithm for MPI application running on MPI cluster. Our analysis shows that this algorithm will improve the performance with MPTCP compared to the default scheduler.

Index Terms — **TCP, Scheduler, Performance, Default Scheduler, Round-Robin Scheduler**

I. INTRODUCTION

Today nearly all computing devices have multiple network interfaces: mobile phones with Wi-Fi and 3G, laptops with Wi-Fi and Ethernet and servers too are equipped with one or more interfaces. TCP is the most common protocol being used today for Internet application but it does not support multi-homing. To utilize all available paths between communicating entities multipath transport protocol is required. The Internet Engineering Task Force (IETF) has adopted Multipath TCP as an extended form of TCP with the support of multi-homing feature.

MPTCP [1, 2] is multipath byte stream protocol which provides reliable and ordered delivery of messages (same as TCP) but it also supports use of all available paths for transferring data among communicating hosts over single connection. MPTCP supports use of multiple IP addresses per one pair of hosts. MPTCP is backwards compatible with applications that are only TCP aware, but has unique features such as traffic routing, network addressing, and connection direction. Today it is being mostly used in datacenters and in wireless networks with the goal of utilization of all available network resources and providing high performance [12, 13, 14]. Linux kernel implementation of MPTCP (on 2.6.x and above) is also available for use in testing, research and in production. Apple iPhones and iPods also have support of MPTCP [9].

One common thing with multipath transport protocols is that sender needs to decide how to distribute data over all available paths. MPTCP congestion control scheme [5] restrict use of congested paths for data transfer but do not schedule data over efficient path. MPTCP congestion control schemes do not fulfill the need of selection of efficient path for data transfer, for that a scheduler is required. Scheduler decides how to distribute data efficiently over multiple available paths. It is what selects the best path among all available paths based on different path characteristics and sends data over them. Scheduler must be capable enough to deal with heterogeneous and dynamically changing characteristics of paths as well as corresponding congestion situation of them.

MPTCP is adopted with the goals of improving performance by using multiple disjoint paths simultaneously for data transfer on single connection, for better utilization of network resources and to improve network capacity. Here MPTCP scheduler needs to deal with paths with various network constraints. So the scheduler, which distributes packets over dissimilar paths, is an important design constraint for efficient performance of MPTCP. The scheduler for MPTCP must take care about different path characteristics such as RTT, delays, buffer size, etc.

In the world of High Performance Computing Clusters [4], MPI [3] (Message Passing Interface) is the de facto standard for achieving communication between various nodes present in the cluster. The goal of MPI is to provide an effective and portable environment where developers can easily build message passing programs for HPC. MPI supports various transport protocols over Ethernet (e.g., TCP, iWARP, UDP, raw Ethernet frames, etc.), shared memory and InfiniBand. The way most HPC are implemented today, they have a primary high bandwidth and low latency network for data communication (usually Infiniband) and a secondary network for management, backup, or fall back purposes. In most cases, it is the network that causes bottleneck for the jobs running on HPC, as a result, networking technologies used in HPCs are constantly evolving and are so advanced, that they are almost exclusively only used in HPC environment.

In this paper, we study default MPTCP scheduler which is part of the Linux Kernel Implementation of MPTCP and address problems and limitations associated with it. To overcome problems associated with default scheduler, we proposed a scheduler based on queue length at subflows and

Manuscript received June, 2015.

Popat Khushi J., *M.E student Computer Engineering (IT Systems and Network Security), SVIT, Vasad, Gujarat, India.*

Jigar A. Raval, *I/C Computer Center, Physical Research Laboratory, Ahmedabad, India.*

Samuel Johnson, *Scientist / Engineer - SC, Physical Research Laboratory, Ahmedabad, India.*

Bhavesh Patel, *Assistant Professor, SVIT, Vasad, Gujarat, India.*

RTT of subflows. The scheduler is designed by keeping in mind about HPC cluster environment and its network requirements. Here we have covered expected outcomes of this proposed algorithm with MPI application. Our analysis shows that it will improve the performance of MPTCP with MPI with compared to default scheduler of MPTCP Linux kernel.

II. BACKGROUND STUDY

A. *Multipath TCP*

MPTCP [1, 2] is an extension of TCP which supports use of multiple available paths for data transfer as well as provides higher resiliency in comparison of TCP. It provides reliable, connection oriented data transfer same as TCP. It follows three-way handshake process same as TCP for connection establishment with MP_CAPABLE option with each packet. Further addition of subflow on this connection can be done by three way handshake process with MP_JOIN option with TCP connection establishment packets. MPTCP supports two kind of data sequence numbering. One 64-bit data sequence number at connection level and another 32-bit data sequence number at subflow level. MPTCP uses coupled congestion control scheme for congestion control which work similar to TCP congestion control but increment and decrement in size of window will be different.

MPTCP need scheduling which was not needed in TCP as it does not support multiple paths for single connection. Next section covers scheduling and its needs as well as different scheduling technique.

B. *Multipath TCP Scheduling*

MPTCP [1, 2] is multipath transport protocol which transmits data using multiple available paths. MPTCP congestion control scheme restricts use of congested paths for data transfer but do not schedule data over efficient path. MPTCP congestion control schemes do not fulfill the needs of selection of efficient path for data transfer, for that scheduler is required. Schedulers decide how to distribute data over multiple available paths. It selects the best path among available paths based on different path characteristics and sends data over the path.

Whenever an MPTCP sender wants to transmit data, the sender desires to make 3 decisions [7].

- First which subflow(s) are available to send data? This decision is made by the MPTCP congestion control scheme which preserves a per-subflow cwnd. The subflows with available cwnd are available for transmitting data.
- Second, if several subflows have available cwnd, a scheduler selects subflow among them to send data.
- Third, after selecting a subflow, the scheduler resolves how much data should be sent on that subflow. The third decision concerns the granularity of the allocation.

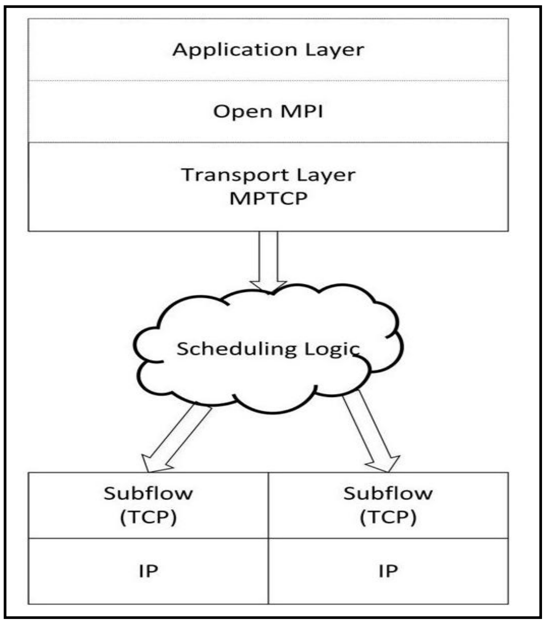

Figure 1 Place of Scheduling locgic in MPTCP stack

Above Figure 1 shows architecture of Multipath TCP with MPI and role of scheduler. It shows that scheduler distributes data at MPTCP level on different TCP subflows. Scheduler has access of all subflows' congestion control attributes, RTT estimation etc.

As the scheduler is responsible for path selection for distribution of data among them, it might be possible that by selection of wrong scheduler may lead to degradation of performance instead of higher performance. Wrong scheduling choices might initiate head-of-line blocking or receive-window limitation, especially when paths are heterogeneous. In such a scenario, the user will scrutinize high delays and lower goodput for its application, resulting in poor user experience. Thus, the schedulers have a considerable impact on the performance of Multipath TCP [6].

C. *Existing Scheduler*

Linux Kernel implementation of MPTCP [9] provides choice between two different schedulers. A Default scheduler and a Round-Robin scheduler.

1) *Default Scheduler*

In the Linux kernel implementation of Multipath TCP, the default scheduler always chooses the subflow with the smallest round-trip-time to send data [8]. Default scheduler checks for RTT of each path and selects path with smallest RTT until congestion window is available after that it will pass data on path with second smallest RTT value. Sending data over the subflow with the smallest round-trip-time is not adequate to attain good performance on memory constrained devices that make use of a small receiving window [9].

It only considers RTT to select path while it is necessary to take care about congestion on path, receiver buffer size, no of packets waiting at subflow level etc to efficiently schedule data in each time.

2) *Round-Robin Scheduler*

Round-Robin Scheduler schedules data over all available subflows in round-robin fashion without taking care about any of path characteristics. As a result it will perform poor in some situations and do not take advantage of heterogeneous network and path characteristics [9]. However, in case of bulk data transmission, the scheduling is not really round-robin, since the application is able to fill the congestion window of all subflows and then packets are scheduled as soon as space is again available in each subflow's congestion window.

When paths have significant delay difference round robin will not efficiently utilize paths because it always tries to equally distribute data among all available paths.

3) Other Existing Solutions

One Delay aware packet scheduling approach is proposed in "DAPS: Intelligent Delay-Aware Packet Scheduling For Multipath Transport [10]". This scheduler aims to reduce the receiver's buffer blocking time taken as a main parameter to improve the Quality of Service (QoS) in wireless environments. In this paper [10], authors developed a model for maximum blocking time because of multipath delay imbalance i.e. to decrease amount of time for which packets need to wait in receiver's buffer for ordered delivery and validated this model with ns-2 simulations. Authors found an approach to match the paths asymmetry and avoid buffer blocking. Authors have implemented and evaluated DAPS in ns-2, and more adapted the scheduling of CMT-SCTP that selects packets to be conveyed over each paths based on their RTT values, in order to support ordered reception. This scheduler takes two paths with different delays and assigns list of sequence numbers to paths on which packets to be transmitted or sent. It is implemented for CMT-SCTP protocol not for MPTCP. For this scheduler authors have considered that both paths have large delay difference as well as cwnd value is stable which is not true for all situations [8].

Another scheduler is proposed in "A Scheduler for Multipath TCP [7]" which tries to approximate the available capacity on each subflow and calculate the number of bytes transmitted over each subflow. This facilitates the scheduler to identify when the subflow is sending too much data and select the other subflow at that time. The scheduler proposed in this paper is implemented in the Linux kernel, but the source code does not seem to have been released by the authors. The performance of the scheduler is evaluated by considering a simulation scenario with very long file transfers in a network with a very small size buffer. It does not represent a real use case for Multipath TCP.

III. Proposed Scheduling Scheme

Here we proposed a scheduling scheme which is based on default scheduler of MPTCP provided with Linux kernel implementation of MPTCP. Below Figure 2 shows us flow chart of proposed scheduler. As shown in figure scheduler go through each available subflow and chooses the best subflow from them based on the availability, congestion situation and RTT. Here scheduler goes through each subflow and chooses the best path among them.

First it checks whether packet comes for retransmission or comes for first time. If packet comes for retransmission and it was earlier transmitted on same subflow then this path will not be selected for same packet and scheduler checks availability of another subflow. If it is first time transmission or not before transmitted on same subflow then another condition will be checked. Next scheduler checks the congestion situation if number of transmitted packets whose acknowledgments yet not received or are on the way are more than the size of congestion window then that path will not be selected because it is already have more packets. Next if congestion window is available and it is more than in flight packets then it will check for the space in its buffer. If number of packets waiting for transmission on that path are more than capacity or space then this path will be rejected. After that next condition will be checked if space is available at sending buffer. Now scheduler will check how many packets are waiting in queue at particular path for transmission and calculate the total processing time. Here we have taken one variable min_proc_time_to_peer which is initialized with maximum value. For the first path if its processing time is less than this variable's value then this path will be selected. If two paths will have same processing time than based on RTT value packet will be transmitted on the subflow with lower RTT value.

This whole process will be repeated for all available subflow and best path will be selected from them.

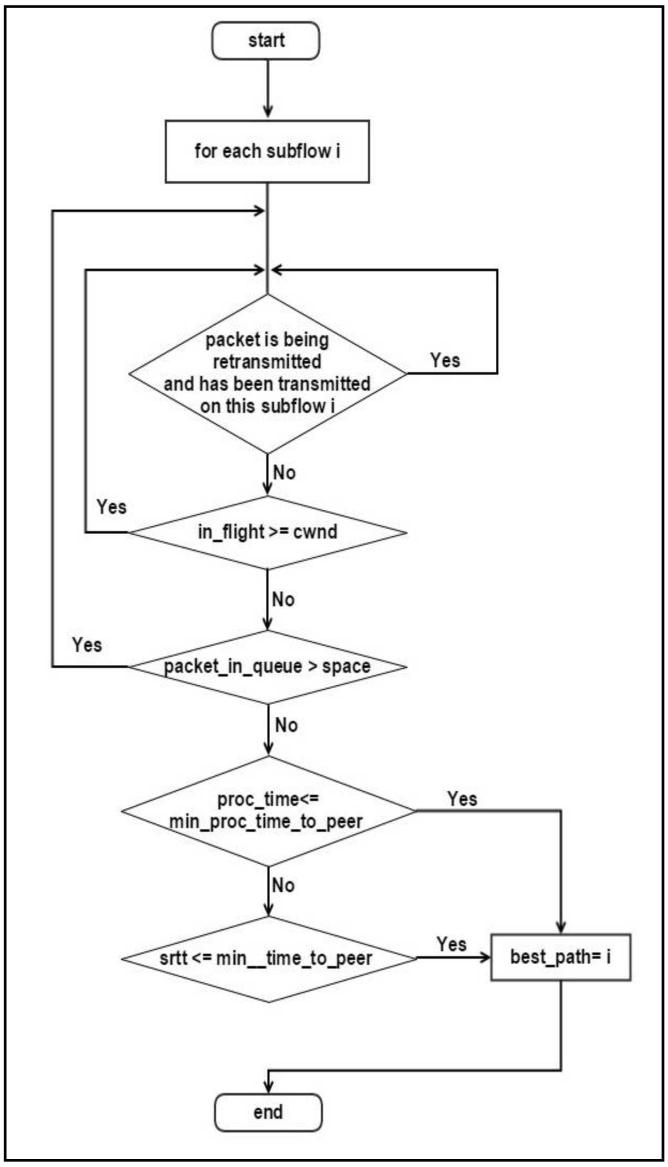
start
for each subflow i
packet is being retransmitted and has been transmitted on this subflow i
Yes
No
Yes
in_flight >= cwnd
No
Yes
packet_in_queue > space
No
proc_time<= min_proc_time_to_peer
Yes
No
srtt <= min__time_to_peer
Yes
best_path= i
end

Figure 2 Proposed Scheduling Flow chart

IV. Expected Outcomes of Proposed Work

We have designed this algorithm by keeping in mind to increase performance of MPTCP with different path characteristics and our theoretical analysis shows that our work will improve the performance of MPTCP with MPI for data transfer.

V. Conclusion

In this paper, we proposed a scheduler which is based on packets waiting at subflow level (queue size), RTT value and congestion situation of subflow. This scheduler is designed by taking care about requirements of MPI applications running on HPC cluster. The performance of this scheduler is measured by running file transfer application of MPI which transfers file from master to compute node and its performance is compared with performance of default scheduler provided with Linux Kernel of MPTCP.

From this we can conclude that default scheduler is not efficient in all situations such as subflows with different delay, buffer size and path capacity. Therefore an efficient scheduler is required which not only based on RTT but also consider other network parameters for path selection. As per our assumption and analysis, our proposed work performs better than default scheduler and considers congestion situation, RTT and packets waiting at subflow for path selection.

Acknowledgment

We would like to thank Physical Research Laboratory (PRL), Navarangpura, Ahmedabad and members of Computer centre of PRL for providing us resources as well as support for carrying out these experiments.

REFERENCES

[1] Ford, A., Raiciu, C., Handley, M. and Bonaventure, O. TCP extensions for multipath operation with multiple addresses. RFC6824, 2014; http://www.rfceditor.org/rfc/rfc6824.txt.

[2] Sebastiean Barre, Christoph Passch, Olivier Bonaventure - "Multipath TCP: From Theory To Practice," in Proceedings of the 10th International IFIP Networking Conference, Valencia/Spain, May 2011, pp. 444–457, ISBN 978-3-642-20756-3

[3] Blaise Barney, Lawrence Livermore National Laboratory, "Message Passing Interface(MPI)", https://computing.llnl.gov/tutorials/mpi/

[4] HPC Cluster networks, High Performance Computing –WIKI [online] http://en.community.dell.com/techcenter/high-performance-computing/w/wiki/hpc-cluster-networks

[5] Raiciu C., Handly M., Wischik M.. RFC 6356: Coupled Congestion Control for Multipath Transport Protocols. RFC 6356 (October 2011)

[6] Behnaz Arzani, Alexander Gurney, Shuotian Cheng, Roch Guerin and Boon Thau Loo –"Impact of Path Characteristics and Scheduling Policies on MPTCP Performance," 2014 28th International Conference on Advanced Information Networking and Applications Workshops

[7] Fan Yang, Paul Amer, Nasif Ekiz – "A Scheduler for Multipath TCP," IEEE 2013

[8] Bonaventure Olivier - "Why is the Multipath TCP scheduler so important" [online] http://blog.multipathtcp.org/blog/html/2014/03/30/why_is_the_multipath_tcp_scheduler_so_important.html

[9] "Multipath TCP-Linux Kernel implementation" [online] http://www.multipath-tcp.org/

[10] Nicolas Kuhn, E. Lochin, A. Mifdaoui, G. Sarwar, O. Mehani, and R. Boreli, "DAPS: Intelligent Delay-Aware Packet Scheduling For Multipath Transport," presented at the ICCC, 2014

[11] Amanpreet Singh, Carmelita Goerg,Andreas Timm-Giel, Michael Scharf –"Performance Comparison of Scheduling Algorithms for Multipath Transfer," Globecom 2012 - Next Generation Networking and Internet Symposium, IEEE

[12] Raiciu Costin, Pluntke Christopher, Barre Sebastien, Greenhalgh Adam, Wischik Damon, Handley Mark. Data Center Networking with Multipath TCP. Hotnets-10 Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Network, October 2010

[13] Raiciu C., Barre S., Pluntke C., Greenhalgh A.,Wischik D. and Handley M. Improving datacenter performance and robustness with Multipath TCP. ACM SIGCOMM Computer Communication Review 41, 4 (2011), 266–277.

[14] Yung-Chih Chen, Yeon-sup Lim, Richard J. Gibbens, Erich M. Nahum, Ramin Khalili, Don Towsley. A Measurement-based Study of Multipath TCP Performance over Wireless Networks. IMC'13 (October 23–25, 2013), Barcelona, Spain.