MPTCP Working Group
Internet-Draft
Intended status: Informational
Expires: August 02,2018

Samar Shailendra
Hemant Kumar Rath
Arpan Pal
TCS R&I
Abhijit Mondal
IIT Kharagpur

January 29, 2018

Extended Socket APIs to control subflow priority in Multipath TCP
draft-samar-mptcp-socketapi-00.txt

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

This Internet-Draft will expire on July 25, 2018.

Copyright Notice

Abstract

This document provides the extended Socket APIs to control subflow priority for Multipath TCP. It also describes an additional data structure for MPTCP to make the subflow priority persistent across subflow disconnection.

Table of Contents

1. Introduction.....2
2. Socket APIs for subflow Priority.....3
2.1. Control Subflow Priority.....4
2.2. Remembering Subflow Priority.....4
3. Security Considerations.....5
4. IANA Considerations.....5
5. Conclusions.....5
6. References.....6
6.1. Normative References.....6
6.2. Informative References.....6
7. Authors' Addresses.....7

1. Introduction

Multipath TCP (MPTCP) [RFC6824, RFC6182] has been designed as a successor to TCP [RFC0793] with complete backward compatibility i.e. it is able to use the same TCP socket APIs as well as communicate with the non-MPTCP enabled hosts. MPTCP is included in the mainline Linux Kernel [MPLINUX]. It has also been used in various devices such as iPhone [I-PHONE] to improve the reliability by concurrently connecting the WiFi and the cellular interfaces. Currently, MPTCP has implemented four path managers to effectively use multiple paths between the source and the destination.

- Default: This path manager does nothing more than accepting the passive creation of subflows.

- Full Mesh: This creates a full-mesh of subflows between all available source and destination interfaces. Usually the subflows are created by the client.

- Ndiffports: This path manager creates multiple subflows between every source and destination pair.

- Binder: This is a relatively newer path manager included in the list. It is based on Loose Source Routing [BOC2013].

MPTCP is completely backward compatible with the TCP socket APIs. However, these socket APIs do not exploit the complete potential of MPTCP as these functionality are not available in TCP [RFC6824]. Hence, it is not possible to provide fine grained control over MPTCP and in some ways it restricts the usability of MPTCP. Since, defining new system call is a big ask for MPTCP and application developers are more used to controlling the behavior of TCP using socket option APIs (setSockOpt and getSockopt), it is quite effective to develop new socket APIs for MPTCP as well to control its behavior.

In their draft [HESID17], Hesmans et al. have proposed MPTCP socket APIs to interact with the underlying socket and control the subflow behavior e.g. getting the list of existing subflows, creation of new subflow, termination of subflows etc.. In this draft, we use their draft [HESID17] as the foundation and provide the extended socket APIs to allow the application to mark a particular subflow as an active or a backup subflow. Such a control is very helpful for the applications which intend to send multiple data streams with different Quality of Service (QoS) [SHA2017] requirements to another application or end host.

By default MPTCP creates every new subflow as an active subflow. Hence, if a backup or low priority subflow gets disconnected and connects again, it becomes active i.e. it does not remember its previous state. In this document we also provide a new data structure to remember the states of subflows. Hence, if a particular subflow between any source destination pair is restored after disconnection, it restores with the same state.

The rest of the document is organized as follows: it gives the details of the socket APIs and the corresponding getSockOpt and setSockopt details. We than provide the details of data structure to remember the subflow priority and the corresponding socket APIs and the socket options.

2. Socket APIs for subflow Priority

In this section we present the new socket APIs for the subflow priority control. Currently, MPTCP provides two possible priorities for each subflow; high priority or active subflow, and low priority or backup subflow. These socket APIs can enable the client or the server to dynamically set the subflow priority.

For all socket APIs, the underlying connections are assumed to Multipath TCP connections, otherwise these APIs return an error and

set the errono as "EOPNOTSUPP".

2.1. Control Subflow Priority

This document addresses a typical requirement for an application to change the priority of a subflow without restarting the entire connection.

The first important step to set the subflow priority is to get the list of available subflows. This can be performed using the "MPTCP_GET_SUB_IDS" socket option defined by Hensman et al.[HESID17, HES2017].

The next step is to set the priority of a particular subflow. The "MPTCP_SET_SUB_PRIO" socket option can be used to set the subflow priority identified by its subflow id. This can be called by using "setsockopt" with "MPTCP_SET_SUB_PRIO" as the socket option and passing a pointer to "mptcp_sub_prio" structure. The "mptcp_sub_prio" structure is defined as follows:

```
struct mptcp_sub_prio {
       __u8     id;
       __u16    low_prio;
   };
```

Here, "id" is the subflow id as returned by the get subflow socket API and "low_prio" is priority value to be defined for the subflow. Note that a subflow is an active subflow if the low_prio flag is set to "0". A typical illustration of this API to set the sublfow priority as backup is as follows:

```
struct mptcp_sub_prio fp = {5, 1}; //subflow id 5 is set to backup.

setsockopt(sockfd, IPPROTO_TCP, MPTCP_SET_SUB_PRIO,&fp, sizeof(fp));
```

On successful return of the above socket API, MPTCP protocol sends this information to remote host using MP_PRIO flag.

2.2. Remembering Subflow Priority

A new subflow in MPTCP is in active state and immediately starts sending the data. If the subflow between a particular source and destination is marked as backup using "MPTCP_SET_SUB_PRIO", it becomes active if the subflow gets disconnected and connects again or a new subflow is created. To handle this situation, two new lists "ActiveInterfaceList" and "BackupInterfaceList" are included in MPTCP to remember the state of the subflows between two end hosts. To resolve any inconsistency between the two lists, the former has been

assigned a higher priority than the later.

To populate the active and backup lists, setsockopt with "MPTCP_SUB_PATH_ACTIVE_LIST" and "MPTCP_SUB_PATH_BACKUP_LIST" respectively can be used. This requires the application to pass the pointer to "MPTCP_SUB_PATH" structure. The "MPTCP_SUB_PATH" is defined as follows:

```
struct mptcp_sub_path {
   sa_family_t sa_family;
   union {
      struct in_addr sin_addr;
      struct in6_addr sin6_addr;
   };
   union {
      struct in_addr din_addr;
      struct in6_addr din6_addr;
   };
};
```

These lists can be updated any time; however the runtime update will not change the state of an existing subflow. Hence, this socket APIs must be called before the subflows are created.

3. Security Considerations

There are no new security considerations for this document.

4. IANA Considerations

There are no IANA considerations in this document.

5. Conclusions

This document provides extended socket APIs for MPTCP to control the subflow priorities. These are expected to be very useful for those applications which want a fine grained control over the data to be sent over the available subflows between the end hosts. These APIs can increase the versatility of MPTCP subflows and also provide an opportunity to the application developers to select the subflows more intelligently. This is expected to be useful for different scenarios and devices e.g. drones [SHA2017, RAO2017] where it is important to segregate control data from user data on different subflow.

6. References

6.1. Normative References

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC793, DOI 10.17487/RFC0793, September 1981, <http://www.rfc-editor.org/info/rfc793>.

[RFC6824] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure,"TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6824, DOI 10.17487/RFC6824, January 2013, <http://www.rfc-editor.org/info/rfc6824>.

[HESID17] B. Hesmans and O. Bonaventure, "A socket API to control Multipath TCP", Internet Draft, July 2017, https://tools.ietf.org/html/draft-hesmans-mptcp-socket-02.

[MPLINUX] "Multipath TCP implementation in the Linux kernel", <http://www.multipath-tcp.org>.

6.2. Informative References

[RFC6182] Ford, A., Raiciu, C., Handley, M., Barre, S., and J. Iyengar, "Architectural Guidelines for Multipath TCP Development", RFC6182, DOI 10.17487/RFC6182, March 2011, <http://www.rfc-editor.org/info/rfc6182>.

[I-PHONE] Apple Inc, "iOS - Multipath TCP Support in iOS7", <https://support.apple.com/en-us/HT201373>.

[BOC2013] L. Boccassi, M. M. Fayed, and M. K. Marina, "Binder: A System to Aggregate Multiple Internet Gateways in Community Networks," in Proceedings of the 2013 ACM MobiCom Workshop on Lowest Cost Denominator Networking for Universal Access. ACM, 2013, pp. 3-8.

[SHA2017] S. Shailendra, K. Aniruddh, B. Panigrahi, and A. Simha, "Multipath TCP Path Scheduler for Drones - A Segregation of Control and User Data," in Proceedings of the 18th ACM International Symposium on Mobile Ad Hoc Networking and Computing. ACM, 2017.

[RAO2017] A. Rao, M. Visali, S. Shailendra, B. Panigrahi, and A. Simha, "Reliable Robotic Communication using MPTCP," in 9th International Conference on Communication Systems and Networks, Jan 2017, pp. 429-430.

[HES2017] B. Hesmans and O. Bonaventure, "An enhanced socket API for Multipath TCP," in Proceedings of the 2016 Applied Networking Research Workshop. ACM, 2016, pp. 1-6.

7. Authors' Addresses

Samar Shailendra
TCS Research & Innovation
Bangalore, India

Email: s.samar@tcs.com

Hemant Kumar Rath
TCS Research & Innovation
Bhubaneswar, India

Email: Hemant.rath@tcs.com

Arpan Pal
TCS Research & Innovation
Kolkata, India

Email: arpan.pal@tcs.com

Abhijit Mondal
Indian Institute of Technology Kharagpur
Kharagpur, India

Email: abhimondal@iitkgp.ac.in