An Initial Prototype of Multi-Connection TCP Transport

Michael Scharf, Thomas-Rolf Banniza
Alcatel-Lucent Bell Labs
{michael.scharf,thomas-rolf.banniza}@alcatel-lucent.com

I. Overview of Multi-Connection TCP Transport

Within the G-Lab NETCOMP project [1], we have developed Multi-Connection TCP (MCTCP) transport as a new multipath transport solution. MCTCP aggregates multiple Transmission Control Protocol (TCP) connections between potentially different addresses into a single session that can be accessed by an application like a single TCP connection. MCTCP encodes control information, as far as possible, in the payload of the TCP connections and therefore requires only minor changes in the TCP implementations, and it is transparent in the single-path case. MCTCP is proposed to the IETF as a simple, modular, and extensible mechanism for multipath transport [2].

As shown in Figure 1, MCTCP starts with an initial TCP connection. As long as multiple paths are not used, an MCTCP transfer is identical to a standard TCP transfer, except for a new TCP option in SYN segments that signals MCTCP support between the session endpoints. Once multi-connection transfer is enabled, data chunks are sent over several TCP connections with a new type-length-value (TLV) framing format. This framing also permits the exchange of control information. The multiple TCP connections operate independently, but the MCTCP session coordinates their congestion control states. Applications can use MCTCP exactly like a single TCP connection, as described in [3]. In summary, MCTCP overcomes the limitations of other multipath transport protocols that currently discussed in the IETF.

II. MCTCP Implementation

We have developed a prototype implementation of our protocol as specified in [2], in order to support IETF standardization. It is based on the Linux operating system and consists of two parts that are shown in Figure 2: The mandatory changes inside the TCP stack are realized by a patch of Linux kernel 2.6.32. In addition, a new shim layer library in the user-space implements most of MCTCP's functions and provides a backward compatible socket interface to applications. The kernel and the user space parts interact by a slightly extended socket interface.

Our current prototype implements the baseline MCTCP protocol, including the connection management, segmentation and reassembly of messages over the different paths, scheduling, and coupling of the congestion control states among the different connections.

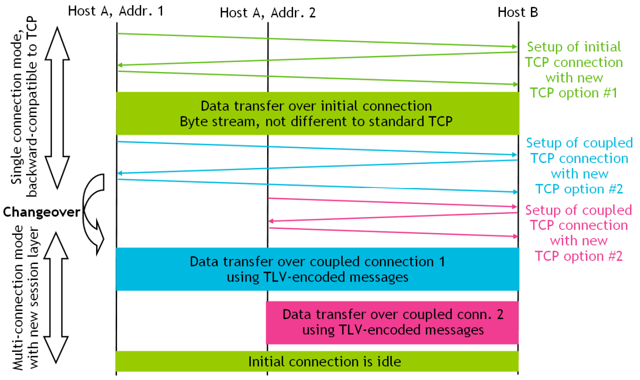
Host A, Addr. 1
Host A, Addr. 2
Host B
Single connection mode, backward-compatible to TCP
Changeover
Multi-connection mode with new session layer
Setup of initial TCP connection with new TCP option #1
Data transfer over initial connection
Byte stream, not different to standard TCP
Setup of coupled TCP connection with new TCP option #2
Setup of coupled TCP connection with new TCP option #2
Data transfer over coupled connection 1
using TLV-encoded messages
Data transfer over coupled conn. 2
using TLV-encoded messages
Initial connection is idle

Figure 1: Illustration of the MCTCP operation

The prototype is still at an early stage and requires further development effort and testing. Still, the realization of multipath transfer with MCTCP can already be demonstrated.

III. Planned Demonstration

In the G-Lab status meeting, we will demonstrate multipath transfer with MCTCP over two paths. The setup will be realized on a laptop that runs several interconnected virtual machines with our MCTCP implementation. We will show the operation of the protocol and the load balancing over the different paths. Due to the early stage of our implementation, the setup will be simple and will not use the G-Lab experimental facilities.

References

[1] BMBF Förderkennzeichen 01BK09

[2] M. Scharf, Multi-Connection TCP (MCTCP) Transport, draft-scharf-mptcp-mctcp-00 (work in progress), Jul. 2010

[3] M. Scharf, and A. Ford, MPTCP Application Interface Considerations, draft-scharf-mptcp-api-01 (work in progress), Mar. 2010

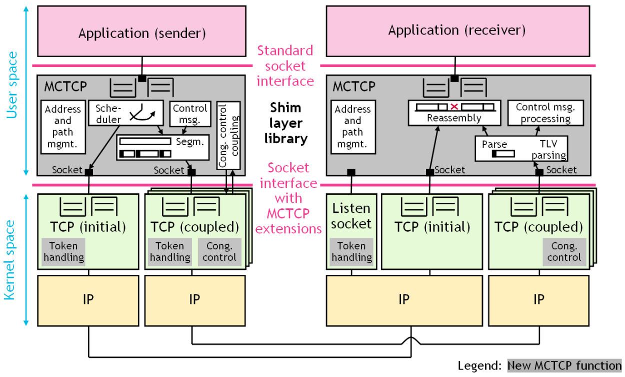

Figure 2: Structure of the implementation of MCTCP