Seamless TCP Mobility Using Lightweight MPTCP Proxy

Georg Hampel[1]
Qualcomm
Bridgewater, NJ, USA
ghampel@qti.qualcomm.com

Anil Rana
Bell Labs, Alcatel-Lucent
Murray Hill, NJ, USA
anil.rana@alcatel-lucent.com

Thierry Klein
Bell Labs, Alcatel-Lucent
Murray Hill, NJ, USA
thierry.klein@alcatel-lucent.com

ABSTRACT
We present a TCP mobility solution for the mobile Internet which enables seamless session end-point migration across multi-provider network environments. The solution can be applied by mobile devices to conduct energy-efficient network selection and dynamic spectrum sharing or by data centers to facilitate service migration across IP domains. We leverage MPTCP's in-band signaling protocol to provide the necessary robustness against firewall- and middle-box policies encountered in these environments. The mobility solution builds on a lightweight MPTCP proxy function which is inserted into the data path and performs header rewriting without packet buffering or stream assembly. The proxy is therefore easily integrated into mobile devices, network routers or data center nodes hence facilitating fast deployment. We discuss the use cases, deployment scenarios and algorithmic challenges of the proxy design. We also present a Linux implementation, demonstrate its functionality and prove the proxy's interoperability with native MPTCP transport-layer implementations.

Categories and Subject Descriptors
H.3.4 [**Information Systems**]: Systems and Software – *Information networks.*

General Terms
Algorithms, Management, Measurement, Performance.

Keywords
Mobility, mobile Internet, traffic offload, TCP, MPTCP.

1. INTRODUCTION
We seek a mobility solution for the mobile Internet, which allows multi-homed devices to seamlessly migrate TCP session end points across multi-operator access environments. Such a technology enables smartphones to hand over between cellular and fixed-wireline operators allowing mobile or multi-homed users to dynamically select the best-performing, lowest-cost or most energy-efficient access link [1]. At the same time, it furnishes cellular operators with a mechanism to dynamically offload traffic flows from their air interface. In the future, such a technology lays the ground for opportunistic network selection by end users and dynamic spectrum sharing among access operators [2]. In another use case, the mobility solution facilitates migration of Internet services between data centers that pertain to different IP domains [3]. While live migration of virtual machines is presently possible it generally remains confined to layer-2 networks.

We demand from our mobility solution to have a realistic deployment path. This means that the benefits from end-point mobility must apply to the entities that are faced with the associated deployment efforts. In our opinion, this factor has not been sufficiently included into past mobility work.

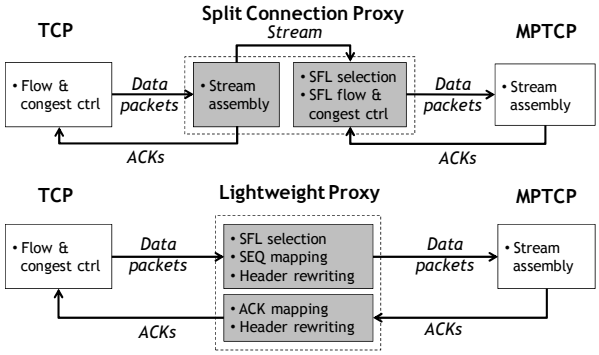


Figure 1. TCP-to-MPTCP protocol conversion. Top: Realization as a split-connection proxy. Bottom: Realization as lightweight proxy.

While a plethora of mobility and/or multi-homing protocols have been developed in recent years they all encounter obstacles that impede proper functionality or deployment for the above use cases. The relevant layer-3 mobility protocols such as Mobile IPv4, Proxy Mobile IPv6 or 3GPP's GTP-based solution restrict mobility to end-user devices, and they rely on inter-access-provider roaming agreements and local roaming anchors [4]-[6]. These prerequisites are usually not available in cellular-to-hotspot handoff scenarios, mostly because fixed wireline operators do not see sufficient economic benefits.

Other layer-3 protocols such as HIP [7] and Mobile IPv6 Route Optimization [8] permit end-to-end signaling circumventing the need for local anchors. The present solutions, however, tend to stall on middle boxes such as firewalls and Intrusion Detection Systems (IPSs) [9], which block the signaling messages (HIP packets, Mobility headers) as well as the user-plane tunneling protocols (IPsec, IPv6 Extension options). The same applies to 3GPP's solution for mobility to non-3GPP domains due to its reliance on UDP encapsulation [10][11]. In all of these cases, middle-box blocking is due to security measures access operators enforce for traffic arriving from untrusted networks.

Finally, TCP-based mobility has been proposed in solutions like TCP Migrate, MSOCKS and Multipath TCP (MPTCP) [12]-[14]. Among those, only MPTCP supports seamless mobility and adequate middle-box compliance. MPTCP, however, faces its

[1] This work was conducted at Bell Labs, Alcatel-Lucent

MobiWac'13, November 3–8, 2013, Barcelona, Spain.

http://dx.doi.org/10.1145/2508222.2508226

own deployment challenge since it demands significant, platform-specific kernel changes on both end hosts. This requirement represents a significant hurdle to the deployment on correspondents such as Internet servers, when the economic benefits are not obvious.

Our contribution: We present a TCP mobility solution that provides seamless session end-point migration while providing sufficient middle-box compliance to operate in multi-operator environments. At the same time, the solution supports multiple deployment scenarios creating an economically viable roll-out path for most use cases.

Our solution leverages MPTCP's signaling framework, which offers the necessary technical features while providing generic mobility support to all applications. To avoid MPTCP's inherent deployment problem, we designed a lightweight MPTCP proxy function (Figure 1), which translates between TCP and MPTCP, and can run on top of a packet filter permitting easy upgrade of end hosts with conventional TCP/IP protocol stacks. The proxy is further compliant with 3GPP gateway processing, which allows using it as a global mobility anchor in the 3GPP operator's network rather than the correspondent.

Finally, end-to-end semantics are retained by avoiding stream assembly or packet buffering. Therefore, the proxy can be used for real-time applications, where TCP is solely applied as a middle-box-compliant wrapper and data are delivered in order of arrival [15].

The main discussion of this paper focuses on the algorithmic challenges of the proxy design, which manages TCP-to-MPTCP translation via straightforward packet-header rewriting. The understanding and meticulous handling of these challenges are instrumental to proper operation and the promised advantages of the proxy-based mobility solution.

The next section discusses the high-level operation of our TCP-based mobility and multi-homing solution. Section 3 addresses the algorithmic challenges of the lightweight proxy design. Section 4 compares our solution to related work. In section 5, we discuss the implementation of our solution, lab setup and measurement results. We close in the conclusion.

2. MPTCP MOBILITY & MULTI-HOMING

2.1 Multipath TCP

MPTCP [14] is a stream-based transport protocol developed by the IETF, which aims to increase the connection's end-to-end throughput by concurrently streaming data along multiple delivery paths. The protocol was designed for operation in wireless access environments, where multiple paths are provided through independent access entities such as cellular and fixed wireline operators. To opportunistically exploit the existing path diversity, special attention was set on MPTCP's compliance with middle boxes such as firewalls, network access translators, performance enhancing proxies and IDSs, which can be encountered in these environments.

MPTCP divides the transport layer into two logical sub-layers referred to as data layer (top) and subflow layer (bottom). The subflow layer sustains an independent data stream on each path, which is (naturally) referred to as subflow. MPTCP applies independent TCP signaling and separate SEQ number spaces on each subflow to make them appear like separate TCP connections on the wire and guarantee passage through middle boxes. The data layer schedules and assembles segments across all subflows. For this purpose, one additional Data SEQ Number space per flow direction is used. The mapping between Data SEQ Numbers (DSNs) and Sublow SEQ Numbers (SSNs) are exchanged between both end points via a TCP option referred to as Direct Sequence Signal (DSS).

MPTCP supplies each subflow with its independent flow- and congestion control. The data layer further couples the subflow congestion-control mechanisms to ensure fairness when concurrent multipath operation competes with conventional TCP connections.

MPTCP conducts all connection management operations in-band via TCP options. Table I summarizes the corresponding TCP options and their specific purpose.

2.2 Split-Connection Proxy

To support interoperation between MPTCP- and TCP-end point, a transport-layer protocol proxy can be employed. Such a protocol proxy creates a split connection where the data stream is first assembled and then retransmitted using a separate flow- and congestion control mechanism (Figure 1, top). Split-connections destroy the connection's end-to-end semantics, and they add delay due to head-of-line blocking at the proxy's stream assembly stage. This delay becomes problematic for real-time applications, which use the underlying TCP carrier as a wrapper to retain middle-box compliance while forbearing in-order assembly at the end host [15].

Split-connection proxies further suffer substantial processing and memory overhead due to stream-assembly and the need for flow and congestion control. The next subsection discusses how these disadvantages can be overcome for mobility use cases.

2.3 Lightweight Proxy for Mobility

Seamless mobility can be regarded as a border case of multipath operation where data transmission is switched between subflows on an occasional rather than frequent base. When switching occurs, new data are sent on the new subflow while in-flight data and their respective retransmissions are handled by the old subflow. Seamless handover therefore resembles multipath operation for a time frame of around one to two Round-Trip-Times (RTTs). For this reason, the sequence-space layering introduced for multipath operation is also needed for seamless mobility. Outside the handover time frame only one subflow carries data while other subflow may be up and remain idle.

Since mobility reduces multipath operation to the timeframe around handover events, transmission can be handled by one single flow- and congestion-control for the entire connection rather than one per subflow. This simplification allows utilizing the native TCP stack for flow- and congestion control and to handle all multipath-related tasks by an independent layer underneath, which we refer to as a Lightweight MPTCP Proxy (Figure 1). The proxy function transforms TCP segments into MPTCP segments and vice versa. For this purpose, it maps between the corresponding sequence spaces and rewrites the TCP headers accordingly. In addition, it provides in-band signaling for subflow setup or tear down and other connection-level functions summarized in Figure 3. Note that the lightweight proxy design retains end-to-end semantics opposed to conventional split-connection proxies. It further avoids the delay associated with head-of-line-blocking during stream assembly which extends its applicability to real-time applications with TCP Minions [15].

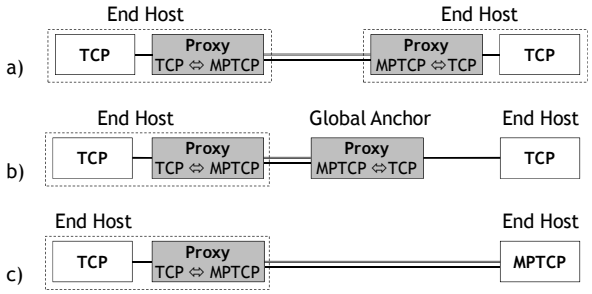


Figure 2. Deployment scenarios for mobility & multi-homing solution using protocol proxies and global

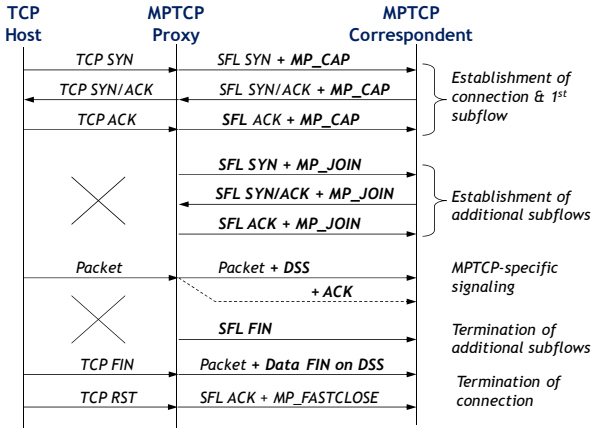


Figure 3. Signaling for establishment and tear down of connection and subflow

2.4 Deployment Options

The lightweight proxy enables a variety of deployment scenarios. For end-user mobility, the proxy can be implemented on top of a packet filter in mobile devices with conventional TCP/IP stacks. Since packet-filter-based solutions are already used for other protocols such as IPsec and tunneling they are readily available for all mainstream smartphone platforms. Furnishing both end hosts with such a proxy hence establishes an end-to-end mobility solution adequate for conversational applications (Figure 2a).

Alternatively, one proxy is placed as a global mobility anchor on a central router such as the packet Data Node GateWay (PDN GW) in 3GPP's Evolved Packet Core (EPC) (Figure 2b). In this scenario, the anchor facilitates mobility support on behalf of MPTCP-unaware correspondents such as Internet application servers. Integrating the proxy into the EPC is rather straightforward since the proxy's header rewriting is in line with typical packet processing tasks performed by these gateways for GTP'based en- and decapsulation. The 3GPP operator further has an economic incentive to deploy such proxy-based anchor functions since they permits dynamic offload of data sessions from the 3G/LTE air interface to unlicensed spectrum.

Finally, the proxy-based solution can interoperate with any native MPTCP transport layer implementation (Figure 2c). In this manner, deployments of lightweight proxies and native MPTCP stack implementations leapfrog each other.

The same principal scenarios also apply to the migration of virtual machines across data centers. MPTCP signaling can further be made compliant with proxy chains where both end hosts as well as intermediate mobility anchors support a proxy function [16].

3. ALGORITHMIC CHALLENGES

The design of the lightweight proxy is faced with a variety of algorithmic challenges due to the following requirements:

- **Seamless mobility**, which demands full support of MPTCP's two-layer sequence spaces,
- **Middle-box compliance**, which demands impeccable TCP behavior on the wire of each path,
- **Interoperability with native MPTCP stacks**, which requires multipath support on the proxy receiver and proper alignment of operation with the protocol,
- **Sequential packet processing** without payload buffering and stream assembly to keep end-to-end semantics intact.

The solutions to these challenges are addressed in the following subsections.

3.1 Connection Management

Since MPTCP uses the same finite state machine (FSM) on data level as TCP the proxy can directly map between the corresponding signaling messages used for connection setup and tear down (Figure 3). For connection establishment, MPTCP applies a SYN/ACK handshake with MP_CAPABLE options enclosed. The proxy conveys the packets of this handshake between TCP and MPTCP by adding or removing the appropriate MP_CAPABLE options. For this purpose, the proxy has to create and store the associated MPTCP keying material.

For connection termination, MPTCP uses Data-FIN and Data-ACK flags carried on DSS options, which directly map to the native TCP FIN- and TCP ACK flags on the TCP section. MPTCP also supports fast connection teardown via the MP_FASTCLOSE option, which is carried on a duplicate subflow ACK and equivalently maps to a native TCP RST.

On the data plane, MPTCP's DSNs map 1:1 to TCP-SEQ Numbers (TSNs). Information on DSNs is carried inside the DSS options which are attached to payload packets. DSS options also carry MPTCP's Data ACK Numbers (DANs), which map 1:1 to the corresponding TCP ACK Numbers (TANs). The processing of DSS options is discussed below.

Additional MPTCP signaling messages such as ADD_ADDR, REMOVE_ADDR, MP_PRIO, etc. are carried on duplicate subflow ACK packets, which are generated by the MPTCP sender and suppressed by the MPTCP receiver. The proxy hence has to forge these packets whenever necessary and drop them upon reception.

Since the proxy communicates connection-state changes between TCP- and MPTCP end points rather than creating a true split connection it has to support one FSM for the entire connection. This FSM and the associated signaling are shown in Figure 4.

3.2 Subflow Management

Each subflow has to appear like an independent TCP connection on the wire with its own sequence number spaces for both flow directions. To avoid confusion between properties related to subflows and those related to the TCP section of the connection, we apply the respective prefixes "SFL" and "TCP" in the following discussion.

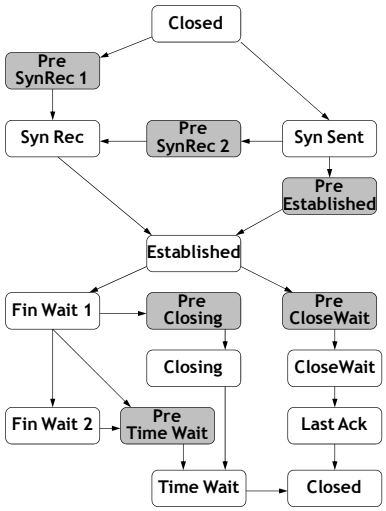


Figure 4a. Finite State Machine of Lightweight MPTCP Proxy.

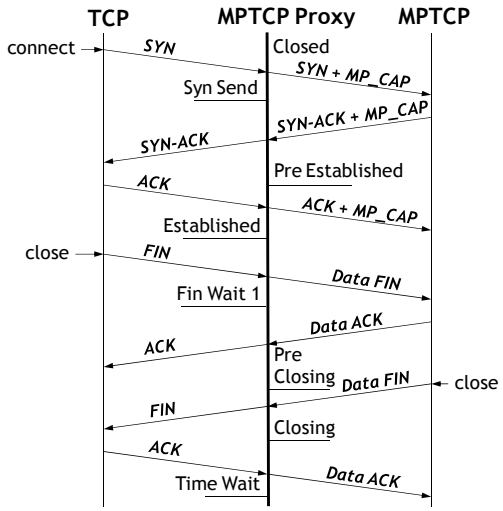


Figure 4b: Signaling flow for connection establishment and termination.

The creation of the initial subflow is overlaid with the SYN/ACK handshake used for connection establishment. The generation of additional subflows and the tear down of established subflows is confined to the MPTCP section of the connection. The proxy has to sustain a separate FSM for each subflow and forge or suppress the respective outgoing or incoming signaling packets. The proxy further has to create the initial sequence number for every subflow and cache state information on the port numbers and IP addresses used by each subflow. The selection of port number and IP addresses follow MPTCP regulations. When forwarding packets from one side to the other, the proxy has to change IP addresses and port numbers accordingly.

3.3 Payload Forwarding

TCP *payload* → **PROXY** → **MPTCP**

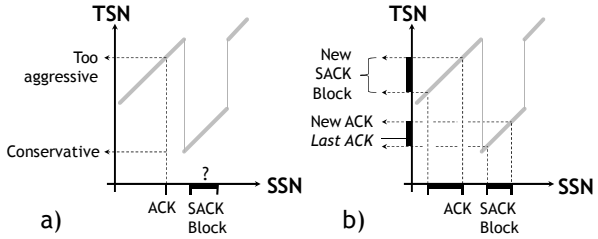


Figure 5. Mapping from SAN to TAN using SEND and RECV tables. 5a: Rreverse lookup of SAN-1. 5b: Proposed projection technique.

For each payload packet arriving on the proxy's TCP side, the proxy has to select a subflow and perform the mapping TSN→DSN→SSN. The TSN→DSN mapping represents a constant offset which can be set to zero; the DSN→SSN mapping depends on the subflow selection, which is determined by mobility management. Packet retransmissions should generally be sent on the same subflow with the same DSN→SSN mapping as the original transmissions. This avoids sequence number gaps on the original subflow, which would lead to blocking by IDSs. If the original subflow is no longer available, retransmitted packets have to be sent on the presently active subflow.

To identify retransmissions, the proxy caches subflow selection and DSN→SSN mapping for each payload packet in a SEND table. To minimize table size, all table entries that are contiguous in the SEQ space and have same DSN→SSN mapping can be merged. This usually results in a single entry in between handovers.

After subflow selection and SSN space allocation, the proxy rewrites the packets SEQ number field to the appropriate SSN value, inserts the DSS option with the corresponding DSN→SSN mapping, updates packet length and checksum fields and transmits the packet on the subflow.

For DSN→SSN mapping, the SSN space allocation should generally be done contiguously, i.e. without producing SSN gaps. In case TCP packets arrive out of order, a contiguous SSN space allocation would lead to a non-monotonous DSN→SSN mapping as shown in Figure 5. Since such non-monotonous mappings create problems for ACKs flowing in opposite direction they should be avoided. This can be accomplished by preemptively reserving DSN→SSN mapping space for the delayed payload packets, which effectively translates out-of-order TCP packets into the equivalent out-of-order SFL packets.

After break-before-make handovers, non-monotonous mappings are often unavoidable. When the active subflow suddenly breaks, there is generally no proper synchronization between the end points on the data that have been successfully delivered on the old subflow. Under such circumstances, SSN space has to be sequentially allocated on the new subflow as TCP packets arrive. This leads to non-monotonous mappings when the TCP sender mixes retransmissions from the old subflow with new payload data.

TCP ← **PROXY** ←*payload* **MPTCP**

For payload running in the opposite flow direction, i.e. arriving at the proxy's MPTCP side, the reverse SSN→DSN mapping can be retrieved from the DSS option carried in the payload packet. In case middle boxes have re-segmented the packet stream, the DSS option may be attached to a later packet resulting in a SSN→DSN

mapping that is out of synch with the payload's actual SSN range. To always assert the availability of proper DSN→SSN mapping information at the proxy, MPTCP permits the remote payload sender to advertise an extended DSN→SSN mapping range that exceeds the packet's payload size. This feature is referred to as "bulk optimization".

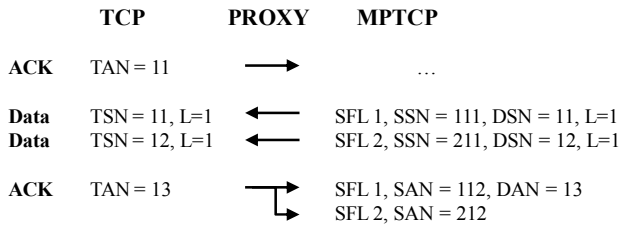
To take advantage of bulk optimization, the proxy extracts the DSS mapping information, enters it into a subflow RECV table and then uses this table to reverse-map the packet header's SSN entry to a DSN value. From this DSN value, the proxy derives the associated TSN, enters it in the packet's SEQ number field, updates packet checksums and forwards the packet.

3.4 ACK Forwarding

TCP *ACK* → **PROXY** → **MPTCP**

While payload forwarding is rather straightforward, the management of ACKs becomes more intricate. When a new TCP ACK arrives at the proxy's TCP side to acknowledge an increment in payload delivery over the preceding TCP ACK, the proxy has to select the appropriate subflow, translate the TAN to a DAN and create an appropriate SAN. The resulting DAN is entered into the DSS option, which is inserted into the packet header.

On subflow layer, one SFL ACK has to be generated for each subflow that has contributed to the TAN increment. This may involve multiple subflows as, for instance, during seamless handover or when the remote host performs multipath operation. An example is shown here, where TSN has been set equal to DSN and the payload holds only one octet:



Since both subflows have contributed to TAN = 13 in this example they both deserve an associated SFL ACK. The inference of SAN from TAN (or DAN) is straightforward in this example and discussed below for more general cases.

Subflows that have not contributed to the payload increment should not be served with SFL ACKs since this would send congestion signals to the peer. When the TCP packet arriving at the proxy also contains payload pertaining to the opposite flow direction, the payload data may have to be sent on a different subflow than the SFL ACK.

Duplicate TCP ACKs arriving at the proxy indicate congestion. Since MPTCP does not support congestion notification on data level, we established an appropriate forwarding rule: Duplicate TCP ACKs should only be forwarded on a subflow if not yet acknowledged payload (highest SSN > SAN-1) or a payload retransmission has traveled in opposite direction on this subflow.

TCP ← **PROXY** ←*ACK* **MPTCP**

When an ACK packet arrives on a subflow with a DAN-carrying DSS option, it can be forwarded to the connection's TCP section by mapping the DAN to the corresponding TAN. MPTCP, however, does not enforce the same strict rules to DAN transmission as TCP does for ACKs. Re-segmenting middle-boxes may further move DAN-carrying DSS options within the packet stream. DAN values may therefore arrive with lower or higher frequency than desirable for proper TCP operation.

Alternatively, TCP ACKs and their respective TAN values can be inferred from subflow- rather than data-level. The MPTCP design document discourages inference of connection-level ACKs from subflow ACKs since it may have detrimental impact on multipath operation in case subflow ACKs are forged by middle boxes. As long as the proxy performs single-subflow transmission, i.e. in between seamless handovers, these concerns do not apply. During seamless handover, however, the same rules have to be obeyed as for multipath operation and TCP ACKs should only be derived from DANs contained in DSS options. In case this invokes excess duplicates, the impact will be small since the handover time frame is short and other handover-related factors dominate performance (e.g. TCP slow start).

3.5 ACK Number Projection

The above discussion has not revealed a general rule set for the translation from TANs to SANs and vice versa. This translation can be derived from the DSN→SSN and SSN→DSN mappings contained in the respective SEND and RECV tables. While the translation becomes a simple reverse lookup for monotonous mappings, it leads to ambiguities in other cases. Figure 5a shows an example where TAN (or equivalently DAN) is derived from SAN through such a reverse lookup function:

$$TAN = \textbf{lookupTSN}\,(\,SAN - 1\,) + 1$$

Applying the same lookup function to a non-monotonous mapping produces too aggressive TANs which acknowledge TSNs that have not yet been delivered. The figure also depicts another lookup function, which delivers an acceptable but very conservative TAN. Obviously, such conservative translation will lead to performance degradation or retransmission of incorrect data in response to duplicate SANs. The situation becomes more complex when Selective ACK (SACK) blocks are contained in arriving packets.

To overcome this problem, we developed an ACK number projection method which handles non-monotonous mappings, incorporates SACK and equally applies to ACK forwarding from TCP to SFL as well as in opposite direction. Without losing generality, we restrict the discussion to the (non-monotonous) TSN→SSN mappings and their reverse translation from SAN→TAN.

In this projection method, an incoming SAN is interpreted as an additional SACK block = [SAN – rwind, SAN], which is added to the array of SACK blocks contained in the incoming packet header (if any). For each SACK block [SAN_L, SAN_R], the corresponding SSN interval = [SAN_L, SAN_R – 1] is projected from the SSN to TSN space based on the mapping graph represented by the mapping table (Figure 5b). This projection leads to a set of disparate SACK blocks [TAN_L, TAN_R] in the TSN space.

If the last transmitted TAN falls into one of these projection blocks, the right edge of that block represents the TAN of the new TCP ACK (Figure 5b). All projection blocks above this TAN value become TCP SACK blocks. If the prior TAN does not fall

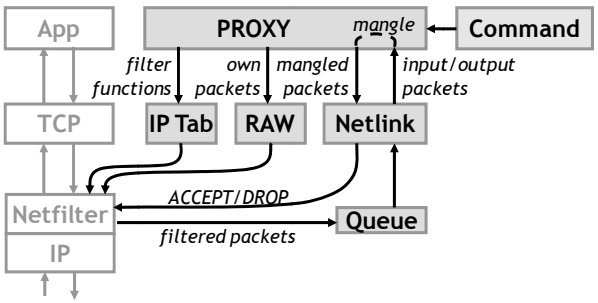


Figure 6. Block diagram of Linux implementation

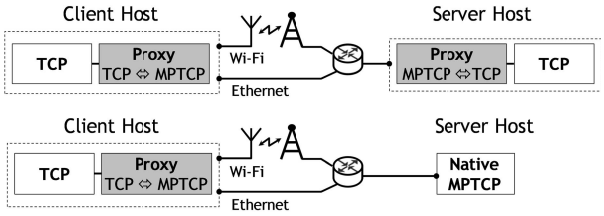


Figure 7. Measurement setups for trials

into any of the projection blocks, a duplicate TCP ACK is sent, and all projection blocks above the prior TAN become TCP SACK blocks. The TCP SACK blocks are inserted into the TCP header of the outgoing TCP ACK. The SFL SACK entries contained in the incoming packet have to be removed.

This projection method guarantees correct and complete inference of ACK information from one sequence space to another for all mapping curves. Obviously, the method can also be applied if SACK is not supported. In this case, only the [SAN – rwind, SAN] block needs to be considered for projection, which leads to the conservative estimate for TAN shown in Figure 5a.

4. RELATED WORK

A variety of publications have addressed the integration of mobility and multi-homing into the TCP end host (e.g. [12]). All of these protocols use the same sequence spaces on all paths which makes them vulnerable to checks on SEQ-number gaps conducted by stateful firewalls and IDSs. Also, incremental deployment of these solutions remains a challenge due to their end-host-based nature.

MSOCKS introduces a TCP proxy for mobility [13]. As our lightweight MPTCP proxy, MSOCKS proposes a SEQ space mapping algorithm referred to as TCP Splice to avoid split connections. TCP Splice, however, is far simpler than our solution permitting only break-before-make rather than seamless mobility.

ROCKS & RACKS introduce a session layer mobility solution for TCP, which is integrated into the end hosts [17]. RACKS, in particular, refers to a packet-filter implementation reminiscent to our lightweight proxy proposal. ROCKS & RACKS also lack support for seamless handovers. Since conducted on session layer, the handover break becomes prolonged due to the setup and tear down of multiple consecutive TCP connections for signaling purposes.

In another publication, we proposed a session mobility solution for stateless applications using HTTP, which can also be realized via packet rewriting [18]. This solution, however, cannot be

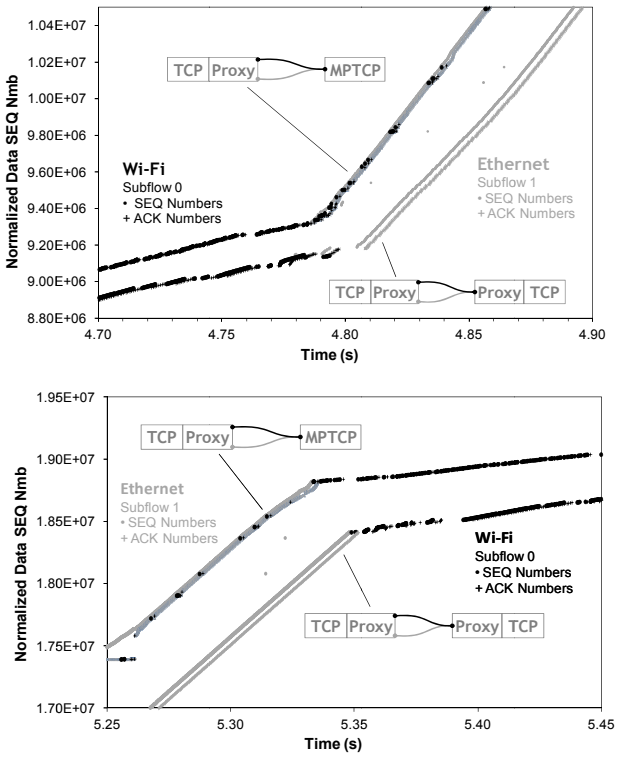


Figure 8. Seamless handover between Wi-Fi and Ethernet on multi-homed client host. Measurements are shown for configuration with two proxies and for proxy on client and native MPTCP on server.

extended to non-HTTP applications and it does not support seamless mobility.

5. MOBILITY TRIALS

5.1 Proxy Implementation

We implemented the MPTCP Proxy on a Linux Platform using the Netfilter/Netlink framework for packet filtering. In our implementation, packet processing is performed in userspace, which allows easy algorithm development and debugging. The final code can then be inserted as a kernel module at a later stage.

Figure 6 shows the block diagram of our implementation. Netfilter selects packets according to a set of filter rules, enters them into a queue where they are picked up by the proxy via a Netlink socket. The proxy processes each packet and returns it to Netfilter. The proxy further sustains a raw socket to send additional signaling packets, and it sets the filter rules for Netfilter via IPtables.

Our implementation further supports a command line tool which interfaces with the MPTCP Proxy for external subflow management, e.g. to add or delete subflows, and to enforce mobility decisions on the remote MPTCP sender. The mobility commands permit seamless handovers as well as break-before-make transitions. Mobility events also require updates to the host's routing tables. For seamless handover, source routing has to be supported.

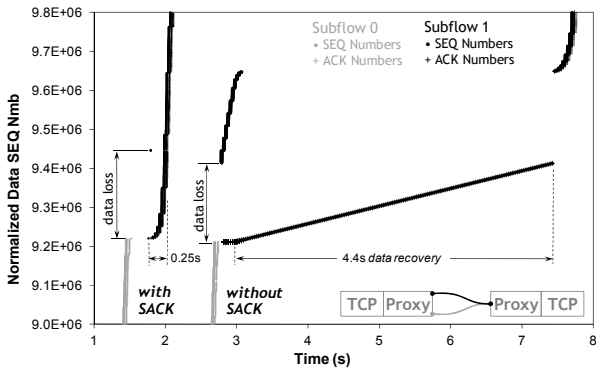


Figure 9. Break-before-make handovers with and without SACK between two Ethernet interfaces on the multi-homed client host.

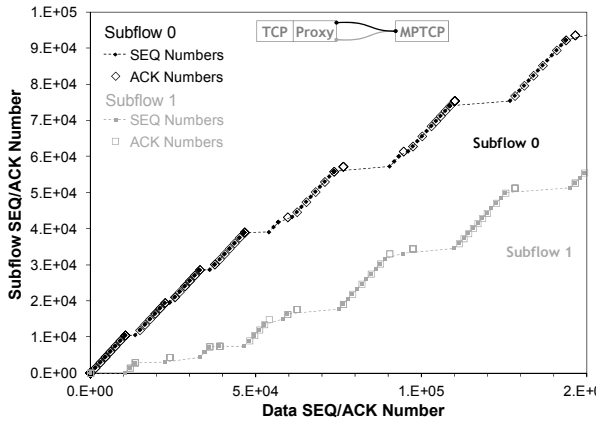


Figure 10. Multipath reception on proxy using two subflows with Ethernet interfaces. The sender is a native MPTCP stack.

The source code of the implementation has been published [19]. The current version supports end-host-based as well as anchor-based operation.

5.2 Trial Setup

Using our proxy implementation, we conducted trials on a lab bench as well as on live networks. In this paper, we focus on lab bench trials to verify functionality and interoperability. Live-network trials are published elsewhere [21].

The lab bench trials consisted of a layer-3 network interconnecting two interfaces of a multi-homed host with a correspondent (Figure 7). In some trials, the multi-homed host sustained one Wi-Fi interface and one Ethernet interface; in others it sustained two Ethernet interfaces. We generally evaluated two different configurations. In one of them, referred to as PROXY⇔PROXY, both hosts hold a native TCP stack together with a lightweight proxy. In the other, referred to as PROXY⇔MPTCP, only the multi-homed host uses a TCP stack and lightweight proxy while the correspondent supports a native MPTCP stack [20]. In both configurations, bulk data are streamed from the correspondent to the multi-homed host.

5.3 Seamless Mobility

For seamless handover trials, two subflows – each using one interface of the multi-homed host – are simultaneously sustained between both hosts. The proxy on the multi-homed host initiates path switching by sending MP_PRIO options to the correspondent. The trial verifies if these messages are properly communicated to the correspondent and acted upon.

Figure 8 shows traces of DSNs and DANs plotted against time for a Wi-Fi-to-Ethernet and Ethernet-to-Wi-Fi handover measured at the proxy of the multi-homed host. On the time scale of 200ms shown in these plots, the traces do not indicate any significant flow interruption. The PROXY⇔PROXY configuration shows a clean handover. In the PROXY⇔MPTCP configuration, the native MPTCP stack generally follows the MP_PRIO commands but it keeps transmitting occasional packets on Wi-Fi when asked to use Ethernet. Also, the MPTCP stack inserts some stray retransmissions from time to time. We don't have any explanation for this behavior.

5.4 Break-before-Make Mobility

In the break-before-make scenarios, transmission is started with only one subflow established on the Ethernet interface of the multi-homed host. Then, handover is imposed on the proxy via command-line interface, which disrupts the existing subflow, initiates establishment of a new subflow on another Ethernet interface and signals the handover via MP_PRIO- and REMOVE_ADDR options to the remote sender. Since the sender is not aware of the break before receiving these messages on the new subflow it keeps streaming data on the old subflow, which are consequently lost. The trials were conducted with and without SACK. Further, a delay of 30ms was added with a network emulator to both paths to simulate realistic access scenarios.

Figure 9 shows measurements of DSN and DAN plotted against time for two trials, one with and another without SACK. In the measurement with SACK, data recovery after the break takes around 250ms while it amounts to ~4.4s without SACK. The poor performance in absence of SACK is due to non-monotonous mappings and conservative ACK translation. This shows that SACK support is essential in deploymenst with two lightweight proxies.

Note that even with SACK, flow interruption can be substantially longer as observed in access environments with high path delay [21]. Under such circumstances break-before-make handover may have significant performance impact. This motivates the support of seamless mobility as we propose in this work.

5.5 Multipath Reception

When a native MPTCP stack communicates with a lightweight proxy it is not bound to the proxy's recommendations on path selection (as seen in subsection 5.3), and it can freely conduct multipath transmission. To guarantee interoperability between both entities, the proxy must be able to handle full multipath reception.

We verified the proxy's multipath reception capability using the PROXY⇔MPTCP configuration with two simultaneously established subflows. To ensure similar delay and throughput characteristics on both paths the multi-homed host used Ethernet interfaces for both subflows. The proxy did not impose any restriction to the peer on subflow selection by setting each subflow's backup flag to zero during subflow establishment.

Figure 10 shows the traces of packet SSNs and SANs for both subflows plotted against the corresponding DSNs and DANs. The measurements show proper multipath operation, which verifies the proxy's multipath reception capability. The slightly unbalanced path utilization seen in this figure may be due to a

discrepancy in path properties and the particular scheduling algorithm applied by the native MPTCP stack.

6. CONCLUSION

We presented a solution to seamless TCP mobility in multi-provider access environments. Our solution addresses the unique technical challenges encountered in such scenarios such as obstruction through path-specific middle-boxes. Our work emphasized on deployability as an additional objective to functionality, which led to a meticulous design recast of an existing protocol, namely MPTCP, rather than a proposal of a new protocol. Given the relevance of our use cases as well as the emphasis on deployability we hope that our solution can gain traction in existing markets.

While our trials have demonstrated the proper operation of our proxy design extended testing involving a broader R&D community is necessary. It would further be desirable to investigate portability of the lightweight proxy to actual mobile platforms such as iOS, Android and Windows Phone 8. We also encourage the MPTCP Working Group to investigate signaling extensions toward enhanced mobility support.

7. REFERENCES

[1] I. Ferdo, “Convergence and competition on the way toward 4G: where are we going?” Radio and Wireless Symposium, 2007 IEEE, Jan 2007.

[2] L.L. Cao, H. Zheng, M. Nekovee, M. Buddhikot, “Market driven sharing of spectrum in infrastructure networks,” in Cognitive Radio Communications and Network: Principles and Practice. Academic Press book, Nov 2009.

[3] K.K. Ramakrishnan, P. Shenoy, J. Van der Merwe, “Live Data Center Migration acrossWANs: A Robust Cooperative Context Aware Approach”, Sigcomm, 2007.

[4] C. Perkins, “IP mobility support for IPv4”, RFC3344, IETF, Aug. 2002.

[5] S. Gundavelli, K. Leung, V. Devarapalli, K. Chowdhury and B. Patil, “Proxy mobile IPv6”, RFC5213, IETF, Aug. 2008.

[6] 3GPP, “General Packet Radio Service (GPRS) enhancements for Evolved Universal Terrestrial Radio Access Network (E-UTRAN) access (Release 11)”, TS 23.401.

[7] R. Moskowitz and P. Nikander, “Host Identity Protocol (HIP) architecture,” RFC 4423, IETF, May 2006.

[8] C. Perkins, D. Johnson & J. Arkko, “Mobility support in IPv6”, RFC6275, IETF, July 2011.

[9] M. Handley, V. Paxson and C. Kreibich, “Network intrusion detection: evasion, traffic normalization, and end-to-end protocol semantics,” Usenix Security 2001, 2001.

[10] 3GPP, “3GPP system to Wireless Local Area Network (WLAN) interworking, wystem description (Release 11),” TS 23.234.

[11] 3GPP, “Architecture enhancements for non-3GPP accesses (Release 11)”, TS 23.402.

[12] A. Snoeren and H. Balakrishnan, “An end-to-end approach to host mobility,” Proceedings of the 6th Annual International Conference on Mobile Computing and Networking, ACM, 2000.

[13] D.A. Maltz, “MSOCKS: An architecture for transport layer mobility”, INFOCOM, March 1998.

[14] A. Ford, C. Raiciu, A. Greenhalgh and M. Handley, “Architectural guidelines for multipath TCP development,” RFC 6182, IETF, March 2011.

[15] J. Iyengar, B. Ford, A. O. Amin, M. F. Nowlan, N. Tiwari, “Minion: Unordered Delivery Wire-Compatible with TCP and TLS,” http://arxiv.org/pdf/1103.0463.pdf.

[16] G. Hampel and T. Klein, “MPTCP proxies and anchors,” http://tools.ietf.org/html/draft-hampel-mptcp-proxies-anchors, Feb. 2012.

[17] V. Zandy, B. Miller, “Reliable network connections”, 8th Int. Conf. Mobile Computing & Network, 2002.

[18] W. Song, G. Hampel, A. Rana, T. Klein and H. Schulzrinne, “MOSAIC: Stateless mobility for HTTP-based applications,” WiMOB, Oct. 2012.

[19] http://open-innovation.alcatel-lucent.com/projects/mptcp-proxy, Go to "Latest File Releases" → "DocManager: Project Documentation" → "Uncategorized Submissions".

[20] MPTCP stack implementation by IP Networking Lab, UCL, Louvain-la-Neuve, Belgium, http://mptcp.info.ucl.ac.be/

[21] G. Hampel, A. Rana, W. Song & T. Klein, “MANTRA: Mobility across Non-Trusting Access Domains”, to be published.