Exploring Mobile/WiFi Handover with Multipath TCP

Christoph Paasch[†]
christoph.paasch@uclouvain.be

Gregory Detal[†]
gregory.detal@uclouvain.be

Fabien Duchene[†]
fabien.duchene@uclouvain.be

Costin Raiciu[‡]
costin.raiciu@cs.pub.ro

Olivier Bonaventure[†]
olivier.bonaventure@uclouvain.be

[†]Université catholique de Louvain
[‡]University Politehnica of Bucharest

ABSTRACT

Mobile Operators see an unending growth of data traffic generated by their customers on their mobile data networks. As the operators start to have a hard time carrying all this traffic over 3G or 4G networks, offloading to WiFi is being considered. Multipath TCP (MPTCP) is an evolution of TCP that allows the simultaneous use of multiple interfaces for a single connection while still presenting a standard TCP socket API to the application. The protocol specification of Multipath TCP has foreseen the different building blocks to allow transparent handover from WiFi to 3G back and forth.

In this paper we experimentally prove the feasibility of using MPTCP for mobile/WiFi handover in the current Internet. Our experiments run over real WiFi/3G networks and use our Linux kernel implementation of MPTCP that we enhanced to better support handover.

We analyze MPTCP's energy consumption and handover performance in various operational modes. We find that MPTCP enables smooth handovers offering reasonable performance even for very demanding applications such as VoIP.

Finally, our experiments showed that lost MPTCP control signals can adversely affect handover performance; we implement and test a simple but effective solution to this issue.

Categories and Subject Descriptors

C.2.5 [**Computer-Communication Networks**]: Local and Wide-Area Networks—*Internet*; C.2.6 [**Computer-Communication Networks**]: Internetworking—*Standards*; C.4 [**Performance of Systems**]: Design Studies; Performance attributes.

General Terms

Measurement, Performance

Keywords

MTCP, Vertical Handover, WiFi/3G, Energy Consumption

CellNet'12, August 13, 2012, Helsinki, Finland.

1. INTRODUCTION

Despite the huge investments, 3G mobile data networks have difficulties in supporting the growing bandwidth consumed by recent smartphones and tablets. Industry experts expect that these bandwidth requirements will continue to grow in the future and the deployment of new cellular technologies such as 4G or LTE may not be sufficient to sustain the demand.

In parallel with cellular data networks, WiFi is being deployed continuously: most enterprise and schools provide WiFi services to their employees and students, some cities have started to deploy WiFi services for their citizens, and many homes use WiFi networks to connect to cable or ADSL modems. Some operators, including BT in the UK or Belgacom in Belgium, have deployed managed WiFi services on most of their clients' ADSL routers. This allows services like FON to aggregate more than 3 million access points in the UK alone [1].

The widespread availability of 3G and WiFi creates obvious opportunities for end-users and operators alike. User experience can be improved if devices use multiple links either simultaneously or alternatively while moving. Mobile network operators would like to be able to "steer" the network their clients are using depending on 3G occupancy, etc.

Meanwhile, the devices themselves use either the 3G or WiFi network at any given point in time, and any handover between the two causes TCP connections to be restarted which disgruntles users.

Multipath TCP (MPTCP) [10] is a TCP extension being standardized at the IETF that enables a single TCP to use multiple interfaces on the client and/or the server. MPTCP works with unmodified applications and runs over today's Internet. In our previous work, we have described why MPTCP is a promising solution for mobility [9] and used simulations to assess the possible benefits.

In this paper we describe our experimental study of MPTCP handover. Our aim is to understand *how* MPTCP will behave in practice, on real wireless networks and with real applications. To perform such evaluation we had to optimize the Linux MPTCP stack in order to better support WiFi/3G handover[1].

A key concern for running MPTCP on mobiles is energy usage: we measure energy consumption of MPTCP on the Nokia N950 to understand this aspect. Our results show that using multiple interfaces provides better throughput but at

[1]Our Linux Kernel Implementation of MPTCP is freely available at http://mptcp.info.ucl.ac.be

higher energy cost (Section 4.4). To capture this tradeoff, we first choose three handover modes and test their performance in practice while handing traffic from WiFi over to 3G (Section 4). We then show that MPTCP allows smooth handovers from WiFi to 3G while maintaining connectivity for applications. Finally, we propose minor changes to the MPTCP protocol that significantly reduce handover delays in certain practical cases (Section 5).

2. MPTCP VERTICAL HANDOVER

Multipath TCP (MPTCP) spreads the traffic over different interfaces of an end-host, moving traffic away from congested links while being fair to single path TCP [14]. A key design choice of MPTCP is its backward compatibility with both the applications and the network; MPTCP preserves the standard socket API that is used by most Internet applications.

An MPTCP connection contains one or more subflows, each of which appears like a regular TCP connection to the network. When the connection starts, MPTCP options are included in the SYN segments to verify that the destination is MPTCP capable and to negotiate a token that will uniquely identify this connection. Additional TCP subflows can be associated to this MPTCP connection by carrying the previously exchanged token in their three-way handshake. MPTCP splits the application's data stream among all the established subflows to maximize the use of all the available resources. Only one subflow will typically be used on each interface in order to improve the reaction to failures and maximize throughput. In a mobile environment, mobile nodes have usually one or more wireless connections to the Internet using a WiFi, 3G and/or 4G interface. Such mobile nodes can leverage Multipath TCP's ability to use multiple interfaces in order to increase the performance but also to preserve established MPTCP connections while moving around. The later is not possible with standard TCP as any change in the IP address of a host forces it to restart all established TCP connections. With MPTCP, a change in the IP address of a host does not force the MPTCP connection to be restarted. An MPTCP connection is associated to a set of underlying TCP subflows. TCP subflows can be added or removed from this set without impacting the MPTCP connection or the application. This ability to add and remove TCP subflows is key in MPTCP's ability to support mobile nodes. We describe it in more detail below.

2.1 Adding and Removing TCP Subflows

Once an MPTCP connection is established, each endpoint knows one of the IP addresses[2] of its peer.

If a mobile client has an additional address, it will just send a SYN packet with a JOIN option to the server's known address. The JOIN option tells the server which connection this subflow belongs to. If the server has an additional address, it cannot directly connect to the mobile client if the latter is behind a NAT. In this case, the server sends an *Add Address* option on an existing subflow indicating the availability of a new address. Upon reception of this option the client will try to establish a new TCP subflow to the newly received address.

[2]MPTCP supports both IPv4 and IPv6. In this paper, we use IP address as the generic term for an IPv4 or IPv6 address. Our implementation supports both IPv4 and IPv6.

Similarly, the *Remove Address* option can be used to inform the peer when one of its addresses has become unavailable (e.g., the mobile node is no more connected to a WiFi access point). Upon reception of this option the destination closes all TCP subflows that are using this address.

These two options are typically sent on separate TCP ACKs[3] and thus may get lost in the network. When losing an interface, the *Remove Address* option is sent on the remaining subflows, which can be none in some cases, e.g., when a node is moving from one WiFi access point to another. Therefore a server might not always be aware of the lost address on the mobile node which, as we will see later, can impact the handover performances.

3. HANDOVER MODES

A mobile node should be able to adapt its protocol stack to its user's requirements. From the user's viewpoint, there are three important factors to be considered. The first factor is the performance of the data transfer. Some users will probably prefer the fastest possible data transfer. The second factor is the battery lifetime. Some users will probably trade performance for longer battery lifetime. The third factor is traffic pricing. Some networks, typically 3G networks are billed in function of the number of transmitted bits or packets. Some users will favor cheaper networks.

An MPTCP implementation should take these factors into account. In our current implementation, we have identified and implemented three modes of operation for MPTCP that correspond to most user needs.

Full-MPTCP Mode refers to the regular MPTCP operations where all subflows are used, i.e., where a full mesh of TCP subflows among the client's and the server's addresses is created. Such mode is mostly intended for users who want to obtain the best data transfer rates.

Backup Mode. MPTCP opens TCP subflows over all interfaces, but uses only a subset of these to transport data segments. The MPTCP protocol allows an endpoint to signal to its peer that it should avoid sending data on a specific subflow by sending the MP_PRIO option on this subflow. This would enable MPTCP to optimize for cost or battery lifetime by sending data segments over the cheaper interfaces. As we will see later, in today's networks it makes sense for MPTCP to prefer the WiFi interface when available and only use the 3G interface when there is no WiFi connectivity.

Single-Path Mode allows a similar behavior as the *Backup* mode, except that at any moment a single subflow is established and used for each MPTCP connection. When the interface goes down, *Single-Path* establishes a new TCP subflow over the other interface. This is possible due to the *break-before-make* design of MPTCP. It allows a short moment during which no subflow is active. MPTCP is able to recover from this interruption by establishing a new TCP subflow and continue the data transmission without disturbing the application. Compared to the *Backup* mode, this mode needs to wait two more round-trip times before the new MPTCP subflow is established and data can be sent.

[3]MPTCP does not treat duplicate ACKs containing such options as a signal of congestion. [4]

4. EVALUATION

In this section we provide a first evaluation of the performance tradeoffs when using the three handover modes presented in the previous section. We perform our measurements in real networks. Using such networks is more difficult than performing lab measurements, but is much more realistic and shows the ability of MPTCP to function in deployed networks. Our measurement setup consists of a client (i3 @2.5GHz 4GB RAM) connected to both a commercial 3G network and through a WiFi interface to an ADSL broadband access provider. The 3G network offers a bandwidth between 1 and 2 Mbps and suffers from huge bufferbloat[4]. The ADSL bandwidth is around 8 Mbps upstream and 450 Kbps downstream with an RTT about 30ms. The client connects to a server (Xeon @2.67GHz 4GB RAM) located in a public-hosting server farm. Both client and server run our MPTCP kernel implementation.

Our measurements focus on two distinct aspects of MPTCP usage: first, we explore the impact of a handover on MPTCP connections using the different modes on both bulk data transfers and a VoIP application. Second, we explore the energy usage of MPTCP when using 3G or WiFi to understand how users might optimize for energy.

Our handover experiments start with a client connected to both WiFi and 3G networks and then after five seconds we disable the WiFi interface on the ADSL router. This emulates a mobile user moving out of WiFi coverage: the client discovers the failure of the WiFi network and switches to only using its 3G connection. This vertical handover from WiFi to 3G introduces an abrupt change in path qualities.

4.1 Download Goodput

We evaluate the evolution of goodput for a simple HTTP application during vertical handover. Fig. 1 shows this goodput averaged over 200ms intervals over 20 measurements for the different handover modes. The x-axis shows the offset compared to the time when the client lost its WiFi connection (the point 0).

For comparison purposes, Fig. 1 also shows the goodput of an *Application-level Handover* which is a modified HTTP client (running over regular TCP) that monitors the changes in the routing table to detect the failure of the WiFi interface. Upon detection of the failure, our application restarts the HTTP download by using the *HTTP Range* header. Supporting such handover requires significant changes to the application, and is specific to the HTTP protocol; in contrast, MPTCP does not require any application-level modification. Further, even if all client applications use HTTP Range, a brief study we performed shows that only 39% of the top ten thousand Alexa websites support this feature. Although feasible in theory, application-level handover cannot always be used in practice.

We can see in Fig. 1 that when the WiFi network becomes unavailable, the *Backup* mode behaves similarly to *Full-MPTCP* which is normal. The TCP subflow on the 3G interface is already established in both cases. The difference between the two modes comes from the fact that in *Backup* mode only the three segments belonging to the three-way handshake have been sent on the 3G interface while with the *Full-MPTCP* mode data segments have also been transmitted on this interface. *Full-MPTCP* recovers quickly because the congestion window on the 3G subflow is larger than in the *Backup* mode, where the congestion window still has the initial value. The *Single-Path* mode and *Application Level Handover* are impacted by the time the 3G interface takes to go from idle to up. Without this impact they behave similarly: they both need to perform a three-way handshake and then respectively reinject and send data into this new subflow/connection, starting with an initial congestion window. After three seconds, all modes reach the average download speed on our 3G network.

[4]We observed up to 2 seconds RTT when sending bursts of data.

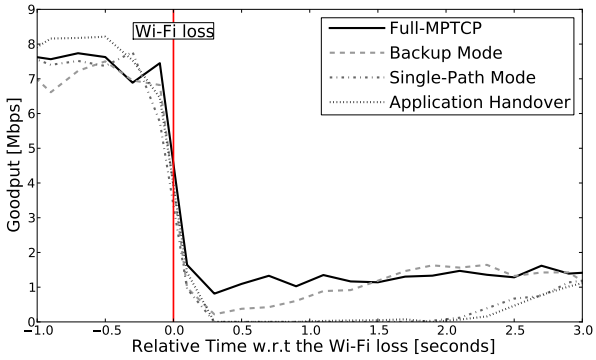


Figure 1: It takes up to three seconds to recover from an address loss when performing handover with MPTCP.

We also performed measurements where we forced the 3G interface to remain active, and found that there were not much differences between the *Backup* and *Single-Path* modes. In this case, the performance impact of the three-way handshake is negligible on average.

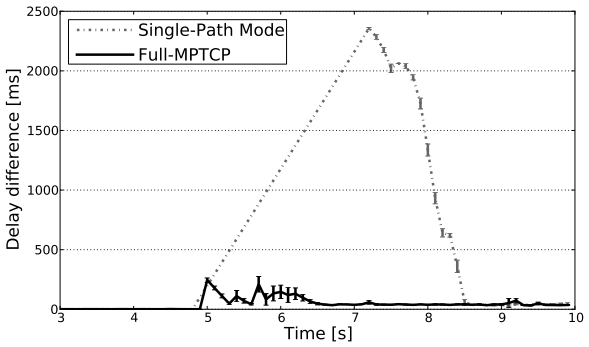
4.2 Application Delay

Variation in application delay is an important metric for streaming or VoIP applications. We measure the application delay by sending blocks of data, tagged with a timestamp. Upon reception of each block, we store the transmission timestamp together with the timestamp at the receiver-side. The evolution of the difference between these timestamps gives us the variation in application delay.

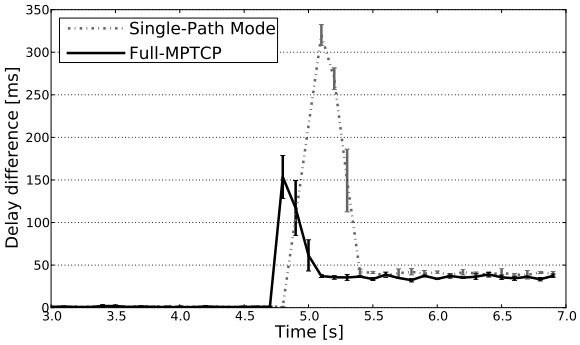
We performed measurements during which the server was transmitting at 500Kbps to the client. Initially WiFi and 3G are enabled and data segments are sent according to the specified handover mode. After around 5 seconds, the WiFi access-point is disabled and traffic has to switch over to the 3G interface.

Fig. 2a and 2b show the impact on the application delay when handing the traffic over from WiFi to 3G. The *Backup* mode is not shown since it is similar to *Full-MPTCP*. A sending rate of 500Kbps does not fill the pipe over the WiFi interface and since MPTCP prefers the WiFi path with its higher bandwidth and lower RTT, no traffic is sent over the 3G network, even in *Full-MPTCP* mode.

As in *Single-Path* mode no subflow is established over 3G prior to the vertical handover event, the 3G interface is idle and thus first needs to come up before establishing the new subflow. This takes up to 2 seconds and its impact on the application delay can be seen in Fig. 2a. When the 3G interface is forced to remain active, the impact of establishing the new subflow over *Single-Path* is less important as can be seen



(a) The 3G modem takes about two 2 sec. to switch from idle to operational-state.



(b) When forcing 3G to stay enabled, the impact of *Single-Path* is less important.

Figure 2: Evolution of the application delay.

in Fig. 2b. *Full-MPTCP* has a slight peak during handover due to two reasons: First, the client needs to announce the lost WiFi-interface with the *Remove Address* option. Second, because only a small number of data segments were sent through the 3G interface, its congestion window is not yet large enough to sustain a sending rate of 500Kbps. The application delay is higher during the slow-start-phase until the congestion window allows a steady rate of 500Kbps.

4.3 Impact on Existing Applications

Since MPTCP does not change the socket API, it can be used transparently by any TCP application. Skype is a commercial Voice over IP application that is able to operate over both TCP and UDP. We experiment with Skype—that has very tight constraints on packet-level delays—to showcase a worst-case for MPTCP handover. We do not imply that Skype should be run over TCP or MPTCP—it is much better to run Skype over unreliable transports such as RTP or UDP if possible.

For our experiment, we force Skype to pass through an MPTCP-enabled HTTP-proxy by blocking UDP and other TCP ports on the client's firewall. Otherwise Skype will by default try to use UDP or do regular TCP over the public Skype-servers. We use the Skype API to play a recorded file and record the received signal. Around the 7^{th} second we turn down the WiFi access point and MPTCP seamlessly performs the handover to the 3G interface without any impact on the application.

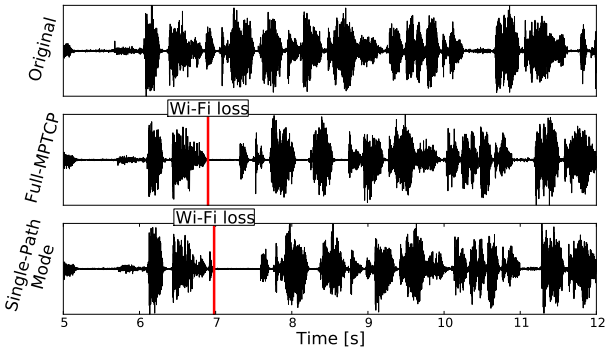


Figure 3: During the failover to 3G a short moment of silence happens during the Skype-call.

We compare the impact on the voice signal during handover from WiFi to 3G with the different handover modes. Fig. 3 shows the amplitudes of the original signal and the signals impacted by the handover. When using MPTCP in either handover modes, the Skype-session does not break and the call continues. A short moment of silence follows the loss of the WiFi interface. This is because MPTCP needs to detect the failure and then reinject the lost data segments in another subflow (*Full-MPTCP*) or in a newly established subflow (*Single-Path*). Hence, Skype plays the received voice signal with some additional delay but resynchronizes the call after some time.

4.4 Energy Consumption

Smartphones have a limited battery lifetime, it is therefore important to understand the impact of using MPTCP on energy consumption of a smartphone over WiFi and/or 3G and compare it to regular TCP. Early works have already shown the potential opportunities to improve a smartphone's energy consumption by using Multipath TCP [8, 9]. These evaluations were solely based on simulations; we fill this gap here.

We have ported our Linux Kernel implementation of MPTCP on a Nokia N950 smartphone and have been able to accurately measure its energy consumption thanks to a modified battery. We measure the total number of Joules consumed during a measurement period and divide it by the number of bits transmitted by the application. This allows us to take into account the impact of the MPTCP overhead on the energy consumption.

To evaluate MPTCP's impact on smartphone energy consumption we download a file from a public MPTCP-enabled web server by using either regular TCP over WiFi or 3G and by using MPTCP simultaneously over WiFi and 3G. We consider two representative scenarios [3]. Our first scenario is the repeated download of one file of 1MB. The second scenario is the repeated downloads of a small file (100KB) with 5 seconds idle time between each download. This two scenarios correspond to large file download and web browsing [3].

At very low speeds, we observed in the first scenario that regular TCP over WiFi consumes roughly half of the energy per bit compared to regular TCP over 3G (Fig. 4). This is because the WiFi interface of the Nokia N950 is able to quickly switch in energy-saving mode. If few packets are

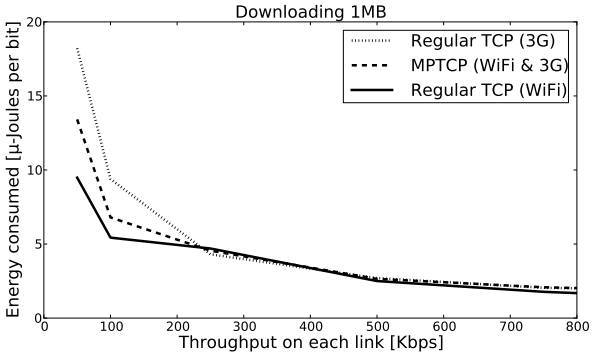


Figure 4: Especially on very slow links using the WiFi interface has a benefit.

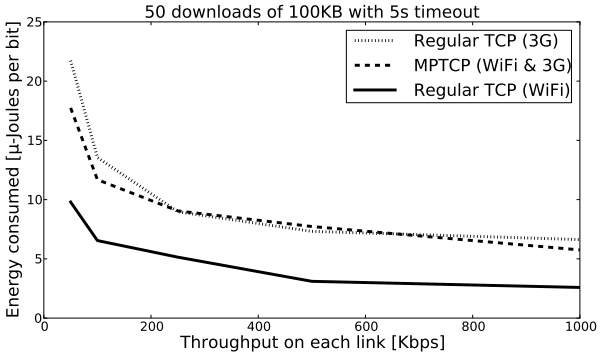


Figure 5: WiFi is much more flexible with respect to varying traffic characteristics.

generated, the WiFi interface is able to save energy between the individual packet bursts. On the other hand, the 3G interface is not able to quickly change its operational mode and thus its energy consumption is higher. As MPTCP uses both interfaces at the same time, the download finishes twice as fast as the other measurements, and thus its energy consumption lays exactly between WiFi-only and 3G-only.

The second scenario, shown in Fig. 5, highlights the impact of the tail-energy as described in [8]. When no more packets are passing on the 3G interface, unlike the WiFi interface, the 3G interface does not directly switch to energy-saving mode. On the Nokia N950 this process may take up to 7 seconds before the 3G interface goes into standby-mode. During these 7 seconds the interface continues consuming energy, the so-called *tail-energy*. As in the scenario reported in Fig. 5 the downloads are repeated every 5 seconds, the 3G interface is unable to switch into energy-saving mode and thus consumes more energy per bit. As MPTCP is using both interfaces it is also influenced by the downsides of 3G's tail-energy.

4.5 Summary of Experiments

We have seen that MPTCP allows unmodified applications such as Skype to continue to function while a handover from WiFi to 3G takes place. To our best knowledge, this is the first time such handover has been tested on real networks and with real applications.

Our energy evaluation on the Nokia N950 shows that using the 3G interface is costly in terms of battery-lifetime, so using *Full-MPTCP* mode is not the best option for users willing to save battery life[5]. On the other hand, *Full-MPTCP* mode offers the smoothest handover. Using the *Backup* mode brings modest performance improvements over *Single-Path* mode during handover when the connection is long enough that it allows the 3G interface to go to sleep. In such cases the *Single-Path* mode uses less energy and is thus preferable. If connections are shorter than 10 seconds and a handover happens, the *Backup* mode is preferable as it gives good performance at modest energy cost.

Given this initial exploration, an interesting question is, how mobile operating systems can optimize user experience and battery-lifetime simultaneously. One option is to always use *Full-MPTCP* mode when the user is actively using the phone (i.e. the mobile is in active mode). When the mobile is sleeping, it makes sense to use less energy hungry modes such as *Single-Path* mode.

5. IMPROVING MULTIPATH TCP

During our measurements we stumbled sometimes upon bad results. In the *Backup* mode the second subflow ended up unused after the first one was lost, and high application delays were observed in *Single-Path* mode. Both issues originated from the loss of the *Remove Address* option which is sent unreliably as explained in Sec. 2.1. In this section we discuss these results and show how we can address them by changing MPTCP to send the *Remove Address* option reliably.

Observations.

Losing the *Remove Address* option has an impact on MPTCP's performance. When receiving the *Remove Address* option, the peer closes the affected subflow and reinjects all unacknowledged data on the second subflow. However, if the *Remove Address* option is lost, the peer continues to send data on all existing subflows, i.e., in this case the first and second subflow. Reinjection of data from the affected subflow will only occur after a *Retransmission TimeOut* (RTO). On congested wireless networks the RTO can easily shoot up to a few seconds, which significantly affects the handover process. Furthermore, the initial subflow will remain active until the expiration of the maximum retransmission-timeout (between 13 and 30 minutes), using precious server resources.

We observed the impact on application delay when losing the *Remove Address* option. We found as expected that in the *Backup* mode, if the *Remove Address* option is lost, the server is not aware that it has to start using the backup-subflow and thus no data is ever received after losing the WiFi interface. If the *Remove Address* option is lost in *Full-MPTCP* mode, the server has to wait until the RTO on the subflow passing by the WiFi-interface. And only after this RTO the data will be reinjected over the second subflow. This increases the handover delay.

The current MPTCP specification does not include any mechanism to ensure a reliable delivery of the *Remove Address* and *Add Address* options. In our current implementation, we use one spare bit in these options that serves as an

[5]Of course, other mobile devices might have different energy consumptions, and the results might be different.

echo bit. We use it as follows. When a host loses an address it sends the *Remove Address* option in a duplicate ACK as described in Sec. 2.1 and starts its retransmission timer. The *Remove Address* option is also automatically added to each outgoing segment. When a host receives a segment containing the *Remove Address* option it then immediately replies with a duplicate ACK that contains the same option with the *echo* bit set. Upon reception of this ACK, the initiator of the *Remove Address* is now certain it was received. We implemented this design in our implementation which consisted of around hundred lines of code; all the experiments we have run use this optimization.

6. RELATED WORK

Over the last decade, multiple research efforts have been conducted to enable end-hosts to deploy mobility for applications in a transparent manner. At the different layers of the OSI-model, protocols such as the *Stream Control Protocol* (SCTP), Mobile IP and MIPv6 have been trying to allow a mobile node to seamlessly perform vertical handover.

Mobile IP (and it's IPv6 counterpart MIPv6) enables transparent mobility at the network-layer [7]. Previous work however has shown that mobility and multipath provides better performance when being plugged at the transport layer [13].

The addition of *Concurrent Multipath Transfer* (CMT) into SCTP [5] was the first step to allow SCTP to split traffic over multiple paths. mSCTP [12, 6] has been proposed to dynamically update a peer's address-list and thus allow handover from one interface to the other, and is similar in spirit to Multipath TCP mobility. Unfortunately SCTP has seen very little deployment, as most applications today still use TCP.

Snoeren et al. proposed Migrate TCP [11] that uses TCP options and DNS updates to migrate the endpoint of a TCP session to a different address; this could be readily used for 3G/WiFi offloading. As we have argued in [9], MPTCP is a better solution because it can use all the interfaces at once which provides increased performance and throughput.

Offloading 3G to WiFi is not a novel idea. Wiffler [2] presents a study of 3G and WiFi coverage while driving and shows how offload can be implemented by changing the applications. Our solution is more general as it enables unchanged applications to benefit from offloading.

7. CONCLUSION AND FURTHER WORK

We have carried out an experimental investigation of Multipath TCP in presence of WiFi and 3G. We have proposed and evaluated three handover modes : *Full-MPTCP*, *Backup* and *Single-Path*. Our experiments in commercial wireless networks demonstrate that MPTCP can play a role for mobile users and also WiFi/3G convergence today. Our measurements show that MPTCP can quickly recover from a WiFi loss in presence of a 3G interface with only a small impact on the application delay and goodput. Our experiments with Skype demonstrate that existing unmodified applications already benefit from MPTCP.

Our further work will be to investigate the performance of MPTCP in other mobility scenarios such as WiFi to WiFi, 3G to WiFi mobility as well as when both hosts are mobile.

Acknowledgement

This work was supported in part by the FP7 CHANGE project and a grant from Google, Inc. Part of the work of Christoph Paasch was performed while visiting Nokia Research as part of the FI SHOK program.

8. REFERENCES

[1] FON WIRELESS, Ltd. FON website available at http://www.fon.com, April 2012.

[2] A. Balasubramanian, R. Mahajan, and A. Venkataramani. Augmenting mobile 3G using WiFi. In *Proc. Mobisys*. ACM, 2010.

[3] H. Falaki, D. Lymberopoulos, R. Mahajan, S. Kandula, and D. Estrin. A First Look at Traffic on Smartphones. In *Internet Measurement Conference (IMC'10)*, pages 281–287, 2010.

[4] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure. TCP Extensions for Multipath Operation with Multiple Addresses. Internet-Draft draft-ietf-mptcp-multiaddressed-06, Jan. 2012.

[5] J. R. Iyengar, P. D. Amer, and R. R. Stewart. Concurrent Multipath Transfer using SCTP Multihoming over Independent End-to-End Paths. *IEEE/ACM Transactions on Networking*, 14(5):951–964, 2006.

[6] S. J. Koh, M. J. Chang, and M. Lee. mSCTP for Soft Handover in Transport Layer. *Communications Letters, IEEE*, 8(3):189 – 191, March 2004.

[7] R. Koodli. Mobile IPv6 Fast Handovers. RFC 5568, RFC Editor, July 2009.

[8] C. Pluntke, L. Eggert, and N. Kiukkonen. Saving Mobile Device Energy with Multipath TCP. In *ACM Workshop on MobiArch*, pages 1–6, 2011.

[9] C. Raiciu, D. Niculescu, M. Bagnulo, and M. Handley. Opportunistic mobility with multipath TCP. In *ACM Workshop on MobiArch*, pages 7–12, 2011.

[10] C. Raiciu, C. Paasch, S. Barre, A. Ford, M. Honda, F. Duchene, O. Bonaventure, and M. Handley. How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP. In *Networked Systems Design and Implementation (NSDI '12)*, 2012.

[11] A. Snoeren and H. Balakrishnan. An end-to-end approach to host mobility. In *Proc. Mobicom*, pages 155–166. ACM, 2000.

[12] R. Stewart, Q. Xie, M. Tuexen, S. Maruyama, and M. Kozuka. Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration. RFC 5061, RFC Editor, September 2007.

[13] D. Wischik, M. Handley, and M. B. Braun. The Resource Pooling Principle. *SIGCOMM Computer Communication Review*, 38(5):47–52, Sept. 2008.

[14] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley. Design, Implementation and Evaluation of Congestion Control for Multipath TCP. In *Networked Systems Design and Implementation (NSDI '11)*, 2011.