Opportunistic Mobility with Multipath TCP

Costin Raiciu
Universitatea Politehnica Bucureşti
costin.raiciu@cs.pub.ro

Dragos Niculescu
Universitatea Politehnica Bucureşti
dragos.niculescu@elcom.pub.ro

Marcelo Bagnulo
Universidad Carlos III Madrid
marcelo@it.uc3m.es

Mark Handley
University College London
m.handley@cs.ucl.ac.uk

ABSTRACT

Host mobility has traditionally been solved at the network layer, but even though Mobile IP has been standardised for 15 years, it hasn't been supported by operators. IP's double role as a location identifier and communication endpoint identifier brings a number of functional and performance problems.

We argue that the best place to handle mobility is at the transport layer. While this is not a new argument, we believe that the emerging standard of Multipath TCP (MPTCP) can be used to solve many issues related to mobility. MPTCP naturally implements *make-before-break*, can be incrementally deployed, is backwards compatible with TCP, and could even ease incremental adoption of IPv6.

Using simulations and indoor experiments with WiFi and 3G, we show that MPTCP gives better throughput, achieves smoother handoffs, and can be tuned to lower energy consumption.

Categories and Subject Descriptors

C.2.6 [**Computer-Communication Networks**]: Internetworking-Standards

General Terms

Design

Keywords

Multipath TCP, Mobility

1. INTRODUCTION

It has become commonplace for mobile devices such as smart phones and tablet PCs to have multiple radios such as WiFi, 3G and Bluetooth. With increasing integration and software-defined radio, we expect devices will support more radio technologies. Each radio technology has its pluses and minuses; 3G has near ubiquitous coverage, but in big cities it often suffers from sufficient congestion to be almost unusable; WiFi gives high transfer rates, but coverage is patchy and for mobile users is often transient. How can we best utilise all the radio links available, so as to get the best coverage and throughput, and use the least battery power while doing so?

Conventional solutions are crude; smart phones typically use one network at a time using a fixed policy such as "use WiFi if available, otherwise 3G". As WiFi and 3G networks each supply an IP address, transition between the two is disruptive, requiring applications to re-establish connectivity. Such simple policies work acceptably well if the user is not mobile. When in a train or car, WiFi connectivity is transient - often good connectivity is available but only for a few seconds[1]. Such policies are too disruptive to use when connectivity (or loss of connectivity) is transient.

Mobile IP [2] might in principle be used to hand off an ongoing connection from one network to another. But even 15 years after Mobile IP was standardised, it is still too rarely deployed for smartphone vendors to use. Worse, because of its very nature as an IP layer protocol, Mobile IP has no access to the information needed to perform optimally in *make-before-break* mobility events. In particular, without rewriting TCP, Mobile IP cannot stripe data between multiple "care-of" addresses for the same TCP connection because the different RTTs and bandwidths will confuse TCP and severely impact performance.

An ideal mobility model would not only be *make-before-break* (easily achieved with multiple radios), but would also maintain more than one active link for as long as is feasible. Thus a download progressing over 3G while on a train would not be transfered to WiFi, but would continue on 3G and add additional download capacity over WiFi whenever that is possible. The goal then is not to perform fast hand-off, but to perform as slow a hand-off as possible whenever both links remain usable. In order to do that, the mobility solution must have access to information about the different available paths, such as RTT and congestion information. These observations lead us to conclude that the IP layer is the wrong place in the stack to support mobility; in fact the transport layer is the lowest layer that has enough information to perform well.

Multipath TCP[3], as currently being standardised in the IETF, might be the mechanism to enable such a mobility model. MPTCP is a set of extensions to TCP that allow a pair of hosts to negotiate MPTCP use, and then to establish multiple parallel subflows using multiple IP addresses for a single connection. Each subflow has its own congestion window which is halved upon loss, but the increases are coupled across all subflows with the purpose of shifting traffic away from congested paths [4]. The sender stripes data to all the working subflows subflows in accordance with the available bandwidth on each path.

In a mobile scenario, an MPTCP connection can be established via any working network interface and IP address. Later, if con-

MobiArch'11, June 28, 2011, Bethesda, Maryland, USA.

nectivity can be achieved using another interface and IP address, a second subflow will be established, and data can now be transferred via both subflows. Later still, if connectivity is lost for one subflow, the remaining one can continue without interruption.

2. MOBILE MPTCP ARCHITECTURE

By far the most common scenario is the mobile host initiating a TCP connection to a server on the fixed Internet, issuing a request, and downloading data, examples including HTTP and IMAP. The proposed MPTCP architecture is optimised for this scenario.

The MPTCP mobility architecture has the following elements: the mobile host, an optional MPTCP proxy, and the remote host.

The *mobile host* supports the MPTCP extensions and is the device that changes its network attachment point, which results in it acquiring or losing IP addresses. The mobile host communicates via TCP with the *remote host*, which may or may not support MPTCP. If the remote host supports MPTCP, the connection can be established directly between the mobile host and the remote host. MPTCP is then used to handle interfaces or IP addresses coming and going in the middle of a connection. A single MPTCP connection can simultaneously communicate using both IPv4 and IPv6 addresses, so it is possible for a mobile host to move seamlessly between IPv4 and IPv6 networks.

If the remote host is not MPTCP capable, which is likely to be the case during the early stages of deployment, the MPTCP proxy comes into play. The MPTCP proxy is a fixed anchor point associated to the mobile host, which supports MPTCP. When the mobile host communicates with a remote host that does not support MPTCP, the mobile host establishes a MPTCP connection to the proxy which in turns establishes a regular TCP connection with the legacy remote host. Mobility of the mobile host is supported though the MPTCP connection with the MPTCP proxy, as in the previous case. The proxy also supports simultaneous movement.

There are additional scenarios that require additional mechanisms, most notably support for non-TCP communications, and for TCP connections incoming to the mobile host.

Dynamic DNS can be used to enable incoming connections, but in an IPv4 world, mobile hosts are almost always behind a NAT. Thus the pressing case for incoming connections is actually for peer-to-peer applications, and these generally encompass an out-of-band rendezvous mechanism followed by the use of simultaneous open from both ends. We will discuss this in more detail later.

The main non-TCP case to consider is for real-time traffic such as VoIP. The IETF is currently standardising multipath extensions to RTP [5] that play an analogous role to that performed by MPTCP for TCP connections.

In the following sections we describe the different scenarios and the details of the different components of the proposed architecture.

2.1 Client-Server Operation

The simplest and in the long run probably the most common scenario is communication initiated by an MPTCP mobile host with an MPTCP-capable remote host. In this case, communication flows directly between the mobile host and the remote host.

A connection is initiated by the mobile host performing the TCP 3-way handshake with the remote host. A TCP option in the SYN and SYN/ACK signals MPTCP capability of the endpoints as well as other relevant MPTCP information. Once the connection is established, there are two basic mobility events that can occur: a new interface is available, or an existing interface ceases to be available.

When a new interface becomes available and an IP address is acquired, if its use is allowed by the host's policy, the mobile host adds the new IP address to the established MPTCP connection. To do this it performs a new TCP 3-way handshake using the new IP address as source address. During the handshake, this new connection is identified as a subflow of the ongoing connection.

At this point, the MPTCP connection has two subflows using the original IP address and the new IP address of the mobile host. Each subflow runs its own congestion control mechanism, but these are coupled. A separate congestion window is maintained for each subflow and each performs slow start. However, in congestion avoidance mode the increment of the window is related to the sum of all the congestion windows of all the subflows. A congested subflow increases its window more slowly than an uncongested one, allowing MPTCP to move traffic away from congestion and allow optimal utilisation of network resources as described in [4]. In any case, the explicit management of the different subflows and their respective windows allows MPTCP to keep track of the data exchanged through each interface and perform flow control, congestion control, error detection and retransmissions accordingly. An additional connection-level sequence number is carried in data packets using TCP options, allowing the receiver to cope gracefully with reordering and delivery of data to the application in order. More details of the subflow operation of MPTCP is in [3].

If an interface on the mobile host goes down, all subflows associated with that interface stop transferring data. Any other subflows continue exchanging data and will retransmit any missing data that was in flight on the failed subflows. If no other interfaces are available, the mobile host waits until a new interface comes up, then instantiates a new subflow on the MPTCP connection, as described earlier, and resumes exchanging data.

A make-before-break handoff is basically the acquisition of a new interface/IP address while the current one is working, followed by the detachment of the original address. With MPTCP, the mobile host adds a new subflow to the ongoing MPTCP connection using the new IP address, and exchanges data though both subflows. When the first interface goes down, data is no longer sent through the initial subflow.

A break-before-make handoff is where all subflows were using a given interface, and that interface then detaches from the Internet. The MPTCP connection is still open, but is stalled. A new IP address then becomes available to the mobile host, whic is used to create a new subflow for the ongoing MPTCP connection. Data is then exchanged through the new subflow.

If the remote host supports MPTCP then this is a complete mobility solution. In the long run, we expect all the main operating systems to support MPTCP. In particular a high adoption of MPTCP from servers is expected due to the significant benefits that MPTCP provides for data centres [6]. However, in the meantime, few servers will support MPTCP. To allow mobility, a proxy solution is required.

2.2 MPTCP Proxies

The mobile host can be configured with the IP address of an MPTCP proxy server. In practice, wireless Internet providers may provide proxy service, and announce this via DHCP. The mobile host can then connect to the proxy server, and indicate to it via an MPTCP option the IP address of the remote host. The proxy then connects to the remote host, establishes the TCP connection, and then becomes a passive relay for the data. The mobile host can then initiate additional subflows to the proxy server as described earlier in the case of an MPTCP-capable remote host. The proxy stripes data to the mobile host across the subflows.

There are several key points to note about this use of a proxy. First, the proxy is application-agnostic; it neither knows nor cares what the application is. Similarly, TCP applications on both the

mobile host and remote host are unaware of the existence of the proxy - all the requests are handled by the mobile host's MPTCP stack. Finally, the proxy will attempt to negotiate MPTCP with the remote host. If this does in fact support MPTCP, the proxy will signal the remote host's IP address to the mobile host as an additional address for MPTCP to use. The mobile host can then establish a new subflow direct to the remote host, drop the proxied subflow, and remove the proxy from the connection. Thus, as remote hosts increasingly support MPTCP, traffic relayed by the proxies decreases.

2.2.1 MPTCP Proxy functionality

Although MPTCP proxies play a key role in incrementally deploying MPTCP for mobile use, the aim is that in the long run they should rarely need to relay traffic. If MPTCP were ubiquitous, and IPv6 were used so NAT traversal were not an issue[1], then the only requirement for a proxy would be to cope with simultaneous move for peer-to-peer connections. For the foreseeable future though, MPTCP proxies will be of importance. Just what should an MPTCP proxy do? There are two distinct modes, depending on whether the server negotiated MPTCP or not.

Case 1: MPTCP-capable Remote Host

If the remote host did negotiate MPTCP, the proxy serves as a packet-level pass-through. If the mobile host initiates a new subflow, the proxy relays the subflow request to the remote host. Thus there is a one-to-one mapping between mobile-host-to-proxy and proxy-to-remote-host subflows. This allows the proxy to simply pass packets through without queueing them, and allows congestion control and retransmissions to work end-to-end.

Case 2: Legacy Remote Host

If the remote host did not negotiate MPTCP, then there can be no subflows that bypass the proxy. This simplifies the situation, but at the same time it requires the proxy to function as a full proxy rather than a packet relay.

The proxy maintains its own congestion window when sending to the remote host, and for each subflow to the mobile host. Packets from the mobile host to the remote host must be reassembled in-order, before being sent to the remote host. Packets from remote host to mobile host do not need to be reassembled in-order - they can simply be relayed onto whichever of the active subflows has free space in its congestion window. The sequence numbers provided by the remote host become MPTCP data sequence numbers, and addition subflow sequence numbers keep track of congestion and retransmissions on each subflow.

2.3 Peer-to-peer Operation

Peer-to-peer operation on mobile devices is complicated by two factors: NATs often prevent direct connection, and simultaneous mobility can cause peers to lose contact.

The state of the art in peer-to-peer NAT traversal is ICE[7], which uses a combination of probing NAT behaviour and proxy techniques to enable connection establishment when both endpoints are behind NATs. In many cases though, connection establishment is not possible without a proxy, because many NATs do not allow TCP simultaneous open.

To enable mobility with MPTCP then, the use of MPTCP proxies is also desirable. Not only does this permit NAT traversal, but it provides at least one subflow with a stable address (albeit a proxied address) so that simultaneous move is not a problem. When a mobile host's interface comes up, a new subflow can be set up from the new address to the proxy, allowing immediate use of that interface. Simultaneously, the host can perform the ICE techniques, attempting to establish a direct path between the end hosts, avoiding the dogleg route through the proxy.

[1]One-to-one IPv6 NATs present no problem

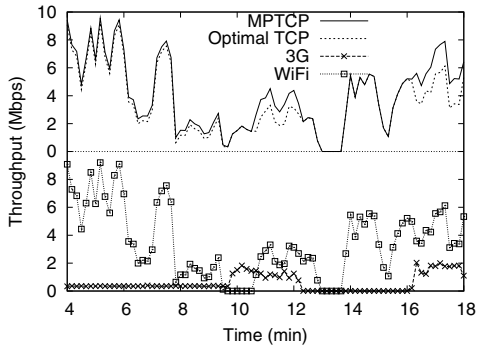

Fig. 1: Comparison of MPTCP with *Optimal TCP* single path strategy; indoor mobility experiment using laptop with WiFi and 3G dongle.

3. EVALUATION

Irrespective of whether a proxy is used, the benefits of MPTCP for mobile devices lie in the ability to overlap use of more than one radio. If mobility is hidden behind a single IP address, as with Mobile IP (also used in 4G networks), then the mobile host is faced with an all-or-nothing choice to switch to a different radio link. If connectivity were binary, then this would be sensible. One could then optimise for performance, cost, power consumption or some function of all three, based on predicted link performance. Unfortunately connectivity on wireless networks is far from binary - we consider here 3G and WiFi. What we really want is to probe the WiFi link without abandoning the 3G link. Obviously the best probe traffic is data traffic that is actually wanted; when it is received, this increases goodput; when it is not received it can be resent on the 3G link. MPTCP provides precisely this capability.

Once more than one subflow has been established, MPTCP must decide whether to use either a single link or more than one link simultaneously. If download performance were the only factor, using all links simultaneously is normally the right answer. On high-power devices such as a car computer or a laptop, using all links simultaneously will give the best user experience; both performance and robustness are improved. However, on energy constrained devices, the selection must also encompass power consumption considerations making the decision algorithm more complex.

3.1 Indoor Mobility Experiment

Fig. 1 shows the results of an indoor mobility experiment using a laptop equipped with WiFi and a 3G dongle. During this experiment we transfered data over both 3G and WiFi. The user moves about one floor of the building with poor 3G reception, walks down the stairs, and onto another floor with better 3G reception. While moving, the throughput achieved over both 3G and WiFi is quite variable, and there are periods where each loses reception completely. The bottom curves show the throughput on each link.

We wish to compare the performance of MPTCP with what single-path *Optimal TCP* would achieve if it could tell in advance which would be the better of 3G and WiFi. Although we have a full Linux MPTCP implementation, we do not have an *Optimal TCP* imple-

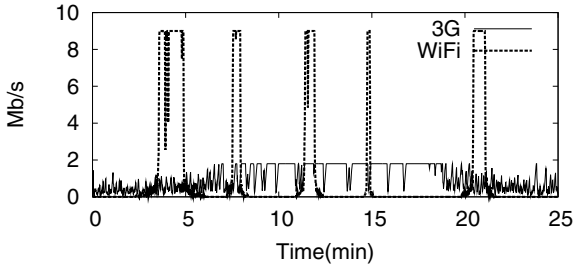

Fig. 2: Synthetic trace of link capacities for a walking scenario used in the simulation for 3G and WiFi interfaces. 3G has better coverage, but is lower capacity, shared with many subscribers. WiFi coverage is patchy, but may offer higher rate when close to the hotspot.

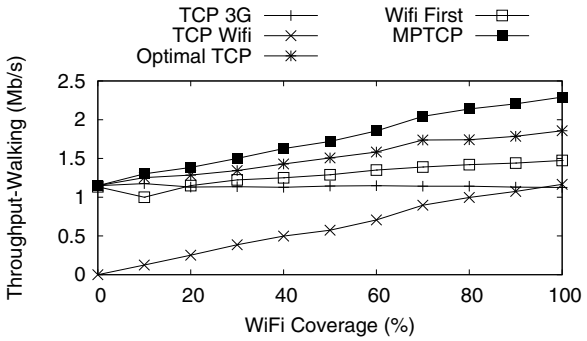

Fig. 3: Throughput comparison for a walking speed scenario.

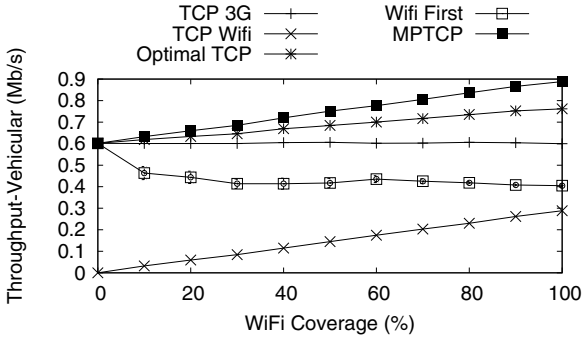

Fig. 4: Throughput comparison for a driving speed scenario.

mentation, nor is the mobility precisely reproducible, so we must resort to a trace-driven comparison. The top curves show the performance MPTCP would get, assuming the wireless link is the bottleneck. This is compared to the performance that a single-path *Optimal TCP* would get using a solution such as Mobile IP if it knew *in advance* which interface would be best for each ten-second period and incurred no switching costs. Over the duration of this trace, MPTCP transfers 622 MB and the *Optimal TCP* transfers 554 MB. MPTCP increases speed by around 12%.

In reality we cannot build an *Optimal TCP*, so any real-world single-path TCP solution is going to perform worse than this, and show greater benefits for MPTCP.

3.2 Outdoor Mobility Simulations

Using the MPTCP stack presented in [4], we simulated two scenarios involving 3G and WiFi. Each scenario uses parameters derived from existing measurement studies.

Based on a 60km/h vehicular scenario from the Boston area [8], we simulated WiFi running at 1Mbps and offering a mean connectivity time of 13 seconds. The reasons for using the lowest bitrate were that it gives the maximum range from the AP, and there is very little time for link rate adaptation to take place. This achieved an average TCP uplink of 240Kbps. A more recent study [9] obtained a downlink TCP throughput of about 280Kbps. These values are likely to improve with the advent of 802.11n, increased urban WiFi deployments, and pedestrian speeds. For 3G we simulated average downlink performance of 600Kbps as in [9], but with 100% coverage. Although these numbers are time and place specific, the precise values turn out to be unimportant to our conclusions.

We also simulated an outdoors walking scenario; the primary difference is that associations with WiFi last several minutes, and therefore the bitrate adaptation has the chance to exploit better signal conditions. For this case we assumed a maximum of 10Mb/s because hotspots are often limited by the DSL backhaul.

Figure 2 shows an extract from one of the traces for the outdoor walking scenario, and illustrates the variable nature of the 3G link and the transient but high-bandwidth associations with WiFi.

A key variable turns out to be the density of WiFi hotspots and open access points. We simulated the full range of WiFi coverage, from none up to ubiquitous. Figures 3 and 4 show how the overall throughput varies depending on WiFi coverage. We show the performance achieved using standard TCP with only one interface as *TCP 3G* and *TCP WiFi*, and compare them against three dynamic strategies: *Optimal TCP*; *WiFi First* – uses WiFi if it is available, otherwise it falls back to 3G; and *MPTCP* – runs the full MPTCP multipath algorithm and congestion control.

At walking speeds the *WiFi first* strategy performs fairly well, but at vehicular speeds it gets 30% less throughput than sticking with 3G; primarily this is due to the low link speed achievable in such transient encounters.

Unsurprisingly, both MPTCP and *Optimal TCP* significantly outperform *WiFi first* or choosing a single interface. MPTCP increases throughput by 50% to 100% depending on the scenario.

In all cases we examined, MPTCP also outperforms *Optimal TCP*, typically by 10-15%. While this difference is not huge, it is a lower bound: the gains will be higher in practice.

3.3 Power Consumption Simulations

Download performance is not the only factor driving mobile networking stacks, as today's smart phones are heavily optimised for low power consumption.

3G and WiFi have different energy consumption patterns [10, 11]. WiFi has a higher cost to manage the connection, but offers a power-save mode, and higher bitrates when close to the AP. [12] shows that the following factors have little or no effect over 802.11n power consumption: higher modulation schemes, channel bonding, transmit power control. MIMO2 and MIMO3 increase power consumption only by 1.3x and 1.7x respectively, thus the recommendation is to use the highest rate possible, racing to sleep mode.

TailEnder [11] finds that 2G and 3G are cheaper than WiFi, but waste 6–12 seconds (controlled by the network operator) in a high power state after the end of a transfer, which could be wasteful for short transactions. 3G power usage depends on the bitrate used, delivery rate, and signal strength. While a higher bitrate and a better delivery rate conform to the "race to sleep" philosophy of saving

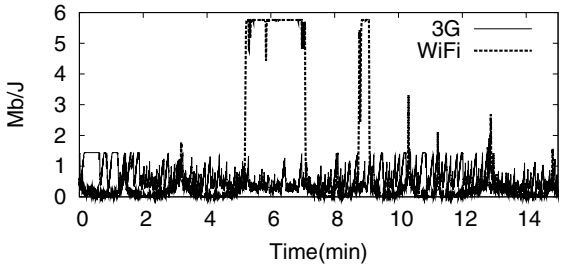

Fig. 5: Energy Efficiency for 3G and WiFi, in a simulated walking mobility scenario.

power, some phones turn up amplifiers in low signal situations [13], drawing up to 50% more power than with better SNR.

All these studies show that power cost of communication in Joules/bit varies widely, depending on technology, usage patterns and signal conditions, and so switching between interfaces depending on local conditions can bring significant power savings.

Based on measurements from existing studies, we simulated a simple energy model where WiFi draws 1.25W while receiving and 75mW while idle; we also simulated scanning and association costs while disconnected (280mW). Our 3G energy model has a ramp up energy of 2.5 Joules and the tail phase lasts 6s. While receiving, 3G draws 650mW at high SNR (close to the cell) and 1W at low SNR. 3G idle draw is 25mW.

Fig. 5 shows the results of simulating the power models described above, in a walking scenario with full coverage for both 3G and WIFI: most of the time 3G offers more bits per Joule; however when WiFi has good signal its efficiency is much higher.

If the user requests the mobile host to download a large file, which interface should it use? A sensible strategy is to use the most energy efficient interface; the device is willing to trade some download speed for increased battery life. Our strategy is to use MPTCP and start downloading using both interfaces for a probing period (10s) and then switch to the most efficient interface for a longer period (100s). After this period ends, another probing phase begins where both interfaces are used, and so on. We simulated this simple strategy together with the simpler strategies of using either 3G or WiFi for the whole file, and measured the overall efficiency.

Algorithm	TCP 3G	TCP WIFI	MPTCP 3G+Wifi	MPTCP Scheduler
Efficiency (Mb/J)	1.16	1.44	1.30	1.55
Throughput (Mb/s)	1.24	1.55	2.79	2.37

As expected, 3G is inefficient overall, WiFi is much better, and MPTCP using both interfaces achieves best throughput but poorer energy efficiency compared to WiFi alone. The simple scheduler increases efficiency by 7% compared to WiFi alone in this case, while also delivering 50% more throughput. We've also implemented a strategy that knows beforehand which interface will be better and switches to it. Surprisingly we found this performed worse than our simple scheduler described above; the reason turned to be the frequent switching back and forth, which created overheads on the 3G link (for ramp up and with the tail energy). When we applied damping the omniscient strategy had similar performance to our simple scheduler.

We've also run the same scheduler on the driving scenario; there WiFi is only briefly available (13s), and is rarely more efficient than 3G. Because of this and the tail energy cost of 3G, it makes no sense to turn 3G off in this scenario. The simple scheduler does not take into account tail energy, and gives worse power consumption than just using 3G. A smarter scheduler could use the mean WiFi association time to decide whether to turn 3G off. Even if the WiFi association time were too small, it might be sensible to also use WiFi when it has higher efficiency. In any event, if vehicular power is available, power saving is of little relevance and both radios should be used.

4. RELATED WORK

Handling mobility at the transport layer was proposed by Snoeren [14], who used TCP options and DNS updates to migrate the ends of a TCP connection to different IP addresses. We also believe that, given the double role of IP as a host identifier, and location identifier, the best place to handle mobility is at the transport layer. As Mobile IP, this solution only allows TCP to use one interface at a time, whereas multipath TCP allows splitting one flow over multiple interfaces. Additionally, MPTCP has an advanced standardisation status and allows incremental deployment.

Combining WiFi and 2G/3G is a well explored subject due to the complementary natures of these two technologies with respect to coverage, speed, and price. Wiffler [9] finds that 3G and WiFi have availabilities of 87% and 11% respectively, but are negatively correlated, resulting in an overall coverage of 96%. Whiffler proposes a system that offloads traffic to WiFi when it is available. MPTCP is a general solution to mobility, and can also be used to offload traffic to WiFi if needed.

Regarding the power consumption of different interfaces, [10] aims at discovering WiFi access to complement the more energy hungry 2G. While the J/bit cost is an order of magnitude higher for 2G, they find the cost of maintaining the connection is lower than for the WiFi. All the mentioned scenarios are naturally exploited by MPTCP with its *make-before-break* model and natural striping of traffic across multiple interfaces.

5. CONCLUSIONS

We proposed a mobility architecture based on MPTCP that can naturally shield the application layer from lower layer handoff implementations, IP address change, and all the issues related to multiple interfaces and carriers. Use of proxies can help with incremental deployment of MPTCP, and even with a mix of IPv4 and IPv6 connections.

We found that MPTCP brings advantages to mobile scenarios in terms of both functionality and performance. Simulations and experiments using 3G and WiFi have shown that MPTCP is able to harvest available bandwidth from the existing interfaces even in highly variable scenarios. Even when compared to an ideal Mobile IP implementation, MPTCP does better as it is able to use both interfaces in parallel. For the cases when battery life is more important, MPTCP can be tuned to find and use the less power-hungry interface.

Acknowledgements

The authors would like to thank the members of the Trilogy project for their insights and help. This work was partly funded by Trilogy, a research project funded by the European Comission in its Seventh Framework program, and by POSDRU/89/1.5/S/62557.

6. REFERENCES

[1] Jörg Ott and Dirk Kutscher. Exploiting regular hot-spots for drive-thru internet. In Paul Müller, Reinhard Gotzhein, and

Jens B. Schmitt, editors, *KiVS*, Informatik Aktuell. Springer, 2005.

[2] C. Perkins. IP Mobility Support for IPv4, Revised. RFC 5944 (Proposed Standard), November 2010.

[3] A. Ford, C. Raiciu, and M. Handley. TCP Extensions for Multipath Operation with Multiple Addresses. Internet-draft, IETF, 2011.

[4] Damon Wischik, Costin Raiciu, Adam Greenhalgh, and Mark Handley. Design, implementation and evaluation of congestion control for multipath tcp. In *Proc. Usenix NSDI 2011*.

[5] V. Singh, T. Karkkainen, J. Ott, S. Ahsan, and L. Eggert. Multipath RTP. Internet draft, IETF, 2011.

[6] Costin Raiciu, Christopher Pluntke, Sebastien Barre, Adam Greenhalgh, Damon Wischik, and Mark Handley. Data center networking with multipath tcp. In *Proc. Hotnets*. ACM, 2010.

[7] J. Rosenberg. Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols. RFC 5245 (Proposed Standard), April 2010.

[8] Vladimir Bychkovsky, Bret Hull, Allen Miu, Hari Balakrishnan, and Samuel Madden. A measurement study of vehicular internet access using in situ wi-fi networks. In *Proc. Mobicom*. ACM, 2006.

[9] Aruna Balasubramanian, Ratul Mahajan, and Arun Venkataramani. Augmenting mobile 3g using wifi. In *Proc. Mobisys*. ACM, 2010.

[10] Ahmad Rahmati and Lin Zhong. Context-for-wireless: context-sensitive energy-efficient wireless data transfer. In *Proc. Mobisys*. ACM, 2007.

[11] Niranjan Balasubramanian, Aruna Balasubramanian, and Arun Venkataramani. Energy consumption in mobile phones: a measurement study and implications for network applications. In *Proc. IMC*. ACM, 2009.

[12] Daniel Halperin, Ben Greensteiny, Anmol Shethy, and David Wetherall. Demystifying 802.11n power consumption. In *Proc. HotPower*. USENIX Association, 2010.

[13] Aaron Schulman, Vishnu Navda, Ramachandran Ramjee, Neil Spring, Pralhad Deshpande, Calvin Grunewald, Kamal Jain, and Venkata N. Padmanabhan. Bartendr: a practical approach to energy-aware cellular data scheduling. In *Proc. Mobicom*. ACM, 2010.

[14] Alex C. Snoeren and Hari Balakrishnan. An end-to-end approach to host mobility. In *Proc. Mobicom*. ACM, 2000.