ABSTRACT

Multipath TCP is a backwards-compatible TCP extension that enables using multiple network paths between two end systems for a single TCP connection, increasing performance and reliability. It can also be used to “shift” active connections from one network path to another without breakage. This feature is especially attractive on mobile devices with multiple radio interfaces, because it can be used to continuously shift active connections to the most energy-efficient network path. This paper describes a novel method for deriving such a multipath scheduler using MPTCP that maximises energy savings. Based on energy models for the different radio interfaces as well as a continuously accumulated communication history of the device user, we compute schedulers for different applications by solving a Markov decision process offline. We evaluate these schedulers for a large number of random application models and selected realistic applications derived from measurements. Evaluations based on energy models for a mobile device with Wifi and 3G radio interfaces show that it performs comparably in terms of energy efficiency to a theoretically optimal omniscient oracle scheduler.

Categories and Subject Descriptors
C.2.1 [Computer-Communication Networks]: Network Architecture and Design—Wireless communication

General Terms
Algorithms, Design, Measurement, Performance

Keywords
Scheduling, Energy-efficiency, MPTCP, Multipath TCP

1. INTRODUCTION

Today, mobile devices are typically equipped with several radio interfaces, such as Wifi and 3G. When multiple interfaces provide connectivity to the Internet, a device is said to be “multi-homed”, i.e., it has several IP addresses and connects to the Internet topology in different topological locations at the same time. When initiating a communication session to another Internet host, a multi-homed device can consequently choose between several paths to that destination IP address.

At first glance, multi-homing appears attractive because it enables communication flexibility. Unfortunately, neither of the two predominant Internet transport protocols (TCP and UDP) support multi-homing well: once a communication session using these protocols has been established over one path, i.e., between one local IP address and one remote IP address, that session is “stuck” to that path until it finishes. It is not possible to “shift” an active communication session to a different path. In addition, if the local or remote IP address becomes unavailable, e.g., because an interface loses connectivity, all sessions over that network path are terminated. It is therefore difficult to exploit the flexibility offered by multi-homing with existing protocols. Protocol extensions such as MobileIP [5] or HIP [4] try to address this problem at the network layer, but have failed to see adoption.

This situation is especially problematic for mobile, battery-powered devices. Different radio interfaces differ significantly in terms of energy consumption, transmission capacity, delay and range, as well as usage cost and reliability. The overall mobile user experience could be greatly enhanced if it was possible to seamlessly leverage the flexibility afforded by these different radio technologies for Internet communication. This is not straightforward: a device can naively try to always use the network path through the “best” radio interface based on its current situation, but especially for longer-lived sessions, that path is unlikely to stay “the best” as time passes. Over time, a device will end up with active sessions across multiple radio interfaces, which is highly undesirable – it wastes energy to keep a radio ready for transmission or reception especially under light communication loads, even more so for multiple radios. A naïve policy that always prefers to use a path via a radio interface that is already in use by other sessions is also problematic, because a session may be established over an inferior path, e.g., one with less capacity.

The root cause of these issues is the lack of support for multi-homing in the traditional Internet transport protocols. If it were possible to shift active sessions between paths, mobile devices could much more aggressively re-map their set of active connections to their set of available radio interfaces. This becomes possible with Multipath TCP (MPTCP), a backwards-compatible extension to TCP currently undergoing standardization in the IETF [2,8]. The main goal of MPTCP is to enable the concurrent use of multiple network paths between two end systems for a single TCP connection, but it can also be used to shift active connections from one path to another.

In order to maximise battery life time, it is important to realise...
2. SCHEDULING ARCHITECTURE

This section describes the scheduler architecture in detail. It consists of three components: First, an MPTCP-capable mobile device, so that multipath transfers can be used with MPTCP-capable servers. Second, an MPTCP-capable proxy that enables the mobile device to use MPTCP for connections to standard TCP servers on the Internet. And third, an MPTCP scheduler on the mobile device that is responsible for controlling interface switching.

that the per-bit energy cost for radio interfaces varies with load [3]. In some devices, Wifi is very energy-efficient for the low throughput range while 3G is more efficient if the throughput is high. However, even on a multi-homed device that is able to seamlessly shift connections, multipath scheduling is not straightforward. Consider for example a data transmission with varying throughput. Let the energy model of the data interfaces consist of three states: transmission energy if data is being actively transmitted, high stand-by energy after transmission has ended, and low idle energy after the stand-by phase is over. In general, the stand-by phase is very short for Wifi and considerably longer for 3G, whereas the energy consumption for the idle phases of 3G and Wifi are similar. If we greedily shift the data stream to the radio interface that would be most energy-efficient for a given load, we do not take the timeout behaviour of radio interfaces into account. Assuming that throughput continuously oscillates between high and low, this policy alternates between 3G and Wifi to exploit the interface with the best energy to bit ratio for transmission. However, every time it switches from 3G to Wifi, the stand-by energy for 3G adds to the transmission energy for Wifi and vice versa. If the changes in throughput oscillate very quickly, this policy can easily waste more energy for stand-by, compared to using even the least energy-efficient interface for the entire transfer. It is therefore not possible to simply shift all sessions to the radio interface that would be most energy-efficient if fully utilised, without paying attention to the actual throughput rates of those sessions and detailed energy models for the interfaces. In other words, we know that it is sometimes more energy-efficient to use one interface, sometimes it is better to use another interface, but how can we optimise MPTCP’s flexible handover of flows between interfaces to maximise those savings?

This paper demonstrates that it is possible to use MPTCP to significantly reduce the energy cost of mobile Internet communication. It proposes a method for generating multipath schedulers by formalising them as Markov decision processes (MDPs). One set of necessary inputs to this process are energy models for the different radio interface technologies which are described as finite state machines and parameterised with on-device measurements. A second set of inputs are application models which are parameterised by continuously monitoring a user’s Internet communication sessions.

Generating these schedulers is computationally expensive. Therefore, the required sets of input data are uploaded from a user’s device to a cloud service, where the scheduler is being derived. Whenever a personalised scheduler has been generated, it is downloaded to the user’s device and takes over MPTCP optimisation from the currently running scheduler instance.

In order to use MPTCP, both end systems involved in a communication session need to support it. This can pose a deployment problem, because it could be more difficult and less beneficial to deploy MPTCP in datacenters or other content servers. When MPTCP is not available end-to-end, ISPs or other third parties that have an interest in improving the Internet connectivity of mobile devices can deploy proxies that translate between singlepath TCP and MPTCP.

2.1 Description of the Architecture

MPTCP is currently being standardised by the IETF [2] and a prototype kernel implementation for Linux is available [11]. MPTCP extends standard TCP so that a single connection between two hosts can transmit segments over multiple paths simultaneously. Each “subflow” of segments along a given path either uses standard TCP congestion control or a coupled congestion controller [8], which links the increase and decrease functions of all subflow congestion windows. The coupled congestion controller improves fairness when several subflows share a joint bottleneck. However, when using MPTCP to shift active connections between the different radio interfaces on mobile devices, only a single subflow will be active most of the time, so either congestion control variant should be suitable. Applications continue to use the standard POSIX socket API and do not need to be modified or even recompiled to use MPTCP once it is enabled in the kernel. Because our scheduler extends only MPTCP, its use is also completely transparent to the applications.

Our energy-efficient scheduler dynamically decides which path, i.e., which radio interface, to use for the current set of active connections. This decision is made every $\Delta t$ time units. It is based on the device’s current state – in our case, this is the measured throughput in the last $\Delta t$ time units and energy state of all interfaces. The scheduler simply looks up the best interface in a table of scheduling rules of the form (state) $\rightarrow$ (action). The table of scheduling rules is precomputed and loaded onto the mobile device which then monitors the state of its interfaces.

Based on this scheduling decision, it initiates the necessary MPTCP protocol actions to shift the affected subflows. Interfaces are identified by IP addresses and paths are identified by IP address pairs. For all active connections, MPTCP will consequently establish new subflows via the new preferred interface and then tear down all subflows over all other interfaces. New subflows are established using the standard SYN, SYN/ACK, ACK three-way handshake. The MPTCP “Join” option specifies which connection it will be part of. To remove a subflow, the “remove address” MPTCP option is used, and the subflow is closed after a standard FIN, FIN/ACK exchange. This procedure is transparent to the application, i.e., the connection remains active, although throughput and RTT can change.

Since very few MPTCP-enabled servers exist on the Internet at the moment, we use a proxy to translate MPTCP communication from and to the mobile device to standard TCP which is supported in the rest of the Internet. The multipath scheduler on the mobile device thus moves flows between interfaces by adding and deleting subflows to the proxy server.

Due to the necessary MPTCP signalling, it takes two RTTs to shift a subflow to a new path, and it takes one RTT to tear down a subflow over a path that is no longer preferred. Because proxies are envisioned to be deployed by mobile ISPs, they are likely to be topologically close to the network edge. This means that the RTTs needed for signalling will be low, resulting in only minor and brief shifting delays.

2.2 Energy-Optimal Scheduling: an Optimisation Problem

Next, we show how to design an online scheduler based on MDP modelling. This scheduler takes interface energy models described as finite state machines (FSMs) and application throughput models for each interface into account when periodically choosing a single interface for transmission. In order to be able to compare its efficiency, we also give a description of an omniscient oracle scheduler that has perfect knowledge over the entire future of the
process and serves as a lower bound. To make it clear that we refer to a scheduling algorithm, we use capital letters, e.g., 3G, WIFI, MDP, or ORACLE.

2.2.1 Energy and Application Model

The first task before deriving an MDP scheduler is to find FSM representations of the energy consumption for all interfaces. Data interfaces usually operate with algorithms that can be easily cast as finite state machines. Figure 2 shows the example FSMs for 3G and Wifi interfaces that were used for the evaluation in Section 3.1. In general, the states of the FSMs can have different energy consumption that itself might depend on the throughput achieved in the state. Timeouts between states – if throughput has remained low for long enough – are common in interface FSMs. The FSMs for each interface are then translated individually into energy models the MDP optimiser can work with. They are also FSMs, but have special structure: In the energy model, state transitions may only be made every Δt time units (in our practical evaluations, we used Δt = 1s). Hence, energy consumption in each state has to be averaged over a time period of Δt. If Δt is chosen small enough, the resulting approximation is close to the actual energy consumption. Timeouts in Δt granularity can be taken into account by adding extra states.

The second step for the scheduler derivation is to generate an application model for a traffic trace from a specific application or user activity. To make the problem tractable for modelling, we use two assumptions: First, we model throughput in each interface as a random variable and apply a Markov approximation, i.e., we assume that the future of the throughput process only depends on the last observed throughput value. The application model depends on the transmitting interface and even on different states in the FSM description of the interface energy model (since different states might have different capacity limits). In general, the energy and application models can be conditioned on major factors influencing the interface energy consumption and throughput behaviour, such as signal strength or channel quality.

2.2.2 Online Scheduling with Incomplete Information – The MDP Scheduler

The online scheduler running on the mobile device has to decide every Δt time units which interface to choose for transmission, in order to minimise the average energy consumption for an application and energy model. MDP theory provides us with tools to efficiently solve for the optimal scheduling rules, assuming the FSM representation of the interface energy model is correct and the Markov approximation for the application model holds.

Formally, an MDP is a 4-tuple \((S, A, P, C)\). \(S\) is a set of states. We model a state \(s\) for the MPTCP scheduler with \(k\) interfaces as a \(k + 1\)-tuple: the first \(k\) state variables represent the energy state of its \(k\) interfaces. Their domain is equal to the set of states of the FSM energy model description for the corresponding interface. The \((k + 1)\)th state variable is the observed throughput during the last \(\Delta t\) time units on the active interface. Its domain is determined by the chosen granularity of transmission speed and the maximum transmission speed set for the application model.

\(A\) is a set of actions. An action or, in our context, a scheduling rule specifies which interface to use exclusively for the next \(\Delta t\) time units. For a device with \(k\) interfaces, it is described as \(a \in \{1, ..., k\}\). The interface identified by \(a\) is chosen as the unique transmitting interface for the next \(\Delta t\) time units.

\(P(s_t|s_{t-1}, a)\) are transition probabilities from state \(s_{t-1}\) at time \(t - 1\) to state \(s_t\) at time \(t\) when taking action \(a \in A\) at time \(t - 1\). In our case, the transition probabilities between different states under a given action are derived from the application and energy models. Therefore, discard all state transition for which one of the energy models for an interface has an invalid state transition according to its energy model. For all remaining transitions, use the throughput transition probabilities from the application model as the overall transition probabilities for the MDP.

\(C(s_{t-1}, s_t, a)\) with \(s_{t-1}, s_t \in S\) and \(a \in A\) is a cost function specifying the cost incurred for reaching state \(s_t\) at time \(t\) after taking action \(a\) in state \(s_{t-1}\) at time \(t - 1\). In our case, this is the sum of the energy consumptions for all interfaces, which can be read off the energy model for all interfaces given the state of their energy model and the incurred throughput.

The global optimisation objective is: minimise the expected sum of discounted costs over an infinite time horizon, i.e.,

\[
\text{minimise } \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t C(s_{t-1}, s_t, \pi(s_{t-1})) | \pi \right]
\]

over all scheduling policies \(\pi : S \rightarrow A\) where \(0 < \gamma < 1\) is a so-called discount factor regulating the time horizon taken into account by the scheduler – the time horizon goes to \(\infty\) as \(\gamma \rightarrow 1\). For the optimisation of average cost, \(\gamma\) should be chosen close to 1. This corresponds to minimising the average energy consumption. The solution to this MDP can be derived by running, e.g., the well-known value iteration or policy iteration algorithms.

In general, energy consumption alone is not all that matters. Soft constraints can be included in the modelling by adding penalty terms for each soft constraint into the cost function. Hard constraints are satisfied by preselecting the interfaces for transmission that are able to meet them. Minimising energy per bit instead of total energy consumption can be covered by adding a battery level state where empty battery is a terminal state. Each state transition reduces the battery level according to the throughput. The optimisation objective becomes maximise throughput.

2.2.3 Offline Scheduling with Perfect Information – The Omniscient ORACLE

In this section, we introduce an omniscient offline scheduler that
serves as a lower bound and a guidance for comparison. The offline scheduler knows the entire future of the process – this is why an omniscient scheduler is traditionally called an oracle. Assuming that there is only one throughput process governing the throughput of all interfaces, we can describe the incurred throughput at all times independently of the transmitting interface as a sequence of numbers. For this sequence, we can work out what the best scheduling is.

We now show how to compute the oracle scheduler for a fixed finite sequence of throughputs. States in the description of ORACLE are tuples of the form (energy state of interface 1, ..., energy state of interface k, throughput, scheduling time slot). An edge between states of neighbouring scheduling time slots exist if and only if at time t state (state of interface 1, ..., state of interface k, throughput, t + 1) can be reached by any state at time t with an action from the action set given the throughput seen in time instant t. We assign weights to all edges according to the energy consumption the action corresponding to the action the edge stands for causes given the throughput of the successor state the edge points to. Let s0 be the initial state of the interfaces and St the set of all possible states for the last time instant. We add an additional final state sf with edges of weight 0 to all states in St. The problem of finding the optimal scheduler with perfect information now reduces to finding the shortest path between s0 and sf. The scheduling decisions are given by the actions corresponding to the edges along the shortest path.

3. PERFORMANCE EVALUATION

So far, we have introduced two scheduling algorithms, MDP and ORACLE. Since we would like to compare them to manual scheduling (because that is what is in use today), we add single interface policies to the comparison that are not allowed to change the interface. In the evaluation, we will specifically use 3G and WIFI. Let El be the average energy consumption of scheduling algorithm a over an infinite time horizon for a fixed application model. Furthermore, let the discount factor γ → 1 be for MDP and assume the Markov approximation of the application model holds. Then, optimisation theory gives us the following guarantee:

\[ E_{ORACLE} \leq E_{MDP} \leq \min(E_{3G}, E_{WIFI}) \] (1)

If the first inequality did not hold, the rules derived by the MDP solver could be used to find a shorter path in the state space of ORACLE. Since the path between s0 and sf is already a shortest path, this produces a contradiction. The second inequality holds because a single interface policy can be described with action rules of the form (any state) → (fixed interface). Since MDP optimises energy consumption over all policies that can be described as (state) → (interface) rules, single interface policies are contained in the feasible set of policies MDP is optimising over. The following evaluation explores the performance gap between these policies.

First, we describe the modelling assumptions for the evaluation in Section 3.1. After that, we test the robustness of MDP for realistic application models in Section 3.2 and for random traffic models in Section 3.3.

3.1 Energy and Interface Model

For this evaluation, we use the 3G and Wifi state space model shown in Figure 2. The 3G interface can send data with high speed in dedicated channel mode (DCH) and with low speed in forward access channel mode (FACH). Since the interface always uses DCH even for low-rate data transmission in our measurements, we model this by restricting data transmission to DCH. Wifi only has two

states: it can transmit or be idle. For our energy model, the functions relating throughput \( x \) [kB/s] and energy consumption [mW] in transmission are:

\[ f_{3G}(x) = 915.096 + 0.352x \] (2)

\[ f_{WIFI}(x) = 873.438 + 0.532x \] (3)

This implies that 3G energy consumption is lower than Wifi for high data rates. However, 3G incurs a DCH timeout, \( t_{DCH} \), leading to a energy consumption of 881.79mW for 5 seconds and a FACH timeout, \( t_{FACH} \), consuming 438.9mW for 2 seconds after transmission has stopped, which prevents agile switching. \( t_{startup} \) is short in general. We used \( t_{startup} = 0 \). Note that DCH and FACH timeouts are set by the 3G operator. Wifi startup and timeout times were less than 100ms, which we approximated by using no startup and timeout delay for Wifi.

We set the time \( \Delta t \) between scheduling decisions to 1 second and restrict the application model of the MDP scheduler to throughput during the last second. Hence, we supply it with transition probabilities for throughput in the last second and throughput in the next second. The optimisation objective is to minimise energy consumption without side constraints over an infinite time horizon. We assume that the stochastic process governing the throughput for all interfaces does not depend on the transmitting interface. This makes the optimisation objective equivalent to \( \text{minimising energy per bit} \). The parameter \( \gamma \) has been set to 0.9, which provided a good trade-off between computation time and quality of the scheduler. Generating the scheduler on a laptop with an Intel Core 2 Duo 1.3GHz processor using one CPU for the computation task takes less than 5 minutes using the value iteration algorithm. The time to generate the scheduler does not depend on the amount of captured data that is used to generate the application model. It mainly depends on the discount factor \( \gamma \) and on the size of the overall state space, i.e. the size of the state space of the energy models and the granularity of the application model. More powerful algorithms for solving MDPs than value iteration are available that can further reduce the computation time.

3.2 MDP Closely Matches ORACLE for Realistic Application Models

We first explore the performance of MDP for realistic traffic. Therefore, we collected traffic traces on a Nokia N900 for four characteristic applications: Low quality (LQ) and high quality (HQ) video streaming, application download and mostly text-based browsing. All traffic traces were collected on Wifi. For high quality video, the lack of processing power for rendering the content on the mobile device was a major bottleneck. To be able to evaluate this
Table 1: Average interface energy consumption for running the schedulers for four example application models generated from measured traffic traces.

<table>
<thead>
<tr>
<th>Application</th>
<th>3G</th>
<th>WIFI</th>
<th>MDP</th>
<th>ORACLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>HQ Stream</td>
<td>708.7mW</td>
<td>604.0mW</td>
<td>545.6mW</td>
<td>533.9mW</td>
</tr>
<tr>
<td>LQ Stream</td>
<td>753.0mW</td>
<td>367.7mW</td>
<td>367.7mW</td>
<td>367.7mW</td>
</tr>
<tr>
<td>Download</td>
<td>204.5mW</td>
<td>163.9mW</td>
<td>163.9mW</td>
<td>161.4mW</td>
</tr>
<tr>
<td>Browsing</td>
<td>327.5mW</td>
<td>139.5mW</td>
<td>139.5mW</td>
<td>139.5mW</td>
</tr>
</tbody>
</table>

Figure 3: ECDF illustrating the deviation, D, in energy consumption from ORACLE for random application models using 3G, WIFI, and MDP.

high-throughput case without secondary effects, the traffic traces were taken on a laptop computer on the same Wifi network and the energy model of the mobile device was applied to the traces. All other traces were directly collected on the mobile device.

Next, we calculated the application model for each application and generated the MDP schedulers. Figure 1 shows a sample run of the MDP scheduler for HQ video streaming. The scheduler combines the energy efficient areas of 3G throughput and Wifi throughput most of the time. Notice the brief spikes in energy consumption that are a result of the online nature of the MDP scheduler. High throughput for a short time is used as an indication that video streaming has begun. ORACLE does not show these spikes, since it knows that it is more efficient to switch to 3G in advance.

We now make the realistic performance gap between 3G, WIFI, MDP, and ORACLE concrete. Table 1 shows the average energy consumption for 3G, WIFI, MDP, and ORACLE for each application model. In comparison to 3G and Wifi, MDP either matches or improves over the best performing interface in terms of energy consumption. The improvement for HQ streaming in comparison to the best-performing interface is 9.7%. This is aligned with our expectation that long-running high-throughput flows enable the scheduler to shift traffic to the more energy efficient 3G interface for that throughput area. However, the main point is that MDP always keeps the gap to the omniscient ORACLE very small. MDP guarantees performance benefits, if they are achievable by any scheduling policy at all, and makes it possible to give performance guarantees for automatically-derived scheduling policies based, e.g., on user-uploaded traces. Savings mainly depend on the hardware. That makes more diversity in terms of throughput dependent efficiency and timeout behaviour for different interfaces attractive.

3.3 MDP is Robust for Random Application Models

In order to study more than selected application models generated from measurements, we now explore the robustness of 3G, WIFI, and MDP for a large number of random application models as a benchmark. When we collected the realistic application traces, we noticed that given the throughput in the last second, the throughput in the next second could be closely approximated by an appropriately parameterised, discretised Beta distribution, if we excluded the case of dropping back to 0kB/s. This led us to a random application model that imitates this behaviour: Our random application model uses $2n + 1$ randomly-chosen parameters, where $n$ is the number of throughput levels in the application model: the first parameter is $p_0$, which controls the burstiness of the application and it is chosen uniformly at random. $p_0$ close to 1 will result in very bursty applications, whereas $p_0$ close to 0 will result in an application that is unlikely to stop transmission. The transition probabilities given the throughput of the last $\Delta t$ time units, $P$ (throughput in the last $\Delta t$ time units, throughput in the next $\Delta t$ time units, action), is formed by combining $p_0$ with a discretised Beta distribution for the non-zero throughputs. $P(x, 0, \cdot) \in [0, p_0]$ is chosen uniformly at random. $P(x, y, \cdot)$ where for $y > 0$ and fixed $x$ is a discretised Beta distribution with parameters $\alpha$ and $\beta$ scaled down by factor $1 - P(x, 0, \cdot)$. $\alpha$ and $\beta$ are distributed uniformly at random in $[0, 10]$ and chosen independently for every $x$.

We generated a large number of random application models and compared the performance of 3G, WIFI, and MDP to the ORACLE policy. Every application was evaluated by measuring the long-term average energy consumption for all algorithms. All schedulers transmit the same number of bytes in total. The deviation, $D$, of a scheduling algorithm for a given application model is defined as the fraction of additional energy use of the scheduler in percent, in comparison to ORACLE. The results are illustrated in Figure 3. On average, MDP’s energy consumption was within 0.9% of ORACLE’s energy consumption, whereas WIFI deviated by 4.5% and 3G by 11.4%. The maximum deviation of MDP for all applications was 6.5%, whereas WIFI was up to 18.5% and 3G up to 84.1% off. In addition, for 74% of all random applications, MDP was not more than 1% off from ORACLE’s energy consumption.

4. RELATED WORK

Research about energy efficient scheduling can be classified in two groups. Many results exist about how to optimise energy performance of a single interface without load balancing on multiple interfaces [1, 9, 10]. In this line of work, MDP-based algorithms to optimise energy consumption of mobile devices taking time variability into account are widely used. For example, [1] solves an energy optimisation problem for a simplified Wifi interface consisting of only of startup and transmission energy. However, while most of the bytes generated by browsing are received, this optimisation framework only takes sending into account.

The second group of articles deals with load-balancing data over multiple interfaces and is closer to our line of work. Although MDP-based modelling is well-suited in this area as well, we are not aware of any paper using an MDP-based transport layer load-balancing scheme to minimise energy consumption with complex energy and application models. In [7], the authors propose to use context information for deciding which interface to use for a single-flow fixed size data transmission on a mobile device. Their proposed algorithm is opportunistic, i.e., it always chooses the best interface at the moment without taking time-varying traffic and complex energy models into account. These simplifications and assumptions are hard to justify for complex energy models and bursty Internet traffic.

Using MPTCP for energy aware scheduling has first been proposed in [6]. The author uses simple heuristics to shift traffic, but
does not take complex energy models or time varying traffic into account. Using a proxy server to enable MPTCP for a service has been examined in [12]. Our contribution is the combination of both ideas to get energy-aware scheduling for all Internet flows, paired with a novel interface scheduler that takes a more realistic energy and application model into account and performs close to optimal.

5. SUMMARY AND FUTURE WORK

We have studied energy efficient interface scheduling using multipath TCP. Our measurements show that different interfaces are energy optimal in different throughput regions, which motivates the use of MPTCP-based flow switching to minimise energy consumption. Simple switching between interfaces depending on the measured throughput is not optimal, since some interfaces use long timeouts or complex finite state machines to control throughput and energy consumption. In addition, because switching interfaces is limited by protocol issues, such as TCP sockets that are bound to a single interface IP address, we propose a scheduling algorithm for an architecture using MPTCP to enable seamless flow switching without interrupting active flows. We described the scheduling problem as a Markov decision process, which enabled us to automatically derive optimal schedulers given an application model and energy models for the interfaces described as finite state machines. Our evaluation shows that the MDP scheduler is robust against random application models and approaches a lower bound for realistic traffic. Because future-generation mobile devices will feature more and more diverse interfaces, fine-grained flow switching using the MDP scheduler can effectively exploit the best properties of all interfaces, to make scheduling more resilient and more energy-efficient.

Several questions remain for future work. We have only looked at a simple state space that only takes throughput into account. Some other alternatives of context information for scheduling have been explored in [7] and have shown to be effective for multipath scheduling. We are currently exploring how to enrich the state space with additional variables, e.g., the channel state of an interface, the average delay during a measurement period, access point or base station information for all interfaces, time and day as well as active application processes. In general, the more state variables the scheduler takes into consideration, the more adaptive it will be regarding its context. However, more state variables also cause a state space explosion, which makes solutions with many state variables difficult to obtain. This leaves the question what a good trade-off between efficiency of the scheduler generation and the scheduler performance is.

In our model, we assume a stationary application model for the entire simulation run. In general, many factors influence the application model in a dynamic way. Similarly, energy models vary depending on, e.g., signal strength and modulation formats, which results in different MDP schedulers for each combination of application model and energy models, depending on all factors that influence them majorly. This means that we could face a large number of different schedulers that have to be deployed to cover all different circumstances. To address this problem, we are currently studying the question of automatically generating good application models by clustering application throughputs for long-term measurements and energy models. Schedulers for each cluster are derived such that the majority of all cases is covered. In operation, the mobile device selects the scheduler closest to its current state determined by context information. If there is no scheduler available close to its state, it will use a default scheduler that has been optimised for the entire set of measurements. The schedulers for our evaluation are 375 bytes in size without compression (3000 states with a binary decision in each state), so even a large number of them will fit comfortably onto a modern mobile device. We suspect that the actual number of schedulers needed to get high energy efficiency in practice is not very large. A large-scale measurement study is currently under way to measure average user application throughput behaviour on mobile devices. This study will allow us to determine how many schedulers a mobile device needs for good performance.

Another open question is how to generate realistic application throughput models. In the evaluation, we used traces from Wifi interfaces and assumed that the throughput process does not depend on the chosen interface, which allowed us to measure throughput on the higher bandwidth interface. In general, throughput distributions depend on many external factors, such as interface bandwidth caps or signal strength. We are currently investigating how to map flow arrival and departure data with flow size information to interface throughputs given context information such as signal strength, modulation format, SSID of the base station or ID of the 3G towers the device is connected to. This procedure allows us to measure application flows independently of the throughput and then map them into expected 3G and Wifi throughputs, which makes the application model more expressive while simplifying its generation at the same time.

6. REFERENCES