Balancing Energy Saving and QoS in the Mobile Internet: An Application-Independent Approach[‡]

G. Anastasi[•], M. Conti[*], E. Gregori[*], A. Passarella[•]

[•]*Univ. of Pisa, Dept. of Information Engineering*
Via Diotisalvi 2 - 56122 Pisa, Italy
{g.anastasi, a.passarella}@iet.unipi.it,

[*]*CNR-IIT Institute*
Via G. Moruzzi, 1 - 56124 PISA, Italy
{marco.conti, enrico.gregori}@iit.cnr.it

Abstract

The scarcity of energetic resources in mobile computers is a very limiting factor. In this paper we propose a solution that tries to balance energy consumption and QoS requirements. Our solution follows an application-independent approach and, therefore, it can be used concurrently, and without modifications, by any network application. Furthermore, our solution is independent from the sub-network technology. We implemented this solution and we extensively tested it. Experimental results have shown that a relevant energy saving (about 70% on average) can be achieved with respect to the legacy approach based on the TCP/IP protocol stack. Furthermore, these savings are obtained without a significant degradation in the QoS perceived by the user. We also compared our application-independent approach with an application-dependent one (i.e., a solution tailored to Web browsing) which performs (slightly) better. However, the application-independent solution still guarantees significant savings, and fits better a general-purpose mobile environment.

1. Introduction

The Internet explosion in the last years has demonstrated that accessing information of some interest in the same moment they are needed is a valuable opportunity. In this context, the concept of *mobility* adds a new dimension: the user is no longer bound to his/her desktop personal computer to access Internet services. It is not visionary to foresee that in a near future millions of users will access Web pages (or read their e-mail) by using a PDA, a pager or a cellular phone. However, integrating mobile devices in the legacy Internet scenario is still a challenging problem for several reasons. Mobile devices have less computational, storage, and energetic resources with respect to desktop computers. Furthermore, they usually connects through wireless links that are characterized by lower bandwidth and greater bit error rates with respect to wired links. Using the legacy Internet solutions may thus result in a non-optimal usage of the system resources. In particular, power saving mechanisms need to be introduced to optimise energetic resources' usage. Finally, the provision of added-value services (e.g., QoS support, security, etc.), that are still an open issue even in the legacy Internet scenario, becomes an extremely challenging problem in the Mobile Internet where the path between the communication's endpoints is not static but changes in time. Among these problems, the scarcity of energy resources is a very limiting factor [8, 11, 16]. Users are not happy if their mobile computer switches off in the middle of a network transaction because the battery is exhausted. At the same time, the user does not like to recharge the battery too often. This paper focuses on strategies to optimise the usage of the mobile-computer battery power.

In principle, energy-related problems in mobile access to the Internet could be solved by either increasing the battery capacity or reducing the energy consumption. Projections on progresses in battery technology show that only small improvements in the battery capacity are expected in next future [18]. As the battery capacity cannot significantly be improved, it is vital that energy utilization is managed efficiently by identifying ways to use less power preferably with no impact on the applications. Strategies for energy saving have been investigated at several layers including the physical-layer transmissions [20], the operating system (techniques can be adopted for hard-disk management [9], CPU scheduling [15], screen blanking [15]), the network protocols and the application level. At the application level some techniques profit of remote tasks' execution [7] (e.g., the mobile computer discards on a fixed host some energy-consuming task, taking only the task's results from it); another approach consists in exploiting the application semantic [12] (e.g., the application compresses the data before exchanging them, or finds some tradeoff between performance and power consumption).

In this paper we investigate energy-saving strategies implemented in software network protocols. We use an *application-independent* approach in the sense that envisaged strategies do not exploit knowledge about the

[‡] The work described in this paper was carried out under the financial support of the Italian Ministry for Education and Scientific Research (MIUR) in the framework of the Projects "Web Systems with QoS Guarantees" and "Internet: Efficiency, Integration and Security".

above applications. Our solution presents to the above layer a standard socket interface, and thus it does not require any modification in the applications. In addition, it is completely independent from the sub-network technology. This work is complementary to a previous one [1] in which we developed an *application-dependent* solution to power saving. Specifically, the application-dependent solution was tailored to Web browsing. The two approaches (application dependent and application independent) share some basic architectural design aspects. Both approaches exploit a network architecture based on the Indirect-TCP model [3]. The mobile computer connects to a fixed host (e.g., a Web server) through a third entity (the Access Point) located at the border between the wireless and wired networks (see Figure 1). With respect to the traditional TCP/IP architecture, the transport connection between the mobile host and the fixed host (e.g., a Web server) is split into two parts. The first one connects the mobile host with the Access Point, while the second connects the Access Point and the fixed host. At the Access Point a software agent (the *Indirect-TCP Daemon*) relays data between the two connections. To achieve power saving the mobile host periodically switches the network interface off. While disconnected, data coming from the Internet and destined to the mobile host are temporarily stored by the *Indirect-TCP Daemon*. To decide when and how long the network interface should be off, both the approaches (i.e., the independent and the dependent one) dynamically estimate the traffic behavior (packet inter-arrival times, idle periods, etc.). Switching off the network interface actually reduces the energy consumption but can heavily increase the User Response Time (URT, i.e., the elapsed time between the generation of a request from the browser, for the retrieval of a Web page, and the rendering of that page at that browser's site), thus negatively affecting the QoS perceived by the user. Thus, a trade off between these two orthogonal performance figures must be reached.

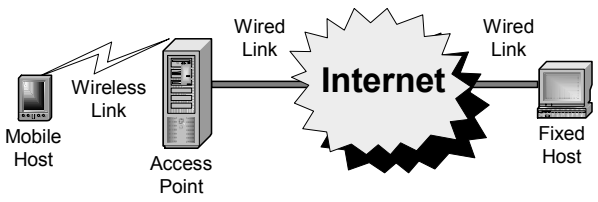

Figure 1. Mobile Internet scenario

The basic difference among the two approaches resides in the algorithms used to decide when, and how long, to keep the network interface off. In the application dependent approach the semantic of the Web application is exploited. Hence, this approach requires a specialized protocol for each application to be supported. For this reason, hereafter we design and evaluate the independent approach that is based on algorithms that do not rely upon any *a priori* application semantic but try to dynamically intercept the behavior of the active application(s). This approach is much more flexible, and it is therefore interesting to compare its performance with those of the application dependent approach that constitute a target reference. It can be expected that the dependent approach provides better power saving characteristics.

We implemented the application-independent solution in a prototype system and extensively tested it. The target of our experiments was twofold: (i) to understand how our solution performs in an actual Internet scenario, with respect to the power savings, and the QoS perceived by the user; and (ii) to compare and contrast it with the application dependent solution. Our experimental results indicated that both the dependent and the independent approach guarantee a significant power saving: the application dependent solution saved, on average, more than 80% of the legacy TCP/IP-architecture energy-consumption, while, in the same application scenario and under similar conditions, the application independent solution saved, on average, around 70% of the legacy TCP/IP-architecture energy-consumption. Furthermore, these reductions in the power consumption were obtained without a significant degradation of the QoS. Specifically, we measured the increase in the User Response Time (URT) caused by our power saving architectures. With respect to the legacy TCP/IP-architecture, the application dependent approach increased the URT of less than 2 seconds, while in the application independent approach the additional URT was less than 2.5 seconds.

The paper is organized as follows. Section 2 presents and discusses previous works. Section 3 is devoted to the definition of our application independent architecture. Section 4 reports some experimental results obtained by using the prototype implementation. Finally, Section 5 concludes the paper.

2. Related works

Among the hardware components of a mobile computer, the network interface accounts for a significant part of the total energy consumption. For devices such as laptops, this portion is approximately 10%, while in small size mobile computers (PDA, hand-held devices, ...) the percentage increases up to 50% [13][1]. Thus, in current mobile computers it is vital to implement smart power-saving strategies tailored to the networking subsystem.

In principle, there are different approaches to limit the power consumed by the network interface. Some researchers focused on the impact of the transmission layer errors on the power consumption. When the bit error rate of the wireless channel is too high, a transmitted

[1] Recall that small mobile computers frequently have no hard-disk, and have limited computational resources, while the network interface must still provide the same functionalities as in a laptop or a PC.

message will be almost certainly corrupted, and this will cause a power wastage. In this case, it is wise to defer the transmission. Other works suggest to limit the number of transmissions, as transmitting consumes more energy than receiving. These strategies apply well in a cellular environment, due to power consumption characteristics of mobile phones. But they do not seem useful in a WLAN environment. In this scenario, the network interface requires nearly the same power in the transmit, receive, and idle states [19, 14]. Thus, the only effective way to reduce the power consumption is to switch the network interface in a sleep status (or, if possible, switch it off) when it is not needed. Furthermore, in WLANs, the exchange of messages should be done at the maximum rate allowed by the wireless channel. This policy reduces the time in which the network interface is on, and therefore the power drained from the battery. The effectiveness of this policy was pointed out in several research works [19, 13, 14]. In the same work it has also been shown that using the legacy TCP/IP architecture (in a mobile Internet scenario) causes a very bad energy utilization at the mobile host. The TCP bandwidth decreases as $1/\left(RTT\sqrt{p}\right)$, where p is the error probability along the TCP connection, and *RTT* is the sum of the wireless and wired links' round trip times [17]. Because the wireless link is typically noisy, many retransmissions may occur, and hence the *RTT* value can be high, thus significantly decreasing the TCP performance. Consequently, the data transfer can be very long, and this results in a great power consumption. Several solutions have been proposed to handle this problem [3, 4]. A possible way is to use the *Indirect-TCP* model [3]. This model was originally proposed to improve the TCP performance in a mobile Internet scenario [3], and then it has been used to handle power saving problems [13, 14, 1].

The approach in [13] implements the power saving strategies at the transport layer: switching the network interface off after an *inactivity* timeout expires (from the last transmission or reception), and resuming it after a *sleeping* timeout, or when the application running on the mobile host generates new data to exchange. The main advantage of this approach is the *application independence*. In the solution proposed by [13] both inactivity and sleeping timeouts have a *fixed value*. As the traffic characteristics dynamically vary, using fixed timeouts may result in poor performance. This can lead both to a weak management of the power consumed by the network subsystem (e.g., if the sleeping timeouts are too small, the network interface is almost always on), and/or to significant degradations in the QoS perceived by the users (e.g., if the sleeping timeouts are too long, the power management subsystem introduces large URTs).

[14] and [1] design power saving strategies by exploiting some knowledge about the behavior of the applications. They use a proxy-based approach, and implement a power saving subsystem tailored to file transfers and to Web service, respectively. This approach provides a nearly optimal power management, because the wireless link is used only when there is some message to exchange and the data can flow at the maximum rate. Moreover, the QoS issue is well addressed. The main drawback is the *dependence* from the application semantic. Thus, this approach fits better a dedicated environment, where the flexibility with respect to the applications is not a critical factor.

3. Network architecture and protocols

In a mobile environment, host mobility is typically handled at the data-link layer (e.g., in WLANs) or at the network layer (e.g., using the Mobile IP). In both cases, it is transparent to the transport layer. Thus, the TCP protocol and the legacy Internet applications can be used even in a mobile environment. Although very simple and costless, this solution presents various drawbacks which heavily impact the power required by the network subsystem of the mobile host.

1. The TCP bandwidth is proportional to $1/\left(RTT\sqrt{p}\right)$, where p is the loss probability on the connection, and *RTT* is the sum of the wired and the wireless round trip times [17]. Because the wireless link is typically noisy, the *RTT* can be very high[2], and the available bandwidth much lower than the wireless-link capacity [14]. A low throughput causes long transfer delays and, hence, the mobile's network interface remains unnecessarily on for long periods.
2. Congestions on the Internet reduce the bandwidth, increasing both p and *RTT*. If the overall throughput is less than the rate available on the wireless link, the effects are similar to those discussed in the point 1.
3. The typical Internet applications (e.g., Web, e-mail, ftp, ...) do not continuously generate traffic on the network: their traffic is characterized by bursts of data (e.g., when the user requests a Web page from the server) followed by idle times (e.g., when the user reads the page that he has just downloaded). During the idle times, the mobile's network interface remains on, and this greatly increases the power consumed without any reason.

To overcome these drawbacks, we extended the *Indirect-TCP model* (see Figure 2) used in previous power-saving architectures, see [14] and [1].

As shown in the figure, the Access Point and the fixed host communicate by using the TCP protocol. On the other hand, the mobile host and the Access Point

[2] Note that the wireless link layer is typically reliable, e.g., IEEE 802.11, and this hides the losses due to channel noise to the transport protocol.

communicate through a high-speed low latency WLAN (e.g., IEEE 802.11). In this case it is more efficient to use a *Simplified Transport Protocol* (STP) that implements only the necessary functionalities. Specifically, STP provides a reliable connectionless type of service relieving the mobile host from the TCP computational burden.

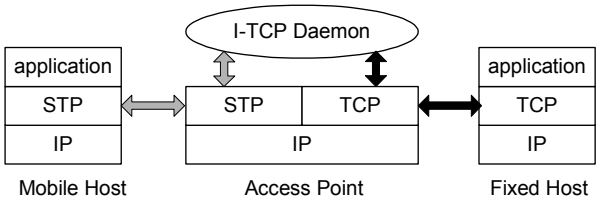

Figure 2. The Indirect-TCP model

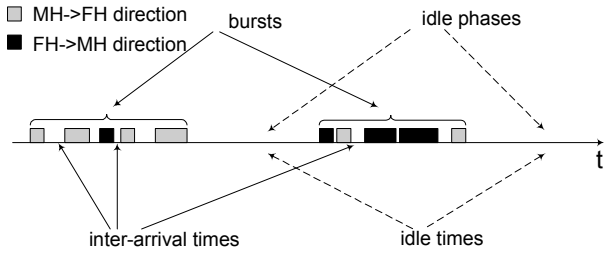

Figure 3. Typical traffic profile, as seen at the mobile transport layer

As noted in point 3 above, mobile hosts do not continuously exchange data, but data transfer phases are characterized by bursts interleaved by *idle* phases during which data are *locally* processed. Figure 3 shows a snapshot of a typical data exchange[3]. Our approach is based on a dynamic estimate of the duration of idle and data transfer phases. Specifically, we measure at run time the data inter-arrival times and the length of the idle times. From these measures, we predict the future traffic behavior, and hence, we decide whether the network interface should be switched off (or not), and when to resume it[4]. Our target is to let the network interface on only when it is energetically convenient, and to transmit data at the maximum rate allowed by the wireless link. This strategy approximates the optimal strategy, and works well if the estimates are accurate. Therefore, the core of our approach are a set of smart algorithms for estimating the traffic characteristics (see Section 3.2 for details). We integrated our power-strategy in the *Indirect-TCP* architecture shown in Figure 2 by introducing, on top of the STP protocol, the *Power Saving Packet Transfer Protocol* (PS-PT), see Figure 4.

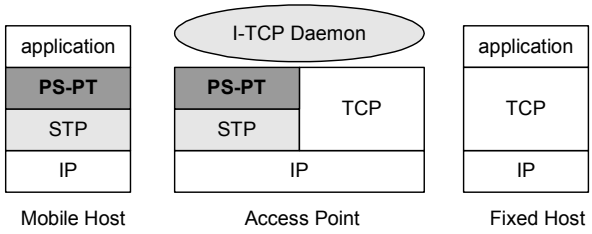

Figure 4. The Power Saving network architecture; evidence on added protocols

When the I-TCP Daemon or the mobile's applications generates a new packet for the mobile host, the PS-PT at the Access Point logs the interval elapsed from the arrival of the previous packet. These logs are used for estimating the traffic inter-arrival times. Specifically, when the transmission queues at both side are empty, the PS-PT estimates when the next packet will arrive. This value is used to decide if the network interface must be switched off. When the network interface is off, the mobile host doesn't know if some packet is waiting at the Access Point. Thus, when the estimated inter-arrival (or idle) time has elapsed, the mobile host must always poll the Access Point for possible new packets, even if its application level has not generated new data to be sent.

We designed algorithms that achieve a tradeoff between the "responsiveness" of the protocol (i.e., if the mobile host polls the Access Point frequently the additional delay introduced by the system is minimum), and its power-saving performance (i.e., if the Access Point is polled rarely, when the mobile host reconnects it is very likely that new data are available thus avoiding power wastage due to useless polls).

3.1. Power Saving Protocols

The core of the PS-PT protocol are the algorithms for estimating when the next packet will arrive. We used an adaptive approach that records the history of the previous inter-arrival and idle times, and relies on this information, to estimate the next inter-arrival or idle time (see Section 3.2 for details). In our prototype implementation we chose to run these algorithms at the Access Point in order to minimize the computational burden (and hence the energy consumption) at the mobile host.

The PS-PT protocol was implemented as a simple master/slave protocol. When there are no more data to be exchanged, the Access Point decide whether it is convenient to the mobile host to switch the network interface off. If so, it sends a "shutdown" command to the mobile host including an indication of the time interval during which the mobile host should remain disconnected. The mobile host uses this interval to set a timer. Upon the timer expiration, the mobile host polls the Access Point

[3] When several applications are concurrently running the resulting traffic profile is the superposition of the single data exchanges.

[4] The network interface has a transient in getting on during which it drains power from the battery but is not available for exchanging data. Therefore, for small inter-arrival or idle times it is energetically convenient to leave the network interface on.

again. In the following, we shall describe in detail the actions performed at the mobile host, and at the Access Point, respectively (the pseudo-code here is optimized for clarity rather than for efficiency).

PS-PT Actions at the Mobile Host

```
 1  OnPacketFromApplications(packet)
 2    timestamp(packet)
 3    if(card is OFF)
 4      stop timer
 5      turn card ON
 6    send packet to Access Point

 7  OnPacketFromAccessPoint(packet)
 8    cmd = extract_command(packet)
 9    if(cmd == OFF)
10      turn card OFF
11      t_i = extract_interval(packet)
12      set_timer(t_i)

13  OnTimerExpired()
14    turn card ON
15    send ON_signal to Access Point
```

Figure 5. PS-PT protocol: actions performed at the mobile host.

Upon reception of a new packet from the above application(s), the mobile host bounds a timestamp to the packet (this timestamp will be used at the Access Point to maintain the history of the arrival times, see line 2). If the network interface is OFF, then the timer used to signal when the mobile host have to reconnect and poll the Access Point is active. The mobile host stops this timer (i.e., the last estimate was too large, lines 3-5) and sends the packet to the Access Point (line 6).

When a new packet from the Access Point arrives, the mobile host checks whether it contains a "shutdown" command (lines 8-9). In this case the packet also includes the time interval during which the network interface should remain OFF. The mobile host switches the network interface OFF, and sets the timer accordingly (lines 10-12). Finally, upon timer expiration, the mobile host polls the Access Point (lines 13-15).

PS-PT Actions at the Access Point

```
16  OnNewPacket(packet)
17    if(card is OFF)
18      timestamp(packet)
19      buffer(packet)
20    else
21      stop timer
22      send/receive data
23      t_i = evaluate_next_interarrival()
24      if(card must get OFF)
25        send (OFF_CMD, t_i - t_PWON) to
            Mobile Host
26      else
27        set_timer(t_i)

28  OnTimerExpired()
29    t_i = update_estimate()
30    if(card must get OFF)
31      send (OFF_CMD, t_i - t_PWON) to
            Mobile Host
32    else
33      set_timer(t_i)
34  OnMobileGetsON()
35    if(there is no data to exchange)
36      t_i = update_estimate()
37      if(card must get OFF)
38        send (OFF_CMD, t_i - t_PWON) to
            Mobile Host
39      else
40        set_timer(t_i)
41    else
42      send/receive data
43      t_i = evaluate_next_interarrival()
44      if(card must get OFF)
45        send (OFF_CMD, t_i - t_PWON) to
            Mobile Host
46      else
47        set_timer(t_i)
```

Figure 6. PS-PT protocol: actions performed at the Access Point.

At the Access Point side, the system records the state of the mobile host's network interface. Upon reception of a new packet (from the Internet) while the mobile host is disconnected, the Access Point buffers the packet and waits for a poll from the mobile host (lines 17-19). On the other hand, if the packet is received while the mobile host is connected, the Access Point relays the packet to the mobile host (if it was received from the Internet, line 22), estimates the next packet arrival time, and decides whether its is convenient to shut down the network interface (lines 23-27). It is worthwhile to recall that the network interface has a transient period t_PWON in getting on during which it is not able to handle data. This implies that the mobile host must be ON t_PWON units of time before the estimated arrival (line 25). If the Access Point estimates that it is convenient for the mobile host to disconnect, it sends a OFF command to the mobile host together with the time interval during which it must remain disconnected (lines 24-25). Otherwise, it sets a timer with the estimated arrival time (lines 26-27). In the latter case the mobile host remains connected. Therefore, if a new packet arrives, the network interface is ON and, hence, the system must stop the timer (line 21).

When the mobile host polls the Access Point, there might be data to exchange or not. In the former case, the Access Point uses the new data to generate a new estimate and performs the same actions described above (lines 41-47). In the latter case, the last estimate provided to the mobile host was too short. Thus, the Access Point updates this estimate and decides what the mobile host must do (lines 35-40). The same situation occurs when the timer expires: the last estimate was too short, but it didn't cause the switching off of the network interface. The mobile host is still connected and the Access Point has to decide whether it is convenient that the mobile host disconnects or not (lines 28-33).

3.2. Algorithm for packet arrivals estimates

As clearly appears from the previous section, our solution relies upon the prediction of the traffic behavior. Therefore, we need an algorithm that provides accurate estimates of packet inter-arrivals and idle times, and is able to adapt quickly to changes in the traffic conditions. The Variable-Share Update algorithm [10] fits these requirements. This algorithm has been proposed as a dynamic algorithm to estimate a generic variable spanning a given range, and is not bound to a specific problem.

Let I be the range of possible values for a variable y that we want to estimate. To predict the value of y, the Variable-Share Update algorithm relies upon a set of "experts". Each expert x_i provides a (fixed) value within the range I, i.e., a value that y can assume. The number of experts to be used, as well as their distribution among the range I, are input data for the algorithm. Each expert x_i is associated with a weight w_i, a real number that measures the dependability of the expert (i.e., how accurately the expert has estimated y in the past). At a given time instant, an estimate of y is achieved as the weighted sum of all experts, using the current w_i value as the weight (i.e., reliability) for the expert x_i. Once the actual value of the variable y is known, it is compared with the estimates provided by the experts, to recalculate and update the weight associated with each expert.

The algorithm is summarized in Figure 7. As shown in the figure, the core of the algorithm is the weights updating algorithm. Updates occur every time a new actual value of the variable y becomes available. First, an error function L is evaluated for each expert: this function measures the deviation of the (prediction provided by the) expert from the actual variable's value. Then, the Variable-Share Update is executed, as follows:

1. each expert *loses* a portion of its weight, according to the deviation from the actual value; the weight w_i becomes w_i' (if $L=0$ the weight is not changed);
2. each expert *shares* a portion of its weight, according to its error function: a *pool* is created by using all the shares (if $L=0$ the expert doesn't share anything);
3. for each expert, the new weight is calculated as the sum of two components: a portion of the weight evaluated in 1, and a portion of the *pool* evaluated in 2. Both components depend on the error function (e.g., if $L=0$, the new weight is the old one, plus a fraction of the pool).

Parameters: $\eta > 0$, $0 \le \alpha \le 1$, n (number of experts)
Variables: x_i (experts), w_i (weights), y (actual variable's value), $\hat{y}$ (estimated variable's value)

Initialization: $w_i = 1/n \quad \forall i = 1,\ldots,n$

Prediction: $\hat{y} = \sum_{i=1}^{n} w_i x_i \Big/ \sum_{i=1}^{n} w_i$

Loss Update: $w_i' \overset{\Delta}{=} w_i e^{-\eta L(y,x_i)}$

Variable share:
$$\begin{cases} pool = \sum_i \left[1-(1-\alpha)^{L(y,x_i)}\right] w_i' \\ w_i = (1-\alpha)^{L(y,x_i)} w_i' + \dfrac{1}{n-1} \left\{ pool - \left[1-(1-\alpha)^{L(y,x_i)}\right] w_i' \right\} \end{cases}$$

Figure 7. Variable-Share Update algorithm

The Variable-Share Update algorithm reduces the weights of those experts that provides bad predictions, and increases the weights of the experts that provide the more accurate predictions. The speed in increasing/decreasing weights is determined by two algorithm parameters: α and η.

This algorithm has been proposed to implement a spin-down technique in hard disks power management [9]. In that context, the update policy guarantees a quick adaptation to changes of the variable's values.

In our problem we have to estimate two variables – inter-arrival times and idle times – that span different ranges. Inter-arrival times are time intervals between two consecutive packets within a burst, while idle times are time intervals between two consecutive bursts of packets. Typically, inter-arrival times are smaller than idle times. We assumed 1 second as the largest value for inter-arrival times: this assumption relies on previous works on Web traffic characterization [6, 2]. Furthermore, we used this value to discriminate between inter-arrival times, and idle times, respectively. Accordingly, we used two different sets of experts for the two quantities, and we considered two non-overlapping intervals for experts values: the first interval ranges from 0 to 1 second, the second one from 1 to 60 seconds.

Choosing 60 seconds as the largest value for an estimated idle time is a tradeoff between power saving and QoS requirements. When an idle phase occurs, the mobile host polls the Access Point with a maximum period of 60 seconds (see Section 3, lines 35-40). Thus, the maximum additional delay introduced by the algorithm to the first packet of the new burst is 60 seconds, which is an acceptable upper bound for applications without real-time requirements.

We used 20 experts for each set. The experts' values are uniformly distributed over the corresponding intervals (experts of the first set are placed every 50 msec, while experts of the second set are spaced by approximately 3 seconds each other).

The update policy shown in Figure 7 needs two additional parameters, α and η. According to [9][5], we

[5] [9] compares several values for α and η assuming a uniform distribution of experts. The experimental results indicate α=0.08 and η=4 as the best choice.

used α=0.08 and η=4. Finally, the algorithm requires the definition of a loss function *L*. This function provides a measure of the deviation of each expert from the actual value of the variable, and its values must lie in [0,1], see [10]. In our implementation we used $L(x_i, y) = |x_i - y| \Big/ \max_i |x_i - y|$ as the error function.

The last steps to build our system requires (i) to decide how to manage an estimate that happens to be too short, and (ii) to decide when to use the first set of experts, and when the second one.

Let us consider a too short inter-arrival time estimate $\hat{y}_i$. When the mobile host polls the Access Point the system must update this estimate (see lines 35-36 of Section 3). As all the experts below $\hat{y}_i$ have clearly provided a wrong prediction, we update the estimate by taking into considering only the other experts in the set. Specifically, the new estimate $\hat{y}_i'$ is the weighted sum of all experts that have an associated value greater than $\hat{y}_i$. In addition, as $\hat{y}_i$ seconds have already elapsed since the previous arrival, the next arrival is expected in $\hat{y}_i' - \hat{y}_i$ seconds. The PS-PT protocol uses this value ($\hat{y}_i' - \hat{y}_i$) to decide whether it is convenient (or not) to the mobile host to switch the network interface off (lines 37-40). The system reuses the same procedure every time it detects that the previous estimate was too short, until the inter-arrival time becomes bigger than the maximum value for that set of experts (1 second). At this point in time, it is assumed that an idle phase has begun, and thus, the second set of experts is used to estimate the length of the idle phase. If even the second set of the experts provide a too short estimate, a similar updating procedure is used: the estimate is updated by considering only experts that provided predictions greater than wrong estimate until the idle phase reaches the maximum value, i.e., 60 seconds. From this time instant onward, the mobile host polls the Access Point periodically every minute, until some data becomes available. The arrival of a new packet is interpreted as the beginning of a new burst, and thus, the algorithm switches again to the first set of experts.

Our updating policy guarantees a smooth increase towards greater estimates. Thus, it minimizes the probability of interpreting an actual inter-arrival time as an idle phase, and so it introduces small delays to the packet transfers (see the experimental results in Section 4).

4. Experimental results

The objective of the proposed solution is twofold: it should allow significant power savings with respect to the legacy TCP/IP approach by minimizing, at the same time, the degradation of the QoS perceived by the users. To evaluate our architecture in an actual network scenario, we used the Web as the testing application. The Web is today one of the most popular Internet application, and it is the candidate to become the killer application even for mobile Internet. Moreover, Web users are sensitive to delays. Hence, it is important to achieve significant power savings, while maintaining acceptable QoS levels, i.e., to minimize the URT increase.

To evaluate the performance of our architecture we defined two indices: a Power Saving Index and a QoS index. The Power Saving Index, *I_ps*, is defined as:

$$I\_ps = \frac{net.int.consumption\ in\ our\ architecture}{net.int.consumption\ in\ TCParchitecture} \quad (1)$$

I_ps measures the percentage of power consumed by our architecture with respect to the legacy one, and, hence, provides an indication of the power saving achieved by using our architecture. The QoS index, named the Page Delay Index (*I_pd*), is defined as:

I_pd = (URT in our arch.) – (URT in TCP arch.) (2)

I_pd measures the additional URT introduced by our architecture with respect to the legacy architecture, and hence provides an indication of the URT increase to pay to achieve power savings.

In our experiments we used SURGE as the application layer at the client side [5]. SURGE is a Web traffic simulator, designed by Barford and Crovella, that models the statistical properties of the traffic generated by a realistic Web user browsing the Internet. Furthermore, at the server side, we used a real Web server. Specifically, in our experiments the Web server was located at the University of Texas at Arlington, while the client was located at the Department of Information Engineering of the University of Pisa (Italy). Hence, our client-server path crossed (congested) intercontinental links, and this allowed us to test our architecture in a congested situation.

To significantly evaluate our architecture, we performed a large set of experiments. Each experiment included 150 file-transfer operations from the Web server to the client[6] (an experiment stopped when the whole page "in flight" arrived at the client). In each experiment, the same set of files were requested in parallel both in our architecture, and in the legacy one. This guarantees the same network conditions in both cases. We ran a set of experiments, where each experiment spanned an entire working day. To increase results' reliability, we replicated the experiments in several working days.

Finally, we compared the results obtained with those related to an *application-dependent* solution developed

[6] Transferring 150 files allowed a significant sampling from the Web file dimensions distribution.

[1]. In [1] we performed an identical set of experiments, evaluating the same indices, under similar network conditions. For the sake of brevity, below we only present the results from a particular day. However, the results obtained in different days exhibit the same statistical behavior.

4.1. Power Saving Analysis

As mentioned in Section 2, the energy drained from the battery by the wireless network interface of a mobile host is almost the same irrespective of its status (receiving, transmitting or idle). Thus, the energy consumption is well approximated by the time the network interface remains on. Accordingly, to evaluate the *I_ps* index (see (2)), we measured this time both in our architecture and in the legacy one.

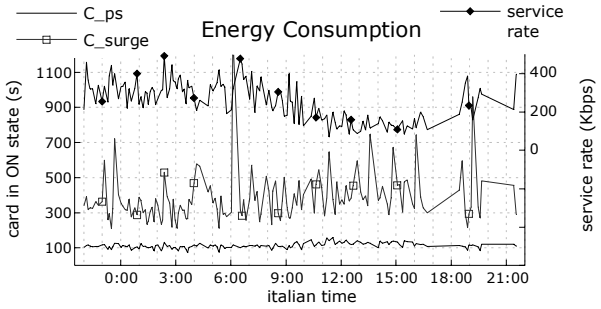

Figure 8. Energy consumption as a function of time.

For each experiment, Figure 8 reports the total time during which the network interface remains on in the legacy architecture (middle curve) and in our architecture (bottom curve), respectively. Figure 8 also includes the *service rate* (top curve) defined as the ratio between the number of bytes transferred on the wired network and the time needed to transfer them. Therefore, the service rate provides a measure of the Web service responsiveness to client's requests. The service rate allows us to understand how the energy consumption depends on the environment conditions.

From Figure 8 it appears that the power consumption in our architecture is almost independent from the service rate. This is a joint effect of: (i) the splitting of the TCP connection in two parts, and (ii) the use of our power-saving policies. Specifically, our power-saving policies allow the network interface to remain on only when there are data to exchange on the wireless link. Therefore, the energy consumption does not depend on the time needed to complete the whole transfer between the remote and the mobile host (this time highly varies with the network conditions).

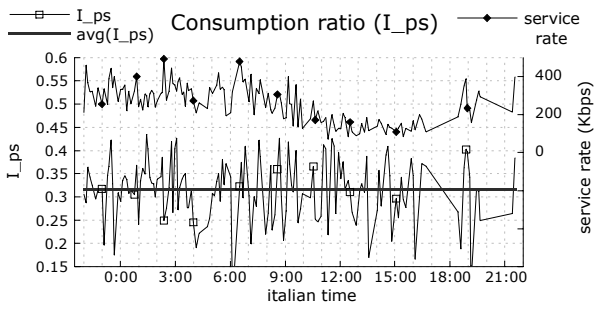

Figure 9. *I_ps* as a function of time

Figure 9 shows the *I_ps* index as a function of time (bottom curve), its average value on the entire day (straight curve), and the service rate (top curve). These curves confirms that our architecture introduces a significant power savings: the energy consumption is always less than 45% of the one in the legacy architecture, its average value is 32%, with values less than 15%. This means that we can save always more than 55%, on average 68%, and with peaks over 85%.

When using an application-dependent approach [1], we obtained a slightly better performance: about 80% saving on average, with peaks over 90%. This difference can be easily explained as the application-dependent approach exploits informations about the traffic characteristics, and the application behavior. On the other hand, the application-independent approach cannot rely on such informations.

4.2. QoS Analysis

Figure 10 shows the additional delay introduced in our architecture to the transfer delay of a single file. Specifically, for each file we measured the additional delay, then we averaged the values on each experiment of a day, and we plotted these average values (bottom curve), together with the average value on the entire day (straight curve). For convenience, we also plotted the service rate (top curve).

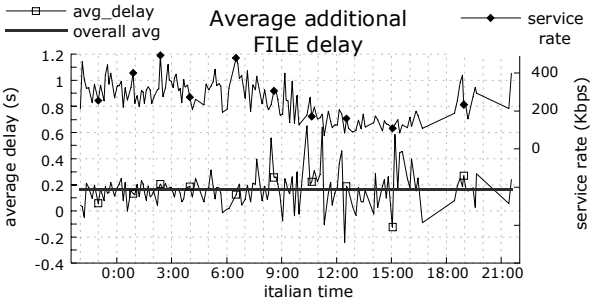

Figure 10. Average additional file delays as a function of time

From Figure 10 it appears that the average additional delay introduced for a single file is no more than 1 second. Furthermore, the average value on all the experiments is below 0.2 seconds.

The file-delay index does not characterize the users' QoS. A user does not perceive the delays related to single files but his/her QoS depends on the URT related to the whole Web page (a page usually contains several files). Figure 11 allows us to evaluate the system performance from this point of view. In each experiment we measured the average additional URT introduced to the downloading of Web pages (average *I_pd,* bottom curve). We also considered the average additional URT over the entire day (straight curve), and the service rate (top curve).

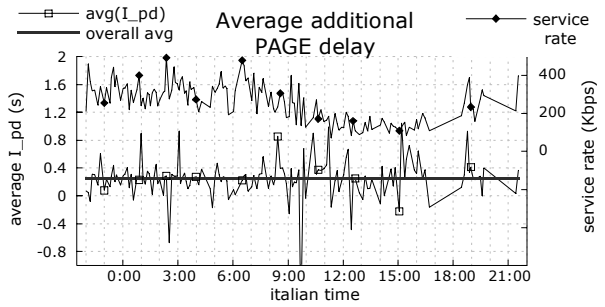

Figure 11. Average I_pd as a function of time

It appears that the additional URT, averaged over the entire day, is slightly higher than 0.2 seconds, i.e., it is only slightly higher than the average additional delay of single files. This means that the delays added to single files of a Web page do not accumulate in the URT of the whole Web page. Moreover, the average *I_pd* in each experiment is no more than 1 second. So, the user doesn't perceive a significant degradation in the QoS.

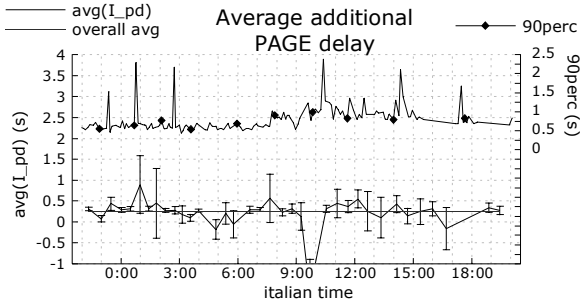

Figure 12. I_pd: confidence interval of the average and 90th percentile (conf. level 95%)

Finally, we examined the upper bounds of the additional URTs. For each experiment, we evaluated the confidence interval of the average and 90th percentile of *I_pd* (confidence level 95%). In Figure 12 we plotted the average *I_pd*, with the related confidence interval (bottom curve), the average additional URT over the entire day (straight curve), and the 90th percentile measured in each experiment (top curve).

From Figure 12 it appears that the average *I_pd* values are small (the values are around 0.5 seconds). Furthermore, the 90th percentile is typically below 1.5 seconds, and always below 2.5 seconds. Therefore, we can say that our system, with a probability of 90%, does not add more than 2.5 seconds, and therefor, the user doesn't perceive almost any degradation in the QoS.

The QoS performances related to the application-dependent system are slightly worse (see [1 for details). In that case we obtained similar values with respect to the average *I_pd*, its confidence interval, and its overall average over an entire day. On the other hand, the application-dependent approach exhibited higher peak values (up to 5 seconds) with respect to the 90th percentile of the *I_pd*. This can be explained by the aggressive power-saving policy adopted by the application-dependent approach: the mobile host's network interface remains off for more time (and, thus, the energy consumption is higher), but this causes an increase in the upper bound of the additional delays. However, on the average, both approaches are equivalent in terms of the QoS perceived by the user. Specifically, with respect to the Web application taken into consideration in both cases, it can be expected that the user does not experience a significant degradation in the QoS.

5. Conclusions

In this work we have designed and evaluated a novel network architecture for reducing the power consumption of mobile hosts in a mobile Internet scenario, while maintaining almost the same QoS level. Our architecture follows an application-independent approach as it does not make any assumption about the traffic profile generated by the applications. Therefore, it fits any Internet application, and can be used concurrently by different applications. Furthermore, it is totally transparent to the upper layer as it presents a standard socket interface. Hence, it can be used without any modifications in the code of legacy Internet applications.

Our architecture exploits the Indirect-TCP model: it splits the transport connection between the mobile and the fixed host. This allows an improvement in the bandwidth available at the transport layer and, consequently, reduces the energy required to transfer a given amount of data.

The core of our approach, is the Variable-Share Update algorithm that predicts packet inter-arrival times and idle period lengths, and manages the network interface of the mobile host accordingly. Specifically, the mobile host switches off its wireless network interface when there are no data to be exchanged over the wireless link. This allows to consume the minimum amount of power for each data communications.

We have implemented a prototype of our architecture and we have extensively evaluated it. We used in the evaluation a Web as application as: (i) the Web is the most popular Internet application, and (ii) Web users are typically delays sensitive. Furthermore, this choice has allowed us to compare the results obtained by using our application-independent approach with those related to the application-dependent approach developed in a previous paper.

The experimental results have shown that the application-independent architecture is able to save, on average, the 68% of the energy consumed by the legacy TCP/IP architecture, with peaks over 85%. Furthermore, this energy saving is obtained without a severe degradation in the QoS perceived by the users. The additional URT introduced for transferring a Web page is no more than 2.5 seconds. The comparison between the application-independent approach, and the application-independent one, has shown that exploiting some knowledge about the application behavior results in a slightly better power saving, but may introduce – in the worst case – larger additional delays. However, in the average case, both approaches have exhibited similar performance in terms of QoS provided to Web users. By considering the high flexibility of the application-independent architecture, these results seem to indicate that this approach is a very promising one for power saving. Experiments with different type of applications (e.g., e-mail), and several applications concurrently active on the mobile host, are however required to further validate its potentialities.

Acknowledgments

The authors wish to express their gratitude to Paul Barford for providing the SURGE traffic generator used in the experiments. Also many thanks to Mohan Kumar for giving the opportunity to use the Web server at the University of Texas at Arlington.

References

[1] G.Anastasi, M.Conti, E.Gregori and A.Passarella, "A Power Saving Architecture for Web Access from Mobile Computers", Proceedings of the *IFIP Networking Conference (Networking'02)*, Pisa (I), May 2002, Lecture Notes in Computer Science, LNCS 2345, pp. 240-251.

[2] M.Arlitt, T.Jin, "Workload Characterization of the 1998 World Cup Web Site", *HPL-1999-35(R.1)*, Internet System and Applications Laboratory, HP Laboratories Palo Alto, September 1999.

[3] A.Bakre, B.R.Badrinath, "Implementation and Performance Evaluation of Indirect TCP", *IEEE Transactions on Computers*, Vol.46, No.3, March 1997.

[4] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, R. H. Katz, "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links", *IEEE/ACM Transactions on Networking*, Vol. 5, N. 6, December 1997.

[5] P.Barford e M.Crovella, "Generating Representative Web Workloads for Network and Server Performance Evaluation", *Proceedings of ACM SIGMETRICS '98*, Madison, WI, pp. 151-160, June 1998.

[6] M.Crovella e A.Bestavros, "Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes", *IEEE/ACM Transaction on Networking*, Vol.5, No.6, pp.835-846, December 1997.

[7] J.Flinn, S.Y. Park and M.Satyanarayanan, "Balancing Performance, Energy, and Quality in Pervasive Computing", Proceedings of the *IEEE International Conference on Distributed Computing Systems* (ICDCS'02), Wien (Austria), July 2002.

[8] G. H. Forman, J. Zahorjan, "The Challenges of Mobile Computing", *Technical Report*, University of Washington, March 1994.

[9] D.P. Helmbold, D.E. Long, B. Sherrod "A Dynamic Disk Spin-down Technique for Mobile Computing", Proceedings of the Second Annual *ACM International Conference on Mobile Computing and Networking*, NY, pp. 130 - 142, November 1996.

[10] M.Herbster and M.K.Warmuth, "Tracking the Best Expert", in the *Proceedings of the Twelfth International Conference on Machine Learning*, (Tahoe City, CA), pp. 286-294, Morgan Kaufmann, 1995.

[11] T. Imielinscki B.R. Badrinath "Wireless Computing", *Communication of the ACM*, Vol. 37, No. 10, October 1994.

[12] A. Joshi, "On proxy agents, mobility, and web access", *ACM/Baltzer Mobile Networks and Applications*, Vol. 5 (2000), pp. 233-241.

[13] R.Kravets e P.Krishnan, "Power Management Techniques for Mobile Communication", *Proceedings of the Fourth Annual ACME/IEEE International Conference on Mobile Computing and Networking* (Mobicom'98).

[14] G. Anastasi, M. Conti, W. Lapenna, "Power Saving Policies for Wireless Access to TCP/IP Networks", Proceedings of the *8-th IFIP Workshop on Performance Modelling and Evaluation of ATM and IP Networks (IFIP ATM&IP2000),* Ilkley (UK), July 17-19, 2000.

[15] J.R. Lorch, A.J. Smith, "Scheduling Techniques for Reducing Processor Energy Use in MacOS", *ACM/Baltzer Wireless Networks*, pp.311-324, 1997.

[16] J.R.Lorch e A.J.Smith, "Software Strategies for Portable Computer Energy Management", *IEEE Personal Communication*, pp.60-73, June 1998.

[17] M.Mathis, J.Semke, J.Mahdavi and T.Ott, "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm", *Computer Communication Review*, Volume 27, number 3, July 1997.

[18] S. Sheng, A. Chandrakasan, R.W. Brodersen, "A Portable Multimedia Terminal", *IEEE Communications Magazine*, December 1992.

[19] M.Stemm e R.H.Katz, "Measuring and Reducing Energy Consumption of Network Interfaces in Hand-Held Devices", *Proc. 3° International Workshop on Mobile Multimedia Communication*, Princeton, NJ, September 1996.

[20] M.Zorzi e R.R.Rao, "Energy Constrained Error Control for Wireless Channels", *Proceeding of IEEE GLOBECOM '96*, pp.1411-1416, 1996.