e-QoS: Energy-aware QoS for Application Sessions Across Multiple Protocol Domains in Mobile Computing

Hanping Lufei
Wayne State University
5143 Cass Avenue
Detroit, MI 48202, USA
hlufei@wayne.edu

Weisong Shi
Wayne State University
5143 Cass Avenue
Detroit, MI 48202, USA
weisong@wayne.edu

ABSTRACT

In this paper we propose a novel energy-aware QoS model, *e-QoS*, for application sessions that might across multiple protocol domains. The model provides the QoS guarantee by dynamically selecting and adapting application protocols. To the best of our knowledge, our model is the first attempt to address QoS adaptation at the application *session* level by proposing a new QoS metric called *session lifetime*. To show the effectiveness of the proposed scheme, we have implemented a case study: instant messaging applications between two PocketPCs. Experiment shows that the session lifetime has been successfully extended to the value negotiated by two PocketPCs with very diverse battery capacities.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Wireless Communication; C.2.2 [**Network Protocols**]: Applications; C.2.8 [**Mobile Computing**]: Algorithm Design and Analysis

General Terms

Design, Performance

Keywords

Energy-Aware, Quality of Service, Application Sessions, Multiple Application Domains, Protocol domain, Session Lifetime, Protocol Adaptation

1. INTRODUCTION

As mobile computing devices and wireless sensors are deployed in large numbers, the Internet will increasingly serve as the interface between people moving around and the physical world that surrounds them [22]. Thus, we expect to see more and more applications having multiple protocol domains involved during one application session. Traditional QoS can hardly satisfy the requirements from the application session level which may dynamically switch between multiple devices and network connections. We define an *application session* as a session period to finish an application level function. For example, in an instant message application session, a user might use a laptop with a cable modem at home, a cell phone with 3G/4G or Bluetooth on the way to the office, a desktop with Ethernet LAN in the office and a PDA with Wi-Fi in the meeting room to talk with another user sequentially. In this scenario, the application session contains several switches between different devices and networks. Although this is an extreme case, it shows that, on the one hand, diverse network connections and heterogeneous devices demand different application protocols, for instance the Gzip protocol for low bandwidth network. On the other hand, both ends of the session could be battery-powered devices. The energy limitation of two ends as well as network and device multi-modalities pose new challenges to traditional QoS models.

We envision that future mobile computing environments could consist of several different application protocol domains. Each domain has its own application protocol set from which the end user can select different protocols based on some proactive judgements to guarantee the pre-negotiated QoS metric. In this context, we propose e-QoS, an energy-aware QoS model for application sessions. If an application session is across multiple protocol domains, the peers on both ends need to negotiate a mutually interested QoS metric through a gateway, which lies between domains (see details in Section 4), before the start of the session. Then the gateway evaluates the candidate protocols for each peer and select one or more protocols according to the peer side information, such as network bandwidth, remaining battery capacity, and so on. Gateway also delivers the protocol modules to the peer so that the protocols can be deployed on the peer side. During the course of the application session, the gateway will monitor the behavior of both peers and the performance of the protocols. Later on, the parameters of the chosen protocols may be adapted or other new protocols could be brought into the session dynamically to satisfy the negotiated QoS metric.

We emphasize energy in the design of the e-QoS model. A new concept, *session lifetime*, which is tied up together with energy, is defined as a new QoS metric. The protocol selection and adaptation methods for maximizing the session lifetime QoS metric are proposed as well. Specifically our contributions of this paper include:

1. *Proposing a general model for application session QoS* — To our knowledge, This paper is the first effort to address the application session QoS management using application protocol selection and adaptation. With the appearance of more and more application level protocols, such as SOAP [31], LDAP [10], and Plugins, their impact on QoS metrics have to be studied and utilized for application session QoS. Dynamically selecting and adapting the necessary application protocols in an on-demand manner is applicable for the future application sessions QoS model.

QShine'06, August 7-9, 2006, Waterloo, Ontario, Canada.

2. *Defining an energy-aware QoS metric, session lifetime, and proposing the enforcement methods* — Energy is managed in terms of a QoS metric, session lifetime, in our model. In this way, it is not only manipulated locally on one end but extended to both ends of the application session. Most of the current energy related efforts only address the client-side management.

3. *Dynamically adapting at the application protocol level* — Most of proposed protocol adaptation methods [1, 9, 21, 25, 30] lie in the network layer. Such systems can cope with localized changes in network conditions but cannot adapt to variations above the network layer.

   Most of existing power management approaches [6, 26, 27, 33] need either extra new hardware or major operating system modifications which are above that some mobile devices can afford. Our model performs entirely in the application level. Furthermore, with the gateway handling most of the computing workload, e-QoS has very lightweight footprint on the user end.

4. *Designing and implementing an energy-aware application: instant messaging between two PockPCs* — Several protocols are developed for this application. Experiment results show that the session lifetime has been successfully extended to the value negotiated by two PocketPCs with very diverse battery capacities.

The rest of the paper is organized as follows. After a brief introduction of terminologies and concepts in Section 2, energy-aware QoS metrics are presented in Section 3. System design is depicted in Section 4. Section 5 evaluates the system using one case study of an instant messaging application between two PocketPCs. Finally, related work and conclusions are listed in Section 6 and Section 7 respectively.

2. TERMINOLOGIES AND CONCEPTS

Before describing the energy-aware QoS metrics, we first introduce some frequently used terminologies and concepts in the following context.

2.1 Session-based Application

On the Internet many applications are session-based. An application session is either a lasting connection at the session layer or application layer between peers, typically a server on one side, and a user on the other side. A session is typically implemented as a layer in a network protocol, like Telnet or FTP. In other cases sessions are maintained by a higher level program using a method defined in the data being exchanged. For example, an HTTP exchange between a browser and a remote host may include an HTTP cookie which identifies state, such as a unique session ID, information about the user's preferences or authorization level, and so on. These kinds of sessions are maintained in application level by application programs. On the contrary, some applications do not have sessions. For instance, sending out an email does not maintain a session in the procedure. In this paper we only consider the session-based application.

2.2 Application-level Protocol

A protocol is the format to express information so that others can understand the information. We divide the protocol into two groups, the network protocol and the application protocol, corresponding to the network layer and application layer respectively. As an example, zip (e.g., Gzip) is a popular application level protocol used in Web content transmission as shown in Figure 1. The Web server compresses the Web page using a zip algorithm then sends it to the Web browser, after the browser successfully receives the compressed Web page, it will use the corresponding decompress algorithm to unzip it and get the original Web page. In this paper, we study the QoS model in the context of application level protocols. Compared with the QoS models in routing and/or MAC layers, the application level QoS model can handle the application-specific quality metrics, such as image sizes and encryption algorithms, and so on. We envision that our approach complements to previous efforts on routing and/or MAC layers very well.

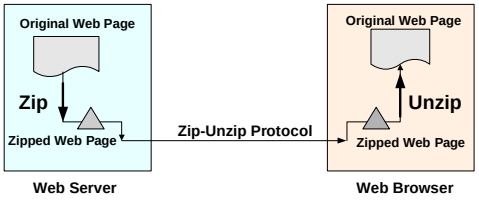

Figure 1: An example of application-level protocols.

2.3 Protocol Domain

In this paper we introduce an overlay concept called *protocol domain*, which is inspired by recent work on delay-tolerant network (DTN) [5] and the proliferation of mobile devices and wireless sensor networks [23]. DTN defines delay-tolerant network gateways interconnected regions that running potentially dissimilar protocol stacks. Although the protocol domain we defined here is similar to region in [5], our protocol domain mainly works at the application level. In each domain, some protocols are available for the application optimization. Figure 2 shows an example of one protocol domain with some peers connected with a protocol domain gateway. The gateway locates on the edge of the domain. It holds those available protocols. Each peer negotiates with the gateway to get a suitable protocol and download it from the gateway according to the application QoS requirement and other situations like the network speed and peer device configurations. The protocol domain is intended to operate above the existing network architectures.

In one protocol domain, there are usually many different peers. Each of them may need different protocols for various application requirements. How to adapt the protocol inside one protocol domain is a key issue. In [14], Fractal, a dynamic application level protocol adaptation approach has been proposed. It uses the mobile code technology for protocol adaptation and leverages existing content distribution networks (CDN) for protocol adaptors (mobile codes) deployment.

3. ENERGY-AWARE QOS METRICS

In e-QoS we extend the QoS to the application session level. It is orthogonal to the network level QoS which may not be able to view the application level information, like the power capacity. With the prevalent of handheld and pervasive computing devices into our daily life, limited battery capacity is a serious impediment to the widespread adoption of running popular applications, e.g., Web Browsing, on this kind of battery powered devices. By observing this dilemma, we consider energy as a major priority in the design of our proposed QoS model. First let us have a look at the energy-aware QoS metrics.

3.1 Session Delay

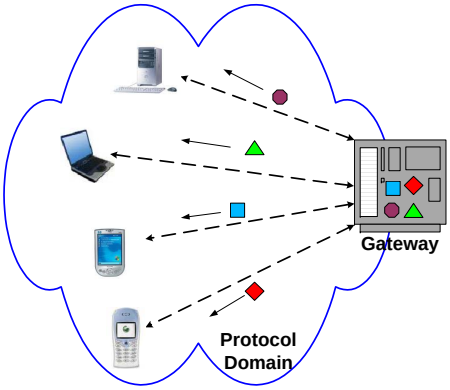

Figure 2: An example structure of a protocol domain.

Delay time is a very sensitive metric for network related application sessions. Session delay is caused by many reasons, such as traffic congestion, low memory, slow hard drive, and so on. Here we only consider the session delay triggered by different algorithms, or in other words, different application level protocols. Utilization of each protocol will incur some delay. Formula (1) shows the evaluation of session delay, which consists of the delay incurred by each involved protocol in the application session. Each individual protocol delay includes several parts, like computing delay and network delay. In [14] several comprehensive formulas and evaluation methods are introduced for delay time of each application protocol.

$$\text{Delay} = \sum_{i=1}^{n} Protocol_i^{delay} \tag{1}$$

3.2 Session Quality

Content quality is another crucial QoS metric especially for some application sessions, like multimedia stream or image transmission. In our model, the quality of each protocol is defined in Formula (2). The original fidelity represents the original data quality, e.g., the original image dimension. The output fidelity is the output data quality after applying the protocol, e.g., the reduced image dimension after the content adaptation protocol. The ratio should be between 0 and 1. The quality of the session is defined as the product of involved protocols qualities in Formula (3).

$$1 \geq Protocol_i^{quality} = \frac{\text{adapted fidelity}}{\text{original fidelity}} \geq 0 \tag{2}$$

$$\text{Quality} = \prod_{i=1}^{n} Protocol_i^{quality} \tag{3}$$

3.3 Session Lifetime

Session lifetime is a new concept we introduced in our paper. As we defined, an application session is the procedure of executing an application function. Then session lifetime is the time period from the start to the end of the application session as shown in Equation (4). Note that the session lifetime is decided by many factors, like the remaining battery capacity, power profiles of involved protocols, even user behaviors, and so on. Some other protocols have constant impact on the session lifetime, for instance, the screen brightness, if we consider it as an application protocol between peer and gateway. Usually, if two ends of an session select the session lifetime as their mutual QoS metric, they will negotiate an expected lifetime value at the first place. Then our QoS model tries to satisfy Formula (5). In order to do this, dynamic selection and adaptation of protocols are necessary. To the best of our knowledge, this is the first energy-aware QoS metric defined for application sessions.

$$Lifetime_{real} = Time_{end} - Time_{start} \tag{4}$$

$$Lifetime_{real} \geq Lifetime_{expected} \tag{5}$$

4. SYSTEM DESIGN

In this section we present the design of the system. A general scenario is shown in Figure 3, there are some protocol domains on the overlay network. The protocol gateway bridges different protocol domains. Note that it is possible that two peers in not adjacent domains want to start an application session. In this situation, there is a path along multiple gateways from one peer to another. How to handle this case is our future work. Next we in turn cover the gateway structure, and protocol adaptation policy.

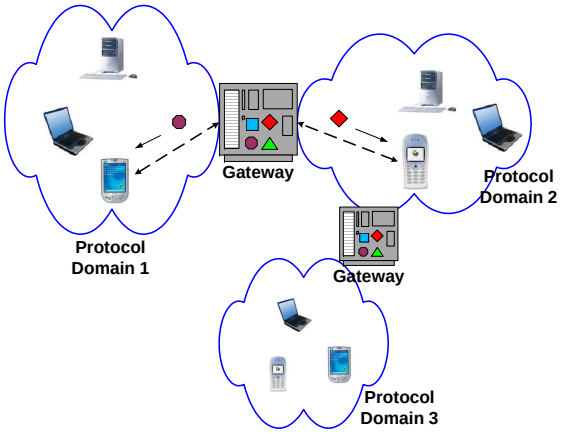

Figure 3: An overview of the system architecture.

4.1 Gateway Structure

Gateway plays an important role in the system. It is in charge of negotiating QoS metrics and protocols with the peers, delivering protocol modules to the peers, monitoring the procedure of application sessions, and adapting the protocols dynamically. In order to finish these functions, gateway needs to know some peer side information, such as remaining energy, network bandwidth etc. We define them into the format of different metadata as shown in Figure 4. The device metadata defines some parameters related to the QoS metrics, like remaining battery percentage, screen brightness, etc. Network metadata is also required in the negotiation procedure. Session metadata records the metadata of included protocols. A general structure of the gateway is shown in Figure 5, which includes a negotiator, a distributor, a session monitor, and a proxy. Each part is running as a daemon on the gateway. Next we will explain the structure and functionality of each module respectively.

The negotiator receives the session and QoS requests from one peer and forwards to the peer on the other side. After both sides make an agreement on the QoS metric, the negotiator will start selecting the proper protocols to satisfy the QoS metric. After the

Device Metadata (DevMeta) = { Remaining battery percent, Screen brightness, CPU speed, Memory size, ... }

Network Metadata (NtwkMeta) = { Network type, Network bandwidth }

Session Metadata (SessionMeta) = { Session ID, *ProtMeta* 1, ... , *ProtMeta* n }

Figure 4: Definitions of metadata.

negotiation is done, it is the distributor's job to deliver the protocol modules to each peer. This is similar to the plugin downloading in Web browser. For the secure execution of the downloaded modules on peer side, several existing security mechanisms can be applied, like digital signature, sandbox, and virtual machine monitor. Therefore we will not propose any new security approach for the deployment of protocols. After the application session starts, the proxy handles protocol translation, content adaptation, session data caching, and so forth. For instance, one peer uses compression protocol to zip the text data while the peer on the other side just uses plain text. Proxy has to zip the text on one way and unzip the text on the other way. There is a cache in the proxy, which is for the temporary storage of the session data in case that peers send or receive data asynchronously. Finally, the monitor begins to monitor the application procedure from the beginning of the session by periodically sampling the peer status. In Section 4.2 the protocol adaptation policy used by the monitor will be presented. Next, we will explain the protocol adaptation policy.

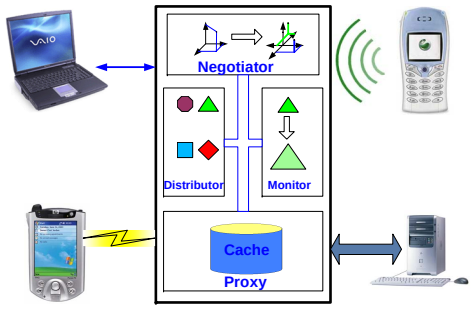

Figure 5: The structure of gateway.

4.2 Protocol Adaptation Policy

Dynamic protocol adaptation is supposed to be executed by the monitor module. One assumption is that the monitor should be able to probe the "pulse" of peers, including device metadata and network metadata. The probe frequency is determined by the monitor according to application sessions. The monitor can pick one sample frequency in the beginning and dynamically change the frequency according to the adaptation effect. For delay and content quality QoS metrics, repeating the selection procedure of the protocol will assure the satisfaction of the metrics, thus it is not the focus of this paper. While for the session lifetime metric, only repeating the selection procedure is not enough, a new adaptation approach is necessary.

We observe that most modern Lithium-Ion batteries follow the similar pattern on the remaining battery versus time curve (Figure 6 in Section 5). By fitting the curve with the linear or polynomial function, the expected current remaining battery percentage, which is the indicator of the session lifetime, can be estimated at any sampling time in the session lifetime. With this reference battery remaining percentage, the monitor module can either adapt the protocol parameters on the proxy or peer side, or find a new protocol for the remaining session lifetime. Obviously, the adapt frequency and the intensity of the adaptation will greatly affect the stability of the system and the user experience, which is our future work. In the next section we will analyze the power profile curve pattern, give out a fast but effective segmented curve fitting linear function.

5. PERFORMANCE EVALUATION

To evaluate the effectiveness and efficiency of our model, we implement a case study, instant messaging between two PocktPCs. The session lifetime is set as the QoS metric. For other QoS metrics, like session delay, our previous work [14] presents enough experimental results. Before move on to the case study, we examine the power profiles of some potential protocols and the segmented curve fitting method.

5.1 Protocol Power Profile

We test several protocol sets on a battery powered PocketPC, HP iPAQ h4150. LCD background light is one culprit for rapid battery draining in daily usage. Choosing correct screen brightness scale is a protocol between a peer and the gateway. Power profiles of different brightness are shown as Figure 6. The x-axis shows the time in thousands of seconds, the y-axis corresponds to the remaining battery percentage. The 100% brightness lifetime which stands up for 3.8 hours is shorter than that of any other brightness options. The smaller the brightness is, the longer the lifetime could be. Finally, 0% brightness lasts up to 40,000 seconds, approximately 11 hours.

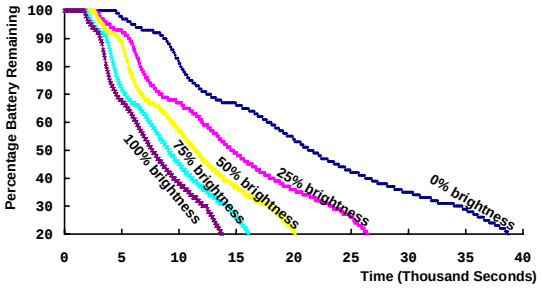

Figure 6: The power profiles of different screen brightness.

Wireless network interface consumes significant power on PocketPC [13] [28]. The HP iPAQ h4150 has an integrated 802.11g wireless card. We test the continuous sending, receiving, and idle power profile as shown in Figure 7. We can see that, first of all, idle state power consumption is small. Its power profile is comparable with that of 0% brightness in Figure 6. However the send and receive lifetime is only roughly one third of the idle case. Then another question is which one is more energy consuming between sending and receiving. Figure 8 answers the question. With the same energy consumed, receiving transfers more than 2,000 MBytes, about 4 times of the transfer size of sending.

Furthermore, for sending and receiving, we tested two different methods as shown in Table 1. Note that *Energy* is presented as the percentage of battery capacity. A lower level parameter, e.g., power reading, may be more useful, however, we think our definition is acceptable since we are using the same device. The first method is that no persistent connection is used. The second one is using persistent connection plus the large chunk. Take the sending scenario as an example, if there are 10 application session data packages each with size of 200 bytes. The total size is 2,000 bytes. In the first method, the connection between peers is setup and tear down 10 times, each time one 200-byte package is sent out. For the second method, the connection is established only once. The large chunk method combines the 10 packages into one large chunk as 2,000 bytes and sends it out. The different energy and time consumptions of these two methods are compared in the table. Similar differences are shown also for the receiving scenario. It is easy to see that the first method, no persistent connection, incurs much more energy and time consumption compared with the second method, persistent connection plus large chunk.

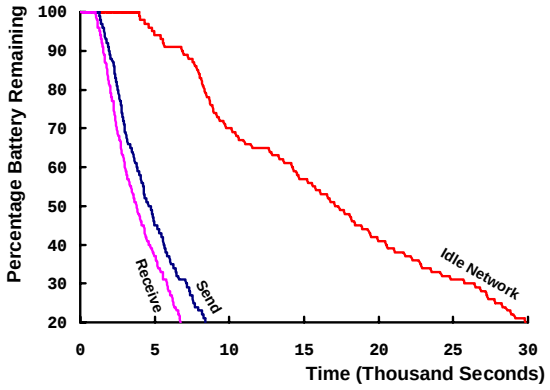

Figure 7: Power profiles of send, receive, and idle network regards time.

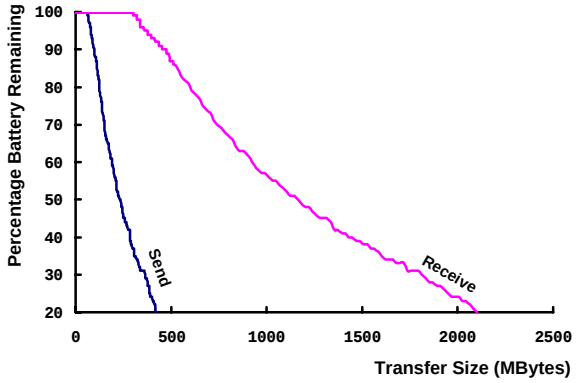

Figure 8: Power profiles of send, and receive regards transfer size.

In summary, the screen brightness and wireless interface are two major energy consumption parts. Based on this observation, we come out several principles for the protocol adaptation: first, choose as low screen brightness as possible; second, reduce the connection times; finally, transfer as much data as possible in one connection. We also evaluate other algorithms power profile for their possible utilization in the case study, such as Gzip, which power consumption is comparable with that of the 100% brightness screen, so that it is too high to be selected. Therefore in our following case study gateway basically selects and adapts protocols related to screen brightness, connection persistence, large chunk, and content adaptation which happens on the proxy side with unlimited power supply. In case study we will give more description about involved protocols.

5.2 Curve Fitting in Protocol Adaptation

Session lifetime has different nature as session delay. If one peer does not send anything out, there is no session delay. But energy keeps reducing as long as the machine is still on. Furthermore, the session lifetime could be seriously compromised if peer starts other unrelated energy consuming process. Consequently, energy and protocol must be monitored and adapted periodically. Our analysis in the above subsection implies that they share a similar pattern, which includes a pure flat start stage followed by a roughly linear regression as shown in Figure 9. Let us use $t1$ to denote the flat stage time period, $t3$ for the total time, b for the bottom battery percent, in previous figure, $b = 20\%$. $t2$ is the knot point between $t1$ and $t3$. m is an adjustment amount bigger than 0 so that the fitting curve between t1 and t3 is not one line but two segmented lines which can fit the real power profile curve more accurately. $\gamma = \frac{t1}{t3}$ and m are relatively stable value set as $\gamma = 11\%$ and $m = 10$. Based on the predicted energy percentage value generated from this curve fitting method, we can adapt the protocol parameter accordingly. Let the initial battery percentage be I, the expected session lifetime be $t3$. For given γ, m, b, I, and $t3$, we can use the following formulas to find $t1$, $t2$, and α, β, which are slopes of the fitting lines of $t2t3$ and $t1t2$.

In case that $I = 100$: $t1 = \gamma \times t3$, $t2 = \frac{1+\gamma}{2} \times t3$, $\alpha = \frac{100-b-2\times m}{(1-\gamma)\times t3}$, $\beta = \frac{100-b+2\times m}{(1-\gamma)\times t3}$

In case that $50 + \frac{b}{2} - m < I < 100$: $\beta = \frac{I-b+2\times m}{t3}$, $t2 = \frac{I-\frac{b}{2}+m-50}{\beta}$, $\alpha = \frac{50-\frac{b}{2}-m}{t3-t2}$

In case that $50 + \frac{b}{2} - m \geq I$: $\alpha = \frac{I-b}{t3}$

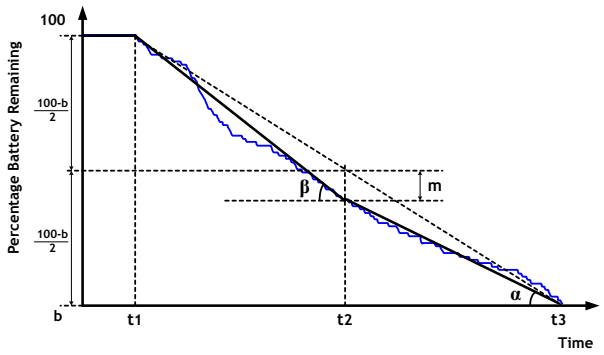

Figure 9: The segmented linear curve fitting method.

5.3 Case Study: PocketPC to PocketPC Instant Messaging

Nowadays more and more communications occur between two handheld devices. In this case, we study the instant messaging session between two PocketPCs with different battery capacities. The experiment platform and hardware configurations are shown in Figure 10, where two PocketPC peers in different protocol domains want to setup an instant message session through the gateway. The two protocols used in this case study are described in Table 2. Be-

	Methods	Size bytes	Energy(batt percent) 80×10^{-6}% of battery capacity	Time seconds
Send	No persistent connection	200×10	10580.47	1041.12
	Persistent connection + Large chunk send	2000	361.53	37.98
Receive	No persistent connection	50000×10	76.15	7.43
	Persistent connection + Large chunk receive	500000	18.59	1.56

Table 1: Energy and time consumption of different send and receive methods.

sides the screen brightness, a message combination protocol is introduced. Usually, in instant message application, people send and receive text information in a short time period. In order to save energy, the message combination protocol prevents the peer from sending and receiving messages too frequently. Instead, it caches the sending messages for a while and sends out multiple messages in one time. The cache in proxy is also used to cache the receiving messages for peers. This technique is very useful in delay tolerant network and peers with intermittent connection. The number of cached messages is called the message combination length, which is decided by the protocol adaptation policy. The bigger it is, the more energy could be saved, also the more delay will be observed and the more bytes have to be transferred in one time.

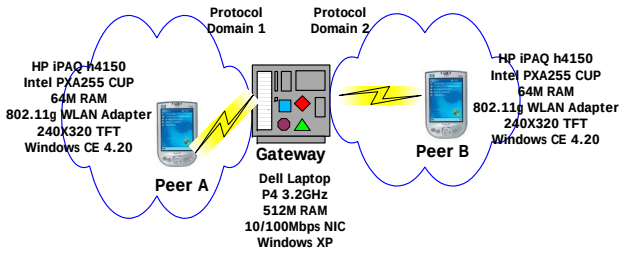

Figure 10: The configuration of experimental platform for case study.

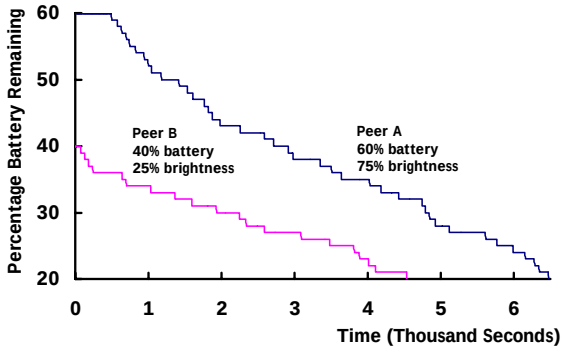

Figure 11: Peer A and B power profile for protocol selection without adaptation.

We assume that peer A has 60% battery and peer B has 40% battery. Their expected session lifetime is 7,000 seconds. Based on the screen brightness protocol, negotiator chooses 75% brightness for peer A and 25% brightness for peer B. We assume both peers send and receive one 256 bytes message in each 3 seconds. Without the protocol adaptation function, none of them can really reach the pre-negotiated session lifetime as we can see in Figure 11. Then the curve fitting is used with parameters as $\gamma = 11\%$, $m = 10$, $b = 20$ and $t3 = 7000$, $I = 60$, $t2 = 1166$, $\alpha = 0.005142$, $\beta = 0.008571$ for peer A, and $t3 = 7000$, $I = 40$, $\alpha = 0.002857$ for peer B.

The protocol adaptation policy works as follows: if the energy is below the predicted value by 1% (of the battery capacity), gateway reduces the screen brightness protocol parameter before increase the message combination length. Otherwise, in case of that the remain energy is beyond the predicted value by 1% (of the battery capacity), the message combination length is reduced if it is not 1 before the screen brightness is increased. The performance of dynamic adaptation for peer A is shown in Figure 12. In the legend, each triangle point is the moment of the screen brightness adaptation. Each circle point marks the message combination adaptation moment. The legend demonstrates the sequences of these two adaptations. Each bar in the legend corresponds to one vertical line in the figure. The legend in Figure 12 shows the adaptation sequences according to the adaptation points marked on the curve. By a series of screen brightness and message combination length adaptation, the session lifetime is extended to 7,000 seconds. The maximum message combination length is 32, which means system combines 32 messages together into one package. If user sends out one message per 3 seconds, he will feel the delay as $32 \times 3 = 96$ seconds, around one and half minutes, which is acceptable.

For peer B, Figure 13 shows that a sequence of message combination length adaptation contributes to the extension of the lifetime. In the legend, the peak length of the combination is 128 message. The corresponding delay is $128 \times 3 = 384$ seconds, approximately 6 minutes. Although it is relatively long, we think it is acceptable for two reasons. First, the session lifetime is greatly extended compared with the original scenario. A tradeoff must be made between energy and delay. Second, most mobile devices experience short disconnected time in the real challenged network [5] for many factors, e.g., weak signal, keeping moving, etc. Hence, a short message delay is acceptable for instant messages on mobile devices. At the end of the session, peer A still has about 30% battery capacity, while peer B has about 500 seconds gap to the specified session lifetime. This means the adaptation policy is over active to peer A but a little more passive to peer B. By tuning the adaptation policy algorithm we believe a more accurate power profile can be achieved.

5.4 Gateway Capacity Performance

Now we are in a position to study the general system capacity of the gateway. Performance of the gateway is greatly determined by the negotiation delay. We setup the gateway on the PlanetLab [24] and use the configuration of case study to examine the negotiation time as shown in Figure 14. It covers the average negotiation time of up to 300 peers sharing one gateway. The x axis is the number of peers. The y axis represents the average negotiation time. Although some fluctuations occur, most of the negotiation times are

Protocol	Priority	Power(batt percent/time)	Delay	Quality	Adaptability
Screen Brightness	0	$\frac{\text{battery percent}}{\text{tested lifetime}}$	0	25%, 50%, 75%, 100%	Yes
Message Combination	1	very small	Long	1	Yes

Table 2: The protocols used in case study.

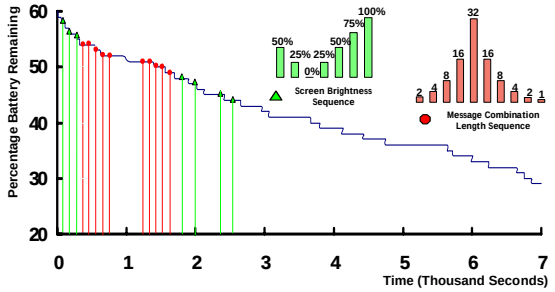

Figure 12: Peer A power profile with adaptation.

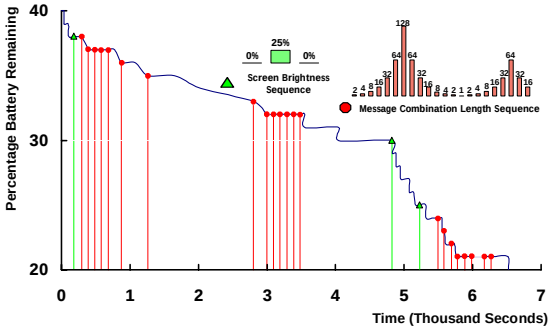

Figure 13: Peer B power profile with adaptation.

between 20 ms to 27 ms. We also show the mean and median line of the measurement data in the figure. Given the fact that the gateway is on a real overlay network built on top of Internet, it is quite normal to see this magnitude of fluctuation. The overall negotiation time remains in a relatively stable range for two reasons. First is the simplicity of energy-aware related protocol selection algorithm. Second is that each peer only needs one time negotiation in the protocol selection stage.

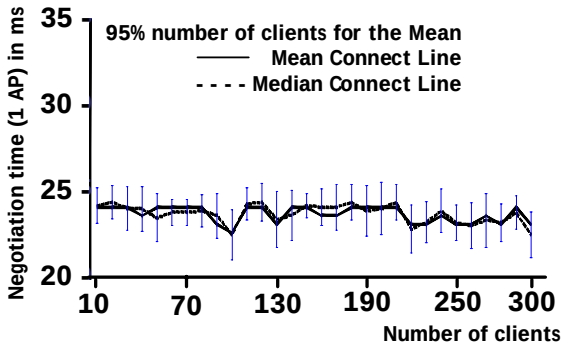

Figure 14: The average negotiation time.

6. RELATED WORK

Our model shares its goals with several recent efforts that aim at power management and application adaptation. We categorize them into three groups as *power management*, *distributed adaptation* and *protocol adaptation*.

Power management Lots of research has emerged to address how to reduce the energy consumption and how to do the power management for laptop, PDA and mobile devices. In terms of working levels, in hardware level, some researchers design the energy-efficient hardware such as [3, 8]. CASTLE [12] uses a hybrid approach that combines a model of the microarchitecture with runtime analysis of processor performance counters to estimate CPU power consumption. In operating system level, EcoSystem [33] explores operating system battery management. It manages energy as any other operating system resource to enforce fairness between applications. *u*sleep [2] implements an aggressive OS-based power management scheme for exploiting idle periods on an Itsy system. From the whole system level, Turducken [27], a hierarchical power management architecture for mobile system, combines diverse platforms, like PDA, wireless sensor node, into a single integrated laptop to reduce the power cost of always-on operation. The Odyssey system [6] makes the trade off between energy and application fidelity. Applications can dynamically modify their behavior to conserve energy. Different from them, our model manages the energy consumption using protocol selection and adaptation from the QoS point of view. Furthermore, it can provide multiple QoS metrics, including not only energy, but also delay and content quality to both sides of a application session.

Some researchers address the power management through network interface. In order to reduce the wireless interface energy consumption, the Wake-on-Wireless (WoW) [26] combines a PDA with a wireless sensor. The device and its wireless network card are shut down when the device is not being used to reduce the idle power. Zhong and Jha [34] propose a low-power low-cost cache device, to which the host computer can outsource simple tasks, as an interface solution to overcome the bottleneck. In [19], an economic model is used as a dynamic, decentralized energy management system which is integrated into the Nemesis OS for energy allocation. The Muse system[4] also employs an economic model to reduce hosting center energy needs by managing server resources. [18] proposes a dynamic game theoretic approach for choosing power optimization strategies for various components. Those efforts either need the new hardware support or use complex algorithms than ours. By naturally using protocol selection and adaptation our model can achieve similar or better performance.

Distributed adaptation Adaptation can be introduced either at the end-points or distributed on intermediate nodes. Odyssey [20], Rover [11] and InfoPyramid [17] are examples of systems that support end point adaptation. Conductor [32] and CANS [7] provide an application transparent adaptation framework that permits the introduction of arbitrary adaptors in the data path between applications and end services. These approaches need some changes to the existing infrastructure for their deployment. Fractal [14] solves the deployment problem by leveraging the existing CDNs technology to distributed protocol adaptors, which are implemented using mobile code. All these previous work complements to our work very well. We are the first to take the energy as a QoS metric for

application sessions.

Protocol adaptation In terms of protocol adaptation, there are network level systems such as [25], in which communicating end hosts use untrusted mobile code to remotely upgrade each other with the transport protocols that they use to communicate. Transformer tunnels [29] and protocol boosters [16] are doing application-transparent adaptation by tuning the network protocol according to the change of network situations. Such systems can deal with localized changes in network conditions but cannot react to changing environments outside the network layer. Our model works at the application layer, it can maximally adapt application level protocols which have no way to be completed in the network layer. Our work is also different from the Web browser plugins, e.g., Realplay, Flash, and so on. Plugin is an application component which completes part of the functionality, incapable of doing protocol adaptation. Our system is a general model to provide the QoS by means of protocol adaptation which has transparency to the client and other characteristics, such as flexibility and extendibility, which plugins do not have.

7. CONCLUSION AND FUTURE WORK

In this paper, e-QoS is proposed to benefit the application session from choosing appropriate protocols according to dynamic end devices and network environments. To the best of our knowledge, this is the first effort on energy-aware QoS on application sessions across multiple protocol domains by means of protocol selection and adaptation, especially for the case that both ends are power limited devices. Session lifetime QoS systems for instant messaging between two PocketPCs have been built in the context of this model. Experiment results show that energy-aware QoS has lightweight system overhead. The proposed adaptation scheme is very effective on energy-aware QoS in terms of the session lifetime. Future work includes finding more power saving protocols especially for interactive application sessions, like instant messaging to reduce the delay time, also integrating this model with end to end service differentiation and access control in a real pervasive computing environment, e.g., a collaborative environment for computer-assisted surgery [15].

8. REFERENCES

[1] B. Baksi, R. Krishna, N. Vaidya, and D. Pradhan. Improving performance of tcp over wireless networks. *Proceedings of the 17th ICDCS*, May 1997.

[2] L. S. Brakmo, D. A. Wallach, and M. A. Viredaz. usleep: A technique for reducing energy consumption in handheld devices. *Proc. Int. Conf. Mobile Systems, Applications, and Services*, June 2004.

[3] A. Chandrakasan and R. Brodersen. Minimizing power consumption in digital cmos circuits. *Proceedings of the IEEE*, p. 83(4):498C523, April 1995.

[4] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle. Managing energy and server resources in hosting clusters. *Proceedings of the 18th Symposium on Operating Systems Principles*, 2001.

[5] K. Fall. A delay-tolerant network architecture for challenged internets. no. Proceedings of ACM Sigcomm 03, Aug. 2003.

[6] J. Flinn and M. Satyanarayanan. Managing battery lifetime with energy-aware adaptation. *ACM Transactions on Computer Systems*, p. 137C179, May 2004.

[7] X. Fu, W. Shi, A. Akkerman, and V. Karamcheti. CANS: Composable, Adaptive Network Services Infrastructure. *Proc. of the 3rd USENIX Symposium on Internet Technologies and Systems (USITS'01)*, pp. 135-146, Mar. 2001.

[8] R. Gonzalez and M. Horowitz. Energy dissipation in general purpose microprocessors. *IEEE Journal of Solid State Circuits*, p. 1277C84, September 1996.

[9] N. C. Hutchinson and L. L. Peterson. The *x*-Kernel: An Architecture for implementing Network Protocols. *IEEE Transactions on Software Engineering* 17(1):64–76, Jan. 1991.

[10] IETF organization. Ldap (v3) revision, 2004, `http://www.ietf.org/ids.by.wg/ldapbis.html`.

[11] A. D. Joseph, J. A. Tauber, and M. F. Kasshoek. Mobile Computing with the Rover Toolkit. *IEEE Transaction on Computers:Special Issue on Mobile Computing* 46(3), Mar. 1997.

[12] R. Joseph and M. Martonosi. Run-time power estimation in high-performance microprocessors. *Proceedings of the 2001 Symposium on Low Power Electronics and Design*, 2001.

[13] R. Kravets and P. Krishnan. Application driven power management for mobile communication. *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking*, Oct. 1998.

[14] H. Lufei and W. Shi. Fractal: A mobile code based framework for dynamic application protocol adaptation in pervasive computing. no. Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium, Apr. 2005.

[15] H. Lufei, W. Shi, and V. Chaudhary. M-CASEngine: A collaborative environment for computer-assisted surgery(abstract). *Proceedings of 2006 Computer Assisted Radiology and Surgery 20th International Congress and Exhibition*, July 2006.

[16] A. Mallet, J. Chung, and J. Smith. Operating System Support for Protocol Boosters. *Proc. of HIPPARCH Workshop*, June 1997.

[17] R. Mohan, J. R. Simth, and C. Li. Adapting Multimedia Internet Content for Universal Access. *IEEE Transactions on Multimedia* 1(1):104–114, Mar. 1999.

[18] S. Mohapatra and N. Venkatasubramanian. Middleware for mobility: A game theoretic approach for power aware middleware. *Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware*, October 2004.

[19] R. Neugebauer and D. Mcauley. Energy is just another resource: Energy accounting and energy pricing in the nemesis os. *Procceedings of the 8thWorkshop on Hot Topics in Operating Systems*, 2001.

[20] B. D. Noble. *Mobile Data Access*. Ph.D. thesis, School of Computer Science, Carnegie Mellon University, May 1998, `http://mobility.eecs.umich.edu/papers/diss.pdf`.

[21] B. D. Noble and et.al. Agile application-aware adaptation for mobility. *Proc. of the 16th ACM Symp. on Operating Systems Principles (SOSP-16)*, Oct. 1997.

[22] NSF Wireless Mobile Planning Group Workshop. New architecture and disruptive technologies for the future internet, Sept. 2005, `http://www.geni.net/wmpg_draft_200508.pdf`.

[23] NSF workshop. Overcoming barriers to disruptive innovation in networking, Jan. 2005, `http://geni.net/barriers_200501.pdf`.

[24] Planetlab, `http://planet-lab.org/`.

[25] P.Patel, A.Whitaker, D.Wetherall, J.Lepreau, and T.Stack. Upgrading transport protocols using untrusted mobile code. *Proc. of the 19th ACM Symposium on Operating Systems Principles*, pp. 1–14, Oct 2003.

[26] E. Shih, P. Bahl, and M. J. Sinclair. Wake on wireless: An event driven energy saving strategy for battery operated devices. *Proceedings of the Eighth ACM Conference on Mobile Computing and Networking*, September 2002.

[27] J. Sorber, N. Banerjee, M. D. Corner, and S. Rollins. Turducken: Hierarchical power management for mobile devices. *Proceedings of the 3rd international conference on Mobile systems, applications, and services*, June 2005.

[28] M. Stemm and R. Katz. Measuring and reducing energy consumption of network interfaces in handheld devices. *IEICE Transactions on Communications*, p. 1125C1131, Aug 1997.

[29] P. Sudame and B. Badrinath. Transformer Tunnels: A Framework for Providing Route-Specific Adaptations. *Proc. of the USENIX Technical Conf.*, June 1998.

[30] P. Sudame and B. Badrinath. On providing support for protocol adaptation in mobile wireless networks. *Mobile Networks and Applications* pp. 43-55, 2001.

[31] W3C Consortium. Simple object access protocol (SOAP) 1.1, 2000, `http://www.w3.org/TR/SOAP/`.

[32] M. Yarvis, A. Wang, A. Rudenko, P. Reiher, and G. J. Popek. Conductor: Distributed Adaptation for complex Networks. *Proc. of the Seventh Workshop on Hot Topics in Operating Systems*, Mar. 1999, `http://lasr.cs.ucla.edu/reiher/papers/yarvis.ps`.

[33] H. Zeng, C. S. Ellis, A. R. Lebeck, and A. Vahdat. Ecosystem: Managing energy as a first class operating system resource. *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems*, 2002.

[34] L. Zhong and N. K. Jha. Energy efficiency of handheld computer interfaces: Limits, characterization and practice. *Proceedings of the 3rd international conference on Mobile systems, applications, and services*, June 2005.