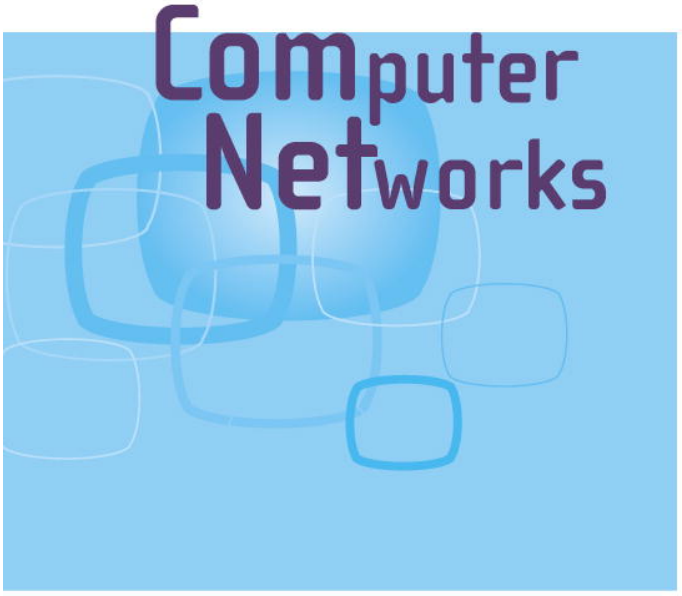
Volume 51 Issue 11 8 August 2007 ISSN 1389-1286



Available online at www.sciencedirect.com
ScienceDirect
Computer Networks 51 (2007) 3125–3141
Computer Networks
www.elsevier.com/locate/comnet

Energy-aware QoS for application sessions across multiple protocol domains in mobile computing

Hanping Lufei, Weisong Shi *

Department of Computer Science, Wayne State University, 5143 Cass Avenue, Detroit, MI 48202, United States

Received 16 November 2006; received in revised form 11 January 2007; accepted 15 January 2007
Available online 25 January 2007

Responsible Editor: I.F. Akyildiz

Abstract

The proliferation of heterogeneous devices and diverse networking technologies demands flexible models to guarantee the quality-of-service (QoS) at the application *session* level, which is a common behavior of many network-centric applications, e.g., Web browsing and Instant messaging. Several QoS models have been proposed for heterogeneous wired/wireless environments. However, we envision that the missing part, which is also a big challenge, is taking energy, a scarce resource for mobile and energy-constrained devices, into consideration. In this paper we propose a novel energy-aware QoS model, *e-QoS*, for application sessions that might across multiple protocol domains, which will be common in the future Internet, rather than an exception. The model provides 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 introducing a new QoS metric called *session lifetime*. To show the effectiveness of the proposed scheme, we have implemented two case studies: Web browsing from a Pocket PC to a regular Web server, and an instant messaging application between two Pocket PCs. In the former case study, our approach outperforms the conventional approach without energy-aware QoS by more than 30% in terms of the session lifetime. In the second case study, we also successfully extend the session lifetime to the value negotiated by two Pocket PCs with very diverse battery capacities.

Keywords: Energy-aware; Quality of service; Application sessions; Protocol adaptation

1. Introduction

With the development of computer and communication technologies, more and more heterogeneous devices, like desktops, laptops, Pocket PCs, and cellular phones are connected to the Internet using diverse networks, like Ethernet, Wi-Fi, Bluetooth, 3G/4G wireless technology. 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 [1]. Thus, we expect to see more and more applications

* Corresponding author. Tel.: +1 313 577 3186; fax: +1 313 577 6868.
E-mail addresses: hlufei@wayne.edu (H. Lufei), weisong@wayne.edu (W. Shi).

1389-1286/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2007.01.006

that having multiple protocol domains involved during one application session. Each domain has its own application protocol set from which the end user can select different protocols for specific application purpose. 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 messaging application session, a user might use a laptop with a cable modem at home, a handheld device with 3G/4G or Bluetooth or even DSRC [2] (Dedicated Short Range Communications) on the highway to the office through intelligent transportation system [3], a desktop with Ethernet LAN in the office and a PDA with satellite radio on the airplane, 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.

To attack the above challenges, we argue that the future mobile computing environment 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 its proactive judgements to guarantee the pre-negotiated QoS metric. Be aware of the necessity of application-level QoS architecture, 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 in intelligent transportation systems* – 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 [4], LDAP [5], 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.
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 [6–10] work at or below the network layer. Such systems can cope with localized changes in network conditions but cannot adapt to variations above the network layer. Any approach that requires the MAC or network layer modifications has to face the deployment challenge since all involved machines have to change their protocol stack in order to take advantage of the improvement brought the approach. Although application level adaptation brings an extra overhead, comparing with the network and MAC layer approach, the evaluation results with two case studies show that the effect of our approach on system performance is in no way to be significant. In terms of power management, most of existing approaches [11–14] 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 two energy-aware application: Pocket PC to Web server browsing and instant messaging between two Pocket PCs* – Several protocols are developed for these two applications. Experiment results show that the session lifetime has been successfully extended to the value negotiated by two peers even 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 two case studies as the Pocket PC to Web server browsing and an instant messaging application between two Pocket PCs. Finally, related work and conclusions are listed in Sections 6 and 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, Gzip is a popular application level protocol used in Web content transmission as shown in Fig. 1. The Web server compresses the Web page using Gzip algorithm then sends it to the Web browser, after the browser successfully receives the zipped Web page, it will use the Gzip algorithm to unzip it and get the original Web page. In this paper, we only address the application level protocol in our e-QoS model. Compared with the QoS models in route or MAC level, the application level QoS model can handle the quality metrics above the route level or the MAC level, for example, the image size.

In this paper, a protocol is described as a collection that consists of the implementation algorithm, the delay, quality, and power profiles as follows:

$$\text{Protocol} = \{\text{Algorithm}, \text{Delay Profile}, \text{Quality Profile}, \text{Power Profile}\}.$$

With this definition, when a protocol is evaluated, its influence on the application session qualities can be quantified. For instance, how much time has to be spent on the execution of the protocol, how much power it will consume, and so on.

2.3. Protocol domain

In this paper we introduce an overlay concept called *protocol domain*, which is inspired by recent

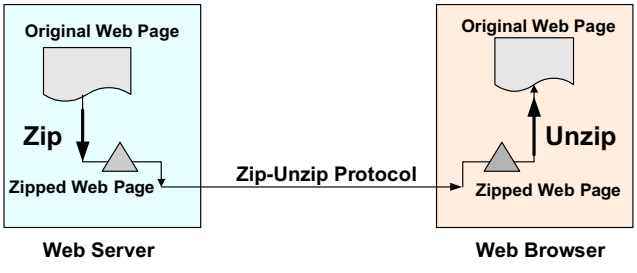


Fig. 1. An example of application-level protocols.

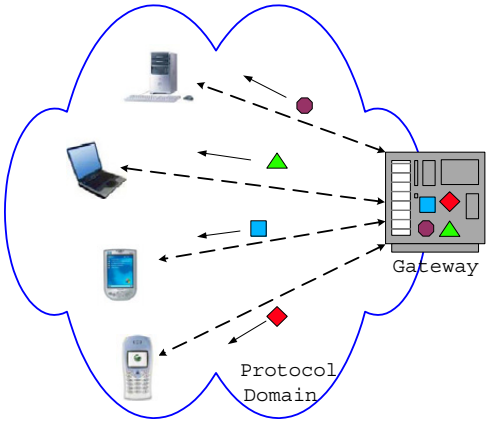


Fig. 2. An example structure of a protocol domain.

work on delay-tolerant network (DTN) [15] and the proliferation of mobile devices and wireless sensor networks [16]. 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 [15], our protocol domain mainly works at the application level. In each domain, some protocols are available for the application optimization. Fig. 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.

The structure and functions of the gateway will be presented in Section 4.2. A typical sensor network is a protocol domain in which all the sensors connect with the sink node that acts like a domain gateway. An ad-hoc network built for disaster rescue is another example of protocol domain. For Internet applications, the current ISP provider could be a good candidate of the gateway because all the service subscribers need to connect to the ISP controlled nodes to access the outside Internet. Although right now the ISP only provides the network layer connection service, we believe with the emergence of more and more diverse computing devices on the network and new applications, providing and adapting application-level protocol to meet application requirements is a must-have function of the future Internet service provider as envisioned in [16].

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

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 [17] several comprehensive formulas and evaluation methods are introduced for delay time of each application protocol

$$\text{Delay} = \sum_{i=1}^{n} \text{Protocol}_i^{\text{delay}}. \quad (1)$$

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 ses-

sion is defined as the product of involved protocols qualities in Formula (3).

$$1 \geqslant \text{Protocol}_i^{\text{quality}} = \frac{\text{adapted fidelity}}{\text{original fidelity}} \geqslant 0, \quad (2)$$

$$\text{Quality} = \prod_{i=1}^{n} \text{Protocol}_i^{\text{quality}}. \quad (3)$$

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 Eq. (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 a 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

$$\text{Lifetime}_{\text{real}} = \text{Time}_{\text{end}} - \text{Time}_{\text{start}}, \quad (4)$$

$$\text{Lifetime}_{\text{real}} \geqslant \text{Lifetime}_{\text{expected}}. \quad (5)$$

3.4. Session QoS space

We define a three dimensional session QoS space to manifest the relations between these three quality metrics. For each point in the space, it corresponds to the selected protocols for the session. In Fig. 3, QoS point P_A represents those protocols, from protocol 1 to protocol n, that are included in the application session. Q_A, D_A and L_A are the corresponding session quality, delay, and lifetime respectively. However different set of selected protocols may affect other two QoS metrics. For example, given L_A as the expected session lifetime, some protocols will be selected based on their power profile so that the total session lifetime will be optimized to be close to L_A. Consequently, we can get the Q_A and D_A related with L_A. It is possible that another QoS point

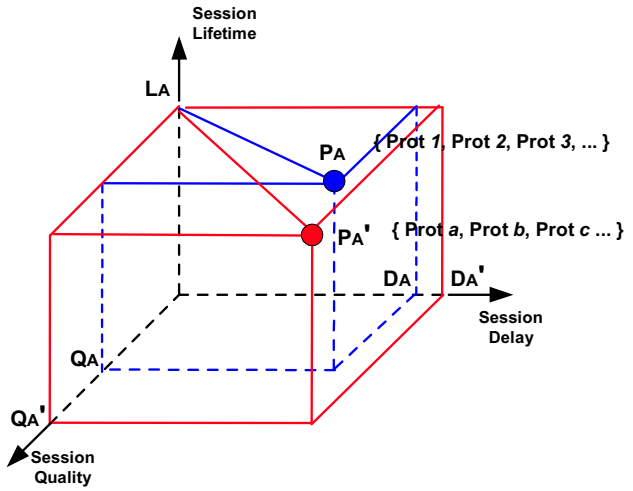


Fig. 3. The session QoS space.

P'_A set of protocols may also achieve the same QoS metric value L_A but with Q'_A and D'_A different from Q_A and D_A. In summary, if the user of the session specifies one QoS metric, there exists a mechanism to select appropriate protocols from the candidates to satisfy the specified QoS metric. In Section 4 we will explain this mechanism in more details.

4. System design

Now we are in a position to present the design of the system. After an overview, we in turn cover the gateway structure, the QoS negotiation procedure, and the protocol adaptation policy.

4.1. System overview

Our system works at the application level. A general scenario is shown in Fig. 4, there are three

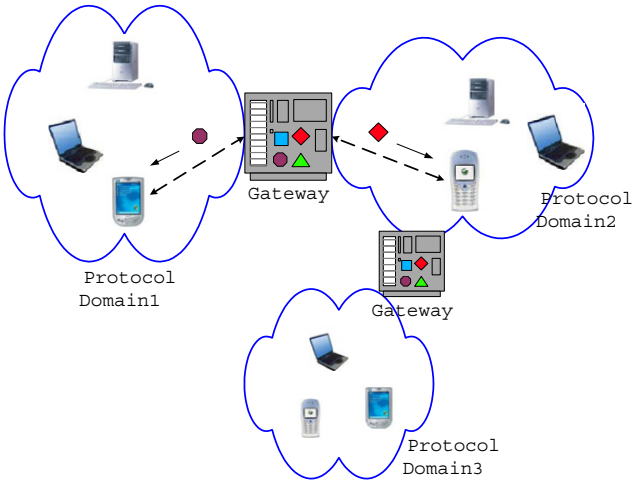


Fig. 4. An overview of the system architecture.

protocol domains on the overlay network. The protocol gateway bridges different protocol domains. If peer A in domain 1 wants to start an application session with peer B in domain 2, they need to negotiate through the protocol gateway which connects domain 1 and domain 2. Based on their mutual QoS requirements the gateway then chooses different protocols according to their diverse conditions like network speed, remaining energy capacity, and so on. After both peers download and install the protocol modules, they can start the application session under the surveillance of protocol gateway. Note that it is possible that two peers in not adjacent domains want to start an application session. For instance the peer in domain 1 wants to talk with a peer in domain 3 in Fig. 4. In this situation, there is a path along multiple gateways from one peer to another. One solution could be that the first and last gateway on the path do the protocol selection and adaptation for his own peer. The gateways in the middle of the path just receive and forward the session data. More complex approach can involve all of the gateways on the path to do the adaptation. How to handle the cases that involve multiple gateways is our next step. Next, we will introduce the structure and functions of the gateway.

4.2. 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 Fig. 5. 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 Fig. 6, 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. In order to show the function of negotiator more clearly, in the next section we will go through the whole QoS negotiation procedure. After the 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

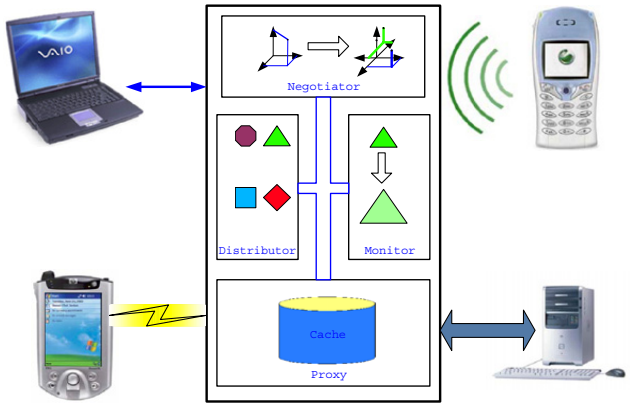


Fig. 6. The structure of gateway.

*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 }

Fig. 5. Definitions of metadata.

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 4.4, the protocol adaptation policy used by the monitor will be presented. Next, we will explain the QoS negotiation procedure.

4.3. QoS negotiation procedure

Generally, the QoS negotiation procedure performs in two steps. First, two peers discuss and settle down a mutual QoS metric. Second, according to peer's situation, like device and network metadata, the gateway selects protocols for each peer. An interactive negotiation protocol (INP) is proposed for the interactions between peers and gateway, as shown in Fig. 7. We assume both the peer side and gateway side understand the protocol definitions. This can be easily implemented by running a light client stub program on the peer side.

At the beginning of the negotiation, the peer A decides to start an application session with the peer B. By specifying the application and desired QoS metric value in the client stub program, peer A first sends `INIT_REQ`, which contains application request and QoS metric value in payload, to the gateway to initialize the protocol negotiation. Each packet has an *INP header* segment, which is used to maintain the interactive negotiation protocol integrity, and we will omit the details in the *INP header*. The gateway then sends back the `INIT_REP` as well as `PEER_META_REQ`, having empty *DevMeta* and *NtwkMeta* to be filled by the peer A, to acknowledge the request and ask some information about the peer. At the same time, gateway will forward the `INIT_REQ` together with `PEER_META_REQ` to peer B to get his feedback about the QoS metric proposed by peer A and his metadata information. After receiving the reply, the client stub program running on each peer will get the content of *DevMeta* and *NtwkMeta* by locally probing the system using system calls. `PEER_META_REP`, containing the collected metadata, is finally sent back to the gateway in step 3. If peer A and B can not make agreement on the QoS metrics, steps 1–3 will be repeated. Based on the QoS metric value and metadata, the negotiator in the gateway will compute and select some protocols for peer A and peer B in step 4. Note that peer A and B may not be provided with the same protocols because of their diverse situations. Then it is the distributor's job to deliver the protocols to each peer in step 5 if the protocol needs the peer side deployment. For some protocols it may not require any deployment on the peer side. For example, gateway only sends image with dimension up to 200×300 to peer B, if incoming image from peer A is bigger, proxy will do the content adaptation before forwarding to peer B. Therefore, peer B may not even know the existence of this protocol happened between him and the gateway. After the security check and protocol deployment, the peer A sends out the `SESSION_REQ` to a proxy and monitor demons that are created by the gateway for this session in step 6. The `SESSION_REQ` contains the real application session related request. From now on the peer A and B continue the application session using the downloaded protocol. The formats of all message types used in INP are listed in the bottom of Fig. 7.

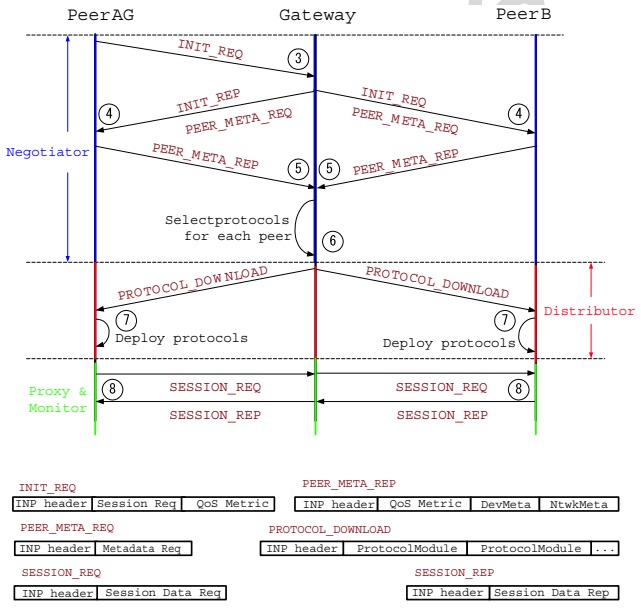


Fig. 7. The interactive negotiation protocol.

Selecting the protocol in step 4 is the key part of the QoS negotiation procedure. Although the monitor module can probe the peers periodically in the application session and adapt the protocol dynamically to fulfil the QoS metric, the accuracy of original selection of protocols greatly decides the probability of finally successful QoS provision. From the QoS space's point of view, given the mutual QoS metric of peer A and B, gateway is able to find a QoS point supported by some protocols to achieve the selected QoS metric and optimize other two as much as possible, as we have seen in Fig. 3

for one peer. Since the QoS metric is the mutual interest of both peers, the gateway needs to combine the two QoS spaces jointly at the mutual QoS metric axis as shown in Fig. 8. Two QoS points that map to the same value on the mutual QoS metric axis will be found in the joint QoS spaces. In the Fractal framework [17], Lufei and Shi have proposed the notion of protocol adaptation tree and an adaptation path search algorithm to find the set of protocols to achieve the minimal delay time. It organizes the protocols as a tree structure in which each protocol is a node except root. Similar approach can be used to evaluate the best session quality. In two QoS joint space, Fractal's method can still be applied to achieve specific delay time or quality metric. However, as far as the session lifetime is concerned, Fractal's method does not perform well because it can not react to the unpredictable user behavior which influence the session lifetime substantially. We tested several protocol's power profiles for some typical battery powered devices and observe that a priority-based protocol selection together with the curve fitting dynamic protocol adaptation is an efficient technique to guarantee the session lifetime QoS metric even in some unpredictable user behavior situations. In evaluation section, we will give a thorough description.

4.4. Protocol adaptation policy

Dynamic protocol adaptation is supposed to be executed by the monitor module. One assumption

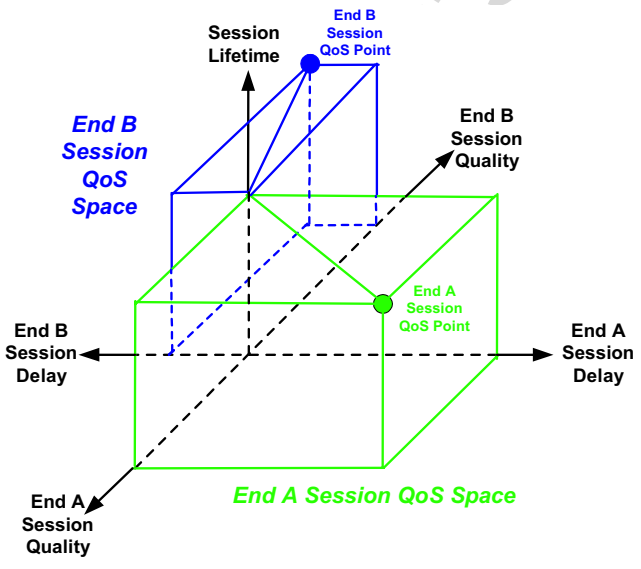


Fig. 8. The joint QoS spaces of two peers.

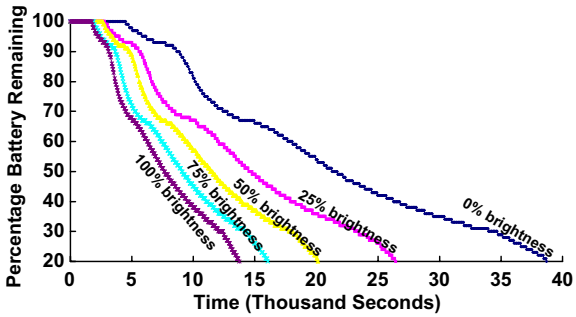


Fig. 9. The power profiles of different screen brightness.

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 (Fig. 9 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 two case studies, Pocket PC to Web Server browsing and instant messaging between two Pocket PCs. The session lifetime is set as the QoS metric. For other QoS metrics, like

session delay, our previous work [17] presents enough experimental results. Before move on to the case studies, we examine the power profiles of some potential protocols and the segmented curve fitting method.

5.1. Protocol power profiles

We test several protocol sets on a battery powered Pocket PC, 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 Fig. 9. 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 h 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 s, approximately 11 h.

Wireless network interface consumes significant power on Pocket PC [18,19]. The HP iPAQ h4150 has an integrated 802.11g wireless card. We test the continuous sending, receiving, and idle power profile as shown in Fig. 10. We can see that, first of all, idle state power consumption is small. Its power profile is comparable with that of 0% brightness in Fig. 9. 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. Fig. 11 answers the question. With the same energy consumed, receiving transfers more than 2000 MBytes, about four 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 2000 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 2000 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.

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

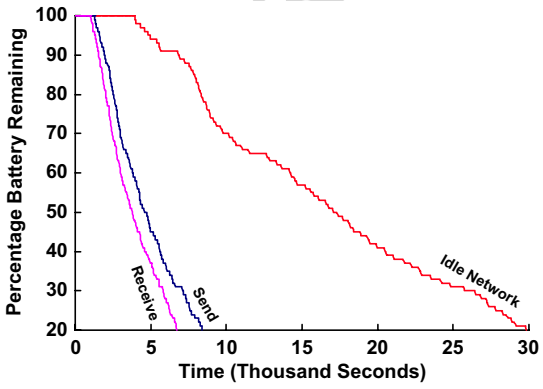


Fig. 10. Power profiles of send, receive, and idle network regards time.

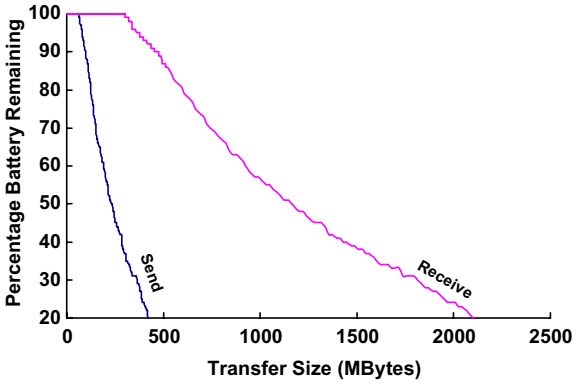


Fig. 11. Power profiles of send, and receive regards transfer size.

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

	Methods	Size (bytes)	Energy (batt percent, 80×10^{-6}% of battery capacity)	Time (s)
Send	No persistent connection	200×10	10,580.47	1041.12
	Persistent connection + Large chunk send	2000	361.53	37.98
Receive	No persistent connection	$50{,}000 \times 10$	76.15	7.43
	Persistent connection + Large chunk receive	500,000	18.59	1.56

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 Fig. 12. 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 $50 + \frac{b}{2} - m \geqslant I$: $\alpha = \frac{I-b}{t3}$.

We use the curve fitting method to represent the energy consumption model for portable devices. The approach looks simple, however, it works well in our two case studies as shown in the following sections. We are aware of several power consumption and management efforts, such as CASTLE [25], EcoSystem [14] and so on. Nevertheless, they need either extra new hardware or major operating system modification, which are above that most mobile devices can afford. On the country, our model performs entirely in the application level, and easy to use. Furthermore, with the gateway handling most of the computing workload, the method has almost unnoticeable lightweight footprint on the mobile device side.

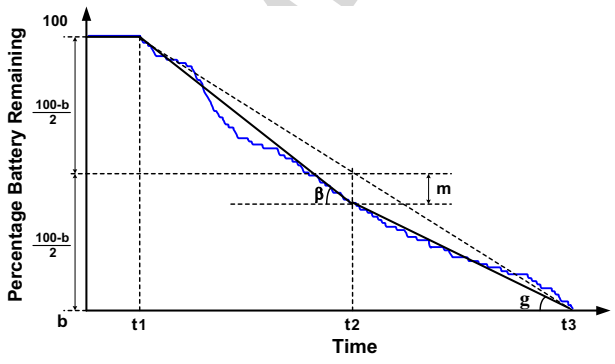


Fig. 12. The segmented linear curve fitting method.

5.3. Case study 1: pocket PC to web server browsing session

In first case study, one peer is a Pocket PC with 100% battery power. On the other side is a Web server with unlimited power. The experimental platform and hardware configurations are shown in Fig. 13. Gateway and Web server are on the same laptop with unlimited AC power supply. There are static Web pages on the Web server. Each page includes one 26KB HTML file and five 600×600

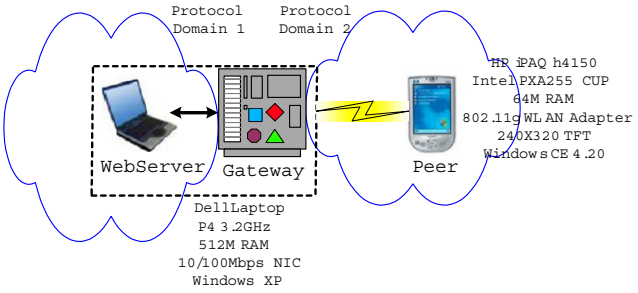


Fig. 13. The configuration of experimental platform for case study 1.

52KB images. In the following experiments, the bottom battery percentage is set as 20%. In the first test, peer has 100% initial battery percentage, 75% screen brightness, no connection persistence, and requests a Web page every 1 minute. As a base case, there is no QoS, no protocol selection and adaptation. The result is shown as the original curve in Fig. 14. The Pocket PC lasts around 11,500 s before the battery reaches the bottom capacity.

The second test uses only the protocol selection function but without the protocol adaptation. The protocols defined for this case study is shown in Table 2. Besides protocol algorithm or name, power, delay, and quality, the priority and adaptability are also listed for each protocol. Since screen brightness is one of the most energy consuming component, its priority is set to 0, as the highest priority, which means the expected session lifetime will be estimated according to the power profile of this protocol. Based on Table 1, connection persistence plus large chunk (CPLC) protocol is the default selection since it reduces energy consumption without the delay and quality deterioration. Content adaptation, for this case, the image size adaptation, does not consume Pocket PC energy since it is done

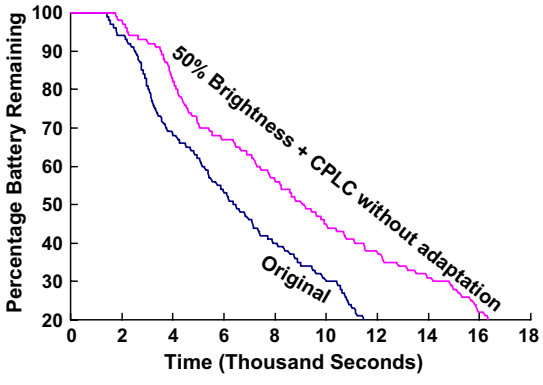


Fig. 14. Power profiles for constant web page request rate.

on proxy. For delay time, we consider it as negligible for the powerful gateway machine unless it is in extreme heavy load. We will address the capacity of gateway in Section 5.5. Regards quality, content adaptation changes the original image dimension into 10%, 50%, or keep the original size.

In the beginning of the negotiation procedure, the stub program on Pocket PC will prompt user to input his QoS metric and desired value as Fig. 15a shows. Suppose user inputs session lifetime and specifies 4 h 26 min and 20 s, or 16,000 s as session lifetime, the negotiator will check the high priority protocol, the screen brightness power profile. Lifetime of 50% brightness is close to the specified session lifetime. Then a message box on peer side informs user to change the screen brightness to 50%. Connection persistence protocol module is also downloaded and deployed on the client side before session starts. Now, the 50% screen brightness and CPLC protocols are selected without adaptation. The energy versus the session lifetime is illustrated as the right curve in Fig. 14. It shows that the selected protocols successfully extends the lifetime to the specified value, just above 16,000 s, which is about 30% more than the original one.

In above scenario, however, we assume that user keeps the sending frequency as exactly one page per minute as the bottom line shows in Fig. 16 in which the x-axis is the time and the y-axis is the user request frequency in the number of Web pages per minute. It is unrealistic to require a user to browse the web page at a constant rate. More practically, if the user changes his behavior by following the dynamically changing rate as the top curve (burst-like) in Fig. 16, it is possible that the previous method will fail to satisfy the expected session lifetime. In this case, the protocol adaptation is the key to alleviate the extra energy drop incurred by the dynamic user behavior and save the session lifetime. Therefore, the top curve in Fig. 16 is used as the user behavior input in the next test. The curve fitting method has $\gamma = 11\%$, $m = 10$, $b = 20$ and input as $I = 100$, $t3 = 16000$. Calculated from the curve fitting equations, $t1 = 1760$, $t2 = 8880$, $\alpha = 0.004213$, and $\beta = 0.007022$. The probe frequency is one sample per 160 s. The adaptation policy works as follows: if the sample energy is less than the predicted value generated from the curve fitting, by 2% (of the battery capacity) or more, we bring the content adaptation protocol into the system and begin reducing the dimension of the images until reach the 10%, then reduce the screen bright-

Table 2
The protocols used in case study 1

Protocol	Priority	Power	Delay	Quality	Adaptability
Screen brightness	0	$\frac{\text{battery percent}}{\text{tested lifetime}}$	0	25%, 50%, 75%, 100%	Yes
Connection persistence + large chunk (CPLC)	1	Very small	0	1	No
Content adaptation	2	0	Very small	10%, 50%, 100%	Yes

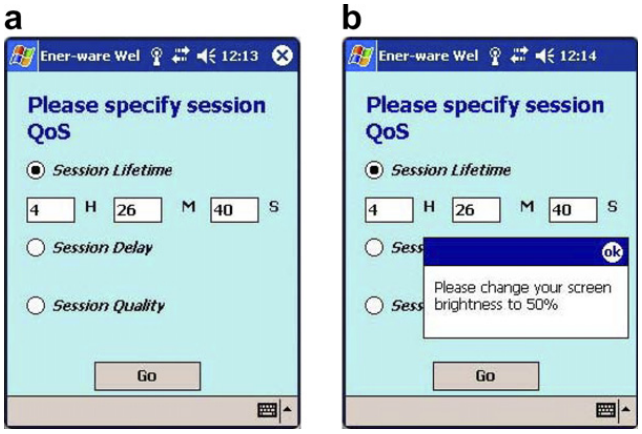


Fig. 15. Screen snapshots of (a) QoS metrics specification interface and (b) screen brightness adjustment message box.

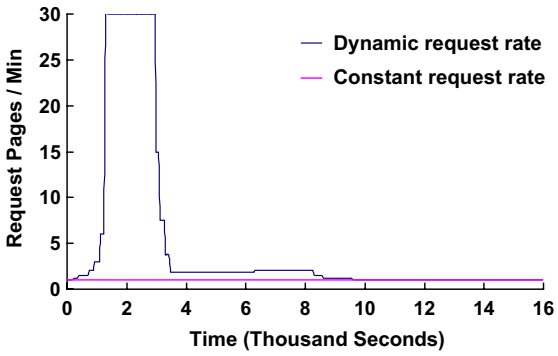


Fig. 16. Dynamic and constant Web page request rates.

ness one level down each time. On the contrary, if the sample energy is more than the predicted value on the fitting curve, by 3% or more, the image size is increased if it is below 100% before brightness is enhanced. The effect of adaptation is clearly shown on the top curve in Fig. 17. It starts with the same protocols, 50% brightness and CPLC as the top curve in Fig. 14. Each triangle point is the moment of content adaptation. Each circle point marks the screen brightness adaptation moment. The legend demonstrates the sequences of these two adaptations. Each bar in the legend corresponds to one vertical line in the figure. By four adaptation periods

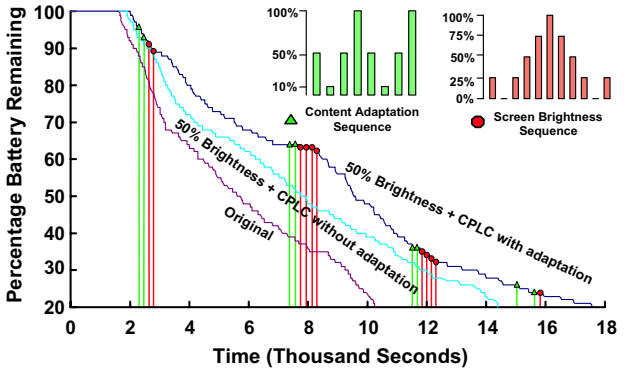


Fig. 17. Power profiles for dynamic request rate.

happened at around 2.5, 8, 12, and 16 thousand seconds, the session lifetime is extended to 17,500 s beyond the expected value, 16,000. For comparison, two approaches used in Fig. 14, original and 50% brightness plus CPLC without adaptation are also tested under the dynamic user behavior. The bottom two curves in Fig. 17 are their performance curves. None of them can really satisfy the specified session lifetime, 16,000 s. The adaptation approach outperforms the original one by more than 7000 s and another one (CPLC) by more than 3000 s.

5.4. Case study 2: pocket PC to pocket PC instant messaging

Nowadays more and more communications occur between two handheld devices. In this case, we study the instant messaging session between two Pocket PCs with different battery capacities. The experiment platform and hardware configurations are shown in Fig. 18, where two Pocket PC peers in different protocol domains want to setup an instant messaging application session through the gateway. The two protocols used in this case study are described in Table 3. Besides the screen brightness, a message combination protocol is introduced. Usually, in instant messaging application, people send and receive text information in a short

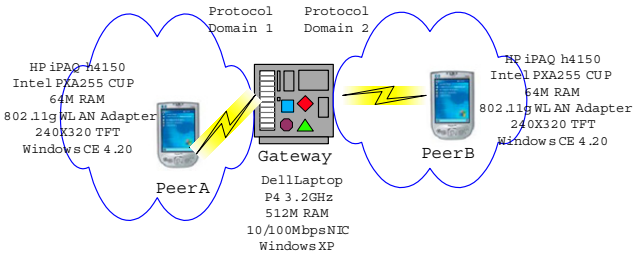


Fig. 18. The configuration of experimental platform for case study.

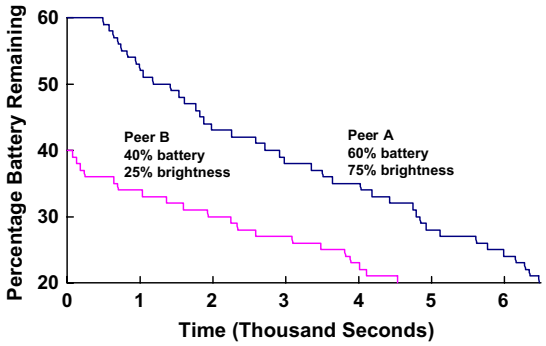


Fig. 19. Peer A and B power profile for protocol selection without adaptation.

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.

We assume that peer A has 60% battery and peer B has 40% battery. Their expected session lifetime is 7000 s. 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 s. Without the protocol adaptation function, none of them can really reach the pre-negotiated session lifetime as we can see in Fig. 19. 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 Fig. 20. 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 Fig. 20 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 7000 s. 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 s, he will feel the delay as $32 \times 3 = 96$ s, around one and half minutes, which is acceptable.

For peer B, Fig. 21 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 $128 \times 3 = 384$ s, 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 trade-off must be made between energy and delay. Second, most mobile devices experience short disconnected time in the real

Table 3
The protocols used in case study

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

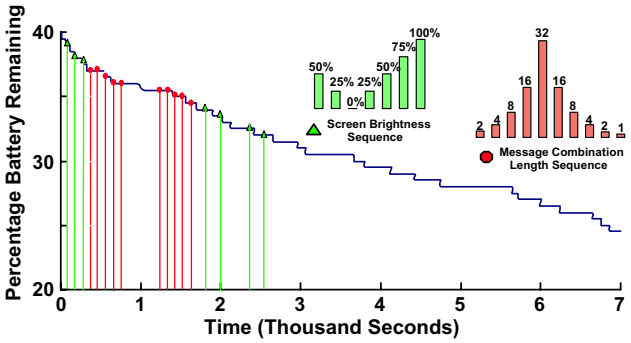


Fig. 20. Peer A power profile with adaptation.

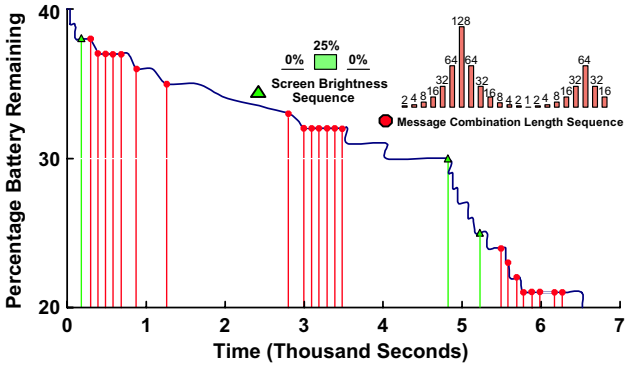


Fig. 21. Peer B power profile with adaptation.

challenged network [15] for many factors, e.g., weak signal, keeping moving, etc. Hence, a short message delay is acceptable for instant messaging applications on mobile devices. At the end of the session, peer A still has about 30% battery capacity, while peer B has about 500 s 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.

It is worth noting that we are aware of that multimedia streaming applications are one of the dominating applications of future Internet, but we do not adopt a multimedia streaming application in our case study because of the following two reasons. First, we observed that possible adaptation protocols for streaming applications include no more than two levels: *user experience adaptation*, such as brightness, window size, frame rate, and so on, and *transmission adaptation*, such as, content adaptation, frame compression algorithm and communication optimization protocols. We argue that the adaption of both brightness and the communication overhead has been shown to be effective for different QoS metrics in two case studies. Thus, we believe that our system will have the similar performance for streaming applications as that of two case studies. Second, the main contribution of this paper is about the QoS space design and the adaptation framework, not the specific adaptation protocol for one specific application. The above two case studies are comprehensive enough to show the benefits. Applying the proposed framework to streaming applications is an interesting direction and deserve future investigation.

5.5. 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 [20] and use the configuration of case study to examine the negotiation time as shown in Fig. 22. 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 between 20 ms and 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.

5.6. Implications and discussions

Based on the evaluation results of two case studies, we derive several implications for building future mobile applications. First, for the screen brightness of the Pocket PC, so far we only use

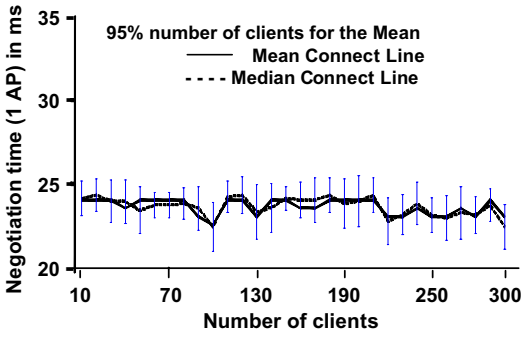


Fig. 22. The average negotiation time.

the setting menu to manually choose different bright levels. An application programming interface would be very beneficial to the energy-concerned mobile applications because the screen brightness can greatly affect the draining time of the battery. Second, HTTP protocol is widely used in Web applications. HTTP/1.1 in RFC 2616 [21] has a key feature as persistent connections, similar as the CPLC protocol used in case study 1. However many Web servers and clients do not really support persistency connection [22]. Our experiments show that persistent connection can save the battery energy significantly. Furthermore, the large chunk approach can enhance the benefit of persistent connection. We argue that all future mobile applications should adopt the persistent connection and large chunk communication for energy saving purposes.

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*.

6.1. Power management

Energy is a scarce resource especially for mobile devices powered by batteries. The speed of battery capacity increasing is much slower than that of energy consumption due to faster processor, larger memory, and bigger screens, and so on. Lots of research has emerged to address how to reduce the energy consumption and how to do the power management for laptops, PDAs and mobile devices. In terms of working levels, in hardware level, some researchers design the energy-efficient hardware such as [23,24]. CASTLE [25] 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 [14] explores operating system battery management. It manages energy as any other operating system resource to enforce fairness between applications. *u*sleep [26] implements an aggressive OS-based power management scheme for exploiting idle periods on an Itsy system. From the whole system level, La Porta et al. [27] optimize the sensor movement for energy efficiency. Turducken [13], a hierarchical power management architecture for mobile system, combines diverse platforms, like Pocket PC, wireless sensor node, into a single integrated laptop to reduce the power cost of always-on operation. Zhu and Cao [28] raise a power-conserving service model for streaming applications over wireless networks. The Odyssey system [11] 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. Jung and Vaidya propose a power control MAC protocol for Ad Hoc networks [29] that allows nodes to vary transmit power level on a per-packet basis. In order to reduce the wireless interface energy consumption, the Wake-on-Wireless (WoW) [12] 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 [30] 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 [31], 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 [32] also employs an economic model to reduce hosting center energy needs by managing server resources. Ref. [33] 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.

6.2. Distributed adaptation

Adaptation can be introduced either at the endpoints or distributed on intermediate nodes. Odyssey [34], Rover [35] and InfoPyramid [36] are examples of systems that support end point adaptation. Conductor [37] and CANS [38] 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 [17] 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.

6.3. Protocol adaptation

In terms of protocol adaptation, there are network level systems such as [9], 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 [39] and protocol boosters [40] 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. Conclusions

In this paper, e-QoS is proposed to benefit the application session across multiple application domains 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 Pocket PCs 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. Possible future directions include 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 the networks.

References

[1] NSF Wireless Mobile Planning Group Workshop, New architecture and disruptive technologies for the future internet, September 2005. [Online]. Available from: <http://www.geni.net/wmpg_draft_200508.pdf>.
[2] dsrc, Dedicated short range communications (DSRC) home. [Online]. Available from: <http://grouper.ieee.org/groups/scc32/dsrc/>.
[3] ITS, Intelligent transportation systems. [Online]. Available from: <http://www.its.dot.gov/>.
[4] W3C Consortium, Simple object access protocol (SOAP) 1.1, 2000. [Online]. Available from: <http://www.w3.org/TR/SOAP/>.
[5] IETF organization, Ldap (v3) revision, 2004. [Online]. Available from: <http://www.ietf.org/ids.by.wg/ldapbis.html>.
[6] B. Baksi, R. Krishna, N. Vaidya, D. Pradhan, Improving performance of TCP over wireless networks, in: Proceedings of the 17th ICDCS, May 1997.
[7] N.C. Hutchinson, L.L. Peterson, The *x*-Kernel: an architecture for implementing network protocols, IEEE Transactions on Software Engineering 17 (1) (1991) 64–76.
[8] B.D. Noble et al., Agile application-aware adaptation for mobility, in: Proc. of the 16th ACM Symp. on Operating Systems Principles (SOSP-16), October 1997.
[9] P. Patel, A. Whitaker, D. Wetherall, J. Lepreau, T. Stack, Upgrading transport protocols using untrusted mobile code, in: Proc. of the 19th ACM Symposium on Operating Systems Principles, October 2003, pp. 1–14.
[10] P. Sudame, B. Badrinath, On providing support for protocol adaptation in mobile wireless networks, Mobile Networks and Applications (2001) 43–55.
[11] J. Flinn, M. Satyanarayanan, Managing battery lifetime with energy-aware adaptation, in: ACM Transactions on Computer Systems, May 2004, p. 137–179.
[12] E. Shih, P. Bahl, M.J. Sinclair, Wake on wireless: an event driven energy saving strategy for battery operated devices, in: Proceedings of the Eighth ACM Conference on Mobile Computing and Networking, September 2002.
[13] J. Sorber, N. Banerjee, M.D. Corner, S. Rollins, Turducken: hierarchical power management for mobile devices, in: Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services, June 2005.
[14] H. Zeng, C.S. Ellis, A.R. Lebeck, A. Vahdat, Ecosystem: managing energy as a first class operating system resource, in: Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems, 2002.
[15] K. Fall, A delay-tolerant network architecture for challenged internets, in: Proceedings of ACM Sigcomm 03, August 2003.
[16] NSF workshop, Overcoming barriers to disruptive innovation in networking, January 2005. [Online]. Available from: <http://geni.net/barriers_200501.pdf>.
[17] H. Lufei, W. Shi, Fractal: A mobile code based framework for dynamic application protocol adaptation in pervasive

computing, in: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium, April 2005.
[18] R. Kravets, P. Krishnan, Application driven power management for mobile communication, in: Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking, October 1998.
[19] M. Stemm, R. Katz, Measuring and reducing energy consumption of network interfaces in handheld devices, in: IEICE Transactions on Communications, August 1997, p. 1125–1131.
[20] Planetlab. [Online]. Available from: <http://planet-lab.org/>.
[21] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Bernes-Lee, RFC 2616: Hypertext transfer protocol, HTTP/1.1, 1999. [Online]. Available from: <http://www.ietf.org/rfc/rfc2616.txt>.
[22] B. Krishnamurthy, M. Arlitt, PRO-COW:protocol compliance on the web – a longitudinal study, in: Proc. of the 3rd USENIX Symposium on Internet Technologies and Systems (USITS'01), March 2001.
[23] A. Chandrakasan, R. Brodersen, Minimizing power consumption in digital CMOS circuits, in: Proceedings of the IEEE, April 1995, 83 (4), p. 498–523.
[24] R. Gonzalez, M. Horowitz, Energy dissipation in general purpose microprocessors, in: IEEE Journal of Solid State Circuits, p. 1277–1284.
[25] R. Joseph, M. Martonosi, Run-time power estimation in high-performance microprocessors, in: Proceedings of the 2001 Symposium on Low Power Electronics and Design, 2001.
[26] L.S. Brakmo, D.A. Wallach, M.A. Viredaz, usleep: a technique for reducing energy consumption in handheld devices, in: Proc. Int. Conf. Mobile Systems, Applications, and Services, June 2004.
[27] G. Wang, M. Irwin, P. Berman, H. Fu, T.L. Porta, Optimizing sensor movement planning for energy efficiency, in: Proc. of International Symposium on Low Power Electronics and Design (IPSLED), August 2005.
[28] H. Zhu, G. Cao, On supporting power-efficient streaming applications in wireless environments, in: IEEE Transactions on Mobile Computing, August 2005, pp. 391–403.
[29] E. Jung, N. Vaidya, A power control MAC protocol for ad hoc networks, in: Wireless Networks, 2005, pp. 55–66.
[30] L. Zhong, N.K. Jha, Energy efficiency of handheld computer interfaces: limits, characterization and practice, in: Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services, June 2005.
[31] R. Neugebauer, D. Mcauley, Energy is just another resource: energy accounting and energy pricing in the nemesis OS, in: Proceedings of the 8thWorkshop on Hot Topics in Operating Systems, 2001.
[32] J.S. Chase, D.C. Anderson, P.N. Thakar, A.M. Vahdat, R.P. Doyle, Managing energy and server resources in hosting clusters, in: Proceedings of the 18th Symposium on Operating Systems Principles, 2001.
[33] S. Mohapatra, N. Venkatasubramanian, Middleware for mobility: a game theoretic approach for power aware middleware, in: Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware, October 2004.
[34] B.D. Noble, Mobile Data Access, Ph.D. dissertation, School of Computer Science, Carnegie Mellon University, May 1998. [Online]. Available: <http://mobility.eecs.umich.edu/papers/diss.pdf>.
[35] A.D. Joseph, J.A. Tauber, M.F. Kasshoek, Mobile computing with the Rover toolkit, IEEE Transaction on Computers: Special Issue on Mobile Computing 46 (3) (1997).
[36] R. Mohan, J.R. Simth, C. Li, Adapting multimedia internet content for universal access, IEEE Transactions on Multimedia 1 (1) (1999) 104–114.
[37] M. Yarvis, A. Wang, A. Rudenko, P. Reiher, G.J. Popek, Conductor: distributed adaptation for complex networks, in: Proc. of the Seventh Workshop on Hot Topics in Operating Systems, March 1999. [Online]. Available from: <http://lasr.cs.ucla.edu/reiher/papers/yarvis.ps>.
[38] X. Fu, W. Shi, A. Akkerman, V. Karamcheti, CANS: composable, adaptive network services infrastructure, in: Proc. of the 3rd USENIX Symposium on Internet Technologies and Systems (USITS'01), March 2001, pp. 135–146.
[39] P. Sudame, B. Badrinath, Transformer tunnels: a framework for providing route-specific adaptations, in: Proc. of the USENIX Technical Conf., June 1998.
[40] A. Mallet, J. Chung, J. Smith, Operating system support for protocol boosters, in: Proc. of HIPPARCH Workshop, June 1997.

Weisong Shi is an Assistant Professor of Computer Science at Wayne State University. He received his B.S. from Xidian University in 1995, and Ph.D. degree from the Chinese Academy of Sciences in 2000, both in Computer Engineering. His current research focuses on mobile computing, distributed systems and peer-to-peer systems, and wireless sensor networks. He has published more than 40 peer-reviewed journal and conference papers in these areas. He is the author of the book "Performance Optimization of Software Distributed Shared Memory Systems" (High Education Press, 2004). He has also served on technical program committees of several international conferences, including the chair of poster track of WWW 2005. He is a recipient of Microsoft Fellowship in 1999, the President outstanding award of the Chinese Academy of Sciences in 2000, one of 100 outstanding Ph.D. dissertations (China) in 2002, "Faculty Research Award" of Wayne State University in 2004 and 2005, the "Best Paper Award" of ICWE'04 and IPDPS'05. He is a member of ACM, USENIX, and IEEE.

Hanping Lufei is a Ph.D. student of computer science at Wayne State University. His current research focuses on QoS, systems security, access control, and trust management in mobile computing environment. He is also interested in computing enhancement for handheld device and resource management in distributed systems. He received his Bachelor degree in 1998 and Masters degree in 2001 from Huazhong University of Science and Technology (HUST) in China and the University of Toledo in USA, both in Electrical Engineering.