When Ambient Intelligence Meets Internet Protocol Stack: User Layer Design

Conference Paper · December 2010

DOI: 10.1109/EUC.2010.66 · Source: DBLP

3 authors, including:

Mehul Motani

National University of Singapore

194 PUBLICATIONS **3,138** CITATIONS

SEE PROFILE

Lawrence Wai-Choong Wong

National University of Singapore

161 PUBLICATIONS **845** CITATIONS

SEE PROFILE

When Ambient Intelligence Meets Internet Protocol Stack: User Layer Design

Yu Lu
NUS Graduate School for Integrative Science and Engineering
Interactive and Digital Media Institute
National University of Singapore, Singapore
Email: lulu@nus.edu.sg

Mehul Motani and Wai-Choong Wong
Department of Electrical and Computer Engineering
Interactive and Digital Media Institute
National University of Singapore, Singapore
Email:{motani,wong_lawrence}@nus.edu.sg

***Abstract*—Recently there has been increasing interest in building networks with Ambient Intelligence (AmI), which incorporates the user-centricity and context awareness. However, both the Internet TCP/IP protocol stack and the seven-layer OSI reference model are not suitable for AmI networks, because they do not specifically take the end-user requirements into consideration in their architecture design. Under the client-server architecture, we propose to explicitly take the end-user into account by defining a new layer called User Layer above the traditional application layer. The User Layer empowers the end-users to influence network performance based on their interaction activities with the networks. We adopt the Model Human Processor (MHP) approach for building the User Model. After that we present an exemplary User Layer implementation to illustrate how the User Layer interacts with the underlying protocol stack and improves end-user's satisfaction with network performance.**

I. Introduction

The seven-layer open systems interconnect (OSI) model and the TCP/IP Internet protocol stack provide layered architectures that partitions complicated network related tasks into different layers. Each layer has specific features and functionalities, with peer interactions at equivalent layers across the networks, and defined interfaces between layers. Both the traditional seven-layer OSI reference model and the TCP/IP Internet protocol stack have played prominent roles in the success of modern computer networks. Many fundamental and respected principles are implemented in those existing architectures such as the packet switching for multiplexing [1], the end-to-end arguments for defining communication protocols [2] and global addressing for routing the datagrams. Sometimes, the existing layered architecture cannot be highly adaptive and Quality of Service (QoS) efficient by sharing state information among different modules and layers. Therefore, researchers propose the cross layer design approach that actively exploits the dependence between protocol layers to obtain performance gains. There are many cross-layer design proposals summarized in [3] and they typically follow some basic approaches, such as merging of adjacent layers or vertical calibration across layers [4]. However, the above-mentioned two network architectures with the common cross-layer proposals do not adequately meet the need of AmI networks, because neither of them directly takes end-users, a critical and key element, into consideration in their architectures. End-users refer to the individual users who employ networked applications on their own hosts.

The networks with Ambient Intelligence (AmI) emphasize on user-centricity and context awareness, where the interaction between the end-user and the networks becomes most crucial. Therefore, we propose to explicitly define a new layer, which is called the User Layer, above the traditional application layer as shown in Fig. 1. The User Layer integrates the end-users into the network architecture, and empowers the end-users to influence network performance based on their interaction behavior. The User Layer aims to meet the end-users explicit or implicit requirement, and eventually improve their satisfaction with network performance. On the other hand, the fast evolving networking technologies are eroding the fundamental principles of the traditional computer networks, although many of them are still valid and quite powerful [5]. As one of the well established design principles, the end-to-end arguments is frequently compromised due to the arbitrary deployment of caches or proxy servers between the clients and the original servers. The User Layer also provides an alternative for the original server to re-establish communication with the end-users through defining some new User Layer protocol.

Different from our prior work [6], we extend the User Layer framework from client side to the server side. Thus the current User Layer operates under the client-server architecture and effectively influences both sides of the networks. Moreover, we adopt Human-Computer Interaction (HCI) and cognitive psychology knowledge to build User Model. Based on the new User Model, we also present a new exemplary User Layer implementation in this paper. All our efforts aims to provide the user-centric, highly adaptive and end-to-end network services to improve the end-user's satisfaction with network performance.

The rest of this paper is organized as follows. In section II, we propose the architecture of the User Layer and introduce its main building blocks. In section III, we describe the Human Information Processing approach for building the model of the end-user. Section IV presents an exemplary User Layer implementation with the first User Layer protocol and corresponding simulation results. We discuss the future work and conclude in section V.

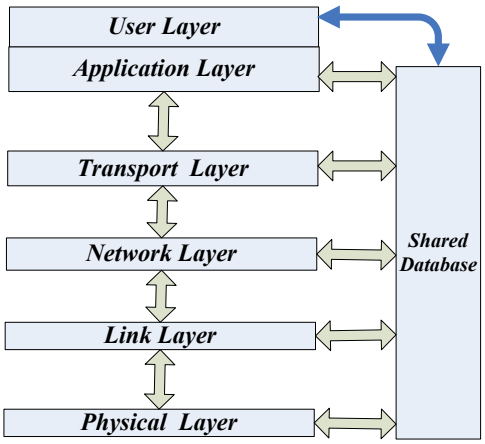

Fig. 1. User Layer above the application layer.

II. System Overview

The User Layer framework operates under the client-server architecture, and consists of three main components as shown in Fig. 2.

A. User Interface

The ***User Interface*** aims to fulfill two main functions:

(1) Monitor and record the interaction activities and behavior between the end-user and each running application. Sometimes, the User Interface is also required to gather the relevant operating system information and other context information in real time. The User Interface may operate in an invisible way to collect those interaction data and work automatically in background.

(2) Provide end-users a direct and efficient way to interact with the networks. For example, end-users can express their own perception about current network performance by simply pressing some button on the User Interface or by other visible ways.

In short, the User Interface undertakes the interaction information gathering task and works at the client side of the User Layer.

B. User Model Generation Subsystem

The ***User Model Generation subsystem*** can work at both the client side and the server side to generate the Individual User Model or the Group User Model, respectively. The main task of the User Model Generation subsystem is to derive and build the User Model for the specific end-user or group of users based on the interaction data collected by the User Interface. The User Model plays a crucial role in the proposed architecture, because it helps the User Layer ascertain end-users' status, intention or preference. In order to generate an effective and accurate User Model, Human-Computer Interaction (HCI) and related cognitive psychology knowledge are required, which will be elaborated in Section III. Moreover, data mining and machine learning methods can also be introduced here to construct the User Model and infer the stable end-user behavior patterns.

C. Control Subsystem

The ***Control subsystem*** can also work at both sides of networks, and establish the feedback loop between the User Layer and the underlying protocol stack. The Control subsystem mainly adjusts the different parameters of the lower layers in accordance with some control rules. The objectives of the control rules aim to improve the end-users' satisfaction and optimize network performance.

III. Modeling the End-User

In order to build an effective User Model for the User Layer, we must first understand the end-users and how they interact with their computers and networks. In Human-computer interaction (HCI) and related cognitive psychology fields, a number of well developed frameworks and models have been applied to explain and describe end-user interaction behavior [7]. The Human Information Processing approach is one of the most successful methods to conceptualize how the end-user's mind works. The basic idea of this approach is that human behavior is a function of several ordered processing stages. Different architectures, such as the ACT [8] and the SOAR [9] models, hold great promise for this field, while the most widely accepted and well-known model is the Model Human Processor (MHP) proposed by Card et al. [10].

As shown in Fig. 3, MHP consists of three interacting subsystems comprising the Perceptual subsystem, the Cognitive subsystem and the Motor subsystem, and each with its own processors and memories. The Perceptual subsystem is equipped with sensors and associated buffer memories for collecting and temporarily storing the external information. The Cognitive subsystem accepts symbolically coded information from the buffer memories of the Perceptual subsystem, and then decides on how to respond. Finally, the Motor subsystem carries out the response and takes action. The MHP models the information processing of the end-user as a sequential or parallel operation of the above-mentioned three MHP subsystems. Furthermore, the Rationality Principle and Problem Space Principle of MHP indicate that the end-user's behavior is based on rational activity, which means the end-user will not randomly and arbitrarily change from one state to another. Besides, all rational activities are served for achieving the end-user's goals, given the task and external information and bounded by the user's knowledge and processing ability.

Based on the MHP and the two basic principles, we can define many different but reasonable end-user states for describing the human-computer complex interaction behavior. In this paper, we just define two general but significant end-user states: ***User Present State*** and ***User Absent State***. Then each running computer application can be associated with only one state, either User Present State or User Absent State.

(1) ***User Present State***: Both the end-user's Perceptual and Cognitive subsystems are turned ON for acquiring and processing the information of the corresponding application.

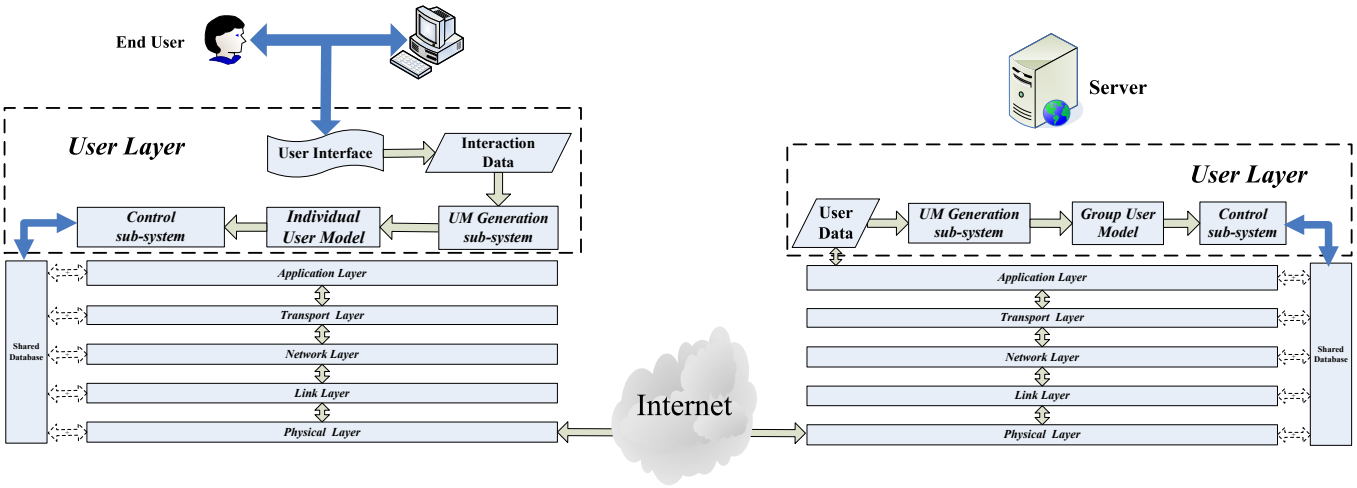

Fig. 2. User Layer block diagram under the client-server architecture.

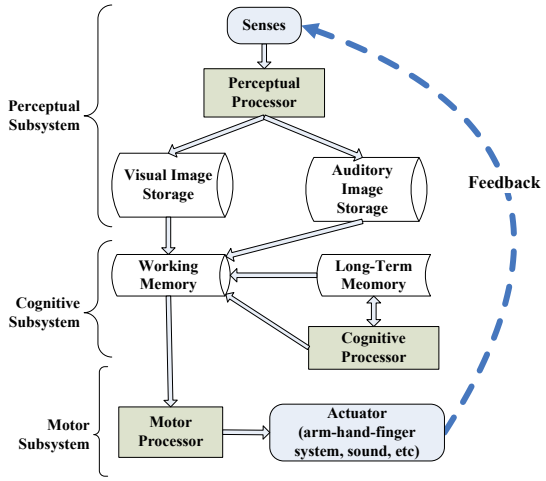

Fig. 3. Model Human Processor (MHP).

(2) ***User Absent State***: All the end-user's three subsystems are turned OFF and thus no interaction between the end-user and the corresponding application.

These end-user states can be applied to both networked and non-networked applications such as Web browser, Instant Messaging programs or Office software. Within the Web browser, each displayed Web page is defined as one single application, while one Web page may generate many different HTTP connections, which can be maintained by different Web servers simultaneously.

In order to identify above two end-user states, the User Layer monitors and detects the end-user's three subsystems in real time:

- For the end-user Perceptual subsystem, MHP shows that the most important buffer memories in the Perceptual subsystem are the Visual Image Store and the Auditory Image Store. Therefore, the User Interface monitors which application is shown on the screen and which application's sound is generating an audio output. In addition, the webcam may also help to monitor the end-user's visual sense by capturing the user's eye movement, when necessary.
- For the end-user Cognitive subsystem, although many researchers attempt to model and build the cognitive architecture [11], it is still difficult to accurately differentiate its status. Fortunately, MHP demonstrates that the Motor subsystem follows the Recognize-Act Cycle of the Cognitive Processor. Thus through monitoring the Motor subsystem behavior, the User Layer might indirectly derive the Cognitive subsystem status.
- For the end-user Motor subsystem, MHP considers the arm-hand-finger system as the most important actuator. From observing the keyboard and the Mouse input information, the User Layer can derive the specific application on which the end-user's Cognitive subsystem is working. Moreover, when the end-user is speaking to the microphone, it is also an important clue for the User Layer to know which application can be associated with the User Present State.

Based on the MHP approach and the above analysis, five ***State Conditions*** require to be verified by the User Interface:

{*Application Displays in the Foreground of the Screen*}
{*Application Generates Output for Earphone or Speaker*}
{*Application Receives the Mouse Input*}
{*Application Receives the Keyboard Input*}
{*Application Receives the Microphone Input*}

Then we propose a simple but effective User Model to determine the end-user state in real time:

(1) IF any TWO or more above State Conditions are TRUE simultaneously for the same running application, THEN the corresponding application holds the ***User Present State***.

(2) IF NONE or only one State Condition is TRUE for the running application, THEN the corresponding application holds the ***User Absent State***.

Above User Model works well in many cases and for many applications, but more complicated User Model can be

constructed for some specific network application or situation through user study. More importantly, for the different User Layer implementations, new end-user states can be defined and employed on the basis of MHP and relevant cognitive science.

IV. Exemplary User Layer Implementation

In order to show the proposed User Layer in action, we show an exemplary User Layer implementation, which aims to reduce the end-user perceived latency when browsing the Web pages. The first User Layer protocol, User State Transfer Protocol (USTP), is also defined for delivering the end-user state from the client side to the Web server side. Besides the example given in this section, the User Layer architecture with the proposed User Model can also be deployed in many other occasions, including the network resource allocation as well as the Cloud Computing data sharing and event notification [12].

A. Problem Description

As the major information resources on the modern Internet, more than 1 trillion Web pages exist in current cyberspace and the number is still rapidly increasing every second. Hypertext Transfer Protocol (HTTP) [13] is the de facto application-layer communication standard for transferring the Web pages. The persistent connections mechanism of HTTP/1.1, also called HTTP keep-alive, allows HTTP clients to send multiple requests over the same TCP connection. As one of the key properties of HTTP, the HTTP persistent connection reduces network congestion from re-establishing TCP connections and conserves the hosts' CPU and memory usage. As a default function, persistent HTTP connection is widely implemented on both the browser and the server sides. HTTP/1.1 [13] specifies that "*Servers will usually have some time-out value beyond which they will no longer maintain an inactive connection*" and "*The use of persistent connections places no requirements on the length (or existence) of this time-out for either the client or the server*". Clearly, HTTP/1.1 does not explicitly define the persistent connection-closing condition but suggests picking a proper timeout value for terminating the persistent connection. In the practical implementation of HTTP/1.1, a fixed timeout value is always imposed: the latest version 2.2.1 of the Apache HTTP Server employs a 5-second and Microsoft IIS uses a 120-second as their default timeout values, respectively. Improperly configuring the timeout value will easily degrade network performance: A small fixed timeout value causes low utilization of the persistent HTTP connection, and thus increase the end-user perceived latency and network burden; On the other hand, a large fixed timeout value will waste and even quickly exhaust Web server resources, which also results in the long and unpredictable end-user perceived latency.

On the other hand, it is an error-prone and time-consuming task for Web administrators to manually pre-configure a rigid and fixed timeout value for their Web servers. There are some research works of tuning the persistent HTTP connection timeout value to achieve optimal Web server performance: Faber [14] and Barford [15] indicated that the Web server should close the persistent connections once the client becomes inactive, but they did not provide any specific approach. Mogul [16] proposed to give higher priority to newly established connections, Sugiki [17] suggested setting higher priority to small RTT connections and prematurely terminate the ones with large RTT. However, none of these previous studies directly address the main problem of HTTP persistent connection timeout mechanism: ***In a HTTP session it is difficult to discriminate between a persistent connection that is being used by the end-user and a persistent connection that is already in a long-term idle status***. Both TCP and HTTP, and even the five-layer Internet protocol stack, cannot provide the relevant end-user's browsing behavior information for controlling the HTTP persistent connection timeout.

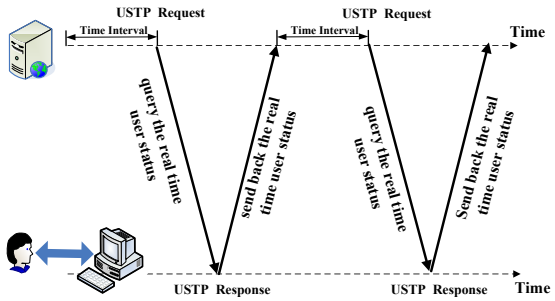

Fig. 4. The User State Transfer Protocol (USTP) workflow.

Therefore, adopting the newly proposed User Layer architecture is a natural and effective way to solve this problem. Consequently, we need to define the first User Layer protocol, User State Transfer Protocol (USTP), to transfer the end-user state to the Web server side.

B. User State Transfer Protocol (USTP)

The User State Transfer Protocol (USTP) is the first User Layer protocol, which is used to deliver real time individual end-user states from the client side to the server side. The USTP presumes a reliable transport and in this case using HTTP persistent connection, which is essentially the TCP connection, as its underlying carrier. The USTP works under the client-server architecture and simply use the request-response message exchange pattern. The overall workflow of the USTP can be illustrated in Fig. 4:

1) After the persistent HTTP connection becomes idle, the Web server side USTP program waits for some time interval and then initiates the request message to the end-user side via the existing persistent HTTP connection.
2) Upon receiving the USTP request, the end-user side USTP program retrieves the current individual end-user state, either User Present State or User Absent State, from the User Model Generation subsystem of the User Layer.
3) The end-user side USTP program encapsulates the current user state in an USTP response message, and sends it back to the Web server side.

4) The Web server side USTP program receives the response message and delivers the user state information to the User Model Generation subsystem for the further processing.
5) After some time interval, the server side USTP program repeats the request via the same persistent HTTP connection, if that connection still exists in the idle status.

Theoretically, the USTP should adjust the time interval between consecutive requests according to the end-user's browsing behavior pattern. Based on the MHP theory, the end-user's browsing behavior primarily depends on his Cognitive subsystem, whose tasks involve learning, retrieving the facts from its long-term memory and solution of problems. Through practical user study and theoretical calculation, MHP shows that the cognitive processing rate has a wide range among different individuals because of their distinct processing capacities. For example, human reading speed ranges from 52 to 652 words per minute; in working memory, the decay parameter varies from 5 to 226 seconds. In other word, even for the same Web page, different end-users require different processing times and the variance magnitude can be several seconds or even larger. In this paper, we simplify the algorithm by setting 7 seconds, the average decay parameter value of the human cognitive subsystem [10], as the time interval value of USTP. In future work, if the Individual User Model Generation subsystem can capture the corresponding end-user's cognitive capacity and roughly predict his processing time, the system performance can be further improved.

Remark: The current USTP operates only in the simplest case, and a more complicated situation occurs when one or more intermediaries are present between the end-user and the Web server such as when a proxy server is in use. Under such situation, a new version of USTP needs to be further discussed and specified.

C. User Layer Control Subsystem

Once the USTP response message successfully transfers the end-user state to the Web server, the User Layer will deliver it to the User Model Generation subsystem to build the Group User Model for the convenience of the server batch processing. In this User Layer example, as mentioned before, we mainly aim to reduce end-user perceived latency when he browses Web pages online. Thus we do not build the Group User Model for the Web server, but simply move each delivered end-user state to the Control subsystem for continue processing. Consequently, the Control subsystem at the server side can adopt the following control rules:

1) IF the User Absent State for the existing HTTP connection arrives, THEN the Control subsystem immediately signals the application layer to ***terminate*** the corresponding HTTP persistent connection.
2) IF the User Present State for the existing HTTP connection arrives, THEN the Control subsystem signals the application layer to ***maintain*** the corresponding HTTP persistent connection and ***wait*** for the next end-user state.

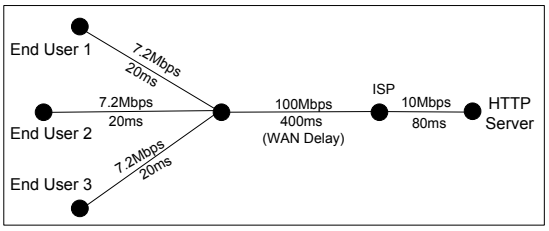

Fig. 5. Networking topology in NS-2.

We can see that the above simple control rules can effectively change the inflexible HTTP timeout mechanism to be adaptive to the end-user's interaction behavior.

D. Simulation Results

We implement the traditional HTTP persistent connection mechanism, USTP and the User Layer Control subsystem in the NS-2 simulator [18]. The simulation topology, as depicted in Fig. 5, consists of multiple end-users and a single HTTP server. The WAN Delay is set to 400ms, which commonly exists especially under the congested network condition. We suppose that the end-user state transition can always be identified by the User Layer and can be modeled as a two-state Markov process. We assume each end-user's sequential HTTP request process follows a homogeneous Poisson process with a rate of 6 requests /minute. We set the mean size of the Web page is 5K bytes and all HTTP requests are the static requests. Since optimizing the Web server's overall performance is out of the scope of this paper, we assume the Web server employs some admission control schemes in [19] [20] to avoid an overload situation. For the first simulation scenario that without the User Layer, we set the timeout value for HTTP persistent connection to 5 seconds, which is the same as the default timeout value of the Apache Server 2.2.1. Because we mainly aim to reduce the end-user perceived latency, we adopt the Web page response time as the performance metric. The Web page response time is the time interval that starts when the end-user sends the Web page request and ends when the end-user receives the last object of that Web page. Other performance metrics such as Web server throughput can be used in the future Internet experiment.

The simulation results in Fig. 6 shows that ***without*** the User Layer, the average Web page response time (end-user perceived latency) is around 2100 ms; while ***with*** the User Layer controlling the persistent HTTP connection, it can be reduced to 1100 ms. Fig. 7 describes the end-user state transition process during the simulation, where the User Absent State continues for about 50 seconds and the User Present State occupies the other simulation time. From this example, we can easily see that the User Layer can effectively reduce the average end-user perceived latency and thus improves the end-user's satisfaction with network performance. More importantly, it empowers the end-users to influence and enhance the network performance based on their own interaction behaviors and activities.

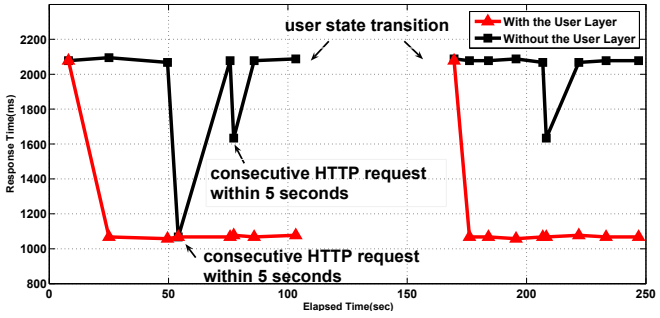

Fig. 6. Comparison results: WITH versus WITHOUT the User Layer.

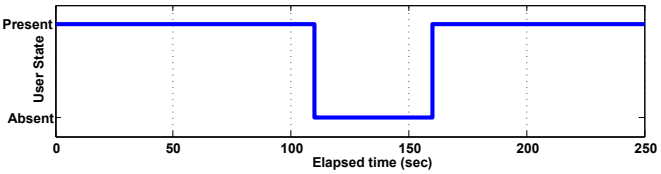

Fig. 7. End-user state transition.

V. Conclusion and Future Work

In this paper, we rethink the design issues of building the networks with Ambient Intelligence. Under the client-server architecture, we propose the specific User Layer and its framework. We introduce the Model Human Processor (MHP) approach and demonstrate how HCI and cognitive science knowledge can be applied in the User Layer design. Then we present an exemplary User Layer implementation for controlling the HTTP persistent connection timeout mechanism and show its feasibility and effectiveness. These efforts go towards supporting a user-centric AmI network design goal.

This initial contribution lays the groundwork for further research. Firstly, with the aim of improving end-users' satisfaction and optimizing network performance, the User Layer will explore interactions with more underlying layers and their protocols such as Transport Layer and TCP. Secondly, we will further develop the User Model Generation subsystem based on the Human Information Processing approach and other cognitive science knowledge. Thus more interaction information between the end-user and the underlying networks will be exposed to the existing User Layer for building the User Model. Thirdly, for the practical application of the adaptive persistent HTTP connection, the proposed control rules will incoporate with the algorithms of performance regulation of Web server [21]–[23] in the future Internet experiment.

Acknowledgment

This work is partially support by project grant NRF2007IDM-IDM002-069 on "Life Spaces" from the IDM Project Office, Media Development Authority of Singapore.

References

[1] D. D. Clark, "The design philosophy of the DARPA Internet Protocols," *SIGCOMM Comput. Commun. Rev.*, vol. 25, no. 1, pp. 102–111, 1995.
[2] J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-end arguments in system design," *ACM Trans. Comput. Syst.*, vol. 2, no. 4, pp. 277–288, 1984.
[3] V. Kawadia and P. R. Kumar, "A cautionary perspective on cross-layer design," *IEEE Wireless Communications*, vol. 12, no. 1, pp. 3–11, 2005.
[4] V. Srivastava and M. Motani, "Cross-layer design: a survey and the road ahead," *IEEE Communication Magazine*, vol. 43, no. 12, pp. 112–119, 2005.
[5] D. D. Clark, J. Wroclawski, K. R. Sollins, and R. Braden, "Tussle in cyberspace: defining tomorrow's internet," *IEEE/ACM Trans. Netw.*, vol. 13, no. 3, pp. 462–475, 2005.
[6] Y. Lu, M. Mehul, and W. C. Wong, "Intelligent Network Design: User Layer Architecture and its application," in *Proc. IEEE SMC*, Istanbul, Turkey, Oct. 2010.
[7] A. J. Julie and S. Andrew, *The human-computer interaction handbook: fundamentals, evolving technologies and emerging applications*. L. Erlbaum Associates Inc., 2003.
[8] R. J. Anderson, M. Michael, and L. Christian, "ACT-R: a theory of higher level cognition and its relation to visual attention," *Hum.-Comput. Interact.*, vol. 12, no. 4, pp. 439–462, 1997.
[9] A. Howes and R. M. Young, "The role of cognitive architecture in modeling the user: Soar's learning mechanism," *Hum.-Comput. Interact.*, vol. 12, no. 4, pp. 311–343, 1997.
[10] K. S. Card, P. M. Thomas, and A. Newell, *The psychology of human computer interaction*. L. Erlbaum Associates Inc., 1983.
[11] A. Newell, *Unified theories of cognition*. Cambridge, MA: Harvard University Press, 1990.
[12] K. Birman, G. Chockler, and R. V. Renesse, "Toward a cloud computing research agenda," *SIGACT News*, vol. 40, no. 2, pp. 68–80, 2009.
[13] R. Fielding, J. Gettys, J. C. Mogul *et al.*, "Hypertext Transfer Protocol – HTTP/1.1," *IETF RFC 2616*, 1999.
[14] T. Faber, J. Touch, and W. Yue, "The TIME-WAIT state in TCP and its effect on busy servers," in *Proc. IEEE INFOCOM*, New York, NY, USA, Mar. 1999.
[15] P. Barford and M. Crovella, "A performance evaluation of hyper text transfer protocols," in *Proc. ACM SIGMETRICS*, Atlanta, GA, USA, May 1999.
[16] J. C. Mogul, "The case for persistent-connection HTTP," in *Proc. ACM SIGCOMM*, Cambridge, MA, USA, Aug. 1995.
[17] A. Sugiki, K. Kono, and H. Iwasaki, "Tuning mechanisms for two major parameters of Apache web servers," *Softw. Pract. Exper.*, vol. 38, no. 12, pp. 1215–1240, 2008.
[18] UCB/LBNL/VINT. The Network Simulator Version 2, ns-2 [online] Available: http://www.isi.edu/nsnam/ns.
[19] S. Elnikety, E. Nahum, J. Tracey, and W. Zwaenepoel, "A method for transparent admission control and request scheduling in e-commerce web sites," in *Proc. ACM WWW '04*, New York, NY, USA, May 2004.
[20] B. Schroeder and M. Harchol-Balter, "Web servers under overload: How scheduling can help," *ACM Trans. Internet Technol.*, vol. 6, no. 1, pp. 20–52, 2006.
[21] T. F. Abdelzaher, K. G. Shin, and N. Bhatti, "Performance guarantees for Web server end-systems: a control-theoretical approach," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 1, pp. 80–96, 2002.
[22] A. Robertsson, B. Wittenmark, M. Kihl, and M. Andersson, "Design and evaluation of load control in Web server systems," in *Proc. IEEE American Control Conference*, Boston, Massachusetts, USA, Jun. 2004.
[23] A. Kamra, V. Misra, and E. Nahum, "Yaksha: a self-tuning controller for managing the performance of 3-tiered Web sites," in *Proc. IEEE IWQOS*, Montreal, Canada, Jun. 2004.