Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 5 (2011) 280–287

Procedia Computer Science

The 2nd International Conference on Ambient Systems, Networks and Technologies (ANT-2011)

The User-Context Module: A New Perspective on Future Internet Design

Yu Lu[a,b,*], Mehul Motani[b], Wai-Choong Wong[b]

[a] *Graduate School for Integrative Science and Engineering, National University of Singapore*
[b] *Department of Electrical and Computer Engineering, National University of Singapore*

Abstract

Advances in ubiquitous sensing and computing technologies, combined with the well developed theories in cognitive science, present major opportunities for the future Internet. We argue that the future Internet should be a user-centric and context-aware communication network, which goes far beyond today's host-centric and store-forward Internet. In this article we propose to explicitly introduce the end-user's context information into the future Internet framework. We, therefore, define a new functional module called the User-Context Module, and describe its architecture and key design principles. We also present two User-Context Module applications to aid gaining an understanding of its operation and impacts.

Keywords: User centered design, User modeling, Computer networks, Protocols.

1. Introduction

The goal of the original Internet, which was built up for the Defense Advanced Research Project Agency (DARPA) around 30 years ago, was to develop an effective technique for multiplexed utilization of interconnected networks and their hosts [1]. With such a host-centric vision, Internet creators built a network infrastructure and successfully connected the worldwide hosts together. One of the fundamental design principles of the original Internet is that it serves as the communication medium between two hosts that desire speaking to each other. Such a design principle directly results in today's Internet which simply regards the end-user, the context environment and the network device as one entity, and hence inevitably ignoring their internal and external relationships and states. In other words, the current Internet conflates dynamic end-users, complicated context environment, and various network devices into one simple concept, namely the Internet host. There is no doubt that such an assumption greatly decreases the complexity of today's Internet, but it essentially excludes the end-user and the related significant context information from today's Internet architecture. Therefore, such a design principle concurrently causes many other problems, such as mobility and security issues.

In our vision, future Internet architecture design should allow end-users, context environment and network devices to show their presence independently. On the basis of such de-conflation efforts, the future Internet is expected to have capabilities of recognizing and utilizing the valuable context information related to the Internet end-user. Such

[*]Corresponding author. Tel.: (+65) 6516 1076.
Email address: `lulu@nus.edu.sg.` (Yu Lu)

doi:10.1016/j.procs.2011.07.037

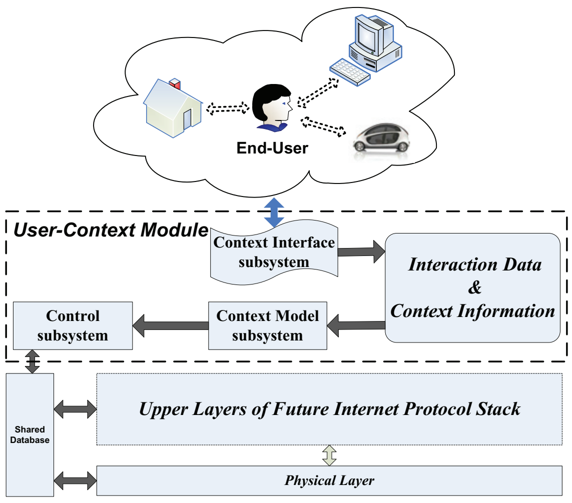
End-User
User-Context Module
Context Interface subsystem
Interaction Data & Context Information
Control subsystem
Context Model subsystem
Shared Database
Upper Layers of Future Internet Protocol Stack
Physical Layer

Figure 1: User-Context Module block diagram with future Internet.

user-centric and context-aware perspectives impose new research challenges. The emerging ubiquitous sensing and computing technology [2] holds great promise for building a completely new level of context-aware environment. Cognitive science [3], which explains how information is represented and processed in the human brain, and the related social behavior science, would eventually empower machines to fully understand the human thinking process and interaction behavior. When such intelligent context-aware environment meets the machine with high cognitive abilities, it creates a feasible way to achieve our vision on future Internet.

Our user-centric and context-aware vision first requires a new naming mechanism with the corresponding supporting architecture. There have been some relevant research proposals within that scope, especially the studies on identifier-locator split architectures, such as the MILSA [4], HIP [5] and LISP [6]. The identifier specifies who the network host is, and the locator explains where the network host is. Simply speaking, the identifier-locator split architectures attempt to use independent name spaces to help the Internet recognize the host and the host address separately. Moreover, some enhanced architecture explicitly name end-user, data and network device in distinct independent realms and are handled by their individual realm managers. The identifier-locator split architectures could enable end-users to be the final destination of the communication, and hence to some extent incorporate end-users into the architecture of the future Internet. All the above-mentioned and other relevant proposals have been well summarized in [7].

We take a fresh approach and introduce the use of end-user context information into the mainstream of future Internet architecture, and further enhance future Internet as a user-centric and context-aware intelligent communication network. To provide the direction for the solution, we propose a new functional module for future Internet protocol stack, called the **User-Context Module**, which is shown in Fig. 1. In this paper, we describe the general User-Context Module architecture and its main building blocks in Section 2. From Section 3 to Section 5, we discuss and present the key design principles and practices for each subsystem of the User-Context Module to help deepen understanding of their operation and impacts. Finally, we conclude in Section 6.

2. System Overview

The system-level block diagram of the proposed User-Context Module is illustrated in Fig. 1, and it consists of three main subsystems: **Context Interface subsystem**, **Context Model subsystem** and **Control subsystem**.

- As shown in Fig.1, the **Context Interface subsystem** directly interacts with the end-user and the surrounding environment. This subsystem is mainly responsible for monitoring, collecting and recording relevant context information of the end-user. The context information covers a wide range of external and internal states of

Table 1: Interaction Conditions and Corresponding Validation Criteria

Interaction Condition			Validation Criteria
End-User	E1	The Perceptual Subsystem is Working on the Web Browser	The end-user's eye-gaze direction towards Web browser
			The end-user wearing earphone or near speaker
	E2	The Motor Subsystem is Working on the Web Browser	The end-user sending the keyboard message to Web Browser
			The end-user sending the mouse message to Web Browser
Web Browser	W1	Presenting Visual/Auditory Information	The Web browser displaying in the device screen foreground
			The Web browser generating audio output
	W2	Retrieving Visual/Auditory Information	The Web browser sending or receiving HTTP message
			The Web browser waiting for HTTP response

end-users, including their real-time tasks and duties, interaction activities with the surrounding objects, as well as their physical locations, identities and preferences.

- The **Context Model subsystem** processes the collected context information by using its context model. The context model refers to the abstract data model for ascertaining the end-user's status, interaction behavior, or any other significant knowledge. During its operation, the Context Model subsystem first classifies the incoming context information, and subsequently sends different context information to the corresponding context model, which has been built in advance. Then the context model processes those delivered context information to generate the so called ***key context knowledge***. The key context knowledge serves as the input of the Control subsystem and thus plays a key role in adjusting the underlying network.

- The **Control subsystem** directly interacts with the underlying network infrastructure according to the derived key context knowledge. With different input key context knowledge, the Control subsystem automatically adjusts different Internet layers, protocols, or parameters to provide optimal network performance for end-users.

In short, a typical User-Context Module consists of the above-described three subsystems. Such specific design explicitly and effectively incorporates the relevant context information of the end-user into the network mainstream infrastructure, and also provides abundant flexibility for different deployment plans. Essentially, the User-Context Module establishes a new feedback loop and communication path between end-users and future Internet protocol stack. We further discuss design issues, principles and practices of these three subsystems in the following sections.

3. Context Interface Subsystem Design

With the aim of monitoring and recording the relevant context information of the end-user, the Context Interface subsystem could leverage on the latest ubiquitous sensing technologies with various intelligent physical sensors. The Context Interface subsystem may operate and perform all its information gathering in an invisible way, and thus without any interruption and disturbance to the end-users.

For example, we are developing a practical Context Interface subsystem on a laptop as shown in Fig. 2. The current Context Interface subsystem could capture the Internet end-user's eye-gaze direction through a built-in or USB Webcam. Hence, it can efficiently sense whether the end-user is watching the screen of the network host. Meanwhile it could detect which Internet application is displayed in the foreground of the host screen and receiving the mouse/keyboard input. In addition, the Context Interface subsystem can also calculate the real-time bandwidth consumption of each running Internet application, and write all those valuable context information and data into the shared database. The Validation Criteria column of Table I shows some context information collected by such a specific Context Interface subsystem. Some relevant works [8, 9] also discuss the technologies and solutions for tracking end-user activities, especially under the Web browsing scenario.

After collecting the real-time context information of the Internet end-user, the Context Interface subsystem delivers it to the Context Model subsystem for further processing. In short, the Context Interface subsystem is dedicated to context information gathering task, and responsible for providing a solution to fully sense the Internet end-user and the surrounding environment.

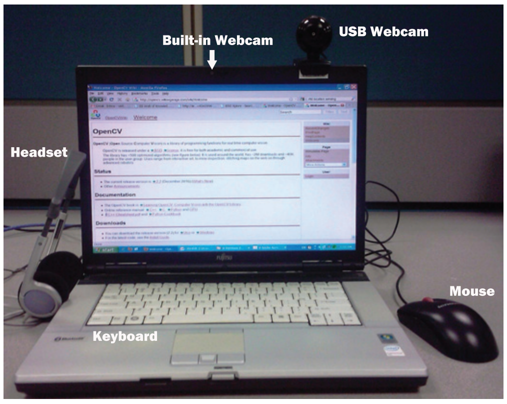

Figure 2: Sensors of the Context Interface subsystem.

4. Context Model Subsystem Design

As indicated earlier, the key component of the Context Model subsystem is the ***context model***, which is used to ascertain the end-user's internal and external states. Constructing an accurate context model requires precise understanding of the end-user, and the relevant relationships with the surrounding environment. Therefore, cognitive science, human-computer interaction (HCI) and even sociology knowledge need to be introduced and employed in this subsystem. Furthermore, thorough user studies and data mining methods are also helpful in building the context model.

4.1. Understanding and Modeling End-User

In cognitive psychology and HCI fields, a number of well developed models have been applied to explain the human internal mental process and interaction behaviors [10]. Those theories and models may provide an efficient way to understand human behavior, especially in terms of human interaction activities within a networked environment.

For example, one of the most widely accepted and elegant models, called the Model Human Processor (MHP) [11], describe the human's cognitive process. The basic idea in MHP is that a human is like a system that can be analyzed in terms of several ordered processing stages and their interrelationships. MHP consists of three interacting subsystems including the Perceptual subsystem, the Cognitive subsystem and the Motor subsystem, and each subsystem works with its own processors and memories. The Perceptual subsystem has multiple sensors (visual, auditory, tactile, etc.) with associated buffer memories for collecting and temporarily storing the external information. The Visual Image Storage and the Auditory Image Storage are the two major buffer memories in the MHP Perceptual subsystem. Therefore, monitoring the corresponding visual and auditory sensors are considered the most direct way to track the human Perceptual subsystem. The Cognitive subsystem accepts symbolically coded information from the Perceptual subsystem and decides how to respond to those external stimuli. Currently it is still difficult to accurately measure the Cognitive subsystem, but the rapid development of Brain-Computer Interface (BCI) and Neuroscience may greatly facilitate this task in the near future. The Motor subsystem carries out all responses and takes actual actions. In the MHP Motor subsystem, arm-hand-finger system is considered as the most frequently-used actuator. More importantly, MHP demonstrates that the cognitive processor follows the recognize-act cycle style, thus monitoring the Motor subsystem behavior can help estimate the status of the Cognitive subsystem.

Similar to the above-described MHP, many other cognitive psychology and HCI approaches model the end-user as the sequential or parallel operation to process information, and has demonstrated their different particular strengths in practice. Thus, we could select one or several of them as the theoretical foundation for constructing the context model.

4.2. Exemplary Key Context Knowledge and Context Model

The key context knowledge, which is directly utilized by the Control subsystem, covers a broad variety of end-user states as well as interaction activities. In this paper, we provide some simple exemplary key context knowledge, which are related to one of the most common Internet activities: Web browsing. As the major information resource on today's Internet, more than 1 trillion World Wide Web pages exist in current cyberspace and the number is still rapidly increasing. A Web browser is an indispensable tool for an end-user to retrieve, present, and contribute information on the World Wide Web. The latest Web browsers are quickly shipping their features with more intelligence, mobile characteristics and non-traditional User Interface. Future Web browser design even separates itself from the conventional Internet device and no longer functions as a pure Internet application. With such rapid development, it is highly possible that Web browsers will still be the indispensable tool for end-users to access information over the Internet in the near future. We, therefore, attempt to first focus on the relationship between the end-user and the Web browser to derive the relevant key context knowledge.

Within the given problem space, there are many ways in which the key context knowledge can be specified depending on the application scenarios. However, for simplicity, we only define two simple but general states as examples and cornerstones of the key context knowledge that suffice in illustrating the underlying proposed user-centric framework, namely:

- **Communicating Status**: The Internet end-user interacts with the Web browser for the purpose of information exchange.
- **Disconnecting Status**: The Internet end-user is detached from the Web browser, and no information exchange occurs between the end-user and the Web browser.

Note that the above defined key context knowledge are applicable to any interaction behaviors between the end-user and the Web browser regardless of types of the network host, end-user's identity, and features of the Web browser. Moreover, they can also serve as the building blocks and foundation for further defining more meaningful key context knowledge.

Based on the above defined key context knowledge and the MHP theory, four necessary interaction conditions are given in Table I. Then the following exemplary **context model** can be built:

(1) IF at least one interaction condition of the end-user (*E1*, *E2*) is true **AND** at least one interaction condition of the Web browser (*W1*, *W2*) is true, THEN the end-user and the Web browser are associated with the **Communicating Status**.

(2) IF no interaction condition of the end-user (*E1*, *E2*) is true **OR** none of the interaction condition of the Web browser (*W1*, *W2*) is true, THEN the end-user and the Web browser are associated with the **Disconnecting Status**.

The Context Model subsystem analyzes the input context information from the Context Interface subsystem, and periodically outputs the real-time interaction status between the end-user and the Web browser to the Control subsystem.

5. Control Subsystem Design

5.1. Objective and Design Principle

With the delivered key context knowledge from the Context Model subsystem, the Control subsystem directly interacts with the Internet infrastructure. For different User-Context Module usage cases, the Control subsystem may adjust different layers, protocols or configuration parameters, with the ultimate goal to enhance underlying network performance, achieve network resource optimization, and consequently improve the end-user perceived QoS. It is also useful if the Control subsystem is able to provide some mechanism to encourage end-users to share their actual key context knowledge that would lead to aggregated improvements in performance and user experiences. Thus the following design principles for the Control subsystem are considered:

- The Control subsystem can actively motivate the underlying networks to maximize the system-level social welfare (aggregated utility).

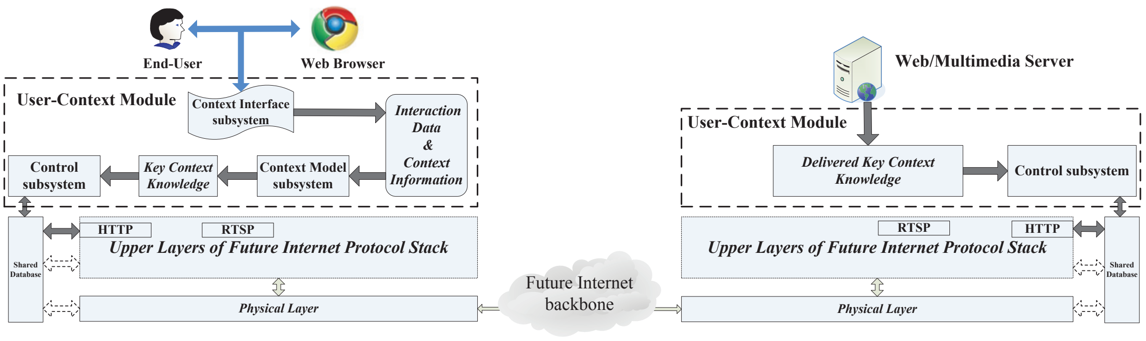

Figure 3: Exemplary User-Context Module under the client-server architecture.

- The Control subsystem can provide some reliable mechanism and framework to encourage every end-user while pursuing his individual benefits to also actively share their key context knowledge.
- The Control subsystem is able to provide both service differentiation and service fairness according to different key context knowledge of the end-users.
- The Control subsystem can adapt to the dynamics of the system, such as arriving end-users or departing end-users.
- The Control subsystem should be highly scalable, which means that it can gracefully handle the growing amount of end-users.

5.2. *Exemplary Control Subsystem (I)*

In Section 4, we provide a simple context model to derive the key context knowledge for an end-user who is Web browsing over the Internet, which are the ***Communication Status*** and the ***Disconnecting Status*** between the end-user and the Web browser. Such key context knowledge can be utilized by many different applications of the User-Context Module. To simply show how the Control subsystem works, we apply such key context knowledge on the Hypertext Transfer Protocol (HTTP) and the Real Time Streaming Protocol (RSTP). As shown in Fig. 3, the Control subsystem directly interacts with both the client side and the Web server side of the underlying protocol stack.

The HTTP protocol is the de facto communication standard for transferring Web pages, and one of its key properties, called the persistent connection mechanism, allows HTTP clients to send multiple requests over the same connection. Too many persistent connections in Web server can easily cause thrashing in the virtual memory system and considerably degrade server performance. In practice, a fixed limit is always imposed on the maximum number of concurrent persistent connections. Moreover, HTTP/1.1 [12] by the Internet Engineering Task Force (IETF) specifies that "*Clients that use persistent connections SHOULD limit the number of simultaneous connections that they maintain to a given server. A single-user client SHOULD NOT maintain more than 2 connections with any server or proxy*". However, today's commercial Web browsers always violate this specification: the default maximum value of Firefox 3.6 is set to 6 parallel persistent connections per server, and 8 persistent connections per proxy. Recently, Internet Explore (IE) also aggressively increases this value from 2 in IE 7 to 6 persistent connections per server in IE 8. The latest Google Chrome also uses at least 6 persistent connections per server. Therefore, the number of live persistent connections in the Web server often becomes a scarce resource resulting in performance bottlenecks. On the other hand, the current HTTP specification does not explicitly define the connection-closing mechanism, and thus in practical implementations, a fixed timeout value for terminating the persistent connection is always imposed. For example, the latest version 2.2 of the Apache HTTP Server employs 5 seconds, while Microsoft IIS uses 120 seconds as their default timeout values. However, setting such a fixed timeout often easily degrades network performance. A small timeout value causes low utilization of the persistent HTTP connection, and therefore increases Web page response time as well as Internet burden. Conversely, a large fixed timeout value may quickly exhaust resources of the Web server and network, and hence result in long and unpredictable Web page response time and possible instability.

Besides, it is an error-prone and time-consuming task for Web administrators to manually configure the timeout parameter for each Web server. There has been some prior work [13] on optimally tuning the HTTP connection timeout value to improve Web server performance. However, none of these previous studies directly addresses the most fundamental issue: **The existing Internet protocol stack cannot provide any end-user context information to help either the Web browser or the Web server decide how to efficiently allocate persistent connections**.

The proposed User-Context Module architecture can offer such context information of the end-user, and thus becomes a natural and effective way to tackle this problem. Utilizing the previously defined key context knowledge and the proposed context model, the Control subsystem at the Web browser side can implement the following illustrative control rules:

(1) IF the **Disconnecting Status** arrives, THEN the Control subsystem at the client side **decreases** the maximum number of the concurrent persistent connections per server and signals HTTP to **terminate** the idle persistent connections.

(2) IF the **Communication Status** arrives, THEN the Control subsystem at the client side **increases** the maximum number of the concurrent persistent connections per server and signals HTTP to **maintain** the corresponding persistent connections until the next key context knowledge reaches.

On the Web server side, the Control subsystem implements the following control rules:

(1) IF the **Disconnecting Status** arrives, THEN the Control subsystem at the server side signals HTTP to gracefully **terminate** the corresponding persistent connections.

(2) IF the **Communicating Status** arrives, THEN the Control subsystem at the server side signals HTTP to **maintain** the corresponding persistent connections.

The above illustrative control rules running on both the client and the server actively enable the network protocol to adapt to the derived key context knowledge, and thus bridge the gap between the Internet end-user and the underlying network infrastructure.

5.3. Exemplary Control Subsystem (II)

Audio and video streaming on Internet has grown exponentially over the past few years, and there are myriad of different multimedia servers, media players being employed based on a group of application layer protocols. The Real Time Streaming Protocol (RTSP) is a popular protocol to stream multimedia content from the multimedia server to its clients. Many online media service providers adopt RTSP as their underlying media control protocol: for example, "MSN Video" [14] provides a variety of Web-based multimedia content, including international news and music video clips, which are always played through the Windows Media Player (WMP) embedded in Web browsers. Communication between WMP and the media servers is done with the Microsoft Media Server (MMS) protocol, which is typically built over RTSP.

The RTSP protocol provides several key methods to control the streaming resources on the remote server, such as SETUP, PLAY, RECORD, PAUSE and TEARDOWN. SETUP method causes the multimedia server to allocate resources for a stream and start an RTSP session; PLAY method starts data transmission on the stream allocated via SETUP; PAUSE method temporarily halts a stream without releasing the server resource. With the above derived key context knowledge, the Control subsystem at the multimedia client side could has the following illustrative control rules:

(1) IF the **Disconnecting Status** arrives, THEN the Control subsystem immediately signals RTSP using PAUSE method to suspend the current streaming process.

(2) IF the **Communicating Status** resumes, THEN the Control subsystem automatically signals RTSP using PLAY method to restart the paused streaming process.

6. Conclusion

In this article, we rethink the design issues for future Internet architecture from a different perspective, where end-user context information is introduced and applied to dynamically adjust network and application parameters to ensure efficient utilization of network resources while enhancing end-user experience. The proposed User-Context Module explicitly incorporates relevant context information into future Internet architecture. The exemplary key context knowledge, context model and control subsystem demonstrate our design philosophy and practice. All of our efforts go towards building a user-centric and context-aware future Internet, and we hope the key ideas behind this work would open up a new realm for innovations on next generation Internet and its related applications.

Acknowledgment

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

Reference

[1] D. D. Clark, The design philosophy of the DARPA Internet Protocols, SIGCOMM Comput. Commun. Rev. 25 (1) (1995) 102–111.
[2] C. A. Da Costa, A. C. Yamin, C. F. R. Geyer, Toward a General Software Infrastructure for Ubiquitous Computing, IEEE Pervasive Computing 7 (1) (2008) 64–73.
[3] M. S. Gazzaniga, The Cognitive Neurosciences III, The MIT Press, 2004.
[4] J. Pan, R. Jain, S. Paul, C. So-in, MILSA: A New Evolutionary Architecture for Scalability, Mobility, and Multihoming in the Future Internet, IEEE Journal on Selected Areas in Communications 28 (8) (2010) 1344–1362.
[5] R. Moskowitz, P. Nikander, Host Identity Protocol (HIP) Architecture, IETF RFC 4423.
[6] D. Farinacci, V. Fuller, et al., Internet Draft: Locater/ID Separation Protocol(LISP), draft-farinacci-LISP-03, August 13, 2007.
[7] S. Paul, J. Pan, R. Jain, Architectures for the future networks and the next generation Internet: A survey, Computer Communications 34 (1) (2011) 2–42.
[8] R. Atterer, M. Wnuk, A. Schmidt, Knowing the Users Every Move: User Activity Tracking for Website Usability Evaluation and Implicit Interaction, in: Proc. WWW, Edinburgh, UK, 2006.
[9] R. W. Reeder, P. Pirolli, S. K. Card, WebEyeMapper and WebLogger: Tools for Analyzing Eye Tracking Data Collected in Web-use Studies, in: Proc. CHI, Seattle, USA, 2001.
[10] A. J. Julie, S. Andrew, The human-computer interaction handbook: fundamentals, evolving technologies and emerging applications, L. Erlbaum Associates Inc., 2003.
[11] K. S. Card, P. Thomas, N. Allen, The psychology of human computer interaction, L. Erlbaum Associates Inc., 1983.
[12] R. Fielding, J. Gettys, J. C. Mogul, et al., Hypertext Transfer Protocol – HTTP/1.1, IETF RFC 2616.
[13] A. Sugiki, K. Kono, H. Iwasaki, Tuning mechanisms for two major parameters of Apache web servers, Softw. Pract. Exper. 38 (12) (2008) 1215–1240.
[14] MSN Video: http://www.msnbc.msn.com/.