IMPROVING QUALITY OF EXPERIENCE AND PROTOCOL PERFORMANCE USING USER CONTEXT INFORMATION

LU YU

A THESIS SUBMITTED FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

NUS GRADUATE SCHOOL FOR INTEGRATIVE SCIENCES AND ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2012

To my parents and departed grandfather.

Acknowledgements

After over seven years postgraduate study, at three different countries, in two different disciplines, I have learned one thing - I could never have done any good research work without the support and encouragement of a lot of people.

First, I would like to express my deepest gratitude to my two advisors, Prof. Wong Wai-Choong, Lawrence and Prof. Mehul Motani, for their continuous guidance and support during my four years PhD study. Their invaluable advice, keen insight, extensive knowledge and enthusiasm have provided me great inspirations and paved the way for my research. They have generously devoted their time and efforts to fostering my independent learning and thinking abilities. If I do take the academic path, I only hope that I can be half the advisor that you have been to me. Whatever path I do take, the philosophy and the thinking skills I have learned from them will definitely benefit all my life. I guess that is why Ph.D. stands for Doctor of Philosophy, and we pursue not only a doctor in pure engineering or science.

I would also like to thank Prof. THAM Chen Khong, Prof. Chua Kee Chaing, Prof. Ge Shuzhi, Dr. Soh Wee-Seng, Prof. Hang Chang Chieh, and Dr. Xiao Wendong, for their professional advices and comments on both my research and my future career plans.

My sincere thanks also goes to Prof. Liu Jinkun for not only his role as my master advisor in Beijing University of Aeronautics and Astronautics, but also his unconditional support for my overseas study in both Singapore and France.

I must thank National University of Singapore for providing me such a precious study opportunity, when my student visa application was unexpectedly rejected by the U.S. embassy around four years ago. I have been very lucky to meet Prof. Justine Clare Burley in Beijing back in 2007, who encouraged me to come to this garden city and my current faculty, NUS Graduate School for Integrative Sciences and Engineering (NGS).

I would like to thank all my lab mates and colleagues in NUS ECE Communications Lab and IDMI Ambient Intelligence Lab, Mr. Wang Hui, Mr. Song Xianlin, Dr. Zhang Xiaolu, Dr. Da Bin, Dr. Chen Qian, Mr. Sun Ju, Dr. Jin Yunye, Mr. Ingwar Wirjawan, Mr. Goh Thiam Pheng and many other good friends. Without you guys to have fun with and complain to, I cannot complete my thesis work and my PhD journey.

Finally, I would like to dedicate this work to my parents and my departed grandfather, who taught me the most important subject and set themselves as the best example: how to care about others more than yourself. They have always been there for me, although I am not a qualified son and grandson. I owe them much.

Contents

Summary

As an effective technique for multiplexed utilization of interconnected networks and their hosts, today's Internet protocol stack does not explicitly take into account dynamic end-users and their context information in its architectural design, which affects Internet performance from both the end-user perspective and the network perspective. On the other hand, the rapid progress in context-aware computing techniques as well as cognitive science greatly facilitates collecting and ascertaining context information of Internet end-users. Proper utilization of the highly abstract and substantive end-user's context information presents major opportunities to further enhance the Internet as a user-centric, context-aware and intelligent communication system. To address these research challenges, a novel functional module, called the User-Context Module, is proposed to explicitly and smoothly integrate an end-user's context information into the five-layer Internet protocol stack. In this thesis dissertation, the research is exploited in three phases: (i) basic architectural design of the User-Context Module; (ii) applications of the User-Context Module; (iii) a resource distribution framework that provides context-driven service differentiation, and also incentivizes context sharing and moderate competition under the User-Context Module.

Firstly, we design the basic architecture of the User-Context Module, which consists of three indispensable subsystems. Two fundamental categories of the advanced context information are defined, and corresponding context models are built for three representative Internet services with the aim of empowering the Internet to capture, understand and utilize end-user's context information.

Secondly, we design and implement two applications of the User-Context Module to demonstrate its operation, implementation and performance. The Internet experimental results show that the two applications can effectively enhance the end-user's quality of experience (QoE) and improve the underlying protocol performance.

Lastly, based on the User-Context Module architecture and the deduced context information, we propose a resource distribution framework that (1) provides service differentiation in allocating limited resources; (2) encourage all Internet clients to provide their actual context information; (3) motivate all Internet clients to adopt a moderate competition policy.

List of Tables

List of Figures

List of Symbols

Symbol	Meaning
V	sending rate of TCP connection
$K_p,\ K_d$	tuning parameters in PD control algorithm
e	error in control system (ideal output - measurable output)
W^{rec}	advertised window size in TCP
f_s	normalized file size in the QoE model
V_d	normalized downloading bandwidth in the QoE model
$i,\ j,\ k,\ L,\ M,\ r$	indices
P	game player (Internet client)
I	set of game players
N	total number of game players
T	time slot over resource distribution process
w	willingness value of the resource owner
b	bidding value of game player
x	assigned resources by the Resource Distribution Algorithm
s	outcome of the Key Context Information
C	class of Internet clients categorized by the resource owner
μ	total amount of available resources

Symbol	**Meaning**
q	duration ratio in the Willingness Update Algorithm
θ	threshold in the Willingness Update Algorithm
η	number of tickets in the Willingness Update Algorithm
l	amplification factor in the Willingness Update Algorithm
$\lceil \cdot \rceil$	ceiling function
$\lfloor \cdot \rfloor$	floor function
G	group of Internet clients divided by Resource Distribution Algorithm
h	final height in the Resource Distribution Algorithm
B^*	bidding strategy profile of all game players
∇	differential operator (gradient)

List of Abbreviation

Abbreviation	**Full Name**
QoE	Quality of Experience
MHP	Model Human Processor
HIP	Human Information Processing
COS	Cumulative Opinion Score
RDA	Resource Distribution Algorithm
WUA	Willingness Update Algorithm
HCI	Human-Computer Interaction
DARPA	Defense Advanced Research Project Agency
OSI	Open Systems Interconnect
HTTP	Hypertext Transfer Protocol
DNS	Domain Name System
TCP	Transport Control Protocol
IP	Internet Protocol
IETF	Internet Engineering Task Force
MILSA	Mobility and Multihoming supporting Identifier Locator Split Architecture
LISP	Locater/ID Separation Protocol

EPIC	Executive Process Interactive Control
GUI	Graphical User Interface
OPENCV	Open Source Computer Vision
QoS	Quality of Service
CS	Communicating State
IS	Inactive State
KCTP	Key Context Transfer Protocol
TLS	Transport Layer Security
XMLP	Extensible Markup Language Protocol
LAN	Local Area Network
WAN	Wide Area Network
RTT	Round Trip Time
RTSP	Real Time Streaming Protocol
ITU-T	International Telecommunication Union Telecommunication standardization sector
PD	Proportional and Derivative
VoD	Video on Demand
CDN	Content Distribution Network
RMI	Remote Method Invocation
KKT	Karush-Kuhn-Tucker

Chapter 1

Introduction

1.1 Motivation

As an effective global system for multiplexed utilization of interconnected networks and their hosts, the Internet has achieved tremendous success in supporting today's Internet services. This is due to many fundamental and respected design principles for building the Internet protocol stack, such as layered architecture for task partitioning, packet switching for multiplexing, end-to-end arguments for defining protocols and global addressing for routing datagrams.

One of the fundamental design principles is that the Internet serves as the communication medium between two hosts that desire to speak to each other [1], where networked hosts work as the delegated representative of Internet end-users [2]. Such a design principle directly results in today's Internet simply regarding its end-user, host and services as one entity, namely the *Internet client*. More specifically, the Internet protocol stack conflates its dynamic end-user, networked host and various running services into one oversimplified concept: *an Internet client that desires communicating*. Fig. 1.1 simply depicts such a design principle for the

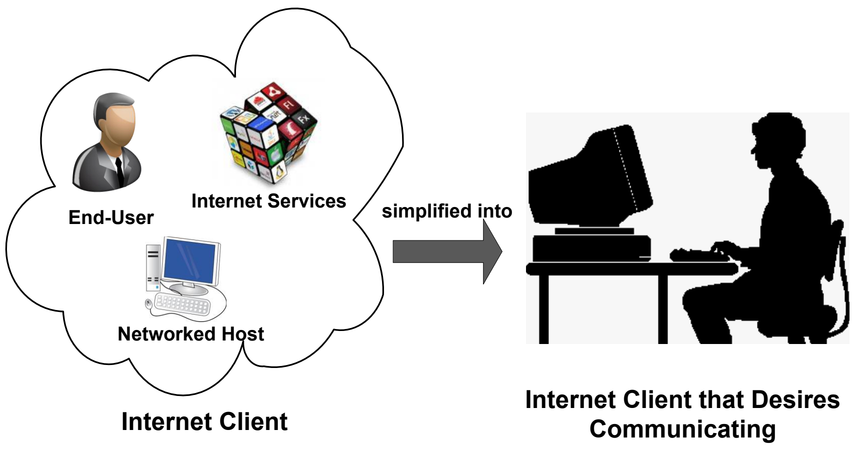


Fig. 1.1: Oversimplification of Internet client

Internet protocol stack and its communication protocols. Note that the end-user refers to the person who uses developed Internet services through a networked host. Internet services span a wide range of online services typically including World Wide Web, file transfer, streaming media as well as electronic mail. Internet application refers to any individual program that supports the corresponding Internet service. Networked hosts range in size from a small netbook through laptop to workstation.

There is no doubt that such a traditional design principle greatly decreases today's Internet complexity, but it essentially excludes the end-user factor from the Internet client entity and even the entire Internet protocol stack. Consequently, communication protocols in the Internet protocol stack inevitably ignores the end-user's presence, preference and any interaction activities with the Internet services and host. As a result, the Internet protocol stack is unable to take advantage of its end-user's information, especially the context information that can be utilized in different communication protocols and services. The absence of the end-user's context information may not only affect the underlying network performance but

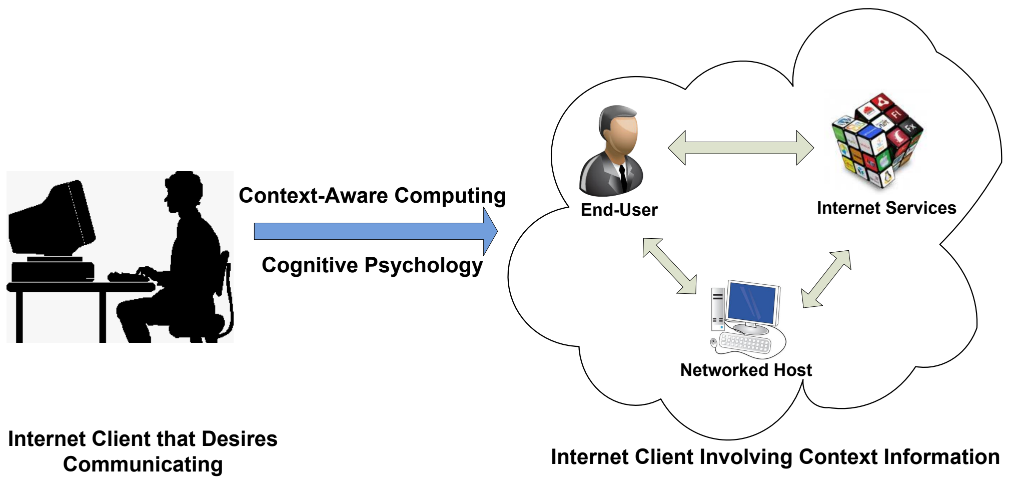

Context-Aware Computing
Cognitive Psychology
End-User
Internet Services
Networked Host
Internet Client that Desires Communicating
Internet Client Involving Context Information

Fig. 1.2: De-conflation of end-user, networked host and Internet services.

also decrease the usability and effectiveness of Internet services. Under many circumstances, it may also cause mobility and security issues.

On the other hand, advances in context-aware computing present major opportunities for empowering the traditional Internet to capture its end-user's presence, activities and other important context information. Briefly speaking, context-aware computing makes use of various sensors and techniques, e.g., wireless network camera and computer vision techniques, to collect a system's physical and environmental information. Such a system then can adapt its operations to the collected context information to increase its usability and effectiveness. There has been an entire body of research dedicated to building context-aware systems for different use cases and applications. For many existing context-aware systems, the Internet serves as a communication carrier to undertake the task of long distance data transmission. However, few prior systems and studies consider explicitly introducing the captured context information into the underlying Internet protocol stack and communication protocols.

Moreover, the developed cognitive models in cognitive psychology [3], which

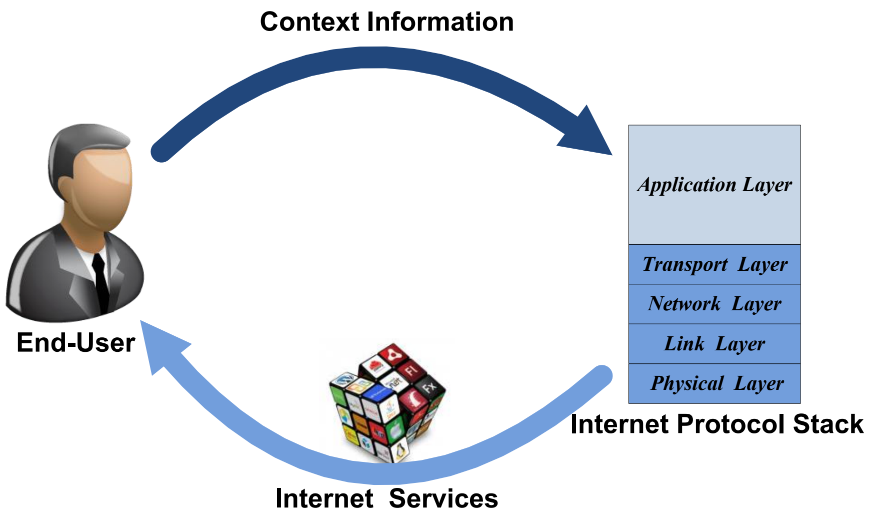


Fig. 1.3: New communication pathway and the closed communication loop.

focus on understanding humans and their activities, can also help capture the end-user's context information. Cognitive psychology is the study of how humans acquire, process and store information and solve problems. Cognitive psychology research as well as some fields in Human-Computer Interaction (HCI) [4] has made great efforts on modeling humans and interpreting their interactions with the external environment.

As shown in Fig. 1.2, combination of existing context-aware computing techniques with the established cognitive models directly helps to restore the oversimplified Internet client, and de-conflate Internet end-user, networked host and Internet services. It would eventually enable the Internet protocol stack and services to fully understand end-users, and actively adapt their operations and performance to the captured context information.

The research we are proposing aims at explicitly incorporating end-user's substantive context information, such as the interaction status between an end-user and different Internet services, into the underlying Internet protocol stack, and further enhancing the Internet as a user-centric, context-aware, and interactive communication system. As illustrated in Fig. 1.3, besides the conventional communication pathway from the Internet protocol stack through Internet services to the client side, a novel communication pathway for transmission of context information from the client side to the Internet protocol stack is proposed. Hence, our work essentially establishes a closed communication loop involving the Internet protocol stack, Internet end-users, and Internet services.

1.2 Research Challenges

Why is introducing end-user's context information into the Internet protocol stack so different from building traditional context-aware systems? The difficulties in enabling a user-centric and context-aware Internet protocol stack stem mainly from the following open issues:

1. What kind of context information is required, and even indispensable, for the Internet protocol stack? How to capture and ascertain such context information?

2. How does the Internet protocol stack utilize and adapt itself to the derived context information?

3. How to motivate selfish Internet clients to actively provide and share their actual context information?

Firstly, any information that can be used to characterize the situations between an end-user and Internet services or host is valid and regular context information. However, only the highly abstract and most substantive context information, which describes end-user's interaction states with the working Internet services, makes sense to the Internet protocol stack. It is because any redundant or invalid context information would easily degrade the performance of the Internet protocol stack, whose key responsibility is to provide end-to-end connectivity service. Hence, only the concise context information that accurately reflects dynamic changes of an end-user's real-time interaction states can be introduced into the protocol stack. In addition, such advanced context information should be acquired and verified from multiple and heterogenous sources.

Secondly, the layered architecture of the Internet provides natural abstractions to deal with the functional hierarchy present in the Internet protocol stack, and the communication protocols running at a particular layer do not need to worry about the rest of the stack. Hence, context information should be cautiously introduced into communication protocols to avoid spoiling the integrity and modularity of the Internet architecture. Improperly introducing the context information would effect the basic functions and operations of the relevant protocols, and even lead to unintended consequences on overall performance of the entire layer.

Last but not least, even though the desired context information has been accurately captured and successfully incorporated into the Internet protocol stack, Internet clients would be reluctant to provide and share their context information, especially the information that may result fewer allocated resources. This is due to the fact that all Internet clients are selfish and rational in nature, and these unconstrained competitors always act in a way to maximize their own benefits. Hence, a systematic mechanism or framework is required to incentivize actual context sharing and moderate competition among Internet clients, when the designed system provides the context-driven service differentiation.

1.3 Thesis Contributions

The contributions of this thesis are listed below:

- Design a functional module, called the User-Context Module, to explicitly and smoothly incorporate the advanced context information of end-users into the Internet protocol stack.
- Construct a group of context models to deduce two fundamental categories of the context information for the representative Internet services.
- Design and implement two practical applications of the User-Context Module, which interact with the distinct communication protocols on different layers to enhance the end-user's Quality of Experience (QoE) and improve the underlying protocol performance.
- Build a resource distribution framework for the User-Context Module to provide context-driven service differentiation and incentivize actual context information sharing and moderate competition among selfish Internet clients.

The first two contributions mainly address the first research problem raised in the previous section, i.e., what is the required context information and how to derive it. The proposed solution, namely the User-Context Module with its three key subsystems, empowers the Internet protocol stack to recognize two fundamental interaction states between an end-user and operating Internet services. For different Internet services, the defined context information can be effectively deduced by the built context models, which leverage on cognitive psychology and first-order rule-based reasoning.

The third contribution are two distinct applications of the User-Context Module, namely the HTTP case and the TCP case. They demonstrate the User-

Context Module's operations and impacts as well as its Control Subsystem's design and implementation. These two applications explore the design space of the User-Context Module and also inform other novel utilization of the deduced context information for the Internet protocol stack. Hence, it takes solid steps towards solving the second problem raised in the previous section, i.e., how to enable the Internet protocol stack to utilize and adapt itself to the deduced context information.

The last contribution is to provide a widely applicable framework with the practical algorithms to encourage selfish Internet clients sharing actual context information and meanwhile reducing the excessive competition among them. The design philosophy behind the proposed framework helps the designers to consider the context owner factor and view the design problem in its entirety when building a new User-Context Module application. The proposed framework addresses the last research problem raised in the previous section.

In short, our research efforts have been made to separate end-users from the conventional oversimplified Internet client, utilize the specific end-user's context information to improve the Internet protocol stack performance and eventually provide services to ordinarily Internet end-users.

1.4 Organization of the Thesis

The thesis is organized in the following manner:

Chapter 2 summarizes the related work from the perspectives of the Internet protocol stack design and the end-user's context recognition, respectively.

Chapter 3 proposes the architectural framework of the User-Context Module through augmenting the traditional Internet protocol stack, and lays a special stress on designing and implementing two subsystems, i.e., the Context Sensing Subsystem and the Context Model Subsystem, to capture and deduce the desired context information.

Chapter 4 presents the first application of the User-Context Module, which mainly introduces the deduced context information into the Application Layer's HTTP protocol. A specifically designed Control Subsystem is designed and implemented for this application. The first application demonstrates how the User-Context Module improves HTTP protocol performance and the end-user's QoE.

Chapter 5 presents the second application of the User-Context Module, which mainly introduces the deduced context information into the Transport Layer's TCP protocol. The second application demonstrates how the User-Context Module improves TCP protocol and enhances the end-user's QoE.

Chapter 6 proposes a supporting framework for the User-Context Module, which provides context-driven service differentiation and incentivizes context sharing and moderate competition among Internet clients.

Chapter 7 contains a summary and suggestions for future research in this direction.

Chapter 2

Background and Related Work

This chapter discusses the background research work related to this dissertation. The review is cross-disciplinary and thus it is classified into three general fields: (1) Internet protocol stack design; (2) recognition of end-user and context information; (3) end-user's QoE. The Internet protocol stack design is first reviewed. Then, we discuss the second field, which mainly includes end-user modeling and context-aware computing. Finally, we give an introduction to the basic concept of QoE. Fig. 2.1 illustrates the organization of this chapter.

2.1 Internet Protocol Stack Design

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 an Internet protocol stack and successfully connected worldwide hosts together.

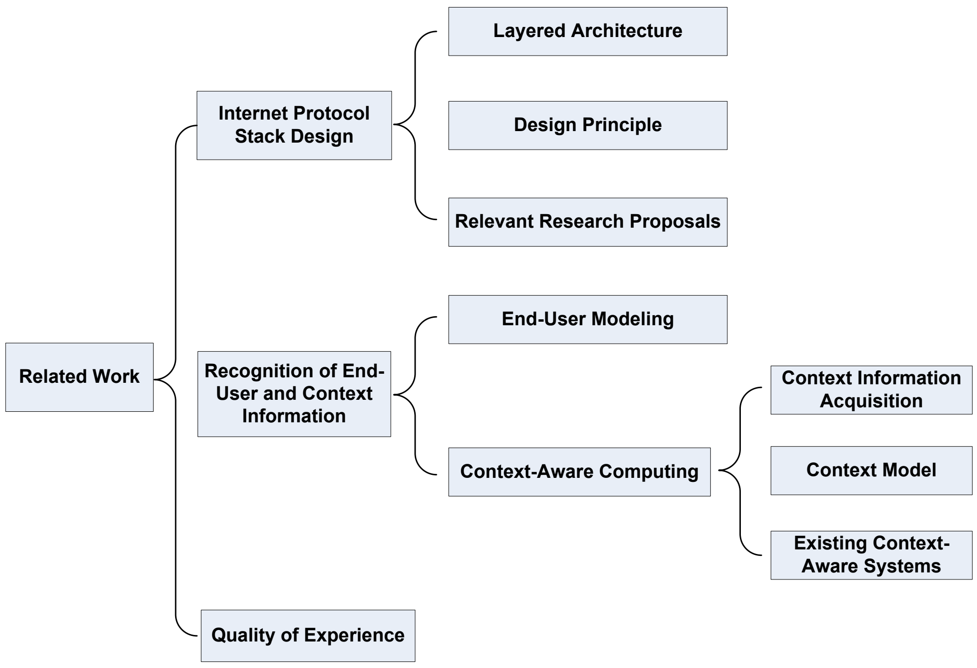

Related Work
Internet Protocol Stack Design
Layered Architecture
Design Principle
Relevant Research Proposals
Recognition of End-User and Context Information
End-User Modeling
Context-Aware Computing
Context Information Acquisition
Context Model
Existing Context-Aware Systems
Quality of Experience

Fig. 2.1: Organization of the Related Work.

2.1.1 Layered Architecture

In order to accomplish complicated data communication tasks, the Internet partitions its methods and protocols into several hierarchical abstraction layers. Each layer has specific features and functionalities with peer interactions at equivalent layers across networks. As shown in Fig. 2.2, the five-layer Internet protocol stack [5] and the seven-layer Open Systems Interconnect (OSI) model [6] are the two commonly used models for organizing and describing the layered architecture of today's Internet. In the five-layer Internet protocol stack, the Application Layer is the top layer and contains all protocols and methods that fall into the realm of process-to-process communications across an IP network, such as Hypertext Transfer Protocol (HTTP) and Domain Name System (DNS). The Transport Layer is mainly responsible for supporting the end-to-end conversation, where Transmission Control Protocol (TCP) is the primary connection-oriented proto-

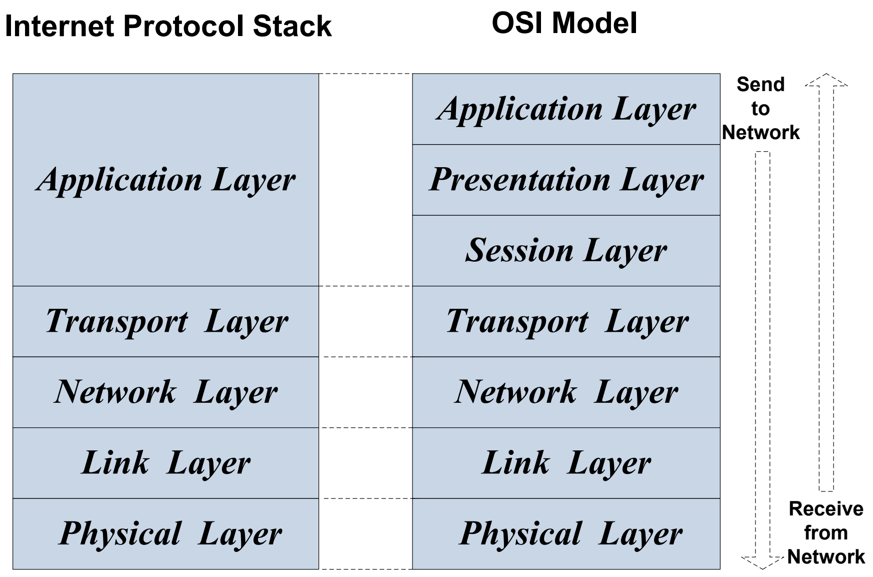


Fig. 2.2: Internet protocol stack and OSI model.

col. The Network Layer is responsible for packet forwarding including routing with an official packet format defined in Internet Protocol (IP). The Data Link layer provides the abstraction of a link, as well as the ability to transmit and receive bits over the link. The Physical Layer handles signals and supports the communication service in bits. In short, such a layered architecture described by the five-layer protocol stack plays a prominent role in the success of the modern Internet. Any new enhancements for the Internet should maintain the integrity and the modularity of the layered architecture.

2.1.2 Design Principles

Since the inception of the Internet, many fundamental and respected principles have been gradually introduced and implemented in its layered architecture and communication protocols, such as packet switching for multiplexing [7], end-to-end arguments for defining communication protocols [8] and global addressing for routing datagrams [6]. Regulated by those established design principles, Internet designers do their work: they design, revise, configure and deploy a variety of communication protocols and Internet services.

One of the fundamental design principles is that the Internet serves as the communication medium between two ***hosts*** that desire to speak to each other [1, 9]. The Internet standard [2] published by Internet Engineering Task Force (IETF) specifies that "*Internet host, or simply 'host', is the ultimate consumer of communication services. A host generally executes application programs on behalf of user(s), employing network and/or Internet communication services in support of this function*". With such a host-centric vision, the Internet protocol stack simply regards the Internet end-user, host and Internet service as one entity, namely the *Internet client*. More specifically, the Internet allows any networked host to be the representative of its end-user, and assumes that any network host always desires to communicate with each other. Such a design principle and assumption greatly reduces the complexity of today's Internet architecture and communication protocol design. However, they inevitably result in the Internet protocol stack oversimplifying the concept of the *Internet client*. Accordingly, the designed communication protocols completely ignore the end-user's presence, interaction state and any other relevant and important information.

2.1.3 Relevant Research Proposals

There have been relevant research proposals within the scope of extending the concept of Internet client, particularly the studies on the identifier-locator split architecture. The identifier specifies who the networked host is, and the locator explains where the networked host is. Briefly speaking, the identifier-locator split

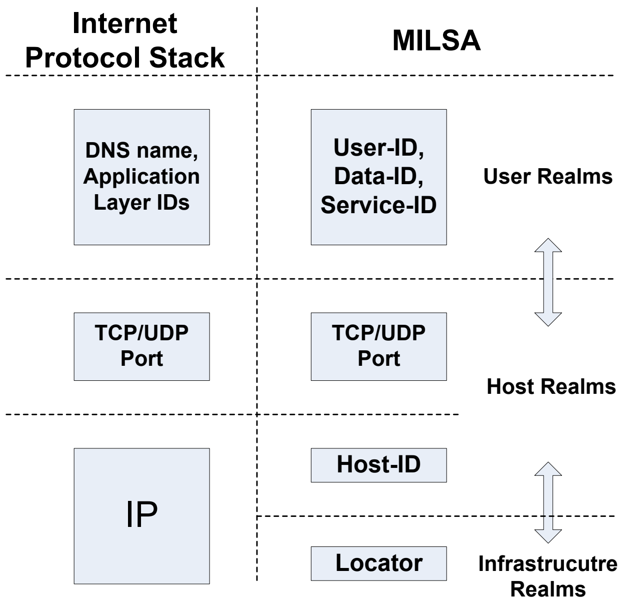


Fig. 2.3: Basic MILSA architecture and its three realms.

architecture attempts to use independent name spaces to help the Internet protocol stack recognize the host and the host address separately. For example, MILSA (Mobility and Multihoming supporting Identifier Locator Split Architecture) [10] introduces a new Host-ID sub-layer into the Network Layer of the protocol stack to separate networked host from its locater. As shown in Fig. 2.3, it defines the independent user realm, host realm and infrastructure realm, which are handled by their individual realm managers.

MILSA and other identifier-locator split architectures, such as HIP [11] and LISP [12], aim to eventually enable Internet end-users, rather than the networked host, be the final destination of Internet services. Hence, to some extent, they incorporate Internet end-users into the architecture of the Internet protocol stack, although none of end-user's context information is included. For more details of the identifier-locator split architecture and relevant research proposals, the reader is referred to [9] and the references therein.

2.2 Recognition of End-User and Context Information

In order to enable the Internet to recognize the Internet end-user's context, we must first understand the end-user himself. After that, we can employ approaches and techniques to capture the required context information. Hence, in this section, we first introduce the particular field related to end-user modeling, and then we review the related work in context-aware computing.

2.2.1 End-User Modeling

Cognitive psychology [13] as well as particular fields in Human-Computer Interaction [4] offer us a group of approaches to model human and interpret human's interaction behavior. The Human Information Processing (HIP) approach [3] in cognitive psychology holds considerable promise to model how an end-user receives, stores, integrates and uses information from the external environment, such as Internet services. The basic idea of the HIP approach is that the human is like a computer or a complex system that can be analyzed in terms of subsystems and their inter-relationships. Fig. 2.4 depicts a basic and abstract model of HIP. Different HIP models have been developed to characterize or predict an end-user's interaction activity and behavior. The most widely known models include Model Human Processor (MHP) proposed by Card et al. [14] and Executive Process Interactive Control (EPIC) [15]. Both models assume that a series of discrete phases compose the information processing, and the output of one phase serves as the input for the next. McClelland's cascade model [16] considers that each phase is continuously active with continuous output values, where only partial information at each phase is transmitted to the next. Besides the discrete and continuous

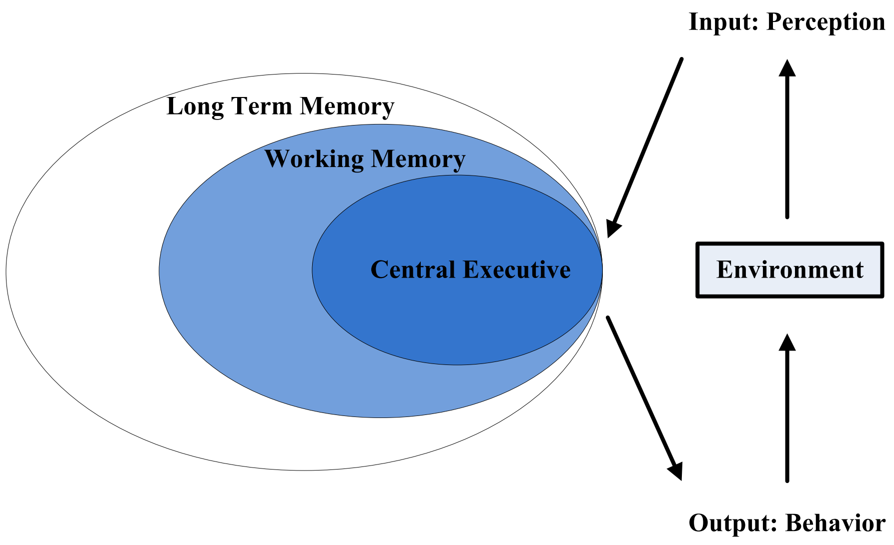


Fig. 2.4: Basic and abstract model of Human Information Processing (HIP).

phase models, Sequential Sampling Models [17] and other applicable HIP models [18] have been proposed and developed. Furthermore, some new approaches start to challenge and improve on the traditional HIP approach, such as the situated cognition [19] and the cybernetic approach [20]. In this dissertation, the proposed context model is based on MHP, not only because it is the most widely known and established HIP model, but more importantly, it offers an efficient way to precisely define an end-user's different interaction states, which can be validated by the specific interaction conditions.

2.2.2 Context-Aware Computing

Besides the established models to describe an end-user, context-aware computing approaches and techniques are also indispensable for the recognition of end-user's context. The ubiquitous computing idea [21] envisioned by Weiser has evolved to a more general paradigm known as context-aware computing. The term *context* refers to any information that can be used to characterize the situation of an entity that is considered relevant to the interaction between an end-user and the application, including the end-user and the application themselves [22]. Context-aware computing enables a system to be aware of its end-user and adapt its operations to the captured end-user's context information.

Context Information Acquisition

Context information acquisition refers to the process of capturing and managing the basic context information from heterogeneous sensors. The sensors can be classified into physical sensors and virtual sensors: physical sensors are the hardware sensors that capture the information from the physical environment, while virtual sensors collect data from software applications including operating systems and Internet services. Different context information acquisition approaches would directly influence the architectural style of a built context-aware system. In general, there have been several context information acquisition approaches, typically including the direct sensor access approach, the context server based approach and the middleware based approach [23]. The middleware based approach uses a method of encapsulation to separate and hide low-level sensing details to ease rapid prototyping and implementing of a context-aware system. The separation of detecting and using context is also necessary to improve the extensibility and the reusability of a context-aware system. The middleware based approach has been widely adopted in the existing context-aware systems, such as SOCAM [24] and Gaia systems [25], which effectively support acquiring, discovering, interpreting and disseminating different context information. Fig. 2.5 illustrates a typical and simplified middleware based context-aware system architecture consisting of Context Sensing Layer, Context Middleware Layer and Context Application Layer. Our User-Context Module architecture also draws upon the design experience

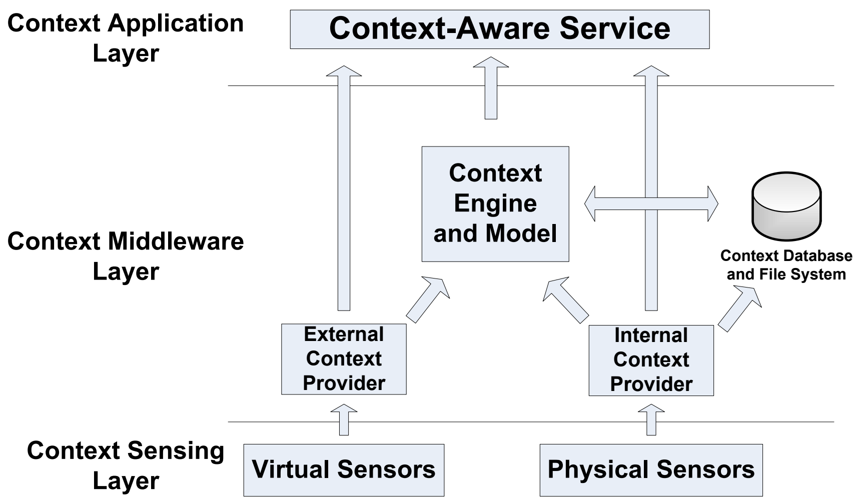


Fig. 2.5: A typical and simplified middleware based context-aware system architecture.

from the middleware based approach for acquisition of Internet end-user's context information.

Context Model

After successfully acquiring basic context information, context models are often required to define, ascertain and store some advanced context data in an application processable form. In general, the existing context models can be classified into several categories, including the logic based model, the ontology based model, the object oriented model as well as the key-value model [26]. The logic based model and the ontology based model are two widely used models in today's context-aware systems. The logic based model often adopts an inference engine, or called reasoning engine, to deduce new facts based on the pre-defined rules and expressions. It has a high degree of formality, and allows addition, update or removal of new facts. The ontology based model directly applies the ontology reasoning techniques, which has high and formal expressiveness. The developed context models for a variety of context-aware systems are well summarized in [27]. Our User-Context Module adopts the first-order rule-based reasoning engine, and thus our context model can be classified into the category of the logic based model.

Existing Context-Aware Systems

We finally provide an overview of the existing context-aware systems. The *Active Badge Location System* [28] is always regarded as the first context-aware system, which utilized an end-user's location context information to forward phone calls to a telephone close to the end-user. In later context-aware systems, end-user's identity, activity, time and other context information are gradually introduced and employed [27]. The latest context-aware systems are always characterized by an intelligent environment, user-centered service and transparency. They deploy various autonomous computational devices and sensors to build a user-centered environment for distinct application scenarios. In most cases, end-users in such an intelligent environment do not notice those integrated devices and sensors while they benefit from the supported applications and services.

With the aim of having "the system adapt to its users", there have been tremendous efforts in building context-aware systems from both the technical and the social perspectives [29–33]. MIT has built a pervasive human-centered computing environment in the Oxygen project [29]. The system deploys multiple embedded computational devices called Enviro21s (E21s) in offices, cars and homes to collect context information. With the hand-held devices called Handy21s (H21s) and the indoor location support, Oxygen's system can assist its users perform a group of tasks in their daily lives. Georgia Tech's researchers have designed an environment that can sense the inhabitants through a variety of sensing technologies in their Aware Home project [30]. One interesting Aware Home initiative called

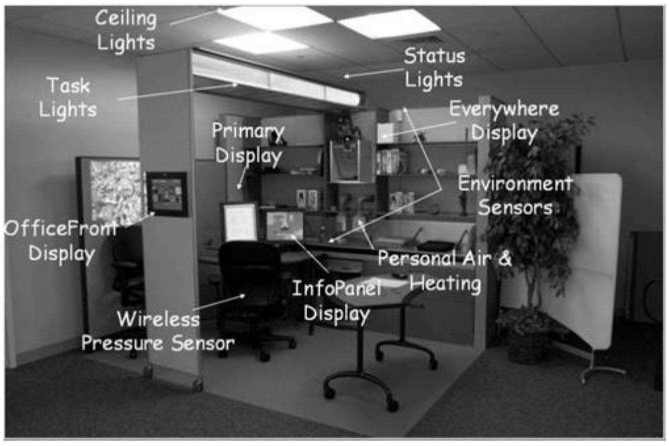


Fig. 2.6: An example of the latest context-aware systems: IBM Blue Space.

“Aging in Place” focuses on developing the technology and applications which enable senior adults to live independently in their homes. IBM has proposed the next-generation workspace solution in its “Blue Space” project [31], which integrated sensors, actuators, displays and wireless networks into one work place. The workspace solution, as shown in Fig 2.6, aims to increase the productivity by deterring unwanted interruptions and facilitating communication among group members.

In the built context-aware systems, the Internet protocol stack has always served as the default long distance data communication carrier. However, limited prior projects consider enabling the Internet to directly utilize the captured end-user’s context information. The context-aware Web service [34] can be regarded as a good attempt in this direction. They mainly employ Web end-user’s context information to support Web content adaptation [35], communication optimization [36] as well as security and privacy control [37]. For example, in Web content adaptation systems, specific context information is always used to customize Web content in a form suitable to the end-user. Nevertheless, existing context-aware Web service systems only utilize context information to adjust high level Internet services, and none of them introduce end-user's context information directly into the Internet protocol stack, or more specifically, the underlying Internet communication protocols.

2.3 Quality of Experience (QoE)

Since one of the main objectives of the proposed User-Context Module application is to enhance the end-user's QoE, it is necessary to first discuss its basic concept and assessment approach.

The ITU Telecommunication (ITU-T) Standardization Sector defines QoE as "*the overall acceptability of an application or service, as perceived subjectively by the end user*" [38]. Other concepts of QoE [39, 40] can be simply interpreted as the end-user's subjective perception on the qualitative performance of communication systems and applications. QoE is currently receiving immense interest from both of the academic and the industrial perspectives. Particular attention is given to assess and measure QoE not only in terms of the traditional Quality of Service (QoS) parameters [41], but a joint consequence of the communication context environment, the characteristics of the service in use and the underlying network performance. Since a large number of variables and information need to be considered, Brooks et al. [42] propose a structured assessment approach to describe end-user's QoE with the following clause:

IF <*Communication Situation*>;

USING <*Service Prescription*>;

WITH <*Technical Parameters*>;

THEN <*end-user's QoE*>.

Such an assessment approach explicitly combines the end-user's usage context information and technical parameters together to measure the QoE. All the attributes in the bracket have many possible options. For example, <*Communication Situation*> takes into account objective communication context related to end-users. The <*Service Prescription*> can be Live Streaming, File Transfer or any other types of Internet services. The <*Technical Parameters*> ranges from the bit rate to the protocol type, and a more complete list is given in [42]. For the <*end-user's QoE*>, the Opinion Score scale from 5 to 1 can be used to describe the end-user's subjective satisfaction on the performance of a given Internet service.

With the structured assessment approach, we can describe and measure the end-user's QoE in a clearer and comprehensive way. The progress on the techniques for enhancing and modeling QoE would impact Internet design and eventually benefit the ordinary Internet end-users.

2.4 Summary

In this chapter, we first review the Internet protocol stack design issue, and then discuss the end-user modeling and the context information recognition. Last but not least, we introduce the definition of QoE and its assessment approach. The traditional host-centric design principle causes that the Internet protocol stack inevitably ignores its end-user's presence, interaction activities and other context information. In order to retrieve and utilize the substantive context information, it is necessary to enable the Internet protocol stack to recognize and understand its end-users. Cognitive psychology provides the required models and framework. Context-aware computing approaches and techniques draws a blueprint for enabling the Internet further adapt to the captured context information. Moreover, the latest context-aware systems demonstrate how to derive advanced context information and utilize them for system-level adaptations. Our investigation of these related work and background knowledge has informed our design objectives, namely enhancing the end-user's QoE and improving the performance of Internet communication protocols.

Chapter 3

User-Context Module Architecture and its Implementation

In Chapter 1, we described the motivation and the research challenges. In Chapter 2, we discussed the related work from the perspectives of the Internet protocol stack design and the context recognition for Internet clients. In this chapter, we present a new functional module, called the User-Context Module, to explicitly and smoothly incorporate the context information of end-users into the Internet protocol stack through augmenting the traditional layered network architecture. The User-Context Module consists of three indispensable subsystems: the Context Sensing Subsystem, the Context Model Subsystem and the Control Subsystem. In this chapter, we also put special emphasis on designing and implementing the Context Sensing Subsystem and the Context Model Subsystem to deduce the specific advanced context information.

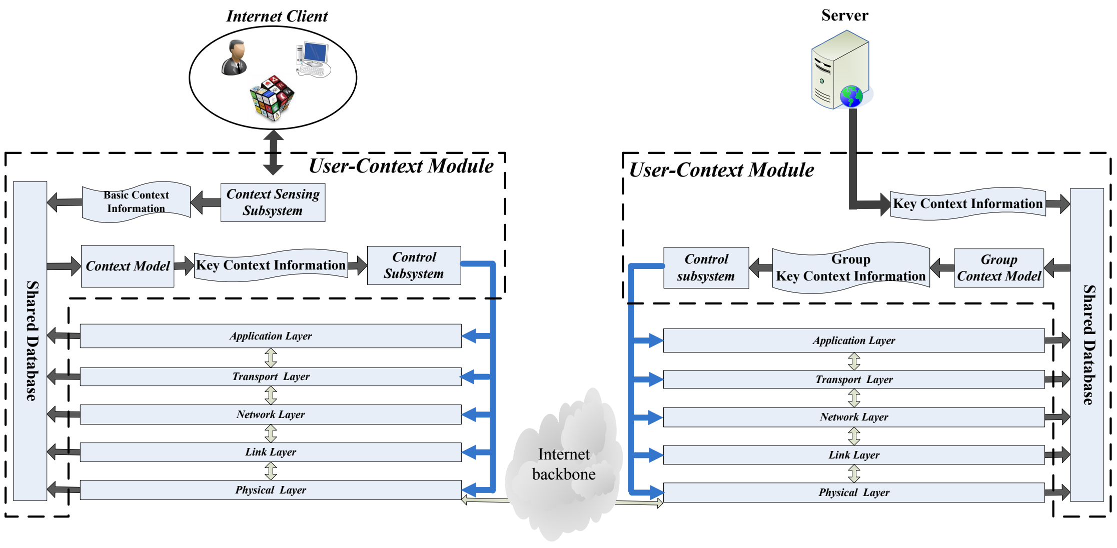


Fig. 3.1: System block diagram of the User-Context Module with the Internet protocol stack.

3.1 Architectural Building Blocks

With the aim of introducing the context information of end-users into the basic infrastructure of the Internet, we propose a functional module, namely the User-Context Module, running on top of the five-layer Internet protocol stack. The User-Context Module mainly operates under the traditional client-server architecture with a strong emphasis on utilizing the interaction information between an end-user and the Internet. The system block diagram is illustrated in Fig. 3.1.

As indicated earlier, the User-Context Module consists of three main components, which are called Context Sensing Subsystem, Context Model Subsystem and Control Subsystem. In general, the Context Sensing Subsystem undertakes the basic context information gathering task and directly works with end-users on the Internet client side. Based on the captured basic context information, the Context Model Subsystem utilizes *context models* to deduce the advanced context information that characterizes interactions within an Internet client. In this dissertation, such advanced context information is termed ***Key Context Infor-***

mation (KCI). With the deduced KCI, the Control Subsystem directly adjusts the Internet protocols and services in different layers to improve underlying network performance. We present the design and the implementation issues of these three subsystems of the User-Context Module in the subsequent three sections.

3.2 Context Sensing Subsystem

3.2.1 Overview of Context Sensing Subsystem

The Context Sensing Subsystem directly interacts with the individual Internet end-user, networked host and their surrounding environment. Since its main functionalities involve monitoring, collecting and recording the basic context information of end-users, the Context Sensing Subsystem mainly operates at the Internet client side as shown in Fig. 3.1. The Context Sensing Subsystem requires a variety of physical sensors and virtual sensors: physical sensors are the hardware sensors that capture the information from the physical environment, while virtual sensors collect data from the software systems, e.g., operating systems running on the networked host and working Internet services. Sometimes, the Context Sensing Subsystem is also equipped with specifically designed User Interfaces to receive manual inputs from Internet end-users. In general, the Context Sensing Subsystem fulfills the following two functions:

(1) Monitor and record the interaction activities and other relevant basic context information between an end-user and Internet services running on a networked host. The interaction activities include the host-oriented information, e.g., which Internet service is currently displaying in the foreground of the network host screen, as well as the user-oriented information, e.g., whether an end-user's eye-gaze direction is towards the networked host

Table 3.1: Basic Context Information from End-Users and Internet Services

	End-User		Internet Service	
Basic Context Information	E1	Eye-gaze direction towards networked host screen	S1	Displaying in the foreground of network host screen
	E2	Wearing earphone or near speaker	S2	Generating audio output
	E3	Touching Mouse	S3	Receiving mouse message
	E4	Touching Keyboard	S4	Receiving keyboard message
	E5	Near Microphone	S5	Receiving audio input

screen. Other relevant basic context information includes the end-user's location, identity and preference. By leveraging on the intelligent physical and virtual sensors, the Context Sensing Subsystem can perform its work and fulfill its tasks in an invisible way.

(2) Besides automatically collecting the basic context information in the background, the Context Sensing Subsystem may also provide the end-user a direct and visible interaction service. For example, an end-user could inform the Internet protocol stack his current state or preference by simply pressing some matching buttons on a specially designed graphical user interface (GUI), or some GUI could show significant underlying network conditions to the end-user through any user-friendly ways.

In short, the Context Sensing Subsystem mainly undertakes the interaction activities and other meaningful basic context information gathering task. All the captured basic context information are delivered to the Context Model subsystem in real-time for further processing.

3.2.2 Implementation of A Context Sensing Subsystem

We implement a Context Sensing Subsystem specifically designed to detect the interaction activities between an end-user and Internet services running on a networked host (laptop or desktop). To enable the designed subsystem more practical and generally applicable, its physical sensors are mainly the default devices on a common network host, including keyboard, mouse and Webcam. Table 3.1 lists the collected basic context information from the end-user side and the Internet service side.

From the end-user side, for example, the Context Sensing Subsystem utilizes a built-in Webcam or a common USB Webcam as the physical visual sensor to capture open eyes on the human frontal face and accordingly estimate whether an end-user's eye-gaze direction is towards the networked host screen in real-time. The Open Source Computer Vision (OpenCV) library [43] and the existing visual tracking algorithms [44, 45], greatly facilitate building such a video-based eye-tracking system. We use the models "haarcascade-frontalface-alt.xml" and "haarcascade-eye-tree-yeglasses.xml" of the Haar Classifier [46] in the OpenCV Library to detect an end-user's frontal face and his open eyes as shown in Fig. 3.2. The Haar Classifier works as a highly efficient and accurate algorithm to detect human facial features, which can analyze a 320 by 240 image with a frame rate of 3 frames per second by using 1.2 GHz AMD processor [47].

From the Internet service side, for example, the Context Sensing Subsystem can automatically monitor and detect which running Internet service is displaying in the foreground of the networked host screen and which Internet service is receiving the mouse/keyboard inputs. Our Context Sensing Subsystem is developed in Visual C++ under Microsoft .NET Framework on Win32 platform, and Fig. 3.3 demonstrates the physical sensors employed in our built Context Sensing

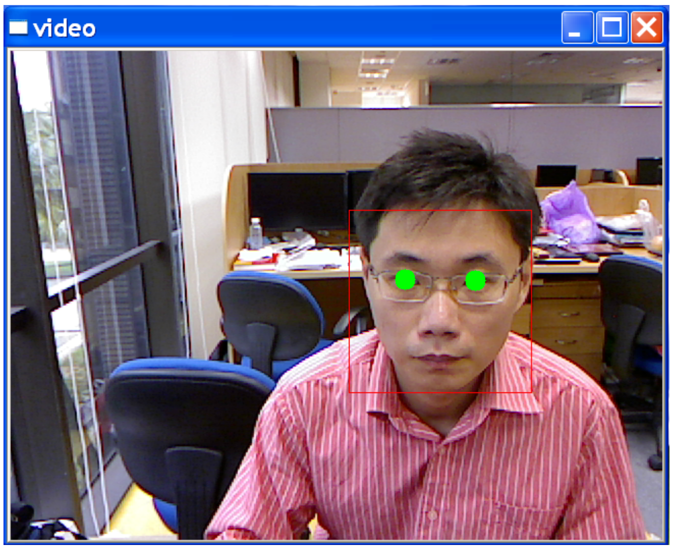


Fig. 3.2: Detecting an end-user's frontal face and open eyes in real-time.

Subsystem.

Besides the basic context information in Table 3.1, the built Context Sensing Subsystem also collects the specific underlying network conditions, which involve the running communication protocols, the critical network configurations and the important Quality of Service (QoS) parameters. For example, the Context Sensing Subsystem periodically samples the bandwidth consumption of each running Internet service at the client side. To measure the bandwidth consumption in real-time, a third party driver called WinPcap [48] is employed to intercept packets flowing through the network adapter installed on the networked host. Some functions have not been fully implemented for the current version of our Context Sensing Subsystem, such as detecting whether an end-user is sitting near the speaker, but a great deal of research has been done for solving such a positioning problem and a variety of RFID location sensing systems have even been commercialized [49]. Moreover, other latest sensing technologies in the latest context-aware systems [27, 34] can also be introduced to the new version of the Context Sensing Subsystem for col-

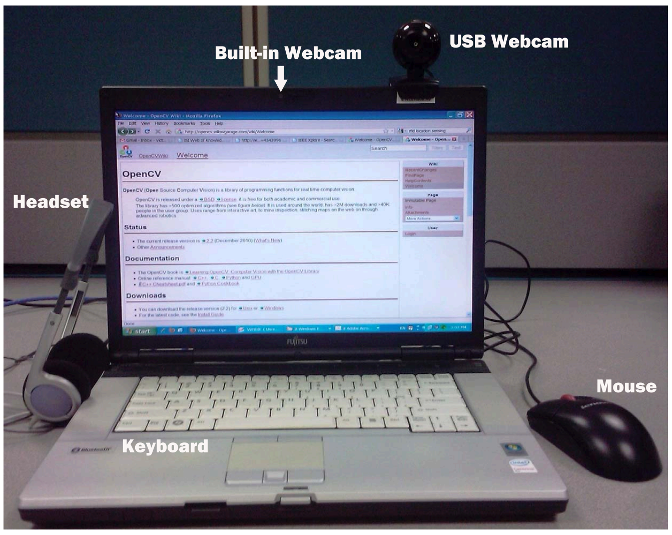


Fig. 3.3: Physical sensors in the built Context Sensing Subsystem.

lecting additional basic context information in a more efficient way. In short, the rapid advancements in ubiquitous sensing and computing technologies greatly facilitate building the Context Sensing Subsystem for the User-Context Module.

Under the framework of the User-Context Module, the basic context information and the underlying network conditions captured by the built Context Sensing Subsystem are directly delivered to the Context Model Subsystem in real-time.

3.3 Context Model Subsystem

3.3.1 Overview of Context Model Subsystem

The Context Model Subsystem plays a key role in the User-Context Module, because it is the component for constructing, hosting and utilizing the context model to deduce the KCI. The context model refers to the abstract data model for ascertaining an end-user's presence, preference and interaction activities. In order to build a reliable and accurate context model, the related cognitive psychology models [3], Human-Computer Interaction (HCI) knowledge [4], as well as data mining and machine learning methods [50] need to be employed. When more complex context information need to be deduced, advanced reasoning approaches such as the ontology reasoning [51] can also be introduced.

Besides the context model, the Context Model Subsystem also includes a shared database, which is used to store and retrieve the captured basic context information and various underlying network conditions. Depending on the built context model, the shared database can also perform the task of data filtering to pick out all irrelevant information before sending data to each built context model. The context model processes those delivered data and finally deduces the KCI, which will be promptly delivered to the Control Subsystem. Note that the Context Model Subsystem can work at both the Internet client side and the server side as shown in Fig. 3.1, where the context model on the server side can be called the group context model and consequently deduces the group KCI. The group KCI is mainly used to help enhance server performance and facilitate server batch processing. We will further present it in Chapter 6.

3.3.2 End-User Modeling

Building a reliable and accurate context model for ascertaining the presence, preference or complex interactions of an end-user is not a straightforward task, and we must first understand and model the end-user himself before further investigating his interaction activities. Fortunately, cognitive psychology and HCI fields offer a variety of well developed frameworks and models to explain and describe the human internal structure and his interaction behavior. The Human Information Processing (HIP) approach in cognitive psychology field is one of the most successful methods to conceptualize how the human mind works when he

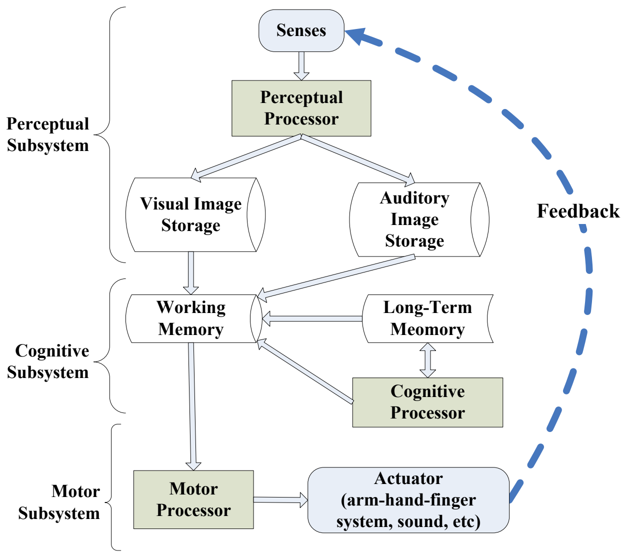


Fig. 3.4: Model Human Processor (MHP) framework.

interacts with the external environment. The basic idea of this approach is that human interaction behavior is a function of several ordered processing stages. In other words, the human is like a system that can be analyzed in terms of subsystems and their interrelationships. Different architectures, such as the ACT [52] and the SOAR [53] models, hold great promise for the HIP approach, while the most widely accepted and well-known one is the Model Human Processor (MHP) proposed by Card et al. [14].

As shown in Fig. 3.4, MHP consists of three interacting subsystems: 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 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 humans as a sequential or parallel operation of these three MHP subsystems. Furthermore, the *rationality principle* and *problem space principle* of MHP indicate that human behavior is based on rational activity, which means a normal end-user will not randomly and arbitrarily change from one state to another. Moreover, all rational activities serve to achieve human's explicit goals, given the task and external information and bounded by his knowledge and processing ability.

Based on the MHP and its two basic principles, we can define many reasonable states to describe the basic end-user status with an Internet service. In this work, we define two important end-user states with an individual Internet service:

(1) ***User Perception State***: Both the end-user's Perceptual and Cognitive subsystems are turned ON to acquire and process the information of the corresponding Internet service.

(2) ***User Halt State***: The end-user's ***three*** subsystems, i.e., the Perceptual, Cognitive and Motor subsystems, are all turned OFF with the corresponding Internet service.

The above defined two end-user states can be applied to most Internet services, and each running Internet service can be associated with only one defined end-user state at a time, i.e., either the ***User Perception State*** or the ***User Halt State***. Note that here the Internet service refers to the smallest unit of service: for example, each open Web page tab of a Web browser is considered as one individual Internet service of the Web browsing service. Other possible situations of an end-user can be simply termed as *Unidentified User State*.

3.3.3 Key Context Information (KCI)

As indicated earlier, the KCI serves as the standard outputs of the context model and the direct inputs of the Control Subsystem. Hence, it plays a crucial role in the User-Context Module and needs to be clearly defined and specified in the context model.

With the above defined end-user states, a variety of KCIs can be defined depending on different usage scenarios. In this work, we define two fundamental categories of the KCI for any Internet client:

(1) **COMMUNICATING STATE** (**CS**): The end-user stays in the User Perception State **AND** the corresponding Internet service keeps working.

(2) **INACTIVE STATE** (**IS**): The end-user stays in the User Halt State **OR** the corresponding Internet service stops working.

The defined two categories of the KCI are applicable to interaction activities between an end-user and different Internet services, regardless of the end-user's identity and the type of networked host. Moreover, they can be used as the cornerstones for further defining and describing more complex KCI. With the explicitly defined KCI, we proceed to build the corresponding context model. In addition, when an end-user stays in the Unidentified User State, the corresponding KCI can be simply named UNIDENTIFIED STATE.

3.3.4 Building the Context Models

From the definitions of the User Perception State and User Halt State, we know that monitoring and recognizing the status of the end-user's three MHP subsystems is the most straightforward way to identify the end-user state and eventually deduce the two categories of the defined KCI:

- For the end-user's Perceptual subsystem, MHP shows that the most important memories for the human perceptual processors are the Visual Image Storage and the Auditory Image Storage. Thus, the Context Model Subsystem should strive to verify whether an end-user perceives any visual or auditory information. Meanwhile, it needs to detect the source of the visual or auditory information among the running Internet services.

- For the end-user's Cognitive subsystem, although many researchers attempt to model and build the cognitive architectures and models [54], 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, we could estimate the ON/OFF status of the end-user's Cognitive subsystem.

- For the end-user's Motor subsystem, the arm-hand-finger system is considered as the most important actuator by MHP. Hence, from observing the interaction activities between the end-user fingers and the keyboard (or mouse) on the networked host, the Context Model Subsystem could deduce whether the end-user's Motor and Cognitive subsystems are turned ON and working. Moreover, MHP takes the human vocal system as another actuator, and thus it is also a significant clue to infer the ON/OFF status of the Motor and Cognitive subsystems.

Based on the above analysis and the defined KCI, five ***Interaction Conditions*** described in Table 3.2 require to be verified by the Context Model Subsystem. Table 3.2 also gives the corresponding ***validation criteria*** for each Interaction Condition, which can be found in Table 3.1. The given validation criteria shows that verifying each Interaction Condition requires two pieces of the

Table 3.2: Interaction Conditions and the Corresponding Validation Criteria

Interaction Condition	Description	Validation Criteria	MHP
(1)	The end-user is perceiving visual information of the Internet service	E1 AND S1	P, C
(2)	The end-user is perceiving auditory information of the Internet service	E2 AND S2	P, C
(3)	The end-user generates mouse input to the Internet service	E3 AND S3	C, M
(4)	The end-user generates keyboard input to the Internet service	E4 AND S4	C, M
(5)	The end-user generates microphone input to the Internet service	E5 AND S5	C, M

P=Perceptual subsystem; C=Cognitive subsystem; M=Motor subsystem.

basic context information from the end-user side and the Internet service side, respectively. Moreover, the rightmost "MHP" column in Table 3.2 demonstrates the related end-user MHP subsystems that have been activated and turned ON. For example, given an end-user's eye-gaze direction towards the networked host screen, i.e., **E1**, and the Internet service displaying in the foreground of the networked host screen, i.e., **S1**, the Interaction Condition **(1)** can be verified: the end-user is perceiving visual information of the Internet service, and his Perceptual and Cognitive subsystems are turned on and working on it.

With the five Interaction Conditions and the defined KCI, the context model can be efficiently built using the first-order rule-based reasoning approach. Since different Internet services require their individual context models, we have chosen three representative Internet services to build the corresponding three context models: *Web Browsing*, *Live Streaming* and *File Transfer*.

- The primary purpose of Web Browsing service is to fetch information on servers and present it to end-users. The Web browser is the corresponding Internet application installed on individual networked host. For example, Mozilla Firefox is such a client side Internet application.

- Live Streaming provides live television or live radio service over the Internet. It always requires a minimum guaranteed bandwidth allocation, because the real-time video/audio programs are sensitive to fluctuations of the received rate. The streaming media player is an Internet application to play back the live multimedia content on the networked host. For example, an IPTV software called QQQTV is such a client side Internet application.

- File Transfer refers to copying a file to or from a remote host over the Internet, and it is also one of the most utilized Internet services. The Internet application implementing the File Transfer Protocol (FTP) can always provide such a service. A client side software called CuteFTP is such an Internet application.

Based on the defined KCI and the given five Interaction Conditions in Table 3.2, three simple but reliable context models for the above-described Internet services are constructed respectively, and then summarized in Table 3.3.

From the built context models in Table 3.3, we see that different combinations of the Interaction Conditions derive the corresponding KCI, where "S" means satisfying the Interaction Condition, "F" denotes failing to satisfy it, "X" means either of the earlier two options, and "n/a" indicates not applicable for that Internet Service. For example, when the Context Model Subsystem has verified that the end-user is perceiving visual and auditory information from a live streaming Internet application, i.e., the *Interaction Conditions (1)* and *(2)* in Table 3.2, then the **Communicating State** between the end-user and that Internet application

Table 3.3: The Context Models for the Three Internet Services

Internet Service	Key Context Information	Interaction Condition				
		(1)	(2)	(3)	(4)	(5)
Web Browsing (Mozilla Firefox)	CS	S	X	S	X	n/a
	IS	F	F	F	F	n/a
Live Streaming (QQQTV)	CS	S	S	X	X	X
	IS	F	F	F	F	X
File Transfer (CuteFTP)	CS	S	n/a	S	X	n/a
	IS	F	n/a	F	F	n/a

CS=Communicating State; IS=Inactive State.

S=Satisfy; F=Fail to satisfy; X=either S or F; n/a=not applicable.

can be directly derived regardless of whether the end-user generates the Mouse, Keyboard and Microphone inputs to that live streaming application, i.e., the *Interaction Conditions (3)*, *(4)* and *(5)* in Table 3.2. Similarly, when the **Inactive State** between the end-user and a live streaming Internet application is deduced, the Context Model Subsystem has to verify that the end-user is not perceiving any visual and auditory information from that application and also not generating any Mouse and Keyboard input to the same application, i.e., failing to satisfy the *Interaction Conditions (1)*, *(2)*, *(3)* and *(4)*, and the *Interaction Conditions (5)* does not need to be considered in this case.

The built context models in Table 3.3 work well for the three representative Internet services in most cases, and more complicated context models with advanced reasoning approaches can be considered for other Internet services and special usage cases. In addition, other possible combinations of Interaction Conditions, which are not described in Table 3.3, simply generate the previously defined *Unidentified State.*

3.3.5 Analysis and Discussion

In order to build reliable and practical context models, the five groups of basic context information in Table 3.1 are collected from the end-user side and the Internet service side, respectively. On the end-user side, we introduce the cognitive framework MHP to model the end-user and systematically describe his interaction behavior. On the basis of the three MHP subsystems, i.e., the Perceptual, the Cognitive and the Motor subsystems, as well as its two principles, we first define two basic end-user states, i.e., the User Perception State and the User Halt State, to specifically describe an Internet end-user. Subsequently, the two defined end-user states, combining with the working status of the corresponding Internet service, are used to further define the KCI. The defined two fundamental categories of the KCI, i.e., the Communicating State (CS) and the Inactive State (IS), serve as the built context model. To deduce the KCI **CS** and the **IS**, the five Interaction Conditions in Table 3.2 are determined to infer ON/OFF states of the three MHP subsystems, and such five Interaction Conditions can be verified by the collected five groups of the basic context information. Those Interaction Conditions with the first-order rule-based reasoning approach eventually establish the context models for the three representative Internet services as shown in Table 3.3. Fig. 3.5 demonstrates the logical structure of the context models, where the built context model is highlighted by a dashed border.

From the User-Context Module perspective, the built context models can effectively deduce the specific KCI for Internet clients. From the Internet design perspective, it is also necessary and significant to build such context models to differentiate between the two basic communication states for Internet clients. As described in Chapter 1, today's Internet follows the traditional design principle that it serves as the communication medium between any two networked hosts

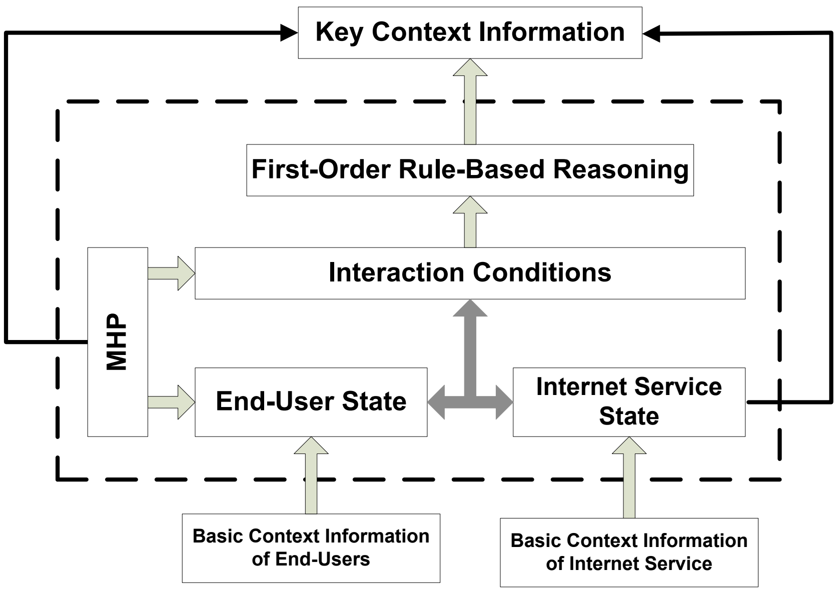

Key Context Information
First-Order Rule-Based Reasoning
Interaction Conditions
MHP
End-User State
Internet Service State
Basic Context Information of End-Users
Basic Context Information of Internet Service

Fig. 3.5: Logical structure of the built context model.

that desire to speak to each other. In other words, the five-layer Internet protocol stack does not explicitly take into account any end-user as well as his context information, and thus essentially conflate dynamic end-user, Internet services and static networked host into one oversimplified concept. For example, Hypertext Transfer Protocol (HTTP) and Transmission Control Protocol (TCP) follow such a traditional principle, and they have been widely used to support various Internet services, including the above-described Web browsing, Live Streaming and File Transfer services. Such a traditional design principle decreases the Internet architecture complexity, but inevitably compromises underlying network performance. Therefore, the built context models together with the User-Context Module architecture essentially provide the Internet an efficient de-conflation solution and take solid steps to separate the Internet end-user from the networked host and running Internet services. More specifically, they empower the Internet to recognize the most important and fundamental interaction states between an end-user and any

Internet service, i.e., the *Communicating State* and the *Inactive State*.

3.4 Control Subsystem

The Control Subsystem is the component for directly interacting with the underlying Internet infrastructure based on the delivered KCI. For different applications of the User-Context Module, the Control Subsystem may interact with different Internet protocols and services in distinct layers, but its main objective is always to improve protocol performance and enhance QoE by dynamically allocating different resources according to the delivered KCI. Note that resources can be of distinct types for different Internet services, and thus the Control Subsystem may work at either or both the Internet client side and the server side.

When interacting with Internet protocols and services, the Control Subsystem does not attempt to modify their internal structures and architectures. In most cases, the Control Subsystem only cautiously chooses proper parameters of protocols or services that are usually accessible and adjustable, and then implement the corresponding ***Control Rules*** to actively tune those parameters. The Control Rules is a set of rules that specify the actions triggered by real-time changes of the delivered KCI. In general, designing a Control Subsystem and the corresponding Control Rules is an application-specific task, but the following three principles should be considered:

- The Control Rules would provide service differentiation according to the delivered KCI, where higher priority is given to the Internet clients in the Communicating State.
- The Control Rules would actively help individual Internet client to enhance its individual utility/payoff or maximize the system-level social welfare.

- The Control Rules would adapt to the dynamics of Internet clients, and would be scalable in terms of gracefully handling rapidly growing Internet clients.

Note that designing a Control Subsystem for the User-Context Module is relatively different from the traditional cross-layer design [55]. The cross-layer design always exploits the dependence between the established protocol layers to obtain performance gains and typically follows some basic structures, such as creating new interfaces or merging of adjacent layers [56], to share and exchange network state information. However, the Control Subsystem under the User-Context Module focuses on actively tuning and managing the accessible parameters in Internet protocols, configurations and services. Hence, in general, the integrity of the conventional Internet layered architecture and protocols can be well maintained, when the Control Subsystem together with the User-Context Module architecture is introduced and implemented.

We present two applications of the User-Context Module in the subsequent two chapters, in which the Control Subsystem mainly works with the upper two layers, and more specifically, HTTP Protocol in the Application Layer and TCP protocol in the Transport Layer. Those two applications would demonstrate the Control Subsystem's operations, practices and impacts.

3.5 Summary

The Context Sensing Subsystem, the Context Model Subsystem and the Control Subsystem compose the core architecture of the proposed User-Context Module. The Context Sensing Subsystem mainly undertakes basic context information gathering task, and closely works with end-users on the Internet client side. The

Context Model Subsystem employs the MHP framework with the first-order rule-based reasoning approach to establish the required context models. The built context models are used to deduce the two categories of the KCI for Internet clients, namely the Communicating State and the Inactive State, for the selected Internet services. The Control Subsystem utilizes the delivered KCI to actively adjust Internet protocols and services according to the specifically designed Control Rules. The User-Context Module architecture with the defined two basic categories of the KCI can be regarded as the first and the crucial step to separate Internet end-user from networked host and Internet service. Moreover, the novel User-Context Module bridges the gaps and establishes a new communication pathway between Internet end-users and the underlying protocol stack. Such modular design not only explicitly and smoothly incorporates end-users and their context information into the Internet, but also provides abundant flexibility for different applications and deployment plans.

Chapter 4

The User-Context Module Application I: HTTP Case

In Chapter 3, we first introduce a novel functional module called the User-Context Module, and then discuss the design and implementation issues of its three main subsystems, especially the Context-Sensing Subsystem and the Context Model Subsystem. As indicated in the same chapter, designing a Control Subsystem with the corresponding Control Rules is an application-specific task. In this chapter, based on the deduced KCI of the Internet client, i.e., the Communicating State and the Inactive State, we present the first application of the User-Context Module. In this application, the Control Subsystem interacts with the Internet Application Layer to improve the protocol performance and the end-user's QoE. More specifically, by adjusting the persistent connection timeout parameter in the Application Layer's Hypertext Transfer Protocol (HTTP), the Control Subsystem effectively reduces the redundant HTTP traffic and the end-user perceived latency in Web browsing. The context model built for the Web browsing service in chapter 3 is adopted to generate the KCI for the Control Subsystem of this application.

4.1 Problem Description

In today's World Wide Web system, HTTP [57] is the de facto communication standard for transferring Web pages. The persistent connection mechanism of HTTP/1.1, also called HTTP keep-alive, allows Web clients to send multiple HTTP requests over the same TCP connection. The persistent connection mechanism reduces network congestion from re-establishing TCP connections and conserves the host's 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 [57] 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 mechanism but suggests picking a proper timeout value for terminating persistent connections. In practical implementations of HTTP/1.1, a fixed timeout value is always imposed. The latest version 2.2.1 of the Apache HTTP Server employs 5 seconds, and the Microsoft IIS uses 120 seconds as their default timeout values. Improperly configuring the timeout value will easily degrade network performance. A small fixed timeout value causes low utilization of HTTP persistent connections, and thus increases the end-user perceived latency as well as the Internet burden. Conversely, a large fixed timeout value would waste and even quickly exhaust the limited Web server resource (e.g. worker threads), which also results in long and unpredictable end-user perceived latency.

There has been limited research work on optimally tuning the persistent connection timeout value of HTTP to improve Web server performance: Faber [58] and Barford [59] indicate that the Web server should close the persistent connections once the client becomes inactive, but no specific approach has been provided. Mogul [60] proposes to give higher priority to the newly established connections, while Sugiki [61] suggests setting higher priority to the small RTT connections and prematurely terminate the ones with large RTT. However, none of these previous studies directly solves the main problem of the HTTP persistent connection mechanism. ***In a Web session, it is difficult for HTTP to discriminate between a persistent connection that is being used by an end-user and a persistent connection that is already in a long-term idle state***. The context model built for the Web browsing service and the deduced KCI, i.e., the Communicating State and the Inactive State, essentially provide HTTP a direct solution to differentiate the above-described two states of HTTP persistent connections. Therefore, adopting the proposed User-Context Module becomes a natural and effective way to address this problem.

Before designing and implementing the Control Subsystem of the User-Context Module for HTTP, it is necessary to first define a Key Context Transfer Protocol (KCTP) to deliver the deduced KCI from the Internet client side to the Web server side.

4.2 Key Context Transfer Protocol

The *Key Context Transfer Protocol* (*KCTP*) is used to deliver the real-time KCI from the Internet client side to the Web server side. The KCTP assumes

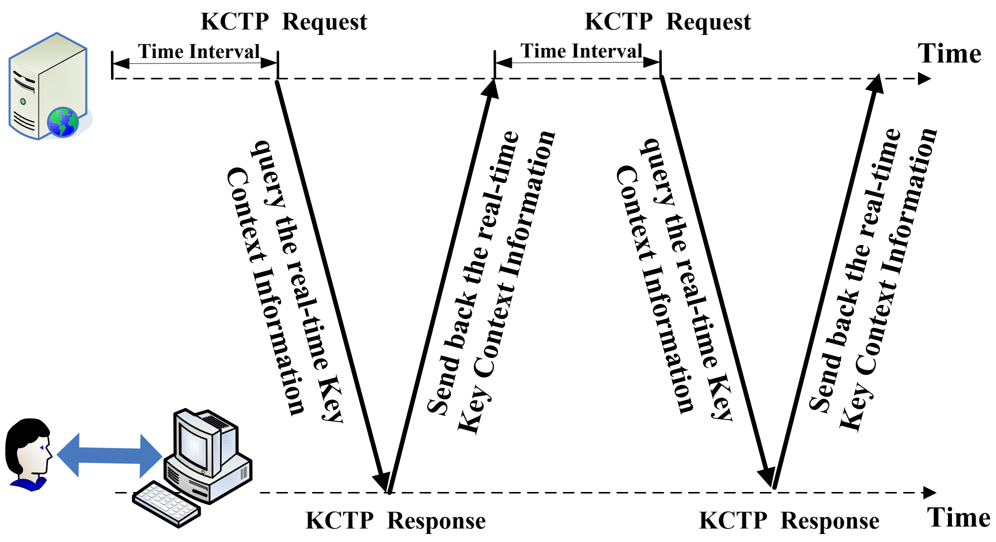

KCTP Request
Time Interval
KCTP Request
Time Interval
Time
query the real-time Key Context Information
Send back the real-time Key Context Information
query the real-time Key Context Information
Send back the real-time Key Context Information
KCTP Response
KCTP Response
Time

Fig. 4.1: Workflow of the Key Context Transfer Protocol.

a reliable transport and in this case using HTTP persistent connection, which is essentially the TCP connection, as its underlying carrier. The KCTP works under the client-server architecture and employs the request-response message exchange pattern. The overall workflow of the KCTP is illustrated in Fig. 4.1:

(1) After the persistent HTTP connection becomes idle, the Web server side KCTP program waits for a time interval and then initiates a request message to the client side via the existing persistent HTTP connection.

(2) Upon receiving the KCTP request, the client side KCTP program retrieves the corresponding real-time KCI from the Context Model Subsystem. The retrieved KCI can be the Communicating State, the Inactive State or any other pre-defined advanced context information of Internet clients.

(3) The client side KCTP program encapsulates that real-time KCI in an KCTP response message, and sends it back to the Web server side.

(4) The Web server side KCTP program receives multiple KCTP response messages from different client sides. Then it delivers all KCIs and relevant information to the Group Context Model Subsystem for further processing.

(5) After some time interval, the server side KCTP program repeats the request via the same persistent HTTP connection, if that connection still exists and remains in the idle state.

Theoretically, the KCTP should adjust the time interval between consecutive requests according to each 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 acquiring the solution of the problem. Through practical user studies and theoretical calculations, MHP shows that the cognitive processing rate has a wide range among different individuals because of their different 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 time and the variance magnitude can be several seconds or even larger. We simplify this cognitive diversity by using 7 seconds, the average decay parameter value of the MHP Cognitive subsystem [14], as the time interval value of the KCTP. In the future work, the Context Sensing Subsystem and the Context Model subsystem could jointly work on capturing the end-user's individual cognitive capacity and predicting his processing time, then the KCTP and the entire system performance can be further improved.

Remark 1: The current version of the KCTP operates only in the simple condition, and a more complicated situation occurs when one or more intermediaries are present between Web clients and Web server, such as when a proxy server is in use. Under such a situation, a new version of the KCTP needs to be specified.

Remark 2: Transferring the private KCI of the Internet clients to the public Web server side may raise security concerns. When necessary, some encryption protocols, such as the Transport Layer Security (TLS), could be adopted to prevent eavesdropping or tampering. Meanwhile, the User-Context Module should also make every effort to process context information locally and avoid any unnecessary transmission over the Internet.

Remark 3: Other interoperable communication mechanisms, such as the XML Protocol (XMLP) [62], can also be employed to undertake the KCI delivery task. The basic workflow is similar to the KCTP, and thus the corresponding modifications may be required.

4.3 The Control Subsystem Design

Once the KCTP response messages successfully transfer multiple clients' KCIs to the Web server side, the group context model will utilize the delivered information to generate the group KCI for the convenience of batch processing by the Web server. In this case, we do not build any group context model but simply move each delivered KCI to the Control Subsystem on the Web server side to continue processing.

With the delivered real-time KCI, the Control Subsystem on the server side adopts the following Control Rules:

(1) IF the Inactive State arrives, THEN the Control Subsystem immediately signals the Application Layer to ***terminate*** the corresponding HTTP persistent connection, i.e., setting the persistent connection timeout parameter to zero.

(2) IF the Communicating State arrives, THEN the Control Subsystem signals the Application Layer to ***maintain*** the corresponding HTTP persistent connection and ***wait*** for the next KCI from the same Web client, i.e., setting the persistent connection timeout parameter to a value larger than the KCTP time interval.

(3) IF the Unidentified State arrives and the Web Server is under the heavy-traffic situation, THEN the Control Subsystem handles it as the Inactive State and immediately terminates the connection. Otherwise, the Control Subsystem treats it as the Communicating State and maintains the connection.

The above Control Rules enable the inflexible HTTP persistent connection mechanism to dynamically adapt to the end-user's real-time browsing behavior, and to influence the underlying network performance. In order to assess the performance gain from both the end-user and the Internet perspectives, we conduct comprehensive experiments accordingly.

4.4 Experimental Setup

4.4.1 Server-side Implementation Issues

We select the Apache HTTP Server in our experiment, as it is a popular open-source Web server. The current Apache HTTP Server 2.2 is configured by writing different Directives in its configuration files, and the HTTP persistent timeout value is set in the main configuration file by the KeepAliveTimeout Directive. The Apache HTTP Server places the KeepAliveTimeout Directive in its main configuration file apache2.conf and sets 5 seconds as its default value. However, any changes to the KeepAliveTimeout Directive can only be recognized by the server when it is started or restarted, because the Apache HTTP Server only reads and processes the main configuration files during its boot-up phase. Furthermore, the KeepAliveTimeout Directive can only simply set the same timeout value for all incoming HTTP requests, so different timeout values cannot be set for different HTTP connections which are initiated by distinct end-users. Since the existing HTTP implementation in Apache Server and KeepAliveTimeout Directive cannot meet our requirements, we therefore disable the KeepAliveTimeout Directive and modify a small part of the Apache source code, where the Apache Server implements the HTTP persistent connection function. The major modifications are made on the file *http_core.c*, which is placed in the Apache source code directory under */modules/http*.

The newly modified Apache HTTP Server can adaptively adjust the timeout value according to the delivered real-time KCI, and it does not require any restart or reboot. Meanwhile, the modified Apache HTTP Server can also set different timeout values for different incoming HTTP requests, which are initiated by distinct Web clients.

We employ the dynamic Web pages as the workload file in our experiment. Thus the PHP code is embedded into the workload HTML files and interpreted by the PHP processor module. We install and configure PHP 5.3.2 module on the modified Apache HTTP Server under the Linux 2.6.28. The average PHP processing time in the experiment is 50 milliseconds, which also takes account of the time to access the database.

4.4.2 Client-side Implementation Issues

In order to emulate multiple end-users' Web browsing scenarios, the experiment requires a specific HTTP request generator to fulfill the following functions:

- The HTTP request generator can emulate the defined Communicating State and the Inactive State: When the Communicating State starts, the generator immediately makes the first HTTP request. After receiving the whole Web page from the server, the generator waits for a certain time interval (end-user processing time) before it sends the next HTTP request. The generator repeats the above procedure until the Communicating State interval ends. Then, during the Inactive State, it simply keeps silent and stops sending HTTP request until the Communicating State resumes.

- To simulate multiple concurrent Internet clients and create heavy-traffic condition, the HTTP request generator should be able to simultaneously open multiple sockets on a single host. Each socket emulates one Internet client and initiates HTTP requests independently.

Existing popular Web workload generators, such as SPECweb2005 [63] and Surge [64] cannot fulfill both of these functions, thus we implement a new HTTP request generator. The new generator is written based on Libwww [65], which is a highly modular and flexible client side Web API for both UNIX and Windows (Win32) platform. We build the new HTTP request generator under the Linux 2.6.28 and all the code is written in C. The new HTTP request generator can open multiple sockets simultaneously, and control the HTTP requests on each socket. The next HTTP request can only be sent after receiving the last HTTP response and waiting for some manually defined time interval. Therefore, the newly built HTTP request generator can emulate multiple Internet clients generating HTTP requests concurrently, while any one of its open sockets simulates single end-user's Web browsing behavior by means of sequentially sending HTTP requests.

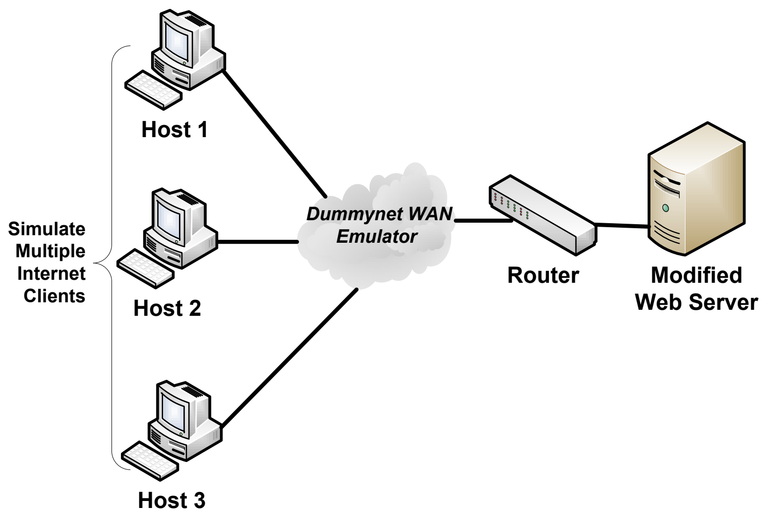


Fig. 4.2: Network topology in the experiment.

4.4.3 Experimental Configuration

Our experimental hardware setup involves several hosts connected to the campus local area network (LAN). Each host is equipped with Duo Intel T7300 2.00-GHz processors, and a 2-GB RAM, and runs Linux 2.6.28. One of the hosts is selected as the Web server and runs modified Apache HTTP Server 2.2.15. Other hosts act as multiple Internet clients and generate HTTP 1.1 requests to the Web server by running the new HTTP request generator. In the experiment, we suppose that the KCI only transits between the ***Communicating State*** and the ***Inactive State***. When in the Communicating State, we assume that each client makes sequential HTTP requests following a homogeneous Poisson process with a rate of 7 requests per minute, and thus the time interval between two consecutive HTTP requests t is exponentially distributed with the mean value of $\frac{60}{7}$ seconds, denoted as $t \sim exp(\frac{60}{7})$. After sending 10 HTTP requests during the Communicating State, each client will automatically transit to the Inactive State. During the Inactive State interval, each client stops generating any HTTP requests to the Web server until the Communicating State resumes. In the experiment, the HTTP request generator periodically alternates between the two states and repeats 3 cycles, which means each client experiences 3 Communicating States and 3 Inactive States. Thus each client sends a total of 30 HTTP requests during the experiment. Since the time interval of the Inactive State varies irregularly, which depends on end-user's personal factors, we simply set this value to 30 seconds in the experiment. To collect more accurate experimental data, we repeated the experiment 3 times with a random starting order of the 3 hosts. Then we average out the observations from all the Web clients.

We consider two experimental conditions: ***light-traffic condition*** and ***heavy-traffic condition***. For the *light-traffic* condition, each host emulates 10 end-users and keeps sending HTTP requests to the server, and thus 3 hosts emulate a total of 30 concurrent end-users. For the *heavy-traffic* condition, all the settings are the same as the light-traffic condition, but each host emulates 100 end-users and thus 300 concurrent end-users in total. Note that the heavy-traffic condition here is different from the server overload situation. The heavy-traffic condition means that the number of concurrent alive clients reaches the maximum number of allowable connections, while the overload situation indicates that the workload persistently exhausts some server resources, such as the server CPU load or the server uplink bandwidth. Since system performance always becomes unstable under the overload situation, we do not consider it in our experiment and we suppose that the Web server employs some admission control schemes, such as in [66, 67] to avoid the overload situation.

In the experiment, we adopt Dummynet [68] to emulate the practical Internet environment. Dummynet is a widely-used tool for enforcing queue and bandwidth limitation, delay and packet loss in network experiments and tests. We enable its delay function and set the configuration parameter to 100 milliseconds for both directions of each link. So the Round Trip Time (RTT) between the client and the Web server is around 250 milliseconds, which consists of the packet-propagation delays and PHP processing time. We also set the packet loss rate to 1%, which is usually caused by the congestion and data corruption along the path of data transmission. The above conditions commonly exist in a Wide Area Network (WAN) as well as the last-hop wireless environments. Since optimizing the Web server's overall performance is out of the scope of this experiment, most of the configuration parameters of the Apache HTTP Server are kept their default settings. For example, the maximum number of HTTP persistent connections that can be processed simultaneously by the Apache HTTP Server is set to 256 in the experiment.

The experiment topology, as depicted in Fig. 4.2, consists of multiple Internet clients and the modified Apache HTTP Server. We also suppose that the real-time KCIs of the Web clients can always be correctly deduced by the implemented Context Model Subsystem.

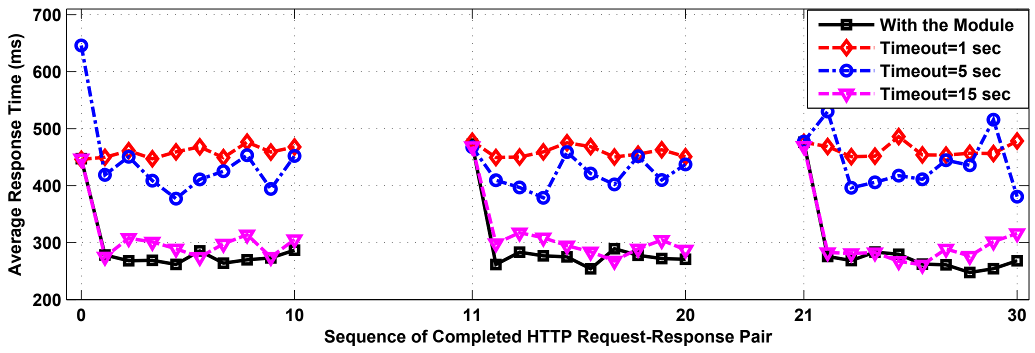
4.5 Internet Experiment Results

Based on the above-described experimental setup, we study the network performance and contrast the results with the case without the User-Context Module. In this User-Context Module application, the Web page response time can be an appropriate performance metric for the end-user perceived latency. The Web page response time is the time interval that starts when the end-user sends the Web request and ends when the end-user receives the last object of that Web page. Besides the Web page response time, we also study the Web traffic statistics and the server aggregate throughput to fully assess the influence of adding the new User-Context Module. The collected Web traffic statistics in this experiment includes the number of the successfully delivered Web pages, and the number of HTTP requests transferred between the server and the clients. The server aggregate throughput is the sum of the Web server generated data rates that are successfully delivered to all the clients.

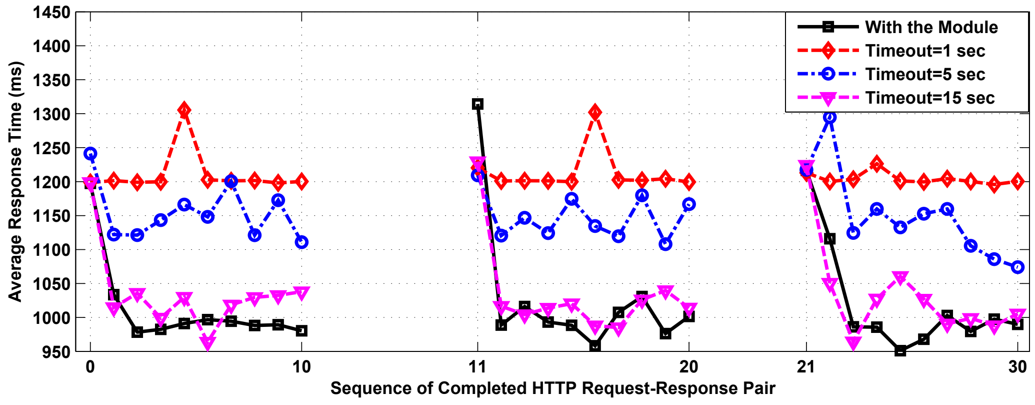
4.5.1 Light-Traffic Condition

We experiment with two groups of Web pages, which are similar to the SPECweb benchmark [63]. In the first group, the mean size of the generated dynamic Web pages is smaller than 5KB, and in the second group the mean size is larger than 50KB. Fig. 4.3(a) shows the average Web page response time of the small size group, while Fig. 4.3(b) shows that of the large size group, and Fig. 4.3(c) describes the KCI transitions during the experiment.

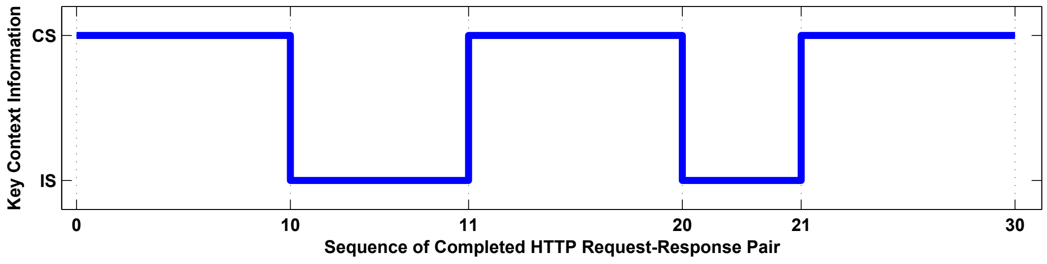
Figs. 4.3(a) and 4.3(b) show that the User-Context Module can significantly shorten the average Web page response time of both groups. In contrast to the results where the fixed timeout value 5 seconds (the default value of the latest Apache HTTP Server) and 1 second are used, the User-Context Module case can save almost 200 milliseconds on average. It roughly equals to the back-and-forth time on the wire, namely one round trip packet-propagation delay. This is because the User-Context Module can actively extend the lifecycle of the HTTP connections when the Web client stays in the Communicating State, and thus avoid multiple unnecessary re-establishments of new HTTP connections. Meanwhile, we see that under the light-traffic condition, the 15 seconds timeout case can also achieve quite short average response time. It is because during the Communicating State, the time interval between the consecutive HTTP requests is always smaller



(a) Small Web page size group.



(b) Large Web page size group.



(c) Key Context Information transition.

Fig. 4.3: Average Web page response time under the light-traffic condition.

than 15 seconds, and thus HTTP persistent connections will not be terminated by the Web server as frequently as the small timeout value cases.

Fig. 4.4 shows the ratio of the total number of HTTP requests sent by the clients to the total number of the successfully transferred Web pages. For both

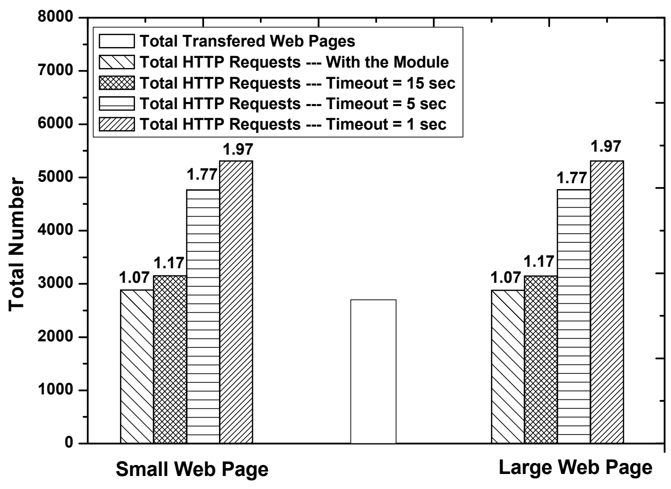


Fig. 4.4: Ratios of HTTP request number to transferred Web page number.

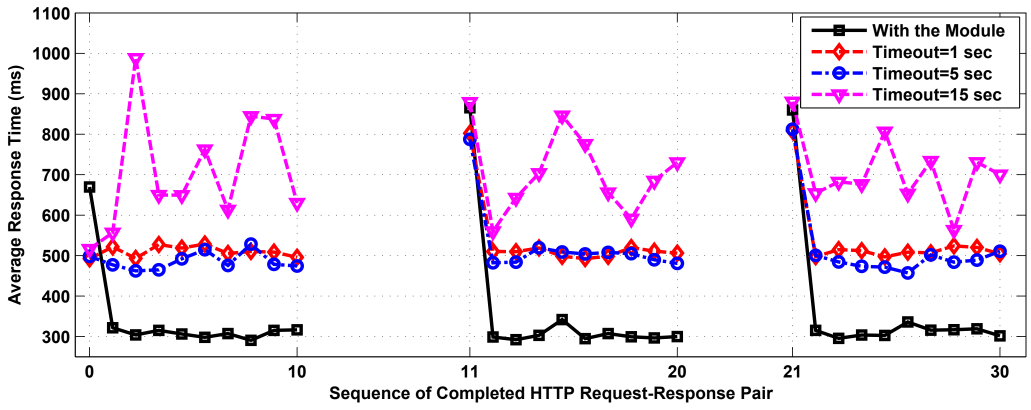
the small and the large page size groups, the User-Context Module case achieves the smallest ratio, i.e., 1.07. The 1 second fixed timeout case results in the highest ratio, i.e., 1.97, which indicates that almost two HTTP requests are required to fetch one Web page. This is because once the small timeout occurs, the Web server will send a TCP segment with the FIN bit set to 1 and enter the FIN_WAIT_1 state. While the Web browser may continue sending new but already invalid HTTP requests through the same connection before it sends the clients' side TCP segment with the FIN bit. Besides the transmission of invalid HTTP requests, the unnecessary re-establishment and closing HTTP connections also significantly increase the burden on both the Web server and the Internet backbone. Note that in practice most commercial Web browsers usually open multiple concurrent HTTP connections for fetching one Web page and different browsers adopt different mechanisms to reduce the unnecessary retransmission. So the absolute value of the ratio may vary case by case, but its relative trend will be the same as shown in Fig. 4.4.

4.5.2 Heavy-Traffic Condition

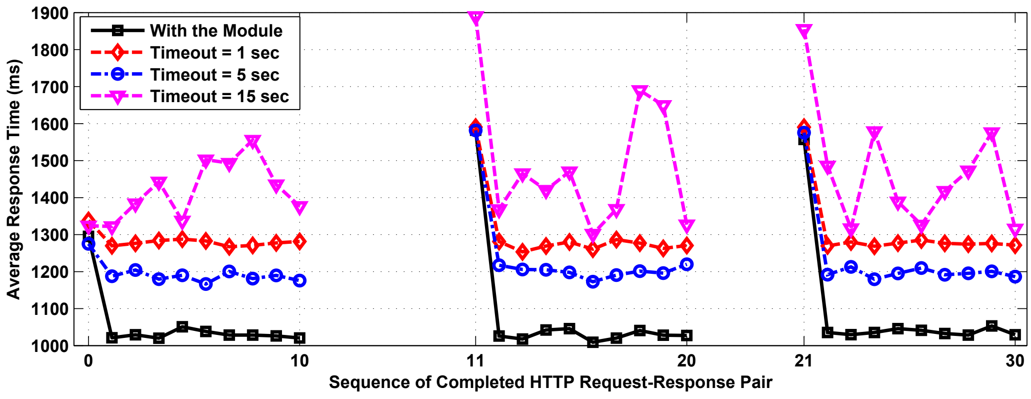
Figs. 4.5(a) and 4.5(b) also show significant reduction in the average Web page response time under the heavy-traffic condition. Similar to the results under the light-traffic condition, the User-Context Module reduces the Web page response time by almost the equivalent of one round trip packet-propagation delay when compared to the 1 second and 5 seconds timeout cases.

Note that under the heavy-traffic condition, the User-Context Module achieves much shorter average Web page response time than the 15 seconds timeout case. This is because with the fast increasing number of Web clients, the new HTTP persistent connections also dramatically increase. With a large fixed timeout value, the Web server cannot terminate inactive connections and allocate the limited server resource to the newly incoming clients in a timely manner. This causes the number of concurrent HTTP persistent connections to easily reach the upper limit of the Apache HTTP Server, which is set to 256 in this experiment. When this happens, new incoming HTTP connections must wait either in the SYN-queue or the ACK-queue of the Apache Web server. Such queuing delay at the server side can easily amount to several seconds and thus greatly influences the Web page response time. From the collected data of the 15 seconds timeout case, we see that the Web page response time of the late arriving clients varies from hundreds to thousands of milliseconds, although the early arriving clients can still attain quite small response time. Consequently, the high and unstable queuing delay experienced by the late arriving clients lead to the large average Web page response time and signification fluctuations in the 15 seconds timeout case, which are evident in Figs. 4.5(a) and 4.5(b).

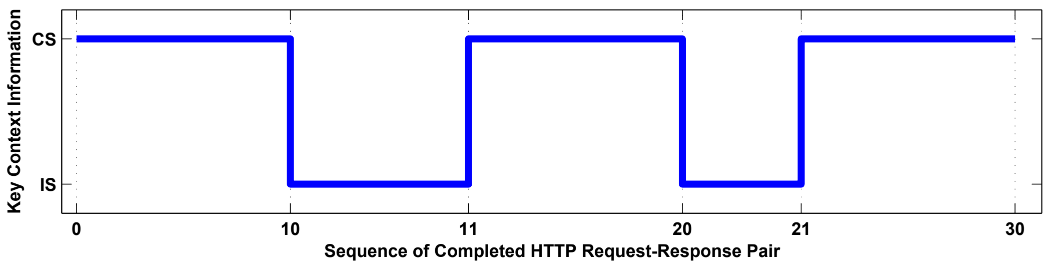
Prior studies [69, 70] have shown that the Web page response time greatly directly influences the end-user's QoE in Web browsing. With the same QoE



(a) Small Web page size group.



(b) Large Web page size group.



(c) Key Context Information transition.

Fig. 4.5: Average Web page response time under the heavy-traffic condition.

rating measurement (called Opinion Scores), the quantitative relationship between the end-user's QoE and the Web page response time has been investigated: ITU-T G.1030 [70] demonstrates that the logarithmic relationship fits well, while Shaikh et al. [69] shows that the exponential relationship gives the best correlation

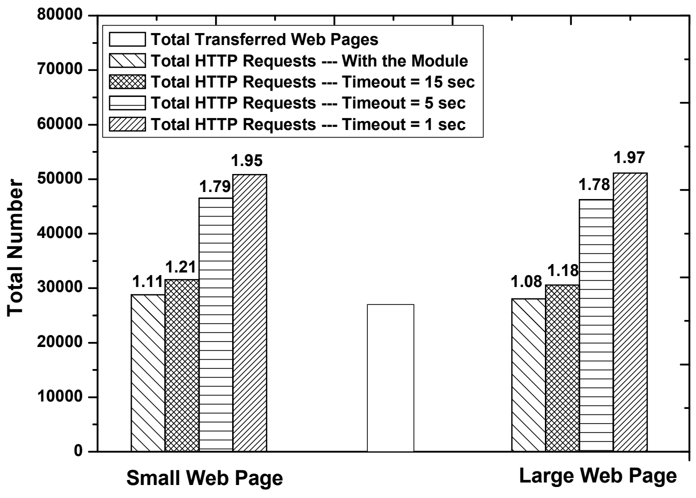


Fig. 4.6: Ratios of HTTP request number to transferred Web page number.

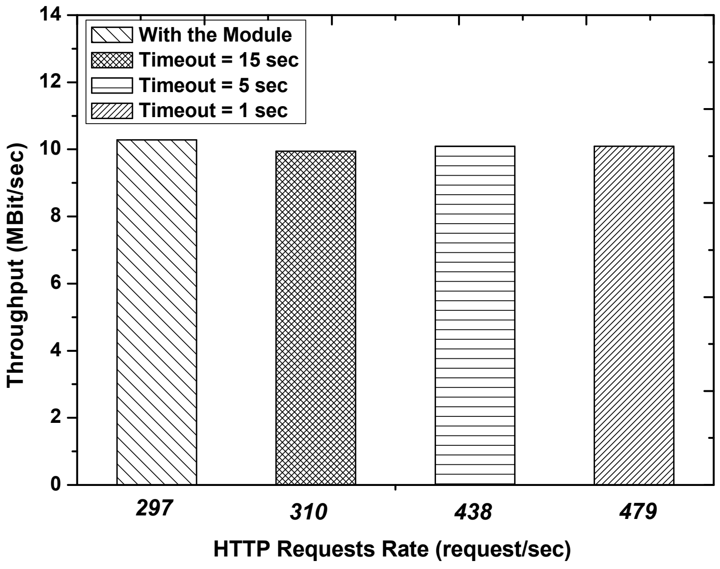


Fig. 4.7: Throughput of Web server under heavy-traffic condition.

result. The two relationships are compared in [41], where a generic exponential relationship between the end-user's QoE and the QoS parameters has been suggested. Those results illustrate that the mathematical relationship between the end-user's QoE and the Web page response time may vary due to the diversity of the participants in user studies and the network configurations. However, all the derived models verify that the Opinion Scores monotonously increases with the decreasing Web page response time. Therefore, it confirms that this User Module application enhances the end-user's QoE through reducing the Web page response time.

Under the heavy-traffic condition, the average throughput of the Web server hosting the large Web page group is shown in Fig. 4.7. We see that introducing the User-Context Module cannot increase the server throughput, which indicates that the aggregate throughput does not greatly depend on the timeout value. However, Fig. 4.7 also shows that with the equivalent aggregate throughput, the User-Context Module case achieves the smallest HTTP request rates, and thus effectively reduces the burden on the server and the Internet backbone. Fig. 4.6 also show the same performance gain in terms of the ratio of the HTTP requests number to the successfully transferred Web page number under the heavy-traffic condition.

From the experimental results under both the light-traffic condition and the heavy traffic condition, we see that the current HTTP persistent connection mechanism, with either small or large fixed timeout values, is unable to achieve optimal operation in terms of reducing end-user perceived latency and cutting down the Internet traffic burden. It is because no Internet client's context information, or more specifically end-user's real-time browsing information, is available for the HTTP protocol.

The Internet experimental results have proven that the proposed User-Context Module is an effective solution, which successfully bridges the gap between the HTTP persistent connection mechanism and Web clients.

4.5.3 Discussions on Delayed and Loss of KCIs

In the above-described Control Subsystem designed for HTTP protocol, the real-time KCIs are required to be delivered from the client side to the Web server side for controlling the persistent HTTP connections. We therefore define the KCTP protocol and utilize it to accomplish the KCI delivery task. Because the implemented KCTP essentially uses the TCP connection as its default underlying carrier, the TCP's retransmission mechanism would provide the reliable data delivery service and thus the KCTP can prevent the loss of KCIs during the transmission process.

On the other hand, the implemented KCTP cannot ensure the timely delivery of real-time KCIs due to network congestion at the Internet backbone. Under such circumstances, there are two possible cases:

- Delayed Communicating State: According to the designed Control Rules, the Control Subsystem would maintain the HTTP persistent connection until the next corresponding KCI arrives. Hence, the delayed CS packet would only result that the server side extends the life span of the current HTTP connection. Because the Web client can still send or receive the data with the same HTTP connection, the delayed CS will not impair the QoE of the corresponding end-user at the client side.

- Delayed Inactive State: According to the designed Control Rules, the Control Subsystem would terminate the HTTP persistent connection when the corresponding IS arrives. The delayed IS information would only result that the Web server continues to maintain the current idle HTTP connection until the delayed IS is delivered. Hence, it will not make any impact on the QoE of the corresponding end-user, because the IS essentially indicates no interaction between the end-user and the Web service at the client side. Moreover, since a Web system normally serves at least hundreds of concurrent clients, single or a few KCIs' delay would have little influence on the performance of the User-Context Module.

In short, a few delayed or loss of KCIs would not impair the QoE of the corresponding end-user and also not deteriorate the overall system performance, given that the Web system always serves a large number of clients.

4.6 Summary

In this chapter, we present the first application of the User-Context Module, which mainly introduces the KCI of Internet clients into the Application Layer's HTTP protocol. The specifically designed Control Subsystem adaptively adjusts the HTTP persistent connection timeout parameter according to the defined Control Rules. Internet experimental results confirm that such design significantly reduces the unnecessary Internet traffic burden and enhances the end-user's QoE.

Meanwhile, implementing such a User-Context Module requires computational burden for deducing and delivering the real-time KCI at the ends of Internet. With the rapid advancement of sensing techniques and computational capability of network hosts, the system overhead would be further reduced and minimized.

This application sets a sample solution for a group of Internet communication protocols to provide service differentiation in the corresponding Internet service according to the deduced end-user usage status. For example, the User-Context Module can also be applied to adjust the session-related variables in Real Time Streaming Protocol (RTSP) the streaming media Internet services. In short, the first application demonstrates the User-Context Module's operations and impacts as well as its Control Subsystem's design and implementation.

Chapter 5

The User-Context Module Application II: TCP Case

In Chapter 4, we presented the first application of the User-Context Module, which introduces the KCI of Internet clients into the Application Layer's HTTP protocol. Such a User-Context Module application is motivated by improving persistent connection mechanism of the HTTP protocol. In this chapter, we present the second application of the User-Context Module, which is mainly motivated by enhancing the end-user's QoE. In this application, the Control Subsystem interacts with the Transport Layer's TCP protocol to dynamically allocate the resource at the bandwidth-limited access link. More specifically, the Control Subsystem manipulates the advertised window size in the TCP protocol based on the captured KCIs of the Internet clients.

5.1 Problem Description

As a connection-oriented and reliable Transport Layer communication protocol, Transmission Control Protocol (TCP) [71] is an indispensable component of the modern Internet protocol stack. Many Internet services and their applications rely on TCP as their transport carrier, such as File Transfer (CuteFTP), Web Browsing (Firefox), Electronic Mailing (Microsoft Outlook), and some Live Streaming services (QQQTV).

Similar to other Internet communication protocols, TCP is also designed with the traditional Internet principle that assumes two network hosts always desire to speak to each other. Hence, an individual TCP stream always intends to maximize its own throughput unless network congestion or receiver buffer overflow happens. It has been proven that when multiple TCP streams compete for the same ***bottleneck link***, the stream with a smaller RTT can always grab a much larger share of that bottleneck link bandwidth than other streams with larger RTT [72]. Therefore, the TCP protocol always favors an Internet application with short RTT regardless of the end-user preference and other influential factors. Such a TCP property can easily impair the QoE of Internet end-users, especially when an end-user wants to prioritize an Internet application having larger RTT. For instance, an end-user may simultaneously open CuteFTP to download a large file and QQQTV to watch online TV. As a live multimedia streaming application, QQQTV always requires a minimum guaranteed bandwidth, but some CuteFTP connections with small RTT can easily grab most of the available bandwidth at the access link, where the last mile bottleneck exists. Hence, with the aim of enhancing the end-user's QoE, the User-Context Module would be a natural and effective solution to enable TCP to provide an appropriate bandwidth prioritization service. Accordingly, the built context models for the Live Streaming service and the File

Transfer service can be used to deduce the indispensable KCI.

Since the main objective of this User-Context Module application is to enhance the end-user's QoE, it is necessary to discuss its assessment approach first.

5.2 Assessment of QoE

As mentioned in chapter 2, assessing and measuring QoE require not only the traditional Quality of Service (QoS) parameters [41], but a joint consequence of the communication context environment, the characteristics of the service in use and the underlying network performance. Brooks et al. [42] propose a structured assessment approach to describe QoE with the following clause:

IF <*Communication Situation*>;

USING <*Service Prescription*>;

WITH <*Technical Parameters*>;

THEN <*end-user's QoE*>.

The <*Communication Situation*> takes into account objective communication context related to end-users. Therefore, the Communicating State, the Inactive State and other properly defined KCI of the Internet client can be introduced into the parameter set of the <*Communication Situation*> attribute. The <*Service Prescription*> can be Live Streaming (QQQTV), File Transfer (CuteFTP) or any other types of Internet services. The <*Technical Parameters*> ranges from the bit rate to the protocol type [42]. For the <*end-user's QoE*>, the Opinion Score scale from 5 to 1 can be used to describe the end-user's subjective satisfaction on the performance of a given Internet service. Based on such a

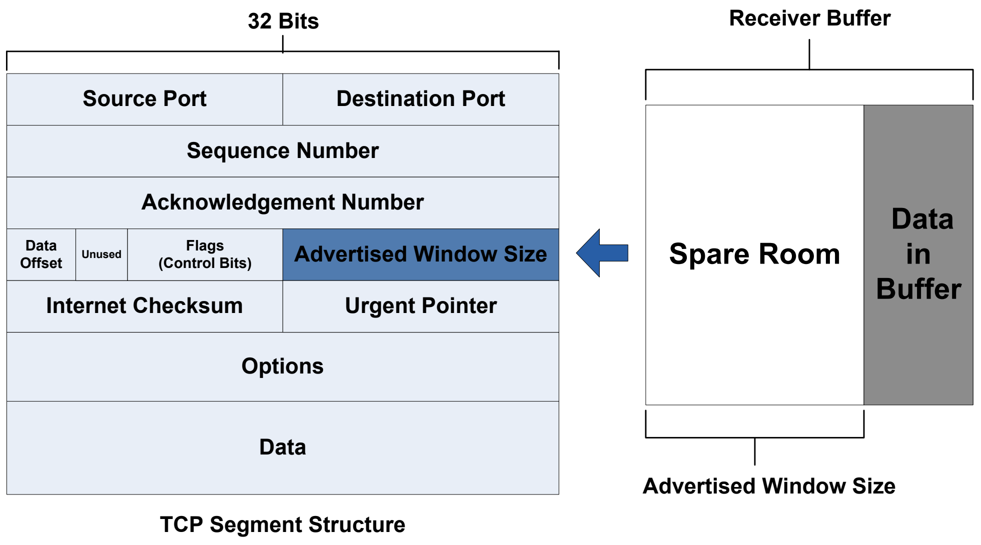


Fig. 5.1: Advertised window size determined by the spare room of the receiver buffer.

structured assessment approach, we design and implement the Control Subsystem for the TCP protocol to enhance the end-user's QoE.

5.3 The Control Subsystem Design

In order to enable TCP to provide a bandwidth prioritization service at the bandwidth-limited access link, the Control Subsystem of the User-Context Module needs to leverage on the flow control mechanism of TCP. The original objective of the TCP flow control mechanism is to obviate the TCP sender overflowing the TCP receiver's local buffer. Different from the well-known TCP congestion control mechanism, the TCP flow control mechanism maintains a variable called *advertised window* at the TCP receiver side. As shown in Fig. 5.1, the advertised window size is always set to the amount of spare room in the buffer, and is included in each TCP acknowledgement returned to the TCP sender. Thus, the advertised window can actually limit the maximum number of bytes a sender is allowed to transmit before receiving the next acknowledgment from the receiver side.

There has been some prior work on adjusting the advertised window size in [73, 74]. Most of them follow a similar mathematical model describing the relationship between the sending rate V of a TCP connection and the corresponding advertised window size:

$$V = \frac{W^{rec}}{RTT}, \tag{5.1}$$

where W^{rec} is the advertised window size in bits and RTT is the average round trip time of the TCP connection in seconds. The above mathematical model sufficiently describes the TCP flow control mechanism, although it simplifies other TCP mechanisms such as the congestion control dynamics [75]. The above model is based on the following assumptions:

- The access link or the last hop link is the bottleneck link of the entire networks, which commonly exists in wired and wireless networks.
- The packet loss probability is small and thus we neglect the effect of the TCP slow start and the time-out mechanism.
- Only long-term bulk-transfer Internet applications and services are considered, since the short-term small sessions are likely to have completed before they can influence the end-user's QoE.

From the TCP model given in (5.1), we see that adjusting the advertised window size at the receiver side can directly influence the TCP sending rate. Therefore, the Control Subsystem can redistribute the limited bandwidth resource at the access link by manipulating the TCP advertised window size on the Internet client side. Furthermore, the TCP model (5.1) also shows that the sending rate

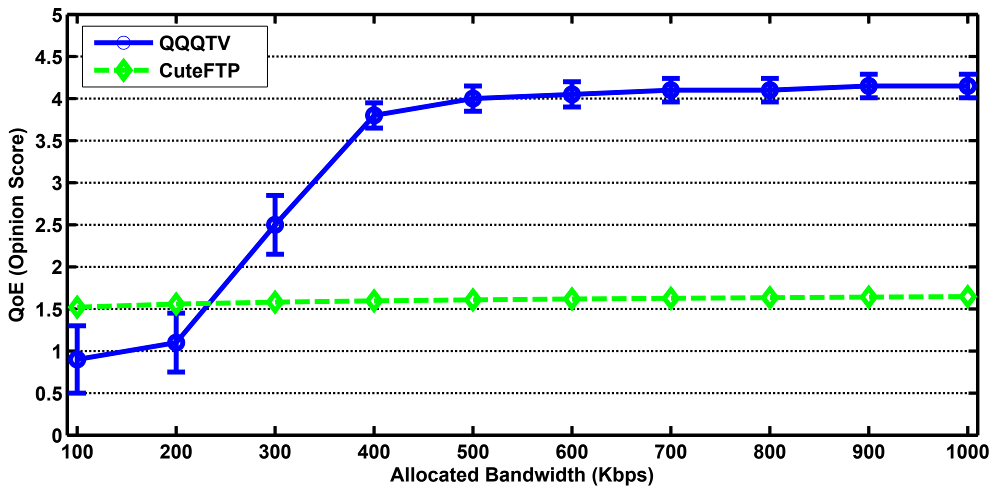


Fig. 5.2: The end-user's QoE on QQQTV and CuteFTP as a function of the allocated bandwidth

V is inversely proportional to the average RTT, and the Control Subsystem can either employ the TCP timestamp option [76] or the method proposed in [77] to calculate the average RTT. Note that implementing such a Control Subsystem at the TCP receiver side for adjusting the advertised window size does not require any changes to the existing TCP protocol.

We set up our experiment with the two Internet applications *QQQTV* and *CuteFTP*: the end-user opens QQQTV for watching the online live TV program (ESPN channel), and meanwhile uses CuteFTP to download a zip file (200 MB). Such usage scenario commonly exists in practice. To enable the access link to be the bottleneck link in the experiment, we employ NetLimiter [78] on the Internet side to limit the overall incoming throughput to 1.0 Mbps. Since the main objective of this User-Context Module application is to enhance the end-user's QoE, we conduct the specific user study to investigate the relationship between the QoE and the bandwidth consumption of QQQTV. The participants are asked to provide their subjective responses about the QQQTV performance on the Opinion Score scale from 5 to 1. The following grades are used: 5 = Excellent, 4 = Good, 3 =

Average, 2 = Poor, 1 = Bad. In total, 4 female and 5 male test users attended the study with an age distribution between 20 and 55 years (mean 32.3 years, median 29.4 years). All participants are rather advanced in terms of watching live streaming multimedia and Internet usage. The solid line in Fig. 5.2 depicts the end-user's QoE on QQQTV as a function of the allocated bandwidth (from 100 Kbps to 1000 Kbps, error bars representing 95% confidence intervals). The outcome of the user study demonstrates that the Opinion Score is Good when the allocated bandwidth is above 400 Kbps, and the Opinion Score drops to almost Poor when the allocated bandwidth is below 320 Kbps, i.e., QQQTV's performance deteriorates to an unacceptable level for the Internet end-users. For the end-user's QoE on CuteFTP, Reichl et al. [79] modeled the QoE on file downloading as a function of the normalized file size f_s and the normalized download bandwidth V_d as follows:

$$QoE = \frac{0.775}{\sqrt{f_s}} \ln(V_d) + 1.268$$

We adopt the above model to describe the relationship between the end-user's QoE and the bandwidth consumption of CuteFTP. Given the fixed file size of 200 MB, the dash line in Fig. 5.2 depicts the variance of the end-user's QoE on CuteFTP with the same range of the allocated bandwidth from 100 Kbps to 1000 Kbps. From Fig. 5.2, we see that allocating enough bandwidth to QQQTV greatly improves the corresponding end-user's QoE, while decreasing the bandwidth for downloading large file has limited impairment to the end-user's QoE on CuteFTP. According to the structured assessment approach to the QoE, Fig. 5.2 can be further expressed as follows:

IF <*Communicating State*>;

USING <*QQQTV*> or USING <*CuteFTP*>;

WITH <*Bit Rate (from 100 Kbps to 1000 Kbps)*>;

THEN <*Opinion Score Results in Fig. 5.2*>.

Note that we take the *Bit Rate* as the main parameter of the attribute <*Technical Parameters*>, because it is the key factor to influence the performance of both QQQTV and CuteFTP. Other technical parameters can be further considered in more complicated usage scenarios.

Based on the end-user's QoE models on QQQTV and CuteFTP as well as the above analysis, we see that QQQTV should be given priority to receive enough bandwidth resource. Hence, under the condition that the two Internet applications are running simultaneously, the following Control Rules can be implemented on the Internet client side:

(1) IF the Communicating State (or the Unidentified State) between the end-user and QQQTV is deduced, and meanwhile the QQQTV bandwidth share is ***lower*** than 320 Kbps, THEN the Control Subsystem immediately ***reduces*** the advertised window size of CuteFTP until QQQTV bandwidth share exceeds 400 Kbps.

(2) IF the Inactive State between the end-user and QQQTV is deduced or QQQTV is terminated, THEN the Control Subsystem ***increases*** the advertised window size of CuteFTP to the initial value.

(3) For other possible situations, the Control Subsystem takes no action.

The above Control Rules ensure that when the end-user is watching QQQTV, QQQTV is always given priority to receive enough bandwidth. Meanwhile, CuteFTP can also keep working with the leftover network resource rather than be forcibly paused or closed. When the end-user stops watching QQQTV, the Control Subsystem will take back the privilege given to QQQTV and allow the running CuteFTP to fairly compete for the limited bandwidth resource again.

Since the second application of the User-Context Module mainly aims to demonstrate our design and motivations, the usage scenario and experimental setup are relatively simple. With more complicated conditions, new Control Subsystem and Control Rules should be re-designed. For example, the Control Subsystem at the server side can also be launched to actively allocate the limited network resource according to the deduced KCI, when a bottleneck exists at the uplink of the server. In addition, some data mining and machine learning algorithms can also be introduced to automatically generate the Control Rules based on the basic context information of Internet clients [80].

To achieve fast response and avoid overshoot, we adopt the widely used discrete PD (Proportional and Derivative) control algorithm to adjust the advertised window size:

$$\Delta W_j^{rec} = K_p e(j) + K_d(e(j) - e(j-1)),$$

where

$$e(j) = R - V_j.$$

The controller output ΔW_j^{rec} is used to adjust the advertised window size at the j^{th} sampling time. R is the target bandwidth and V_j is the allocated bandwidth at the j^{th} sampling time. K_p and K_d are the tuning parameters in the PD control algorithm. The sampling interval can be about five RTT of the adjusted TCP connection.

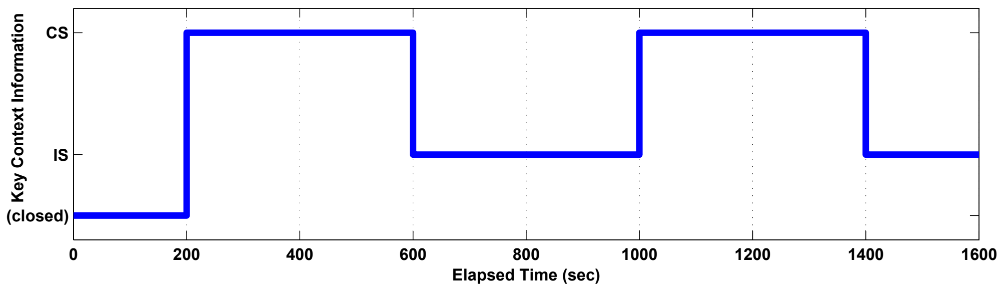


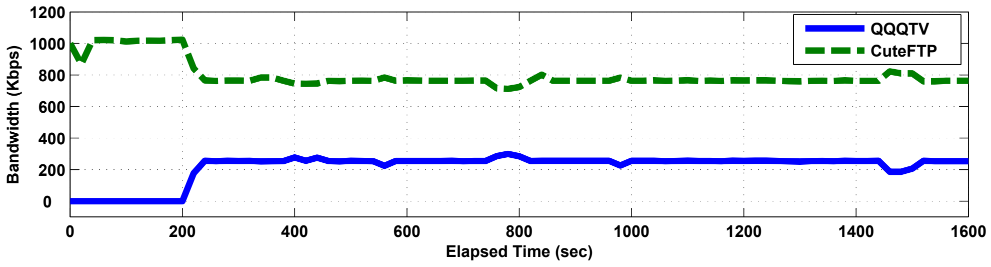
Fig. 5.3: Key Context Information transition on QQQTV.

5.4 Experimental Results and QoE Enhancement

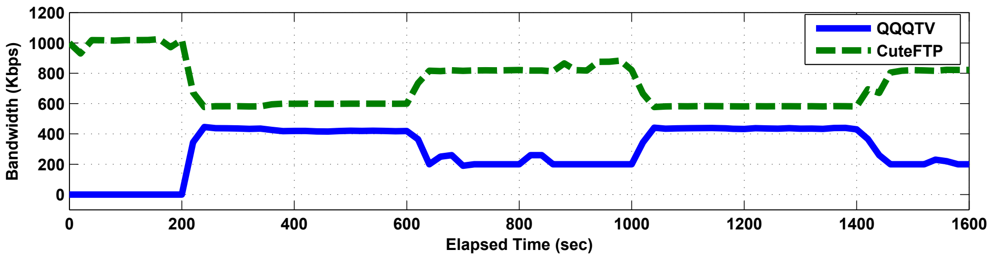
To illustrate the feasibility and effectiveness of the above-described solution, we implement the defined Control Rules and the function to measure the real-time bandwidth consumption for each TCP connection on the Internet client side. The experiment lasts 1600 seconds, and CuteFTP starts its downloading task from the beginning to the end. Meanwhile, the KCI transitions on QQQTV are shown in Fig. 5.3.

Fig. 5.4(a) shows the bandwidth distribution of the bottleneck link (1Mbps in total) without the User-Context Module, when QQQTV and CuteFTP run simultaneously. We see that CuteFTP always captures most of the limited bandwidth resource, i.e., nearly 800Kbps, because it has relatively smaller average RTT and enough advertised window size in each TCP acknowledgement. Therefore, QQQTV always cannot receive the minimum guaranteed bandwidth, which greatly impairs the corresponding end-user's QoE. In practice, the end-user has to manually shut down or pause CuteFTP to facilitate normal watching QQQTV.

Fig. 5.4(b) illustrates the bandwidth distribution of the bottleneck link when the User-Context Module is implemented. We see that when the end-user starts watching QQQTV, i.e., the Communicating State between the end-user and QQQTV,



(a) Bandwidth allocation without the User-Context Module



(b) Bandwidth allocation with the User-Context Module

Fig. 5.4: A comparison between the system without and with the User-Context Module.

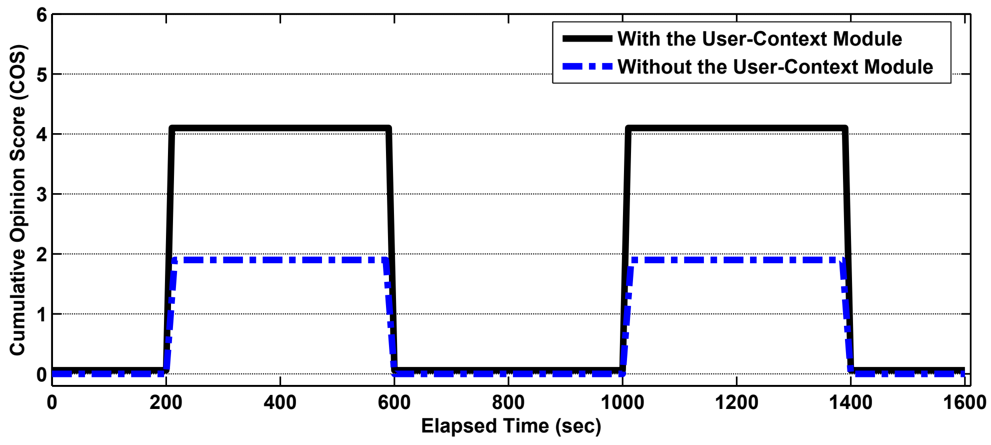
the Control Subsystem can automatically decrease the bandwidth consumption of CuteFTP until QQQTV bandwidth share exceeds 400 Kbps. When the end-user temporarily stops watching QQQTV, i.e., the Inactive State between the end-user and QQQTV, the Control Subsystem then releases the constraints on the advertised window size of CuteFTP, and takes back the privilege given to QQQTV. During the entire process, the Control Subsystem guarantees that the full bottleneck link capacity is utilized by the two running Internet applications and no resource wastage.

On the basis of the above experimental results, we further investigate its influence on the end-user's QoE. As indicated earlier, the end-user's QoE is a joint consequence of the technical parameters (traditional QoS parameters), the communication context environment and the characteristics of the network services in use. Fig. 5.2 has depicted the end-user's QoE on QQQTV and CuteFTP when the parameter of the <*Communication Situation*> is the Communicating State. When the parameter of the <*Communication Situation*> is the Inactive State, the Opinion Score in the <*end-user's QoE*> can be simply assigned "0" regardless of the types of Internet services and the variances of technical parameters. The main reason is that the Inactive State indicates no interaction between the end-user's all three MHP subsystems and the corresponding Internet service, and thus it results in little influence on the end-user's subjective satisfaction. Given the KCI transition on QQQTV depicted in Fig. 5.3, we further consider the two specific scenarios with regard to the KCI transition on CuteFTP:

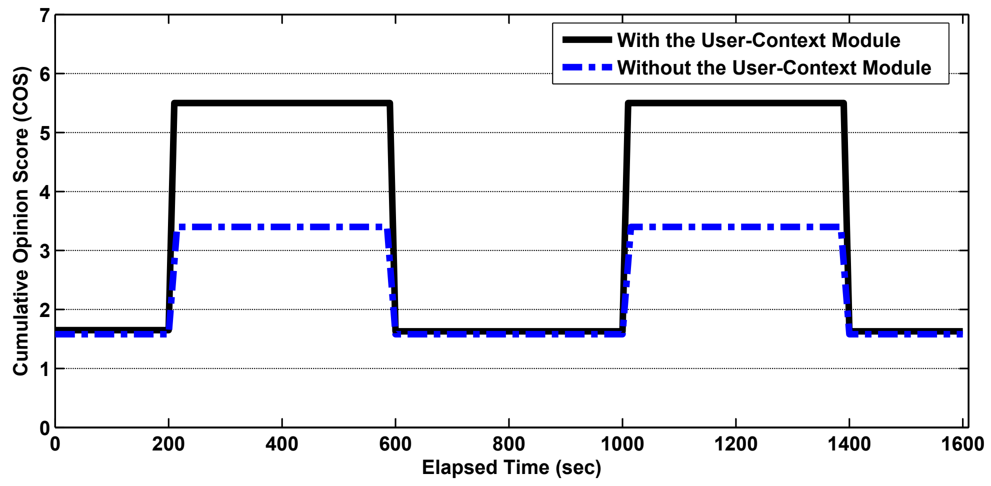
Scenario A: CuteFTP is always associated with the Inactive State, which means the end-user is unaware of CuteFTP downloading from the beginning to the end.

Scenario B: CuteFTP is always associated with the Communicating State, which means the end-user is aware of CuteFTP downloading even when the end-user is watching QQQTV.

Based on the bandwidth distribution results in Figs. 5.4(a) and 5.4(b), we can calculate the Cumulative Opinion Score (COS), i.e., the sum of the Opinion Score on QQQTV and CuteFTP, for the above-described two usage scenarios, respectively. Fig. 5.5(a) illustrates the variances of the COS under the Scenario A. Note that under the scenario A, the Opinion Score on CuteFTP stays at zero, because the parameter of the <*Communication Situation*> for CuteFTP is always the Inactive State. Hence, Fig. 5.5(a) essentially depicts the end-user Opinion Score on QQQTV. Therefore, we clearly see that introducing the User-Context Module effectively prevents the QoE on QQQTV falling down to the Poor level and maintains it at a high satisfaction level during the period when the end-user



(a) COS under the Scenario A



(b) COS under the Scenario B

Fig. 5.5: Cumulative Opinion Score (COS) under the two extreme scenarios

is watching QQQTV. Fig. 5.5(b) demonstrates the COS under Scenario B. We see that the User-Context Module can still dramatically increase the COS when QQQTV is in the Communicating State, although the end-user Opinion Score on CuteFTP would be slightly decreased because less bandwidth is allocated to it. Such a result indicates that when the User-Context Module lowers down CuteFTP bandwidth share, it does not influence much on the overall QoE of the Internet end-user even when he is constantly aware of the CuteFTP downloading task.

The given scenarios are the two extreme cases depicting the KCI between the end-user and CuteFTP. In reality, the situation is more likely to frequently switch between the two states, and even turns to the Unidentified States, caused by human complex internal conscious and unconscious psychological and cognitive factors [4]. However, with the derived positive COS results in both extreme cases as shown in Figs. 5.5(a) and 5.5(b), we can tentatively conclude that the proposed User-Context Module with the defined Control Rules effectively enhances the end-user's QoE in practice. Therefore, the User-Context Module application fulfills the initial design objective. Noted that the designed Control Subsystem can only address the access link bottleneck problem, and the main system overhead is still the real-time KCI deduction at the client side. In addition, different from the first HTTP case, the Control Subsystem designed for the TCP protocol directly manipulates the advertised window size at the client side, and thus the KCIs are not required to be transferred to the server side. Accordingly, there would be no loss or delayed KCIs to influence the QoE and system performance of the User-Context Module.

5.5 Summary

In this chapter, we present the second application of the User-Context Module, which mainly introduce the KCI of Internet clients into the Transport Layer's TCP protocol. With the aim of enhancing the end-user's QoE, the Control Subsystem manipulates the advertised window size in TCP to actively re-distribute the access link bandwidth for prioritizing the specific Internet service. The experimental results with the corresponding QoE assessment approach confirm that the second User-Context application effectively enhances the Internet end-user's subjective satisfaction. Furthermore, such an application demonstrates a systematic way to design the Control Rules for enhancing the end-user's QoE and construct the QoE model through the user study. Accordingly, a wide range of the similar User-Context Module applications can be developed, and then evaluated by the given structured QoE assessment approach.

Chapter 6

A Resource Distribution Framework Incentivizing Context Sharing and Moderate Competition

In the previous chapters, we have introduced the core architecture of the User-Context Module, which consists of the Context Sensing Subsystem, the Context Model Subsystem and the Control Subsystem. In the Context Model Subsystem, the two fundamental categories of the KCI have been defined and subsequently deduced by the context models built for different Internet services. Based on the context models built for Web browsing, Live Streaming and File Transfer, we present the two applications of the User-Context Module to improve Internet performance from different aspects. Both applications essentially make use of the captured KCI to manage different types of the limited resources. In this chapter, we propose a resource distribution framework that provides context-driven and QoE-aware service differentiation, which means that starving clients are prioritized in resource allocation to enhance the corresponding end-user's quality of experience (QoE). Moreover, the framework actively motivates each Internet client to consistently provide its actual context information and to adopt moderate competition policies, given that all clients are selfish but rational in nature.

6.1 Motivations and Examples

As illustrated in the User-Context framework and its applications, the deduced KCI can be directly used to help the Internet to differentiate between clients that are really resource-starved and clients that are just ordinary resource consumers. The proposed User-Context Module essentially introduces the KCI into the resource distribution process and provides service differentiation in allocating the resource. More specifically, the Control Subsystem adaptively allocates the limited resources to real starving Internet clients based on the real-time KCI. Such a design could effectively improve the protocols performance and enhance the end-user's QoE.

On the other hand, another critical issue is to motivate the individual client to provide truthful and actual context information. Normal operations of the User-Context Module require that Internet clients provide their actual KCI in a timely way. In many cases, the limited resources are located on the server side or remote end of the network, and accordingly Internet clients are required to share their KCIs with the remote resource owner. However, Internet clients are assumed to be rational and selfish in nature, and therefore they may not be willing to provide their KCIs, especially the negative ones (e.g., Inactive State), which may lead to fewer allocated resources or a lower priority.

Moreover, the selfish nature results in the Internet clients competing aggressively for any limited resource over the Internet, typically including the resource held by servers. To further aid understanding of the above-described issues, we take two practical Internet systems as the illustrative examples, namely the World Wide Web system (Web system) and the Streaming Media system.

6.1.1 Web System Example

The Web system adopts the client-server architecture and leverages on the HTTP protocol for transferring Web pages between the Web server and the Web clients. On the Web server side, the child process usually creates multiple worker threads to handle any incoming HTTP connection requests: normally, one worker thread only serves one HTTP connection at a time on a first-come-first-served basis. Too many worker threads in Web server can easily cause thrashing in virtual memory system and considerably degrade server performance. In practice, a fixed limit is always imposed on the maximum number of worker threads: for example, the default maximum number in an Apache HTTP Server 2.2 is set to 256. Therefore, the worker threads held by the Web server always become the limited resource in the Web system. On the Web client side, HTTP/1.1 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 frequently violate this restriction: the default maximum value of Firefox 3.6 is set to 6 parallel persistent connections per server, and 8 persistent connections per proxy as default settings. Recently, the latest Internet Explorer as well as Google Chrome also aggressively adopts at least 6 parallel persistent connections per server as their default settings. Hence, the limited worker threads in the Web system are often subjected to excessive competition from aggressive and unconstrained Web clients.

As described in the previous chapters, today's Internet simply assumes that all the end-users behind their network hosts desire to communicate with the other end. The traditional Web system also assumes that all the allocated worker threads are being used by the end-users through the established HTTP connections, and thus it usually handles all incoming HTTP requests equally and maintains a first-in, first-out (FIFO) queue with the drop-tail queue management [81]. To handle the established HTTP connections, the Web system has to use the fixed timeout mechanism for releasing the worker thread [61]. As shown in the previous chapters, when the User-Context Module is introduced and implemented, the Web system can effectively differentiate between the worker threads that are being used by the real end-users and the worker threads that are just grabbed by the aggressive Web browsers. With such crucial context information, the User-Context Module can provide many possible service differentiation solutions. However, since providing the Inactive State (IS) information to the Web server may directly result in fewer and even no allocated worker threads, any rational and selfish Web client may not be willing to share its negative KCI. Accordingly, any well-designed Control Rules cannot properly work and become impractical.

6.1.2 Streaming Media System Example

The Streaming Media system also faces similar problems, and the uplink bandwidth on the streaming server side is always the most expensive and limited resource. For example, as a popular video on demand (VoD) and video-sharing system, YouTube mainly adopts the traditional client-server architecture and leverages on Content Distribution Networks (CDNs) despite some P2P-based local distribution proposals [82]. YouTube servers have to handle clients that normally request multiple clips at a time while demanding shortest buffer delays [83]. However,the system cannot provide any service differentiation in distributing its uplink bandwidth according to the end-user's viewing behavior and other relevant context information. Purchase of the uplink bandwidth imposes substantial costs on YouTube, and such high cost is one of the main reasons it is acquired by Google in 2006 [84]. When the User-Context Module is introduced and implemented, the Streaming Media systems, such as YouTube, could dynamically distribute its uplink bandwidth in an optimal way according to the watching behavior and other relevant context information of the clients. However, if providing the Inactive State or other negative context information would only result in less allocated bandwidth, the rational and selfish YouTube clients would not provide it or only provide the positive KCI for their own benefits. Under such circumstances, the User-Context Module cannot properly operate, even though the KCI can be accurately deduced on the YouTube client side by dedicated context models.

6.2 Objectives of the Framework

The analysis and the examples illustrate that the User-Context Module requires a systematic mechanism to address all the above-described open issues. We therefore propose a novel resource distribution framework with the three explicit design objectives:

1. The framework should provide *service differentiation* in allocating limited resources in terms of the deduced KCI.

2. The framework should encourage selfish and rational Internet clients to pro-

vide their *actual KCIs*, including the negative Inactive State.

3. The framework should motivate all Internet clients to adopt a *moderate competition* policy.

In this chapter, we first introduce the basic workflow of the resource distribution process. Its three-step distribution procedure can be used to evolve a variety of Internet services and systems. We then address the general design principles for the two indispensable algorithms used in the resource distribution process, and sequentially corresponding concrete algorithms that are conceptually simple and widely applicable. Based on the given two algorithms, we provide theoretical insights of the framework workflow, and show that the framework with its associated algorithms can effectively incentivize context sharing and moderate competition among the selfish but rational Internet clients. Finally, we implement and test the proposed framework on a Web system to validate the framework performance.

6.3 Framework Workflow

Assume that μ basic units of the limited resource are held by the server (or server cluster), which is termed *resource owner* in this framework. The limited resource can be of any type, such as worker thread, bandwidth, CPU time, memory, etc. A finite set of Internet clients, denoted by P_i, $i \in I = \{1, 2, ..., N\}$, compete for the given limited resource. All Internet clients update and transfer their latest KCIs to the resource owner through interoperable communication mechanisms, such as the XML Protocol (XMLP) [62], JAVA RMI (Remote Method Invocation) or any other pre-defined protocols or approaches. The resource owner maintains a database to store and manage the delivered KCIs with the timestamp of its recent update. Since clients only need to update their newly changed KCIs

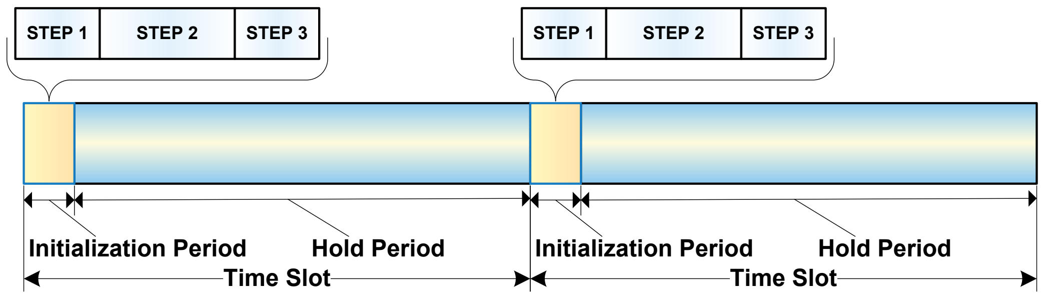


Fig. 6.1: Time slot divided into the Initialization Period and the Hold Period.

to the resource owner, synchronization between Internet clients and the resource owner is not required. On the resource owner side, the time domain is divided into fixed-sized time slots T_j, $j \in \{1, 2, ..., +\infty\}$, and as shown in Fig. 6.1, each individual time slot can be further divided into two parts: an Initialization Period and a subsequent Hold Period. The resource distribution process only occurs in the Initialization Period, but its result effects the entire Hold Period and part of the next Initialization Period. Normally, the Initialization Period only occupies a small portion of its time slot length, e.g., 5% to 10%.

Within each Initialization Period, the interaction steps between the resource owner and Internet clients, i.e., the basic workflow of the resource distribution framework, can be described as follows:

STEP 1: According to the current and historical KCI, the resource owner first performs the ***Willingness Update Algorithm (WUA)*** to calculate its willingness value for each Internet client. The willingness value, say $w_i(T_j)$, reflects the amount of resource that the resource owner is willing to offer to client P_i during the current time slot T_j. After performing the WUA, the resource owner immediately informs each client the assigned willingness value.

STEP 2: After receiving the assigned willingness value, each client, say P_i, takes a proper strategy to select a bidding value $b_i(T_j)$ and sends it back to the resource owner. The bidding value $b_i(T_j)$ reflects the amount of the resource that client P_i expects to obtain from the resource owner during the current time slot T_j. Meanwhile, based on its bidding value $b_i(T_j)$, a set of control rules on the client side need to be determined on a case-by-case basis.

STEP 3: With all the received bidding values as well as the original willingness values, the resource owner executes the ***Resource Distribution Algorithm (RDA)*** to obtain the final resource distribution result. The result $x_i(T_j), \forall i \in I$ is the amount of the resource finally assigned to client P_i for the current time slot T_j. Based on the final resource distribution result, a set of control rules on the server side need to be determined on a case-by-case basis.

Remark 1: If any individual client cannot provide its bidding value in a timely way before STEP 3 starts, the resource owner then assumes that the client uses the given willingness value as its bidding value.

Remark 2: The basic unit of the limited resource is selected depending on the resource characteristics and the usage case. For example, in the Web system, the single worker thread can be chosen as the basic unit of the limited resource. In the YouTube streaming system, 512 Kbps can be set as the basic unit for the uplink bandwidth on the server side, since the YouTube servers currently use the "block sending" method with the constant block size of 64 KB [84].

The above three-step procedure defines the basic workflow of the resource distribution framework, which is illustrated in Fig. 6.2. STEP 1 and STEP 3 of the framework workflow require the ***Willingness Update Algorithm*** and the

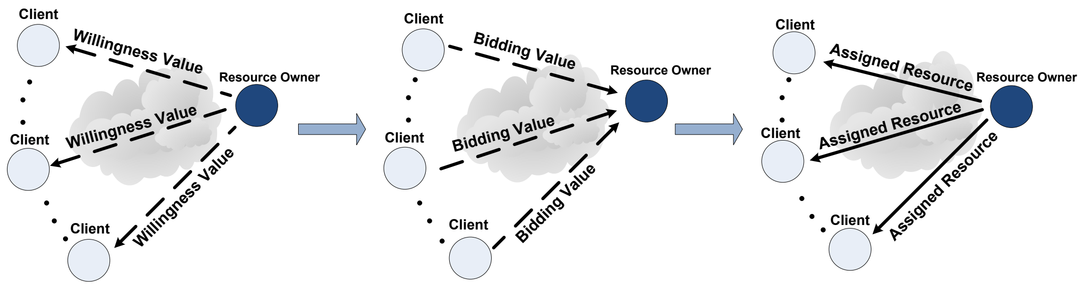


Fig. 6.2: Three steps in the basic workflow of the resource distribution framework.

Resource Distribution Algorithm, which will be discussed in the following sections, respectively. STEP 2 requires a proper bidding strategy, which will be discussed in the theoretical analysis section.

6.4 Willingness Update Algorithm

In STEP 1 of the workflow, the willingness value $w_i(T_j)$ reflects the amount of resource that the resource owner is willing to offer to client P_i during time slot T_j. The main objective of introducing the willingness value concept and the WUA is to make a preliminary resource distribution based only on the KCIs of Internet clients. The following design principles for the WUA are proposed:

- Group all Internet clients into multiple classes according to their current and historical KCI.
- Incentivize the prioritized class by assigning its members higher willingness values, while the prioritized classes should take into account both positive and negative KCI.
- The sum of the assigned willingness values equals to the total amount of the available limited resource.

In previous chapters, we have built the context models under the User-Context Module architecture to deduce the two fundamental categories of the KCI, namely the Communicating State (CS) and the Inactive State (IS). We assume that each client updates its KCI in a timely manner, either the CS or the IS with the time stamp to the database located on the resource owner side. For each Internet client, say P_i, we first introduce a new variable called the *duration ratio* which is defined as $q_i(\tau) = \frac{t_i^{CS}(\tau)}{t_i^{IS}(\tau)}$, where $t_i^{CS}(\tau)$ and $t_i^{IS}(\tau)$ are the cumulative times spent by client P_i in the CS and the IS over the previous τ time slots, respectively. At the beginning of each time slot, say T_j, the resource owner categorizes all clients into four classes according to the client's current KCI, denoted as $s_i(T_j)$, and its duration ratio $q_i(\tau)$:

$$\begin{aligned} C_1 &= \{P_i : s_i(\mathrm{T}_j) = \mathrm{CS}\ \&\ q_i(\tau) \leq \theta\} \\ C_2 &= \{P_i : s_i(\mathrm{T}_j) = \mathrm{IS}\ \&\ q_i(\tau) > 0\} \\ C_3 &= \{P_i : s_i(\mathrm{T}_j) = \mathrm{CS}\ \&\ q_i(\tau) > \theta\} \\ C_4 &= \{P_i : s_i(\mathrm{T}_j) = \mathrm{IS}\ \&\ q_i(\tau) = 0\}, \end{aligned}$$

where θ is a threshold parameter that needs to be specified by the resource owner. Classes C_1 and C_3 include all clients currently in the CS, while C_1 requires a small $q_i(\tau)$, i.e., a high proportion of the IS duration over the previous τ time slots. Classes C_2 and C_4 involve all clients currently in the IS, while C_4 requires that its members keep staying in the IS during the previous τ time slots. In principle, class C_1 has the highest priority among all classes. In other words, the client currently in the CS would receive large willingness value from the WUA, given that it spent enough time in the IS over the previous τ time slots. On the other hand, class C_4 has the lowest priority, because its members never transit back to the CS over the previous τ time slots. The priority order of classes C_2 and C_3

may alter depending on the usage case and the resource type. Such a classification essentially facilitates the resource owner incentivizing both the CS update and the IS update from Internet clients.

By leveraging on the above described four classes, we present a conceptually simple implementation of WUA based on the lottery scheduling [85]. The lottery scheduling is a simple randomized allocation mechanism: the allocation rights are represented by lottery *tickets* that are distributed among the participants. Each allocation is determined by holding a lottery. The reward is granted to the participant having the winning ticket in every round. In the proposed WUA, the clients in the same class, say C_r, receive an equal number of tickets, denoted as η_r, $r \in \{1,\ 2,\ 3,\ 4\}$. Given N_r is the total number of clients in class C_r, the following pseudo-code describes how the WUA calculates the willingness values for the current time slot T_j.

Algorithm 1 Willingness Update Algorithm (WUA)

```
Input: $C_r$, $N_r$ and $\eta_r$, $r \in \{1, 2, 3, 4\}$, $\mu$, $l$.
Output: Willingness values $w_i(T_j)$, $i \in I$.
 1: $w_i(T_j) = 0$, $i \in I$;
 2: Provide $\eta_1$, $\eta_2$, $\eta_3$ and $\eta_4$ lottery tickets to each client in classes $C_1$, $C_2$, $C_3$ and
    $C_4$, respectively;
 3: $K_N = \eta_1 * N_1 + \eta_2 * N_2 + \eta_3 * N_3 + \eta_4 * N_4$;
 4: $\mu_l = \mu * l$;
 5: for $l = 1 \to \mu_l$ do
 6:     Randomly pick one ticket from a total of $K_N$ tickets, denoted by $\lambda$;
 7:     if the player $P_i$ has the ticket $\lambda$ then
 8:         $w_i(T_j) = w_i(T_j) + 1$;
 9:     end if
10: end for
11: $w_i(T_j) = w_i(T_j)/l$, $i \in I$;
```

The WUA first clears the willingness values assigned in the previous time slot. It then distributes different number of tickets to each client according to its class, and calculates the total number of tickets used for the current time slot, i.e., K_N.

By multiplying an *amplification factor* l with the total amount of resource μ, the WUA obtains the value μ_l and accordingly holds μ_l rounds of lottery. In each round, the willingness value of the winning client is increased by 1. Finally, the willingness values are all divided by the same factor l to ensure that their sum equals to μ. Note that if the original μ is sufficiently large, the amplification factor l can be simply set to 1 in the algorithm.

Theoretically, the probability ρ that a client given η_r tickets will win a lottery with a total of K_N tickets is simply $\rho = \frac{\eta_r}{K_N}$. After μ_l identical lotteries, the expected willingness value of that client is $\mu_l * \rho$, with a variance $\mu_l * \rho(1-\rho)$. Accordingly, the assigned willingness value $w_i(T_j)$ follows the binomial distribution, which can be denoted as $w_i(T_j) \sim B(\mu_l,\ \rho)$. The corresponding coefficient of variation equals to $\sqrt{\frac{(1-\rho)}{\mu_l \rho}}$, which indicates that the disparity between the actual assigned willingness value and its expected value decreases with $\sqrt{\mu_l}$. Briefly, the expected willingness value assigned to a client is proportional to its share of the total ticket number. Hence, the resource owner can prioritize class C_r by simply providing more tickets to its clients, i.e., increasing η_r. In practice, the time span parameter τ and η_r, $r \in \{1,\ 2,\ 3,\ 4\}$ can be a constant or dynamically configured by the resource owner in terms of its priority policy and real-time workload.

Remark 1: For the new clients that request to join the resource distribution process, they have to wait until the new time slot starts. However, their KCI during the waiting time slot, say T_{j-1}, can be regarded as being in the IS, and thus they would be grouped into class C_1 in their first time slot T_j.

Remark 2: Besides the lottery scheduling algorithm, other algorithms and mechanisms, which fulfill the task of dynamic priority assignment and adjustment, can also be used to design new WUA.

6.5 Resource Distribution Algorithm

In STEP 3 of the framework workflow, the resource owner executes the Resource Distribution Algorithm (RDA) to obtain the resource distribution result of the current time slot. The following design principles for the RDA are proposed:

- An Internet client, who requests a reasonable amount of the resource, should be allocated a fair portion of the limited resource. The Internet client, who behaves aggressively, should be allocated less or even no resource.
- Any two Internet clients, who both adopt moderate bidding strategies and receive the same willingness value, should be allocated similar amount of the limited resource.
- The final resource allocation result should achieve a high level of satisfaction from the perspectives of both Internet clients and the resource owner.
- The RDA should strive to preserve the scalability, efficiency and responsiveness of the original system and its services.

Assume that the willingness values from the resource owner and the bidding values from Internet clients are given, we present a practical RDA based on the so-called water filling algorithms [86, 87] to demonstrate the described design principles. Each Internet client, say P_i, is treated as a bucket with an area $b_i(T_j)$ and a width $w_i(T_j)$ as shown in Fig. 6.3. Each bucket has a bottom thickness $\frac{b_i(T_j)}{w_i(T_j)}$, and accordingly its total height amounts to $\frac{2b_i(T_j)}{w_i(T_j)}$. The height of the bucket reflects the aggressiveness level of the client: higher bucket indicates more aggressiveness.

The main task of the RDA is to divide all the buckets (clients) into three groups according to their height: the "moderate" group, the "normal" group and the "aggressive" group, denoted as $G_1 = \{P_1, ..., P_L\}$, $G_2 = \{P_{L+1}, ..., P_M\}$ and

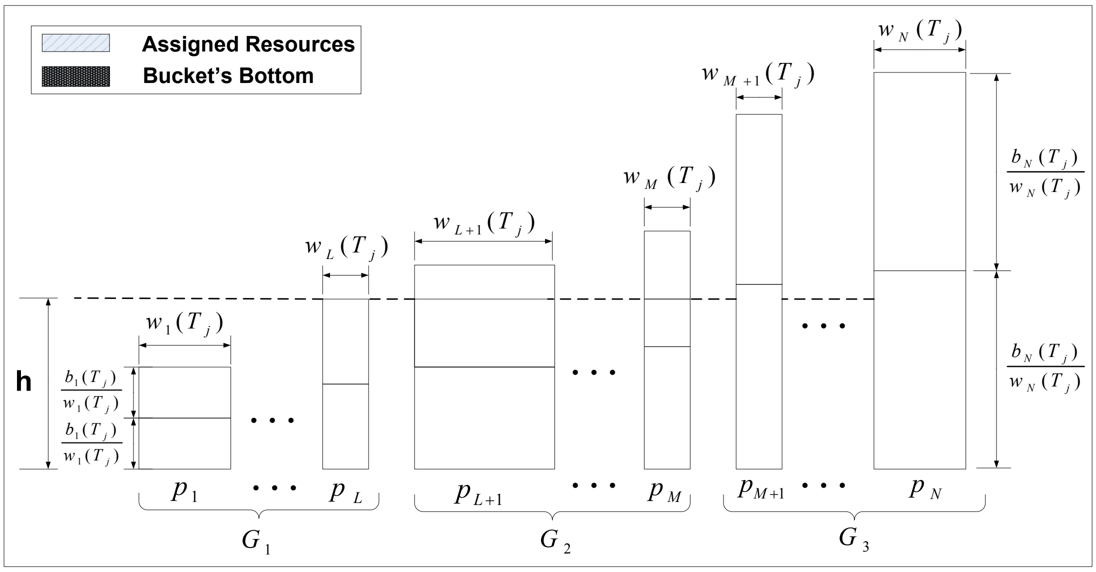


Fig. 6.3: Three bucket groups in the given Resource Distribution Algorithm (RDA).

$G_3 = \{P_{M+1}, ..., P_N\}$, $1 \leq L \leq M \leq N$. For the clients in group G_1, the RDA fulfils all their demands, i.e., offering their bidding amounts of the resource. For the clients in group G_2, the RDA partially satisfies their demands by offering a certain amount of resource, which ensures that all buckets in group G_2 reach the same **final height**, denoted by h. For the clients in group G_3, the RDA does not offer any resource to them. Fig. 6.3 illustrates the general distribution result of the RDA. The RDA can be expressed by the pseudo-code in Algorithm 2.

In Routine 1, the RDA successively selects a bucket from the shortest one and assumes it to be the last member of group G_1. Then the RDA calculates the corresponding amount of the required resource α: if α is less than the available amount of resource μ, the selected bucket would be assigned to group G_1 and the same procedure is applied to the next bucket; otherwise the RDA calculates the leftover resource and jumps to Routine 2. In Routine 2, the RDA successively selects a bucket from the tallest one and assumes it to be the first member of group G_3. Then it calculates the corresponding amount of the required resource

Algorithm 2 Resource Distribution Algorithm (RDA)

```
Input: $\mu$, $b_i$ and $w_i$, $\forall i \in I$ for the current time slot .
Output: Three groups $G_1$, $G_2$, $G_3$, and final height $h$ in group $G_2$.
Init: $low = \frac{2b_1}{w_1}$, $high = \frac{b_N}{w_N}$, $b_0 = 0$, and sort all clients in ascending $\frac{b_i}{w_i}$ order, denoted by $\left\{\frac{b_1}{w_1}, \frac{b_2}{w_2}, \dots, \frac{b_N}{w_N}\right\}$.
Routine 1:     /* pick out all clients in $G_1$ */
for $k = 1 \to N$ do
  $\alpha = \sum_{i=1}^{i=k} b_i$;
  $j = k + 1$;
  while $low > \frac{b_j}{w_j}$ do
    $\alpha += (low * w_j - b_j)$;
    $j++$;
  end while
  if $\alpha \le \mu$ then
    $L = k$;     /* assigned $P_k$ to $G_1$ */
    $low = \frac{2b_{k+1}}{w_{k+1}}$, $\alpha = 0$;
  else $\{\alpha > \mu\}$
    $\mu -= \sum_{i=0}^{i=k-1} b_i$;
    $\mu' = \mu$, exit for;
  end if
end for
Routine 2:     /* pick out all clients in $G_3$ */
for $k = N \to L + 1$ do
  if $high \ge \frac{2b_{L+1}}{w_{L+1}}$ then
    $high = \frac{b_{k-1}}{w_{k-1}}$;     /* assigned $P_k$ to $G_3$ */
  else if $high < \frac{2b_{L+1}}{w_{L+1}}$ then
    $\beta = \sum_{j=L+1}^{k} (high * w_j - b_j)$;
    if $\beta \ge \mu$ then
      $high = \frac{b_{k-1}}{w_{k-1}}$; /* assigned $P_k$ to $G_3$ */
    else $\{\beta < \mu\}$
      $M = k$;
      exit for;
    end if
  end if
end for
Routine 3:     /* calculate the final height $h$ in group $G_2$ */
$h = \frac{b_M}{w_M} + \frac{\mu - \sum_{i=L+1}^{M-1} (\frac{b_M}{w_M} * w_i - b_i)}{\sum_{i=L+1}^{M} w_i}$;
```

β: if β is larger than the leftover resource, the selected bucket would be assigned to group G_3 and the same procedure is applied to the next bucket; otherwise the RDA jumps to Routine 3. Since Routine 1 and Routine 2 have picked out all the buckets in groups G_1 and G_3, the rest of buckets would be automatically assigned to group G_2. In Routine 3, the RDA calculates the final height h in group G_2 with the leftover resource. After that, the resource owner can simply distribute the resource in terms of the three groups derived by the RDA:

$$x_k(T_j) = \begin{cases} b_k(T_j), & \forall k \in [1,\ L]; \\ w_k(T_j) * h - b_k(T_j), & \forall k \in [L+1,\ M]; \\ 0, & \forall k \in [M+1,\ N]. \end{cases}$$

Note that the prerequisite of running the given RDA is $\sum_{i=1}^{N} b_i > \mu$, which means that the sum of all bidding values exceeds the total amount of resource. When $\sum_{i=1}^{N} b_i \leq \mu$, the resource owner can simply regard all clients as "moderate" clients and offers their bidding amount of resource, i.e., $x_i(T_j) = b_i(T_j)$, $\forall i \in I$.

6.6 Theoretical Analysis of the Framework

In the previous two sections, we have presented the WUA and the RDA, which are required in STEP 1 and STEP 3 of the framework workflow respectively. In STEP 2 of the framework workflow, a bidding strategy needs to be independently determined by the individual Internet client. In this section, we demonstrate that Internet clients are motivated to actively share their actual Key Context Knowledge and moderately compete from the theoretical perspective. In addition, we also prove that the distribution results of the framework always maximize a particular form of the social welfare function with the given WUA and the RDA.

Since the concepts of Nash equilibrium and non-cooperative game are used in our analysis, we first briefly introduce the related parts of game theory.

6.6.1 Non-Cooperative Game and Nash Equilibrium

Game theory [88] is a mathematical tool for modeling and analyzing the strategic interactions among rational decision makers (players), and subsequently provides insight into various competitive environments and mechanisms. As one of the main branch of game theory, non-cooperative game describes the situation where each selfish player makes decisions independently and acts to maximize his own benefit [89]. The outcome of the non-cooperative game is termed as the Nash equilibrium, which essentially indicates that no individual player can unilaterally improve his payoff/utility given that the other players adopt the existing Nash equilibrium. One of the important applications of the non-cooperative game theory is to help design the mechanism that leads independent and selfish players towards a system-wide desirable outcome [90]. The details of the non-cooperative game theory and its examples of can be found in [91].

6.6.2 Theoretical Analysis

The basic workflow of the framework determines the three-step interaction process between the resource owner and its clients. Such an interaction process can be modeled and analyzed as a non-cooperative game: all Internet clients can be regarded as the game players; each game player can independently choose a bidding strategy to maximize its own payoff; the given WUA and RDA jointly work as the utility function and the final resource distribution results are the payoffs for each game player. Hence, we adopt the non-cooperative game theory tool to analyze the resource distribution process.

Lemma 1: Under the proposed framework with the given WUA and RDA, any Internet client, say P_c, who bids the assigned willingness value, i.e., $b_c(T_j) = w_c(T_j)$, can be guaranteed to receive its bidding amount of resource, i.e., $x_c(T_j) = b_c(T_j)$, regardless of other clients' bidding strategy.

Proof: The proof is given in the Appendix.

Lemma 2: Under the proposed framework with the given WUA and RDA, the bidding strategy profile $B^*(T_j) = \{b_c^*(T_j) : \ b_c^*(T_j) = w_c(T_j), \ \forall c \in I\}$ is the unique pure-strategy Nash equilibrium in time slot T_j.

Proof: The proof is given in the Appendix.

Proposition 1: Under the proposed resource distribution framework with the given WUA and RDA, the best policy for any individual Internet client is to share its actual KCI, i.e., either the CS or the IS, and meanwhile adopt a moderate bidding strategy to compete for the limited resource.

Proof: As mentioned earlier, in general, all Internet clients are rational and selfish in nature, and thus they always attempt to acquire more resource regardless of others. The proposed framework with the given WUA and RDA addresses it from both the context sharing and resource bidding aspects:

1. *Context Sharing*: in the given WUA, the highest prioritized class C_1 requires a high proportion of the IS duration over the previous τ time slots. Meanwhile, classes C_2 and C_3 have the same priority in the WUA. Hence, for any rational Internet client temporarily in the IS and not starving, the best policy is not to hide the IS but quickly update it to the resource owner. As a result, when its CS resumes, such a client will be most probably classified into class C_1 and accordingly receive a higher willingness value. Lemma 1 shows that the higher the willingness value received, the more resource can be guaranteed to gain from the resource owner. In other words, timely

update of negative KCI to the resource owner would be incentivized by allocating more resource when the client transits back to the positive KCI. To prevent Internet client from manipulating its KCI update to improve its payoff, the value of the time span parameter τ used in the duration ratio can be dynamically configured by the resource owner and not be made public to the clients. In short, motivated by such a specific service differentiation mechanism, the best policy for any rational Internet client is to actively share its actual KCI.

2. *Resource Bidding*: when any selfish client, say P_c, attempts to acquire more resource by adopting aggressive bidding strategies, i.e., $b_c(T_j) \gg w_c(T_j)$, Lemma 2 shows that such a client would deviate itself far from the system unique Nash equilibrium $B^*(T_j)$. As a result, the client cannot gain more resource to improve its payoff, but receives less or even no resource from the resource owner. Because adopting aggressive bidding strategies suffers a significant reduction in the finally allocated resource, the best policy for any rational Internet client is to adopt a moderate bidding strategy.

In short, with the given WUA and RDA, the proposed resource distribution framework can effectively motivate context sharing and moderate competition among Internet clients.

Lemma 3: Under the proposed framework with the given WUA and RDA, the distribution results $X = \{x_i(T_j) : \ \forall i \in I\}$ solves the following optimization problem:

$$\begin{aligned} &\max \prod_{i=1}^{N} \left(\frac{x_i(T_j)}{b_i(T_j)}+1\right)^{w_i(T_j)} \\ &\textit{subject to} \quad 0 \le x_i(T_j) \le\ b_i(T_j),\ \forall i \in I, \\ &\qquad\qquad\quad \sum_{i=1}^{N} x_i(T_j) \le \mu, \end{aligned} \tag{6.1}$$

where $w_i(T_j)$ and $b_i(T_j)$, $\forall i \in I$, are the willingness values and the bidding values, respectively, in time slot T_j.

Proof: The proof is given in the Appendix.

Proposition 2: Under the resource distribution framework with the given WUA and RDA, the resource distribution results maximize a social welfare function in all time slots.

Proof: On the resource owner side, the latest willingness value, say $w_i(T_j)$, can reflect the resource owner's satisfaction degree with Internet client P_i in terms of its current and historical KCI. On the Internet client side, the value $\frac{x_i(T_j)}{b_i(T_j)}$, i.e., the ratio of the finally assigned resource to its initial bidding amount, can reflect the satisfaction degree of client P_i with its resource distribution result in the current time slot T_j. Hence, we can choose function $\prod_{i=1}^{N} (\frac{x_i(T_j)}{b_i(T_j)}+1)^{w_i(T_j)}$ to simply describe the social welfare in time slot T_j, which considers the satisfactions from both the resource owner and all Internet clients.

Given a condition that each client cannot receive more resource than its bidding amount, maximizing the above social welfare function is equivalent to optimization problem (A.6). Lemma 3 has proven that the resource distribution result X in any given time slot solves optimization problem (A.6). Hence, the resource distribution results always maximize the selected social welfare function in all time slots.

Remark: The social welfare can be modeled by other functions on the condition that they consider the satisfactions of the resource owner as well as Internet clients.

6.7 Illustrative Case and Experimental Results

To demonstrate how the proposed framework operates in practice, we take the Web system as an illustrative case. As mentioned earlier, the resource owner, namely the Web server, holds a limited number of the worker threads, which often face excessive competitions from the Web clients. In this case, the Web client refers to the individual end-user and his Web browser(s). We assume that the KCI of each Web client, i.e., the CS and the IS, can be deduced in a timely way and delivered to the Web server. At the beginning of each time slot, i.e., in STEP 1 of the Initialization Period, the Web server first executes the given WUA to obtain the willingness values for all Web clients and immediately informs each client of their assigned willingness value. In STEP 2, each Web client needs to decide how many worker threads to bid for the current time slot, and the bidding value is essentially the number of parallel HTTP connection requests sent by the Web browser. Considering the given RDA running on the Web server side, any rational Web client, say P_i, would behave moderately and choose a bidding value $b_i(T_j)$ close to $w_i(T_j)$. In this case, Web client P_i can simply adopt a bidding strategy as follows:

$$b_i(T_j) = \max\{1,\ \lceil w_i(T_j) \rceil\}, \tag{6.2}$$

where $\lceil \cdot \rceil$ is the ceiling function. Accordingly, the control rules implemented on the client side actively adjust the number of the parallel HTTP connections that the Web browser sends to the Web server. Given that $x_i^r(T_{j-1})$ is the number of the established HTTP persistent connections between the Web browser and the Web server over the previous time slot T_{j-1}, the following control rules are implemented on the Web browser of client P_i based on bidding strategy (6.2):

(1) IF $b_i(T_j) > x_i^r(T_{j-1})$, THEN the Web browser immediately **initiates** $b_i(T_j) - x_i^r(T_{j-1})$ new HTTP connection requests to the Web server.

(2) IF $b_i(T_j) \leq x_i^r(T_{j-1})$, THEN the Web browser takes **no** action.

The above control rules indicate that the Web client does not need to perform the connection termination tasks, which are left for the server side. In ideal circumstances, $x_i^r(T_{j-1})$ equals to the resource distribution result of the previous time slot, i.e., $x_i^r(T_{j-1}) = x_i(T_{j-1})$.

In STEP 3, the Web server collects all the bidding values and executes the given RDA to obtain the resource distribution results for the current time slot, i.e., $x_i(T_j)$, $\forall i \in I$. Given that $\beta_i(T_j) = \lceil b_i(T_j) - x_i(T_j) \rceil$, the following control rules can be implemented on the Web server side:

(1) IF $\beta_i(T_j) > 0$, THEN the Web server **gracefully terminates** $\beta_i(T_j)$ established HTTP persistent connections with the Web client P_i.

(2) IF $\beta_i(T_j) = 0$, THEN the Web server takes **no** action on client P_i.

The above control rules on the server side essentially enable the Web server to take back the worker threads from the aggressive Web clients and accomplish the result of the RDA. Note that the given RDA guarantees the distribution result $x_i(T_j) \leq b_i(T_j)$, $\forall i \in I$, and thus it is unnecessary to consider the case $\beta_i(T_j) < 0$ in the above control rules.

Remark: Considering that the given WUA and RDA are running on the Web server side, it is reasonable for any commercial Web browsers to stop arbitrarily increasing the limit of parallel persistent connections per server, but adopt a proper competition policy similar to the moderate bidding strategy (6.2).

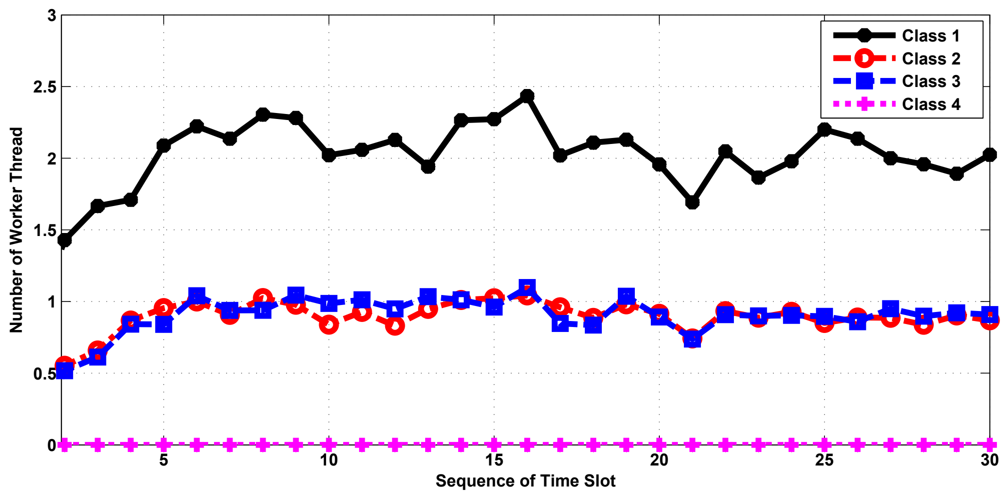
We have implemented the proposed framework as well as the above described control rules on a conventional Web system. On the Web server side, we have selected Apache HTTP Server 2.2.15, as it is a popular open-source Web server. In order to implement the control rules and the two algorithms, we have modified a small part of the Apache source code, which implements the HTTP protocol and the thread pool management. Then we re-compile the server under the Linux 2.6.28 and connect it to a MySQL database recording all the Web clients' KCI with the timestamp. On the Web client side, we use a HTTP request generator to emulate multiple Web clients. Each client switches between the CS and the IS and follows a similar state transition model given in [92], where the user sessions are exponentially distributed. In the experiment, we set the CS duration t_{cs} is exponentially distributed with the mean value of 20 seconds, denoted as $t_{cs} \sim exp(20)$, and the IS duration t_{is} is exponentially distributed with the mean value of 62.5 seconds, denoted as $t_{is} \sim exp(62.5)$. In the IS period, each Web client stops generating HTTP requests and keeps silent. In the CS period, each Web client makes sequential HTTP requests following a homogeneous Poisson process with a rate of 30 requests per minute. More specifically, the probability that a Web client sends k HTTP requests during the CS period satisfies $\Pr(k) = \frac{e^{-0.5t_{cs}}(0.5t_{cs})^k}{k!}$.

The total amount of the limited resource is set to 256 units, namely the default maximum number of parallel worker threads allowed in an Apache HTTP server. The time slot length has been set to 10 seconds equally, where STEP 1 and STEP 2 of the Initialization Period are required to be completed in 800 milliseconds. In the given WUA, the threshold parameter θ and the duration parameter τ are set to 3.0 and 2, respectively. The ticket numbers given to each class, i.e., η_r, $r \in \{1,\ 2,\ 3,\ 4\}$, are set to 4, 2, 2, 0, respectively. The amplification factor l used in the WUA is set to 10. In addition, a small positive constant is added to the $t_i^{IS}(\tau)$ to avoid a zero denominator when calculating the duration ratio $q_i(\tau)$.

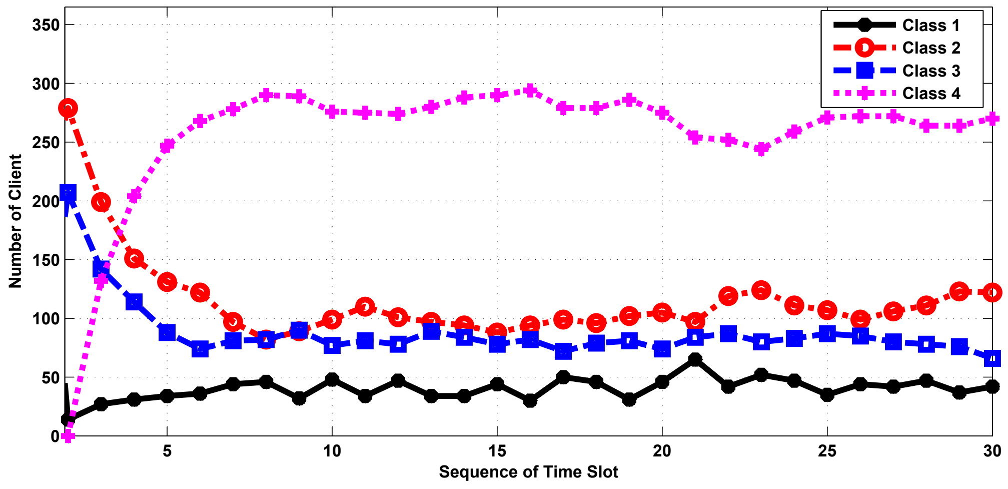
Our experimental hardware setup involves five hosts equipped with Duo Intel T7300 2.00-GHz processors and 2-GB RAM. One host runs the modified Apache HTTP Server, and others run the HTTP request generator to act as multiple Web clients. We adopt Dummynet [68], a widely-used tool to enforce queue delay and packet loss, to emulate operating in a wide area network (WAN) environment. We enable its delay function and set it to 50 milliseconds for both direction of each link. Thus, the round trip time (RTT) is around 150 milliseconds, which includes the database access time. We select four typical scenarios to demonstrate the framework performance and its important properties. The experiments run for 30 time slots each round, and we repeat 10 times for each example to average out fluctuations caused by the random variables in the algorithms.

Example 1 (Service Differentiation). We first consider 500 clients competing for the 256 worker threads. All of clients share their actual KCI including the negative IS, and adopt the moderate bidding strategy (6.2). Fig. 6.4(a) illustrates the average number of worker threads finally assigned to the individual client in the four classes categorized by the WUA. Fig. 6.4(b) illustrates the number of clients in each class. Fig. 6.5 demonstrates the enhancement of the average end-user's QoE on Web browsing.

In the traditional Web system with FIFO queue and drop-tail queue management, 500 concurrent clients competing for 256 worker threads would result in at least half of clients waiting in the pending connection queue or are simply blocked. Under the proposed resource distribution framework, Fig. 6.4(a) shows



(a) Average number of worker threads allocated to individual client in different classes



(b) Number of clients in the four classes

Fig. 6.4: Service differentiation under the resource distribution framework.

that the Web clients who are currently in the CS and hold a high proportion of the IS duration over the previous 2 time slots, i.e., the members of class C_1, obtain around 2 worker threads on average from the Web server. The Web clients who actively transit from the CS to the IS in the previous 2 time slots, i.e., the members of class C_2, can also obtain 1 worker thread on average. Meanwhile, the clients who are currently in the CS but hold a low proportion of the IS duration,

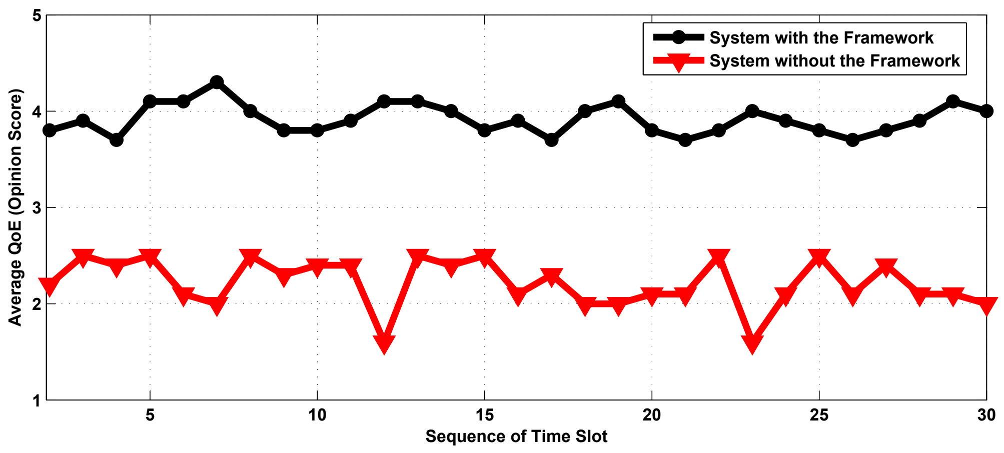


Fig. 6.5: A comparison of the average end-user's QoE on Web browsing.

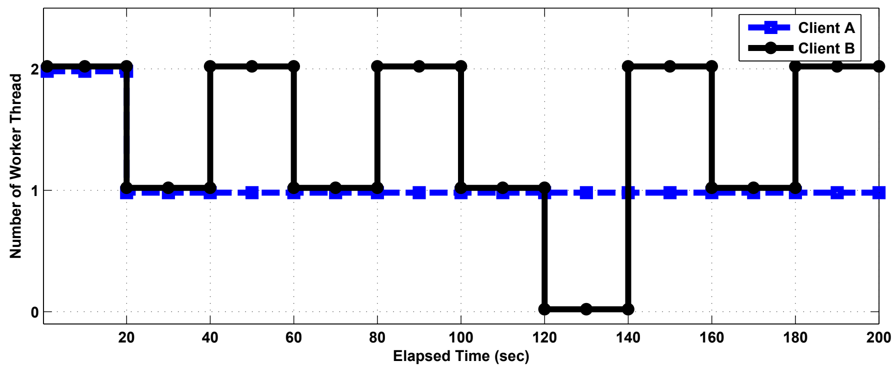
i.e., the members of class C_3, also obtain nearly 1 worker thread, since the same number of tickets are assigned to classes C_2 and C_3 in the WUA. The Web clients who have stayed in the IS for the previous 2 time slots, i.e. the members of class C_4, receive nothing from the resource owner.

For assessing the end-user's QoE, prior studies [69, 70] have systematically investigated the quantitative relationship between the QoE and the Web page download time. Shaikh et al. [69] shows that the exponential relationship gives the best correlation result: $QoE = 4.836 * \exp(-0.15T)$, where T is the Web page download time. Meanwhile, the Opinion Score has been used to rate the QoE: 5 = Excellent, 4 = Good, 3 = Average, 2 = Poor, 1 = Bad. As indicated earlier, the above mathematical relationship is valid only when the end-user keeps interacting with the Web service, namely the clients that are in the CS. Otherwise, the QoE would be always 0 no matter how much resource is allocated to the clients that are in the IS. Hence, in each time slot, we simply compute the average download time of all the Web clients in the CS, and then obtain the corresponding QoE value by using the given quantitative relationship. For the purpose of comparison, we also conduct the experiment and analysis on a conventional Web system with the FIFO queue and drop-tail queue management. Fig. 6.5 depicts the average end-user's QoE in both cases: we clearly see that the system with the proposed framework maintains a much higher QoE than the conventional Web system. The main reason is that the proposed framework provides service differentiation and successfully allocates the limited worker threads to the starving clients, i.e., the members in classes C_1, C_2 and C_3, while the traditional Web system only uniformly treats all the incoming HTTP requests that would result in large queueing delay for the starving clients. Moreover, the Web page transmission time can also be reduced when multiple worker threads, e.g., 2 worker threads for the member of class C_1, can be allocated to the same starving client. Note that it is difficult to establish a quantitative relationship between the end-user's QoE and the number of the allocated worker threads due to lack of reliable models of the Web server and the Internet.

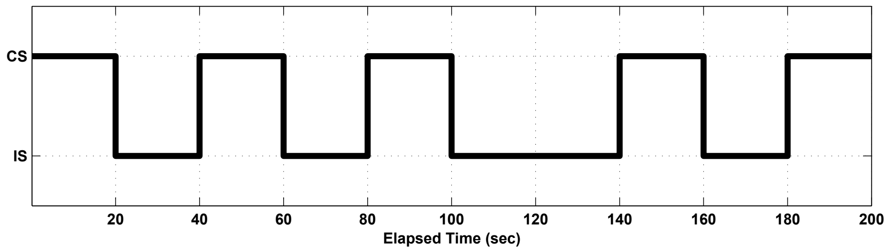
In short, the experimental results confirm that the framework effectively provides the service differentiation in terms of the current and historical KCI, and significantly enhances the end-user's QoE.

Example 2 (Context Sharing). We still consider 500 clients, among which client A and client B always have the same KCI during all time slots as shown in Fig. 6.6(b). Client A purposely never updates its IS to the resource owner but fraudulently informs the resource owner the CS. Client B honestly updates its KCI transitions to the resource owner. Both of them adopt the moderate bidding strategy (6.2). Fig. 6.6(a) demonstrates the final resource distribution results of the two clients.

As shown in Fig. 6.6, during the 1st and the 2nd time slots, both client A and client B are allocated 2 worker threads, because the WUA groups the new clients into class C_1. Both client A and client B transit from the CS to the IS before



(a) Amount of the resource allocated to client A and client B



(b) Key Context Information transitions of client A and client B

Fig. 6.6: A comparison between the honest client and the dishonest client.

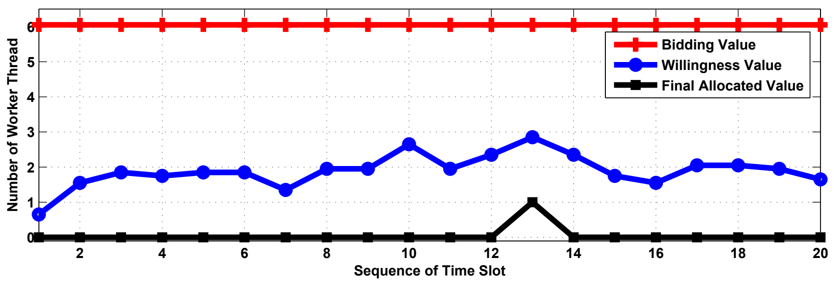
the 3rd time slot arrives, while only client B updates its transition to the resource owner. Accordingly, client A and client B are grouped into classes C_3 and C_2 respectively and are allocated 1 worker thread over the next 2 time slots. Then both clients switch back to the CS before the 5th time slot comes, and client B also provides timely updates it to the resource owner. Because client B shares its IS between the 3rd and the 4th time slot, it has a lower duration ratio $q_B(\tau)$ over the 5th and the 6th time slot. Hence, client B can be grouped into class C_1 by the WUA and is allocated 2 worker threads during its CS period. Meanwhile, client A cannot receive any more worker threads and still stays in class C_3 over the 5th and the 6th time slot, because it never shares its negative IS and thus its duration ratio $q_A(\tau)$ is kept high. In all the subsequent time slots, client B is always allocated more resource than client A when it transits back to the CS, which is due to client

B actively sharing its negative IS with the resource owner. Hence, the framework effectively encourages clients to provide their actual KCI.

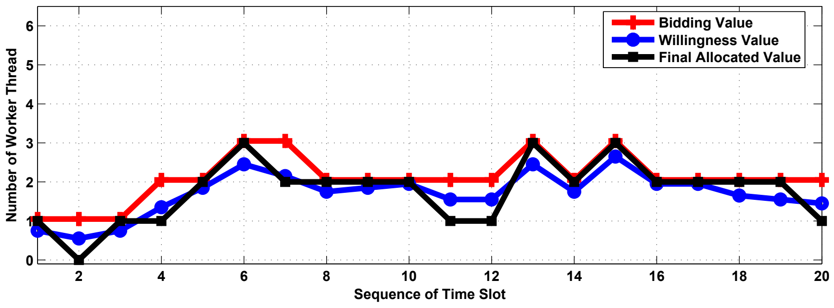
Note that during the 13$^{\text{th}}$ and the 14$^{\text{th}}$ time slots, client B has entered the long idle status, which indicates that its end-user has not been interacting with the corresponding Web page for more than 2 time slots. Hence, it is reasonable that the framework allocates less or even no worker thread to client B during this period. The saved worker threads are assigned to the starving clients by the Web server. In addition, dynamically adjusting the number of lottery tickets for each class, i.e., η_r, and the threshold parameter θ in the WUA can enable the incentivization mechanism to be more adaptive and responsive.

Example 3 (Moderate Competition). We still consider the 500 concurrent clients, among which client A and client B always receive the similar willingness values from the resource owner. Client A adopts an aggressive policy that keeps sending 6 parallel HTTP persistent connection requests to the Web server, i.e., bidding 6 worker threads. Client B adopts the given moderate bidding strategy (6.2) and the corresponding client-side control rules. Both of them provide their actual KCI to the Web server. The received willingness values, the bidding values and the final distribution results of client A and client B are shown in Fig. 6.7(a) and Fig. 6.7(b), respectively.

As shown in Fig. 6.7(a), client A adopts an aggressive strategy and its bidding value 6 is much higher than the given willingness values. As a result, client A obtains almost no worker thread from the Web server in each time slot. On the contrary, client B adopts the moderate strategy that the bidding value is always close to the assigned willingness value and the corresponding control rules. As a result, client B successfully gains a reasonable number of worker threads in each time slot as shown in Fig. 6.7(b). From the final distribution results of



(a) Client A (aggressive)

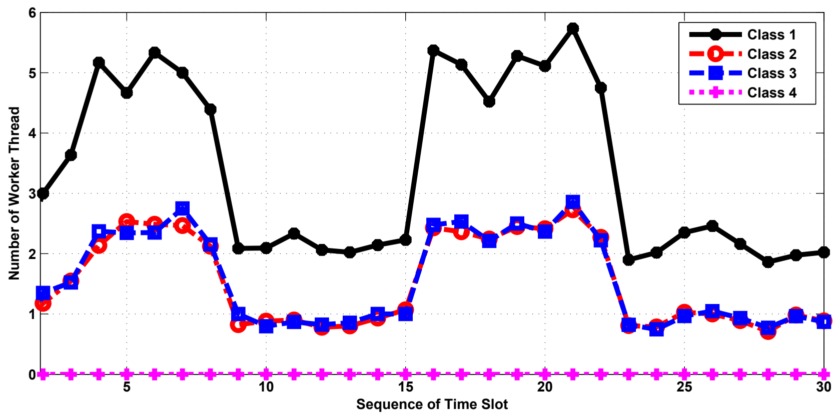


(b) Client B (moderate)

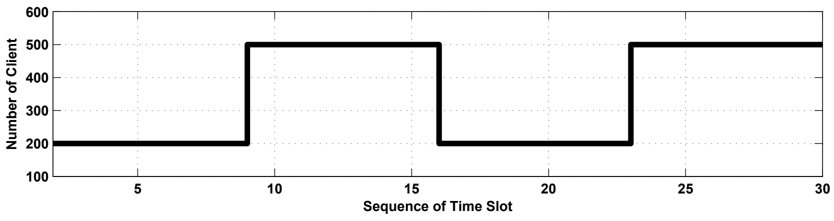
Fig. 6.7: A comparison between the aggressive client and the moderate client.

the two comparable clients, we see that the framework guarantees the moderate clients to receive their fair portion of the limited resource, and meanwhile penalizes the aggressive clients by decreasing the allocated resource. Hence, it effectively incentivizes moderate competition among Internet clients.

Example 4 (Adaptability). We further investigate the performance of the framework under a dynamic condition where there are significant fluctuations in the total client number. As shown in Fig. 6.8(b), only 200 clients compete during the first 8 time slots. From the 9th time slot onwards, the total client number dramatically increases to 500 and holds for the subsequent 8 time slots. From the 16th time slot onwards, the total client number drops back to 200 for another 8 time slots and then increases to 500 again.



(a) Average number of worker threads allocated to individual client in different classes



(b) Fluctuations in the total client number

Fig. 6.8: Framework adaptivity in terms of the total client number.

Fig. 6.8(a) illustrates the system performance under such dynamic conditions. We see that the framework can still effectively provide the expected service differentiation: when the total number of clients is 200, the individual client in class C_1 receives around 5 worker threads on average, and the individual client in classes C_2 or C_3 obtains more than 2 worker threads on average. When the total number of clients dramatically increases to 500, the framework quickly adjusts its willingness values and the distribution results are similar to Example 1, which also demonstrates the stability of the framework. In short, this example illustrates that the proposed framework can gracefully handle both the heavy workload and the light workload situations.

6.8 Summary

In this chapter, we present a novel resource distribution framework that provides context-driven and QoE-aware service differentiation. The framework also incentivizes context-sharing and moderate competition under the User-Context Module architecture. By leveraging on the selfish but rational nature of Internet clients, the proposed framework explicitly allocates the limited resource to the starving Internet clients. A three-step framework workflow as well as the required WUA and RDA algorithms are sequentially introduced. We analyze the Internet clients behavior by formulating a noncooperative game and prove that the framework guides all clients (game players) towards a unique Nash equilibrium. We have proven that under the proposed framework, the best policy for any Internet clients is to provide their actual KCI and self-enforce moderate competition policies. Furthermore, we prove that the distribution results computed by the framework maximize a social welfare function. We demonstrate the motivation, operation and performance of the framework by presenting a Web system example. The experimental results in four scenarios demonstrate that the improved Web system performance and confirm that all the three design goals are achieved. The proposed framework mainly addresses the limited resource located on the server side, and it cannot be directly applied on the case of the intermediate nodes. More sophisticated framework workflow may be required for the new case, although the similar system architecture can still be utilized.

Chapter 7

Conclusion and Future Work

In this thesis dissertation, we have devoted our research efforts on incorporating the context information of end-users into the Internet protocol stack and services. In this final chapter, we summarize our work and suggest several areas that merit future research.

7.1 Research Summary

7.1.1 The User-Context Module Architecture

How to empower the Internet protocol stack to capture, understand and utilize an end-user's context information is the most fundamental problem in our research. Our solution to this problem is a new functional module, called the User-Context Module, working with the five-layer Internet protocol stack. The User-Context Module architecture mainly consists of three key components, which are the Context Sensing Subsystem, the Context Model Subsystem and the Control Subsystem. Such design adopts the modularity approach to decompose the solution into the "loosely coupled" parts, and each has realized a specific and indispensable function:

- The Context Sensing Subsystem mainly undertakes the basic context information gathering task and closely works with end-users on the Internet client side.
- The Context Model Subsystem employs logic based context models to deduce the well-defined KCI based on the captured basic context information. The deduced two categories of KCI represent the highly abstract and substantive context information for a variety of Internet services.
- The Control Subsystem directly adjusts the Internet communication protocols and services in different layers, and accordingly enables the Internet protocol stack adapting itself to the deduced KCI.

The three subsystems operate together as a coherent entity to improve the performance of the Internet protocol stack and enhance the subjective satisfaction of Internet end-users. In the previous chapters, the thesis has laid particular stress on the design and implementation issues of the User-Context Module architecture and its three subsystems.

7.1.2 The Key Context Information and Context Models

In order to provide the desired context information to the Internet protocol stack and service, we first introduce the cognitive psychology model to help define the KCI, and then construct the corresponding context models to deduce it. More specifically, we first define two basic end-user states, i.e., User Perception State and User Halt State, based on the MHP in cognitive psychology. Such two end-user states, combining with Internet services working status, are used to further define the two categories of KCI, i.e., Communicating State and Inactive State. With the five given Interaction Conditions, the corresponding context models are built using the first-order rule-based reasoning approach to deduce the two categories of KCI for different Internet services. The well-defined KCI and the context models directly help the User-Context Module to understand and ascertain the two fundamental interaction states between end-users and the Internet protocol stack.

7.1.3 The Applications of the User-Context Module

The modular design of the User-Context Module provides abundant flexibilities for various applications and deployment plans. We have presented two applications of the User-Context Module, which introduce the deduced KCI into the Application Layer's HTTP protocol and the Transport Layer's TCP protocol, respectively.

In the first User-Context Module application, the Control Subsystem adaptively adjusts the HTTP persistent connection timeout parameter to reduce the end-user perceived latency in Web browsing and the unnecessary HTTP traffic burden on the Internet backbone.

In the second User-Context Module application, the Control Subsystem manipulates the advertised window size in TCP protocol to dynamically allocate the limited bandwidth at the access link. Such an application effectively enhances the end-user's QoE and the TCP performance.

Both applications have demonstrated the User-Context Module's operations and impacts as well as its Control Subsystem's design and implementation. Furthermore, the design philosophy behind the two applications can be generalized and applied in new User-Context Module applications, whose main objectives includes, but not limited to, improving protocol performance and enhancing the QoE.

7.1.4 The Resource Distribution Framework

In most cases, the prerequisite of running a practical User-Context Module application is that Internet clients actively provide truthful KCI, including the negative ones. Hence, we have proposed a novel resource distribution framework that not only provides the context-driven and QoE-aware service differentiation, but also incentivizes the actual context sharing and moderate competition among selfish and rational Internet clients. The framework workflow and the algorithm design principles are introduced to provide the guidelines of building and evolving new User-Context Module applications. The given WUA algorithm and RDA algorithm can effectively encourage selfish Internet clients to provide the deduced KCI, especially the negative *Inactive State*. Meanwhile, the framework incentivizes moderate competition by penalizing aggressive Internet clients.

7.2 Future Research Directions

Introducing the context information of end-users into the Internet protocol stack has a large exploration space and many issues warrant further investigation.

7.2.1 Advanced End-User Models and KCI

Human's cognitive mechanism, which decides how to respond to the external stimuli, greatly and directly influences the Internet end-user's interaction behavior. Despite the difficulties in explaining and measuring human cognitive capability, the latest progress of cognitive psychology, brain-computer interface (BCI) as well as neuroscience hold great promise for building more accurate end-user models. With advanced end-user models, new KCI can be accordingly defined and introduced into the User-Context Module to further describe interactions among end-user, networked host and Internet service. The defined KCI, i.e., Communicating State and Inactive State, can also serve as the building blocks of newly introduced KCI. All KCI should keep highly abstract and be cautiously used to avoid impairing the basic functionalities of the Internet architecture and communication protocols. When building context the corresponding models to deduce new KCI, advanced context modeling and reasoning approaches, such as the ontology reasoning, can be considered and employed. Moreover, the computational overhead for collecting and deducing new KCI at client side should also be considered and carefully assessed.

On the Internet server side, methods to describe group KCI and group context models require further study with the aim of facilitating batch processing and enhancing server performance. Moreover, the mechanisms and algorithms to dealing with the large-scale delayed KCIs on the server side are also needed to be considered and designed.

7.2.2 More Applications of the User-Context Module

Currently, the User-Context Module mainly works with the two protocols in the upper two layers, i.e., the Application Layer's HTTP protocol and the Transport Layer's TCP protocol. To further enhance the Internet as a user-centered and context-aware communication system, the User-Context Module is expected to explore its interactions with many other communication protocols on all the five layers. For example, the Control Subsystem of the User-Context Module could seek to adjust the session-related variables in Real Time Streaming Protocol (RTSP) on Application Layer or the security-related variables in Internet Protocol Security (IPsec) on Network Layer. To meet new Control Subsystem's requirements, the Context Sensing Subsystem and the Context Model Subsystem may also need to be re-designed or upgraded .

7.2.3 Context Usage in Future Internet Architecture

Our User-Context Module provides a solution of introducing context information into the legacy Internet architecture, where any major changes to the established layered architecture and protocols are almost impossible. Hence, the KCI can only be utilized to interact with the accessible and manageable components in the existing Internet communication protocols and services. Otherwise, any newly introduced context information would effect the Internet integrity and modularity, which leads to unintended consequences on overall performance of the protocol stack.

Currently, there are a number research projects on architectural design for the next generation Internet using the so called *clean-slate* approach. The clean-slate approach means designing the Internet from scratch without being bounded by the constraints of the established Internet architecture and protocol stack. For example, the identifier-locator split architecture, which was described in the chapter of the related work, also adopts the clean slate design approach. Under the identifier-locator split architecture, we suggest that the User Realm could explicitly include the well defined KCI with the User-ID and the Service-ID.

To be generally applicable in the future Internet, the User-Context Module could expose a set of application programming interfaces (API) that any protocols could use to extract the captured and relevant context information. Such a general API solution requires three indispensable conditions for the future Internet architecture: (1) control subsystem with the control rules; (2) adjustable and critical parameters in protocols; (3) well-defined control interfaces of protocols that are able to tune the adjustable parameters. Some of the proposed architectures essentially have satisfied all the above three conditions. For example, SILO architecture [93] consists of a control agent with its policy (control subsystem with the control rules), the control knobs (the adjustable parameters) and the service-specified control interfaces. For the future protocols that frequently use the specific context information, we also suggest that they can reserve 2 to 8 bits in their header for the standardized context information.

7.3 Conclusion

The main contributions and novelty of this work can be summarized as follows:

- The designed User-Context Module firstly reveals the fact that properly selected context information of end-users can be directly used in the underlying Internet protocol stack to help enhance QoE and improve protocol performance.
- The designed User-Context Module explicitly demonstrates how to capture, ascertain and utilize the required context information of end-users.
- The designed User-Context Module effectively provides context-driven service differentiation, and incentivizes actual context sharing and moderate competition among Internet clients.

We hope this work can inspire Internet designers and open up a new realm for innovations on both the existing and future Internet design.

Bibliography

[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] R. Braden, "Requirements for Internet Hosts – Communication Layers," *IETF RFC 1122*, 1989.

[3] R. Lachman *et al.*, *Cognitive psychology and information processing: An introduction.* Lawrence Erlbaum Associates, 1979.

[4] A. Sears and J. A. Jacko, *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications, Second Edition (Human Factors and Ergonomics).* Lawrence Erlbaum Associates, 2002.

[5] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach.* Pearson Education, Inc., 2008.

[6] H. Zimmermann, "OSI Reference Model–The ISO Model of Architecture for Open Systems Interconnection," *IEEE Transactions on Communications*, vol. 28, no. 4, pp. 425–432, 1980.

[7] 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.

[8] 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.

[9] S. Paul, J. Pan, and R. Jain, "Architectures for the future networks and the next generation Internet: A survey," *Computer Communications*, vol. 34, no. 1, pp. 2–42, 2011.

[10] J. Pan, R. Jain, S. Paul, and C. So-in, "MILSA: A New Evolutionary Architecture for Scalability, Mobility, and Multihoming in the Future Internet," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 8, pp. 1344–1362, 2010.

[11] R. Moskowitz and P. Nikander, "Host Identity Protocol (HIP) Architecture," *IETF RFC 4423*, 2006.

[12] D. Farinacci, V. Fuller, *et al.*, "Locater/ID Separation Protocol(LISP)," *IETF Internet Draft*, 2012.

[13] U. Neisser, *Cognitive Psychology.* New York, NY: Meredith, 1967.

[14] S. K. Card, T. P. Moran, and A. Newell, *The psychology of human computer interaction.* L. Erlbaum Associates Inc., 1983.

[15] D. E. Meyer and D. E. Kieras, "A Computational Theory of Executive Cognitive Processes and Multiple-Task Performance," *Psychological Review*, vol. 104, no. 1, pp. 3–65, 1997.

[16] J. L. McClelland, "On the time relations of mental processes: A framework for analyzing processes in cascade," *Psychological Review*, vol. 88, pp. 375–407, 1979.

[17] R. Ratcliff and P. L. Smith, "A comparison of sequential sampling models for two-choice reaction time," *Psychological Review*, vol. 111, pp. 333–367, 2004.

[18] T. V. Zandt, H. Colonius, and R. W. Proctor, "A comparison of two response time models applied to perceptual matching," *Psychological Bulletin and Review*, vol. 7, pp. 208–256, 2000.

[19] K. P. L. Vu and R. W. Proctor, *Handbook of human factors in web design.* Mahwah, NJ: Erlbaum, 2004.

[20] D. D. Schmorrow, *Foundations of augmented cognition.* Mahwah, NJ: Erlbaum, 2005.

[21] M. Weiser, "The Computer for the Twenty-First Century," *Scientific American*, September, 2002.

[22] A. K. Dey, "Understanding and Using Context," *Personal Ubiquitous Computing*, vol. 5, no. 1, pp. 4–7, 2001.

[23] H. Chen, *An Intelligent Broker Architecture for Pervasive Context-Aware Systems.* PhD thesis, University of Maryland, 2004.

[24] T. Gu, H. K. Pung, and D. Q. Zhang, "A Service-oriented middleware for building context-aware services," *Journal of Network and Computer Applications*, vol. 28, no. 1, pp. 1–18, 2005.

[25] M. Roman *et al.*, "A Middleware Infrastructure for Active Spaces," *IEEE Pervasive Computing*, vol. 1, no. 4, pp. 74–83, 2002.

[26] T. Strang and C. Linnhoff-Popien, “A Context Modeling Survey,” in *Proc. UbiComp*, (Nottingham, UK), Sept. 2004.

[27] M. Baldauf, S. Dustdar, and F. Rosenberg, “A Survey on Context-aware Systems,” *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 2, no. 4, pp. 263–277, 2007.

[28] R. Want, A. Hopper, V. Falcao, and J. Gibbons, “The Active Badge Location System,” *ACM Transactions on Information Systems*, vol. 10, no. 1, pp. 91–102, 1992.

[29] MIT Oxygen. [Online] Available: http://oxygen.lcs.mit.edu/, 2004.

[30] Georgia Tech Aware Home. [Online] Available: http://awarehome.imtc.gatech.edu/, 2010.

[31] IBM Blue Space. [Online] Available: http://www.research.ibm.com/bluespace/, 2003.

[32] R. Chen *et al.*, “Modeling the Ambient Intelligence Application System: Concept, Software, Data, and Network,” *IEEE Transactions on Systems, Man, and Cybernetics-Part C*, vol. 39, no. 3, pp. 299–314, 2009.

[33] A. Padovitz, S. W. Loke, and A. Zaslavsky, “Multiple-Agent Perspectives in Reasoning About Situations for Context-Aware Pervasive Computing Systems,” *IEEE Transactions on Systems, Man, and Cybernetics-Part A*, vol. 38, no. 4, pp. 729–742, 2008.

[34] H. L. Truong and S. Dustdar, “A Survey on Context-aware Web Service Systems,” *International Journal of Web Information Systems*, vol. 5, no. 1, pp. 5–31, 2009.

[35] B. Han, W. Jia, J. Shen, and M. C. Yuen, “Context-awareness in Mobile Web Services,” *Parallel and Distributed Processing and Applications*, vol. 3358, pp. 519–528, 2008.

[36] I. Matsumura *et al.*, “Situated Web Service: Context-aware Approach to High-Speed Web Service Communication,” in *Proc. IEEE Conf. on Web Services*, (Chicago, IL), Sept. 2006.

[37] C. D. Wang, T. Li, and L. C. Feng, “Context-aware Environment-Role-Based Access Control Model for Web Services,” in *Proc. IEEE Conf. on Multimedia and Ubiquitous Engineering*, (Busan, Korea), April 2008.

[38] ITU-T. Rec. P. 10 /G. 100, Amendment 2: New Definitions for Inclusion in Recommendation ITU-T P.10/G.100, 2008.

[39] R. Jain, "Quality of Experience," *IEEE Multimedia*, vol. 11, no. 1, pp. 95–96, 2004.

[40] Nokia, "Quality of Experience (QoE) of mobile services: Can it be measured and improved," white paper, finland, 2004.

[41] M. Fiedler, T. Hossfeld, and P. Tran-Gia, "A generic quantitative relationship between quality of experience and quality of service," *IEEE Network*, vol. 24, no. 2, pp. 36–41, 2010.

[42] P. Brooks and B. Hestnes, "User measures of quality of experience: why being objective and quantitative is important," *IEEE Network*, vol. 24, no. 2, pp. 8–13, 2010.

[43] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library.* O'Relly Media, Inc., 2008.

[44] Z. Ramdane-Cherif and A. Nait-Ali, "An adaptive algorithm for eye-gaze-tracking-device calibration," *IEEE Transactions on Instrumentation and Measurement*, vol. 57, no. 4, pp. 716–723, 2008.

[45] M. Betke, J. Gips, and P. Fleming, "The Camera Mouse: Visual Tracking of Body Features to Provide Computer Access for People With Severe Disabilities," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 10, no. 1, pp. 1–10, 2002.

[46] R. Lienhart and J. Maydt, "An Extended Set of Haar-Like Features for Rapid Object Detection," in *Proc. IEEE ICIP*, (New York, USA), September 2006.

[47] P. I. Wilson and J. Fernandez, "Facial feature detection using Haar classifiers," *Journal of Computing Sciences in Colleges*, vol. 21, no. 4, pp. 127–133, 2006.

[48] F. Risso and L. Degioanni, "An architecture for high performance network analysis," in *Proc. IEEE ISCC*, (HAMMAMET, Tunisia), July 2001.

[49] H. Liu, H. Darabi, P. Banerjee, and J. Liu, "Survey of Wireless Indoor Positioning Techniques and Systems," *IEEE Transactions on System, Man and Cybernetics - Part C: Applications and Reviews*, vol. 37, no. 6, pp. 1067–1080, 2007.

[50] I. H. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques.* San Francisco: Morgan Kaufmann, 2005.

[51] T. Gu, X. H. Wang, H. K. Pung, and D. Q. Zhang, "An ontology-based context model in intelligent environments," in *Proc. Communication Networks and Distributed Systems Modeling and Simulation Conference*, (San Diego, USA), Jan. 2004.

[52] J. R. Anderson, M. P. Matessa, and C. Lebiere, "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.

[53] 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.

[54] A. Newell, *Unified theories of cognition.* Cambridge, MA: Harvard University Press, 1990.

[55] V. Kawadia and P. R. Kumar, "A cautionary perspective on cross-layer design," *IEEE Wireless Communications*, vol. 12, no. 1, pp. 3–11, 2005.

[56] 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.

[57] R. Fielding *et al.*, "Hypertext Transfer Protocol – HTTP/1.1," *IETF RFC 2616*, 1999.

[58] 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.

[59] P. Barford and M. Crovella, "A performance evaluation of hyper text transfer protocols," *ACM SIGMETRICS Performance Evaluation Review*, vol. 27, no. 1, pp. 188–197, 1999.

[60] J. C. Mogul, "The case for persistent-connection HTTP," in *Proc. ACM SIGCOMM*, (Cambridge, MA, USA), Aug. 1995.

[61] 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.

[62] N. Mitra and Y. Lafon, "Simple object access protocol (soap)." [online] Available: http://www.w3c.org/TR/soap, 2007.

[63] SPECweb. [online] Available: http://www.spec.org/web2005, 2005.

[64] P. Barford and M. Crovella, "Generating representative Web workloads for network and server performance evaluation," in *Proc. ACM SIGMETRICS*, (Madison, Wisconsin, USA), June 1998.

[65] W3C, "Libwww: The w3c protocol library." [online] Available: http://www.w3.org/Library, 2004.

[66] 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*, (New York, USA), May 2004.

[67] 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.

[68] Dummynet. [online] Available: http://info.iet.unipi.it/ luigi/dummynet/.

[69] J. Shaikh, M. Fiedler, and D. Collange, "Quality of Experience from User and Network Perspectives," *Ann. Telecommun.*, vol. 65, pp. 47–57, 2010.

[70] ITU-T. Rec. G. 1030: Estimating end-to-end performance in IP networks for data applications, 2005.

[71] J. Postel, "Transmission Control Protocol," *RFC 793*, 1981.

[72] T. V. Laskshman and U. Madhow, "The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss," *IEEE/ACM Transactions on Networking*, vol. 5, no. 3, pp. 336–350, 1997.

[73] P. Mehra, C. D. Vleeschouwer, and A. Zakhor, "Receiver-Driven Bandwidth sharing for TCP and its Applications to Video Streaming," *IEEE/ACM Transactions on Multimedia*, vol. 7, no. 4, pp. 740–752, 2005.

[74] N. T. Spring *et al.*, "Receiver based management of low bandwidth access links," in *Proc. IEEE INFOCOM*, (Tel Aviv, Israel), March 2000.

[75] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose, "Modeling TCP Reno performance: a simple model and its empirical validation," *IEEE/ACM Transactions on Networking*, vol. 8, no. 2, pp. 133–145, 2000.

[76] V. Jacobson, R. Braden, and D. Borman, "TCP Extensions for High Performance," *Information Science Institute, RFC 1323*, 1992.

[77] F. Berzosa *et al.*, "Receiver-based RTT Measurement in TCP," *U.S. Patent 7330426 B2*, 2008.

[78] L. Software, "Netlimiter: Internet traffic control and monitoring tool," 2010. [online] Available: http://www.netlimiter.com/.

[79] P. Reichl *et al.*, "The Logarithmic Nature of QoE and the Role of the Weber-Fechner Law in QoE Assessment," in *Proc. IEEE ICC 2010*, (Cape Town, South Africa), May 2010.

[80] Y. Lu, M. Motani, and W. C. Wong, "Intelligent Network Design: User Layer Architecture and its application," in *Proc. IEEE SMC*, (Istanbul, Turkey), Oct. 2010.

[81] J. Wei and C. Z. Xu, "eQoS: Provisioning of Client-Perceived End-to-End QoS Guarantees in Web Servers," *IEEE Transactions on Computer*, vol. 55, no. 12, pp. 1543–1556, 2006.

[82] X. Cheng and J. Liu, "NetTube: Exploring Social Networks for Peer-to-Peer Short Video Sharing," in *Proc. IEEE INFOCOM*, (Rio de Janeiro, Brazil), April 2009.

[83] M. Zink, K. Suh, Y. Gu, and J. Kurose, "Characteristics of YouTube network traffic at a campus network - Measurements, models, and implications," *Computer Networks*, vol. 53, no. 4, pp. 501–514, 2009.

[84] S. Alcock and R. Nelson, "Application Flow Control in YouTube Video Streaming," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 2, pp. 25–30, 2011.

[85] C. A. Waldspurger, *Lottery and Stride Scheduling: Flexible Proportional-Share Resource Management.* PhD thesis, MIT, 1995.

[86] J. L. Boudec, "Rate adaptation, congestion control and fairness: A tutorial." [online] Available: http://icapeople.epfl.ch/leboudec, 2008.

[87] R. T. B. Ma, S. C. M. Lee, J. C. S. Lui, and D. K. Y. Yau, "Incentive and Service Differentiation in P2P Networks: A Game Theoretic Approach," *IEEE/ACM Transactions on Networking*, vol. 14, no. 5, pp. 978–991, 2006.

[88] G. Owen, *Game Thoery, 3rd Edition.* San Diego, CA: Academic, 1995.

[89] B. Wang, Y. Wu, and K. J. R. Liu, "Game Theory for Cognitive Radio Networks: An overview," *Computer Networks*, vol. 54, no. 14, pp. 2537–2561, 2010.

[90] K. Akkarajitsakul, E. Hossain, D. Niyato, and D. I. Kim, "Game Theoretic Approaches for Multiple Access in Wireless Networks: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 13, no. 3, pp. 372–395, 2011.

[91] K. Ritzberger, *Foundations of Noncooperative Game Theory.* New York, Oxford University Press, 2002.

[92] P. Gill *et al.*, "Characterizing user sessions on YouTube," in *Proc. Annual Multimedia Computing and Networking Conference*, (San Jose, CA, USA), Jan. 2008.

[93] R. Dutta *et al.*, “The SILO Architecture for Services Integration, controL, and Optimization for the Future Internet,” in *Proc. ICC*, (Glasgow, Scotland), June 2007.

Appendix A

Proof of Lemma 1, 2 and 3.

Lemma 1: Under the proposed framework with the given WUA and RDA, any Internet client, say P_c, who bids the assigned willingness value, i.e., $b_c(T_j) = w_c(T_j)$, can be guaranteed to receive its bidding amount of resource, i.e., $x_c(T_j) = b_c(T_j)$, regardless of other clients' bidding strategy.

Proof: In STEP 1, the given WUA satisfies that the sum of the assigned willingness values equals to the total amount of resource, i.e., $\sum\limits_{i=1}^{N} w_i(T_j) = \mu$. Hence, we have

$$w_c(T_j) = \frac{w_c(T_j)}{\sum\limits_{i=1}^{N} w_i(T_j)} * \mu. \tag{A.1}$$

In STEP 3, the given RDA classifies all clients into three groups $G_1 = \{P_1, ..., P_L\}$, $G_2 = \{P_{L+1}, ..., P_M\}$ and $G_3 = \{P_{M+1}, ..., P_N\}$, where $1 \leq L \leq M \leq N$. The given RDA under the framework guarantees that all the members in group G_2 reach the same ***final height*** h by offering the members a certain amount of resource. Consider P_L and P_M are the last members of group G_1 and group G_2, we have

$$\begin{cases} \frac{2b_L}{w_L} \leq h < \frac{2b_{L+1}}{w_{L+1}} \\ \frac{2b_M}{w_M} \leq 2h < \frac{2b_{M+1}}{w_{M+1}}. \end{cases}$$

where the time expression T_j can be omitted within any individual time slot. Because all clients in group G_1 receive the amount of their bidding value, the amount of the resource assigned to group G_1, denoted by μ_1, satisfies

$$\mu_1 = \sum_{i=1}^{L} x_i = \sum_{i=1}^{L} b_i \leq \sum_{i=1}^{L} \frac{h}{2} * w_i.$$

The amount of the resource assigned to group G_2, denoted by μ_2, satisfies

$$\mu_2 = \sum_{i=L+1}^{M} x_i = \sum_{i=L+1}^{M} (h * w_i - b_i) < \sum_{i=L+1}^{M} \frac{h}{2} * w_i.$$

The amount of the resource assigned to group G_3, denoted by μ_3, satisfies

$$\mu_3 = \sum_{i=M+1}^{N} x_i = 0.$$

Hence, the amount of the resource assigned to all clients satisfies

$$\mu_1 + \mu_2 + \mu_3 < \sum_{i=1}^{M} \frac{h}{2} * w_i. \tag{A.2}$$

Because only the clients in group G_1 receive their bidding amount of resource, i.e., $x_i = b_i$, we need to prove that any client, say P_c, whose bidding value $b_c = w_c$, must be assigned to group G_1 by the given RDA. We consider the two cases that P_c is assigned to group G_2 and group G_3 respectively.

(1) When client P_c is assigned to group G_2, i.e., $h < \frac{2b_c}{w_c} \leq 2h$: consider $b_c = w_c$ and (A.1), we have $h < \frac{2\mu}{\sum_{i=1}^{N} w_i}$. Together with (A.2), we get

$$\mu_1 + \mu_2 + \mu_3 < \sum_{i=1}^{M} \frac{h}{2} * w_i < \mu * \frac{\sum\limits_{i=1}^{M} w_i}{\sum\limits_{i=1}^{N} w_i} < \mu.$$

The above inequality shows that the total assigned resource is less than the total available resource, which conflicts with the basic design principle of the given RDA. Hence, it is impossible that client P_c is assigned to group G_2 by the RDA.

(2) When the player P_c is assigned to group G_3, i.e., $2h < \frac{2b_c}{w_c}$: consider $b_c = w_c$ and (A.1), we have $h < \frac{\mu}{\sum\limits_{i=1}^{N} w_i}$. Together with (A.2), we get

$$\mu_1 + \mu_2 + \mu_3 < \sum_{i=1}^{M} \frac{h}{2} * w_i < \frac{\mu}{2} * \frac{\sum\limits_{i=1}^{M} w_i}{\sum\limits_{i=1}^{N} w_i} < \mu.$$

The above inequality also conflicts with the given RDA, and thus it is also impossible that client P_c is assigned to group G_3 by the RDA.

To sum, any client, who bids the given willingness value, can only be assigned to group G_1 by the RDA, and accordingly receives its bidding amount of resource regardless of other clients' bidding strategies.

Lemma 2: Under the proposed framework with the given WUA and RDA, the bidding strategy profile $B^*(T_j) = \{b_c^*(T_j) :\ b_c^*(T_j) = w_c(T_j),\ \forall c \in I\}$ is the unique pure-strategy Nash equilibrium in time slot T_j.

Proof: From Lemma 1 and the given strategy profile B^*, we have

$$\sum_{c=1}^{N} b_c = \sum_{c=1}^{N} x_c = \mu, \tag{A.3}$$

where the time slot expression T_j is also omitted. (A.3) shows that the limited resource is just used up and all clients are assigned to group G_1 by the RDA. Consider Lemma 1, no individual client, say P_c, could gain more resource by a unilateral deviation from its initial bidding strategy $b_c = w_c$, given that all the other clients insist on their own bidding strategy. Therefore, the strategy profile B^* is one pure-strategy Nash equilibrium of the competition game in time slot T_j.

Next, we further prove the uniqueness of the derived Nash equilibrium. Assume that there exists another pure-strategy Nash equilibrium $\tilde{B} = \{\tilde{b}_c :\ \forall c \in I\}$ and the corresponding distribution result $\tilde{X} = \{\tilde{x}_c :\ \forall c \in I\}$. The Nash equilibrium $\tilde{B}$ must satisfy the conditions

$$\begin{cases} \tilde{b}_c \geq b_c^* = w_c, & \forall c \in I; \\ \tilde{x}_c = x_c^* = w_c, & \forall c \in I. \end{cases} \tag{A.4}$$

Otherwise, the client, say P_c, which receives $\tilde{x}_c < w_c$, can improve its payoff $\tilde{x}_c$ by unilaterally change its bidding strategy to $\tilde{b}_c = w_c$, which guarantees $\tilde{x}_c = w_c$. (A.4) demonstrates that any other Nash equilibrium requires at least one client be assigned to group G_2 and no client be assigned to group G_3 by the RDA. Hence, there are three possible cases.

(1) Only one client, say P_M, in group G_2, i.e., $\tilde{b}_M > b_M^*$ and $L + 1 = M = N$: in this case, there always exists a small positive constant δ, such that the following

condition can be satisfied:

$$\frac{2b_1^* + \delta}{w_1} < \frac{x_M^* + \tilde{b}_M - \delta}{w_M}.$$

Hence, client P_1 can always improve its payoff from x_1^* to $x_1^*+\delta$ by unilaterally increasing its bidding value from b_1^* to $b_1^* + \delta$. More generally, when only one client is assigned to group G_2, any clients in group G_1 can gain more resource by cautiously adding a small positive constant to its initial bidding value. Hence, no Nash equilibrium exists in this case.

(2) Multiple but not all clients in group G_2, i.e., $\tilde{b}_k > b_k^*$, $\forall k \in [L+1,\ M]$, and $1 < L+1 < M = N$: in this case, any client in group G_1, say P_1, can also improve its payoff from x_1^* to $x_1^* + \delta$ by unilaterally increasing the bidding value from b_1^* to $b_1^* + \delta$, given δ satisfying the condition

$$\frac{2b_1^* + \delta}{w_1} < \frac{x_{L+1}^* + \tilde{b}_{L+1} - \delta}{w_{L+1}}.$$

Hence, no Nash equilibrium exists in this case as well.

(3) All clients in group G_2, i.e., $\tilde{b}_c > b_c^*$, $\forall c \in I$ and $0 = L < M = N$: in this case, the given RDA guarantees

$$\frac{x_1^* + \tilde{b}_1}{w_1} = \frac{x_2^* + \tilde{b}_2}{w_2} = \cdots = \frac{x_N^* + \tilde{b}_N}{w_N} = h'. \tag{A.5}$$

where h' is the final height in group G_2. (A.5) indicates that there always exists a positive constant $\varepsilon < \tilde{b}_1 - b_1^*$, such that client P_1 can improve its payoff by unilaterally decreasing its bidding value from $\tilde{b}_1$ to $\tilde{b}_1 - \varepsilon$. More generally, when all clients are assigned to group G_2, any client can improve its payoff by cautiously reducing its bidding value. Hence, no Nash equilibrium exists in this case as well.

In short, no pure-strategy Nash equilibrium exists in all possible cases. Therefore, the given strategy profile $B^*(T_j) = \{b_c^*(T_j) : \ b_c^*(T_j) = w_c(T_j), \ \forall c \in I\}$ is the unique pure-strategy Nash equilibrium within any individual time slot T_j, $j \in [1, 2, ..., +\infty]$.

Lemma 3: Under the proposed framework with the given WUA and RDA, the distribution results $X = \{x_i(T_j) : \forall i \in I\}$ solves the following optimization problem:

$$\begin{aligned}
&\max \prod_{i=1}^{N} \left(\tfrac{x_i(T_j)}{b_i(T_j)} + 1\right)^{w_i(T_j)} \\
&\textit{subject to} \quad 0 \leq x_i(T_j) \leq\ b_i(T_j),\ \forall i \in I, \\
&\qquad\qquad\quad \sum_{i=1}^{N} x_i(T_j) \leq \mu,
\end{aligned} \tag{A.6}$$

where $w_i(T_j)$ and $b_i(T_j)$, $\forall i \in I$, are the willingness values and the bidding values in time slot T_j.

Proof: After the logarithmic transformation of the given objective function, the optimization problem can be expressed equivalently as follows:

$$\begin{aligned}
\min \quad & -\sum_{i=1}^{N} \log{(x_i + b_i)}^{w_i} \\
\textit{subject to} \quad & \sum_{i=1}^{N} x_i - \mu \leq 0, \\
& x_i - b_i \leq 0, \\
& -x_i \leq 0,
\end{aligned}$$

where $\forall i \in I$ and the time slot expression T_j is omitted. It is a convex optimization problem, as the new objective function as well as all inequality constraints are continuously differentiable and convex. In addition, because the inequality constraints satisfy Slater's condition, then strong duality holds, i.e., the optimal duality gap is zero. Therefore, the Karush-Kuhn-Tucker (KKT) conditions are not only necessary, but also sufficient conditions for the points to be primal and dual optimal. In short, to prove $X = \{x_i : \forall i \in I\}$ solves the original optimization problem, if and only if it satisfies the following KKT conditions:

$$\begin{cases} \sum_{i=1}^{N} x_i - \mu \leq 0; \\ x_i - b_i \leq 0 \ , \quad i = 1, 2, ..., N; \\ -x_i \leq 0 \ , \qquad i = 1, 2, ..., N; \\ \lambda_i \geq 0, \quad i = 0, 1, 2, ..., 2N; \\ \lambda_0 * (\sum_{i=1}^{N} x_i - \mu) = 0; \\ \lambda_i * (x_i - b_i) = 0, \quad i = 1, 2, ..., N; \\ \lambda_{i+N} * (-x_i) = 0, \quad i = 1, 2, ..., N; \\ \nabla(-\sum_{i=1}^{N} \log{(x_i + b_i)^{w_i}}) + \lambda_0 \nabla(\sum_{i=1}^{N} x_i - \mu) + \sum_{i=1}^{N} \lambda_i \nabla(x_i - b_i) \\ + \sum_{i=1}^{N} \lambda_{i+N} \nabla(-x_i) = 0, \end{cases}$$

where all λ_i, $\forall i \in I$ are the Lagrange multipliers. The KKT conditions can be further simplified to the following equivalent conditions:

$$\begin{cases} \{\sum_{i=1}^{N} x_i - \mu < 0 \ \& \ \lambda_0 = 0\} \ \ or \ \ \{\sum_{i=1}^{N} x_i - \mu = 0 \ \& \ \lambda_0 \geq 0\}; \\ \{x_i - b_i < 0 \ \& \ \lambda_i = 0\} \ \ or \ \ \{x_i - b_i = 0 \ \& \ \lambda_i \geq 0\}; \\ \{x_i > 0 \ \& \ \lambda_{i+N} = 0\} \ \ or \ \ \{x_i = 0 \ \& \ \lambda_{i+N} \geq 0\}; \\ \frac{-w_i}{x_i + b_i} + \lambda_0 + \lambda_i - \lambda_{i+N} = 0; \\ 0 \leq x_i \leq b_i, \end{cases} \tag{A.7}$$

where $\forall i \in I$. To prove that the final distribution result $X = \{x_i : \ \forall i \in I\}$ always satisfies the above derived KKT conditions, two possible cases need to be considered.

(1) $\sum_{i=1}^{N} x_i = \mu$: in this case, the framework executes the given RDA to divide all clients into the three groups, i.e., the "moderate" group G_1, the "normal" group G_2 and the " aggressive" group G_3, and then assign them the corresponding

amount of resource. We consider the most general situation that all three groups are co-exist. Note that the given RDA under the framework guarantees that all the members in group G_2 reach the same final height h by offering a certain amount of resource, i.e., $h = \frac{x_i+b_i}{w_i}$, where $L+1 \leq i \leq M$.

The "moderate" group G_1 receives $x_i = b_i$, and accordingly the KKT conditions require $\lambda_0 \geq 0$, $\lambda_i \geq 0$ and $\lambda_{i+N} = 0$, where $1 \leq i \leq L$.

The "normal" group G_2 receives $0 < x_i < b_i$, and accordingly the KKT conditions require $\lambda_0 \geq 0$, $\lambda_i = 0$ and $\lambda_{i+N} = 0$, where $L+1 \leq i \leq M$.

The "aggressive" group G_3 receives $x_i = 0$, and accordingly the KKT conditions require $\lambda_0 \geq 0$, $\lambda_i = 0$ and $\lambda_{i+N} \geq 0$, where $M+1 \leq i \leq N$.

Considering all the above necessary conditions together, we have

$$\begin{cases} \lambda_i = \frac{w_i}{x_i+b_i} - \frac{1}{h}, & 1 \leq i \leq L; \\ \lambda_i = 0, & L+1 \leq i \leq N; \\ \lambda_{i+N} = 0, & 1 \leq i \leq M; \\ \lambda_{i+N} = \frac{1}{h} - \frac{w_i}{b_i}, & M+1 \leq i \leq N; \\ \lambda_0 = \frac{1}{h}. \end{cases}$$

The above solution guarantees that the resource distribution result $X = \{x_i : \forall i \in I\}$ satisfies the derived KKT conditions (A.7), and therefore it is also the solution of the initial optimization problem.

(2)$\sum\limits_{i=1}^{N} x_i < \mu$: in this case, the resource is not completely used up, which indicates $\sum\limits_{i=1}^{N} b_i < \mu$. Accordingly, the resource owner does not need to execute the given RDA, but simply distributes the resources $x_i = b_i$, $\forall i \in I$. Let $\lambda_0 = 0$, $\lambda_{i+N} = 0$ and $\lambda_i = \frac{w_i}{2b_i}$, where $\forall i \in I$, the derived KKT conditions (A.7) can be satisfied as well. Hence, the distribution result also solves the initial optimization problem.

Appendix B

List of Publications

1. Yu Lu, Mehul Motani, and Wai-Choong Wong, "When Ambient Intelligence Meets the Internet: User Module Framework and its Applications," *Computer Networks (Elsevier)*, vo. 56, no. 6, pp. 1763-1781, 2012.

2. Yu Lu, Mehul Motani, and Wai-Choong Wong, "A QoE-Aware Resource Distribution Framework Incentivizing Context Sharing and Moderate Competition," *IEEE/ACM Transactions on Networking*, *under review.*

3. Yu Lu, Mehul Motani, and Wai-Choong Wong, "The User-Context Module: A New Perspective on Future Internet Design," in *Proc. International Conference on Ambient Systems, Networks and Technologies (ANT)*, Niagara Falls, Ontario, Canada, Sept., 2011.

4. Yu Lu, Mehul Motani, and Wai-Choong Wong, "When Ambient Intelligence Meets Internet Protocol Stack: User Layer Design," in *Proc. IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, Hong Kong SAR, China, Dec., 2010.

5. Yu Lu, Mehul Motani, and Wai-Choong Wong, "Intelligent Network Design: User Layer Architecture and its application," in *Proc. IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Istanbul, Turkey, Oct., 2010.