A QoE-Aware Resource Distribution Framework Incentivizing Context Sharing and Moderate Competition

Yu Lu, *Member, IEEE*, Mehul Motani, *Member, IEEE*, and Wai-Choong Wong, *Senior Member, IEEE*

***Abstract*—We contend that context information of Internet clients can help to efficiently manage a variety of underlying resources for different Internet services and systems. We therefore propose a resource distribution framework that provides quality of experience (QoE) aware service differentiation, which means that starving clients are prioritized in resource allocation to enhance the corresponding end-user's QoE. The framework also 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. We analyze the Internet client's behavior by formulating a non-cooperative game and prove that the framework guides all clients (game players) towards a unique Nash equilibrium. Furthermore, we prove that the distribution results computed by the framework maximize a social welfare function. Throughout this paper, we demonstrate the motivation, operation and performance of the framework by presenting a Web system example, which leverages on the advanced context information deduced by a context-aware system.**

***Index Terms*—Context awareness, internet services, quality of experience, resource distribution, World Wide Web.**

I. Introduction

ADVANCES in context-aware computing greatly facilitate the traditional Internet to capture the end-user's presence and interaction activities with client-side software or tools. In general, context-aware computing makes use of various sensors and techniques, e.g., wireless network camera and computer vision techniques, to enable a system to be aware of the state of the end-user and other relevant context information. The system then can adapt its operations to the captured context information with the aim of increasing its usability and effectiveness. There has been an entire body of research dedicated to building such context-aware systems [1]. By leveraging on existing context-aware systems, the Internet could easily acquire its client's various context information and accordingly further enhance itself as a user-centric and context-aware communication system. A number of efforts have been made to introduce the context information into the Internet system, especially for enabling adaptable Web service systems [2].

However, few prior studies consider utilizing the Internet client's context information to specifically manage the underlying resources, particularly if such resource is limited and subjected to excessive competition among Internet clients. More explicitly, the context information of Internet clients 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. Introducing such context information into the resource distribution process can help properly allocate the limited resources to the real starving Internet clients, and accordingly enhance the end-user's perception on the performance of the corresponding Internet service, i.e., the QoE of the end-user.

To aid understanding of our motivation, we take the contemporary World Wide Web system as an illustrative example. On the Web server side, the system process only creates a restricted number of worker threads to handle the incoming HTTP connection requests, because too many worker threads can easily cause thrashing in the virtual memory system and considerably degrade server performance. For example, the default maximum number of threads in an Apache HTTP Server 2.2 is set to 256. Meanwhile, most Web servers handle their clients equally by maintaining a first-in, first-out (FIFO) queue and adopt a fixed timeout mechanism. If a Web system can capture the valuable real-time context information, e.g., the end-user is currently browsing its Web pages or the end-user is distracted by other irrelevant matters, then the system could allocate the limited worker threads in an optimal way according to the captured context information.

The streaming media system, e.g., YouTube, can be another good 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) [3]. YouTube servers have to handle clients that normally request multiple clips at a time while demand shortest buffer delays [4]. 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 [5]. Similarly, if a YouTube system can capture the valuable real-time context information, e.g., the end-user is watching the specific video clip and buffering others, the system can provide the corresponding service differentiation in distributing its expensive uplink bandwidth according to the captured context information. The above two examples show that utilizing the Internet client's context information holds great promise to help Internet systems to prop-

Manuscript received May 14, 2012; revised November 12, 2013 and July 27, 2014; accepted February 20, 2015; approved by IEEE/ACM Transactions on Networking Editor V. Misra.

Y. Lu is with the Institute for Infocomm Research, A*STAR, and National University of Singapore, Singapore (e-mail: luyu@i2r.a-star.edu.sg).

M. Motani and W.-C. Wong are with the Electrical and Computer Engineering Department, National University of Singapore, Singapore, 119260.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2015.2409895

erly manage different types of limited resources, which are typically located on the server side.

On the other hand, another critical issue is to motivate the individual client to provide truthful and actual context information. The Internet client refers to the entity consisting of the Internet end-user and the client-side software, such as the Web end-user and the Web browser. The Internet clients are assumed to be selfish but rational in nature, and therefore they would not be willing to provide their context information, especially the negative ones that 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.

We still take the Web system as a typical example. On the Web client side, HTTP/1.1 [6] specifies that a single Web client (Web browser) should not maintain more than two HTTP persistent connections, i.e., concurrently grabbing more than two worker threads, with any given server or proxy. However, the selfish Web clients frequently violate this: the Web browser Firefox 3.6 sets six parallel persistent connections per server and eight persistent connections per proxy as default parameters. Similarly, the latest version of Internet Explorer and Google Chrome also employ at least six parallel persistent connections per server as default settings. In short, the limited resources, e.g., the worker threads and the uplink bandwidth, often faces excessive competition from selfish Internet clients. Moreover, the selfish nature prevents them from actively providing their actual context information, especially the negative ones.

To address the above issues, we propose a resource distribution framework with three explicit design objectives: 1) provide *QoE-aware service differentiation* in allocating limited underlying resources; 2) encourage all Internet clients to provide their *actual context information*; 3) motivate all Internet clients to adopt a *moderate competition* policy. More explicitly, our work makes four main contributions:

1) Our context-aware vision motivates a new design for allocating limited resources in terms of the significant context information of Internet clients. The proposed three-step distribution procedure can be used to evolve a variety of Internet services and systems.
2) Based on the proposed framework workflow, we first address the general design principles for the two indispensable algorithms used in the resource distribution process, and then present corresponding concrete algorithms that are conceptually simple and widely applicable.
3) 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.
4) We have implemented and tested the proposed framework on a Web system to validate the framework performance.

The rest of this paper is organized as follows. Section II reviews the related work. Section III presents the critical and fundamental context information deduced by an existing context-aware system. Section IV provides a novel resource distribution framework and the theoretical analysis. Section V demonstrates the framework implementations on an illustrative Web system and the experimental results. Section VI concludes the paper.

II. Related Work

The ubiquitous computing idea [7] 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 [8]. Context information acquisition is the process of capturing and managing the basic context information from heterogeneous sensors, and there have been several context information acquisition approaches, typically including the middleware based approach, the context server based approach and the direct sensor access approach [9]. The middleware based approach uses a method of encapsulation to separate and hide low-level sensing details, which eases rapid prototyping of a context-aware system and improves system extensibility and reusability. The middleware based approach has been widely adopted in the context-aware system design, and the system normally consists of the Context Sensing Layer, the Context Middleware Layer and the Context Application Layer.

In existing context-aware systems, the Internet always serves as the default long distance data communication carrier, while limited prior studies consider enabling the Internet to adapt itself to the captured context information. The context-aware Web service system [2] can be regarded as an attempt in this direction. The context-aware Web service system mainly employs the end-user's context information to support Web content adaptation [10], communication optimization [11] as well as security and privacy control [12]. For example, an end-user's preference information is often used to customize the Web content in a form suitable to the client. Different from the prior work that mainly adjusts high-level Internet services, our work focuses on introducing the context information into the underlying resource distribution process. One of our main objectives is to dynamically assign limited resources to real starving Internet clients, and accordingly enhance the end-user's QoE.

The ITU Telecommunication Standard (ITU-T) defines QoE as "the overall acceptability of an application or service, as perceived subjectively by the end user" [13]. The QoE is a joint consequence of the technical parameters (QoS parameters), the end-user's communication context information and the characteristics of the network service in use. Brooks *et al.* [14] propose a structured assessment approach to QoE with the following clause:

$$\begin{aligned}&\text{IF } \langle Communication\ Situation\rangle;\\&\text{USING } \langle Service\ Prescription\rangle;\\&\text{WITH } \langle Technical\ Parameters\rangle;\\&\text{THEN } \langle End\text{-}User\text{'s QoE}\rangle.\end{aligned}$$

The attributes in the bracket have many possible options: $\langle Communication\ Situation\rangle$ takes into account the objective communication context between the end-user and the service; $\langle Service\ Prescription\rangle$ can be any type of Internet services; $\langle Technical\ Parameters\rangle$ can be a range of different operating or performance parameters, such as bit rate, delay, etc., and a more complete list is given in [14]. For $\langle End\text{-}User's\ QoE\rangle$, the Opinion Score scale from 5 to 0 can be employed to describe the end-user's subjective satisfaction on the performance of the given Internet service. With such a structured assessment

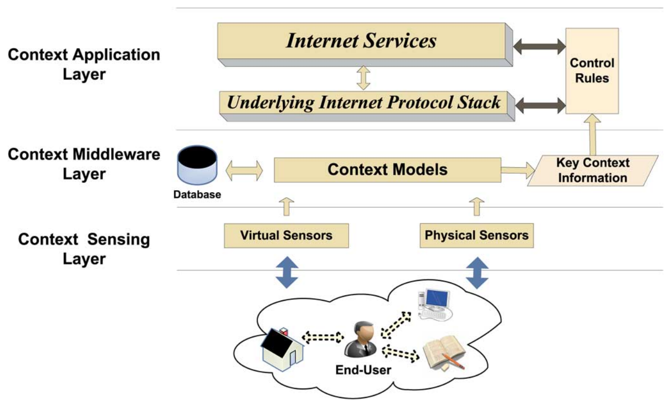


Fig. 1. The built middleware based context-aware system for the Internet.

approach, we can describe and measure the end-user's QoE in a clearer and comprehensive way.

In order to analyze the distribution process of the proposed framework, the non-cooperative game tool and the Nash equilibrium concept are used in this paper. As one of the main branch of game theory [15], the non-cooperative game theory describes the situation where each selfish player makes decisions independently and acts to maximize his own benefit. 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 non-cooperative game theory is to help design the mechanism that leads independent and selfish players towards a system-wide desirable outcome [16].

III. Key Context Information

We draw upon the design experience from the middleware based approach and implement a context-aware system capable of capturing the context information of Internet clients [17]. The built context-aware system also consists of the Context Sensing Layer, the Context Middleware Layer and the Context Application Layer, which are simply depicted in Fig. 1. The Context Sensing Layer is deployed on the client side with a number of physical sensors and virtual sensors. Physical sensors are the hardware sensors that capture the information from the physical environment, such as a Webcam and indoor location sensors, while virtual sensors collect data from the operating system and the Internet protocol stack. For example, the physical sensor Webcam has been used to track the end-user's eyes-gaze direction. The mouse movement and the keyboard inputs are monitored by the virtual sensors implemented in the operating system.

Within the Context Middleware Layer, the context inference engine, or called the context model, performs the context abstraction and reasoning task to translate basic context data into highly abstract and substantive context information, which is termed as the ***Key Context Information*** (KCI). Utilization of the KCI greatly facilitates building context-aware Internet services and disseminating the context information over the Internet. In our context-aware system, two fundamental KCI categories have been defined:

1) **User Communicating State (UCS)**: The end-user interacts with the specific Internet service and the information exchange occurs between them.
2) **User Inactive State (UIS)**: The end-user is detached from the specific Internet service and no information exchange occurs between them.

The above-defined KCI covers a wide range of interaction scenarios. For example, an Internet end-user browsing the Web page will be translated into the **User Communicating State** with the corresponding Web service; the end-user's attention is distracted away from the Web page will be translated into the **User Inactive State** with the Web service. The KCI can be deduced in real-time by the Context Models on the client side, which uses the centralized management module to locate and retrieve the information in both push and pull modes. Hence, the KCI can be served to the Context Application Layer in two ways: 1) Internet services directly query and gain the specific KCI; 2) Internet services subscribe to the specific KCI change notification service.

The Internet services and its protocol stack are located within the Context Application Layer, where they do not necessarily need to know the details of the captured raw context data but directly make use of the deduced KCI. Furthermore, a set of control rules are required to trigger the actions when the UCS or the UIS is delivered to the Context Application Layer. In principle, designing and implementing the control rules is a service-specific task, where minor changes on the original service and its underlying communication protocols may be required.

The defined KCI, i.e., the UCS and the UIS, belongs to the parameter set of the $\langle Communication\ Situation\rangle$ attribute for assessing the QoE. Specifically, given $\langle Communication\ Situation\rangle$ is the UIS, $\langle End\text{-}User's\ QoE\rangle$ can only be set to 0 regardless of the values in the other attributes. The main reason is that the UIS implies no interaction between the end-user and the Internet service, and hence the service performance as well as the assigned resource has little influence on the end-user's subjective satisfaction. For example, when the end-user is in the UIS with the YouTube service, the corresponding end-user's QoE would be always 0 no matter how much uplink bandwidth on the server side is allocated to this client. Based on such fact, the *QoE-aware resource distribution* should provide higher priority in allocating any limited resource to the Internet clients in the UCS, and thus enhance their end-user's QoE.

Base on the two categories of the KCI deduced by the built context-aware system, we propose a novel QoE-aware resource distribution framework incentivizing context sharing and moderate competition.

IV. A Resource Distribution Framework

The proposed resource distribution framework consists of a basic three-step workflow, and two algorithms running in the first step and the third step, respectively. In this section, we first present the workflow, and sequentially describe the design issues of the two required algorithms called the Willingness Update Algorithm (WUA) and the Resource Distribution Algorithm (RDA). Based on the deduced KCI, i.e., the UCS and the UIS, we design a conceptually simple and concrete WUA and RDA respectively. Finally, we provide a theoretical analysis of the proposed framework with the designed algorithms.

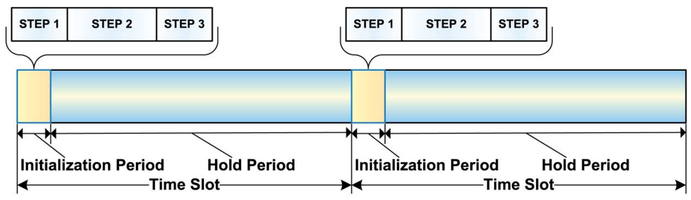


Fig. 2. Time slot divided into the initialization period and the hold period.

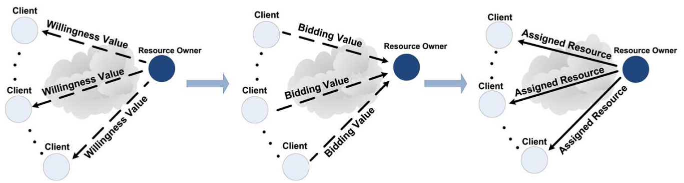


Fig. 3. Three steps of the resource distribution framework workflow.

A. 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 different types, such as worker thread, bandwidth or CPU time. A finite set of Internet clients, denoted by $P_i, i \in I, I = \{1, 2, \ldots, N\}$, compete for the given limited resource. All Internet clients transfer and update their latest KCI to the resource owner through interoperable communication protocols or mechanisms, e.g., XML Protocol (XMLP) [18] or JAVA RMI (Remote Method Invocation) [19]. The resource owner maintains a database to store and manage the delivered KCI with the timestamp of its recent update. Since the clients only need to update their newly changed KCI to the resource owner, synchronization between the Internet clients and the resource owner is not required. On the resource owner side, the time domain is divided into fixed-size time slots, denoted as T_j, $j \in \{1, 2, \ldots, +\infty\}$. As shown in Fig. 2, each individual time slot is further divided into two parts: Initialization Period and Hold Period. The resource distribution process only occurs in the Initialization Period, while its results effect the entire subsequent Hold Period. The Initialization Period should only occupy a small portion of the time slot length, e.g., 5% to 10%. Within each Initialization Period, the interaction steps between the resource owner and the Internet clients, i.e., the basic workflow of the resource distribution framework, can be described as follows:

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 the 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.
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.
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.

The above three-step procedure describes the basic workflow of the resource distribution framework, which is illustrated in Fig. 3. STEP 1 and STEP 3 of the framework workflow require the *WUA* and the *RDA*, which will be discussed in the following subsections, respectively. STEP 2 requires a proper bidding strategy, which will be discussed in the theoretical analysis subsection.

The proposed three steps need to be sequentially executed, and each step is an independent and indispensable process. At STEP 1, the resource owner calculates the preliminary resource distribution results based only on its clients' historical and current context information, and then informs each client. At STEP 2, each client takes the initiative in requesting the limited resource. At STEP 3, the resource owner calculates the final distribution results based on the outcomes of both STEP 1 (clients' context information) and STEP 2 (clients' request behaviors).

Remark 1: If any individual client cannot timely provide its bidding value 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 [5].

B. 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 KCI of the 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.

To demonstrate the above design principles, we present a simple and practical WUA, which specifically works with the deduced UCS and UIS. For each Internet client, say P_i, we first introduce a new variable called the *duration ratio* which is defined as $q_i(\tau) = t_i^{UCS}(\tau)/t_i^{UIS}(\tau)$, where $t_i^{UCS}(\tau)$ and $t_i^{UIS}(\tau)$ are the cumulative times spent by client P_i in the UCS and the UIS 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 $q_i(\tau)$:

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

where θ is a threshold parameter that needs to be specified by the resource owner. Classes C_1 and C_3 include all the clients currently in the UCS, while C_1 requires a small $q_i(\tau)$, i.e., a high proportion of the UIS duration over the previous τ time slots. Classes C_2 and C_4 involve all the clients currently in the UIS, while C_4 requires that its members keep staying in the UIS during the previous τ time slots. In principle, class C_1 has the highest priority among all classes. In other words, a client currently in the UCS would receive large willingness value from the WUA, given that it spent enough time in the UIS 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 UCS 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 classification essentially facilitates the resource owner incentivizing both the UCS update and the UIS update from the Internet clients.

By leveraging on the above described four classes, we present a conceptually simple implementation of WUA based on the lottery scheduling [20]. The lottery scheduling is a simple randomized allocation mechanism: the allocation rights are represented by lottery *tickets* that are distributed among the participates. 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.

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 $\rho = \eta_r / K_N$. 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 $\sqrt{(1-\rho)/\mu_l \rho}$, 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.

Algorithm 1 Willingness Update Algorithm (WUA)

Input: C_r, N_r and η_r, $r \in \{1, 2, 3, 4\}$, μ, 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:    ifthe player $P_i$ has the ticket $\lambda$ then
8:        $w_i(T_j) = w_i(T_j) + 1$;
9: $w_i(T_j) = w_i(T_j)/l, i \in I$;
```

As a non-deterministic algorithm, the proposed WUA allocates resources in an unpredictable order by holding lotteries, and thus each client in the same class is given the same chance of winning the willingness values. Moreover, the proposed WUA inherently allows multiple outcomes, some of which may have a low probability but still possibly occur. For example, when resources are extremely scarce, the clients in classes C_2, C_3 and C_4 still have some chances of winning their own willingness values. Lastly, the algorithm is conceptually simple to demonstrate the WUA design principles, and requires no more special operations and configurations to trivially support dynamic system environments.

Remark 3: 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 UIS, and thus they would be grouped into class C_1 in their first time slot T_j.

Remark 4: Besides the lottery scheduling algorithm, many deterministic algorithms, e.g., Deterministic Stride Scheduling [20], can also be used to design a new WUA, which may provide a better accuracy, consistent behavior and easier implementation.

Remark 5: The threshold parameter θ in the proposed WUA is used to restrict the number of clients classified into class C_1. Note that an overly large value of theta would lead all clients in the UCS to be assigned into class C_1, while an unduly low value of theta would result all clients in the UCS to be assigned into class C_3. The value of θ is greatly affected by the duration distribution of the UCS clients, and we will further illustrate it in Section V.

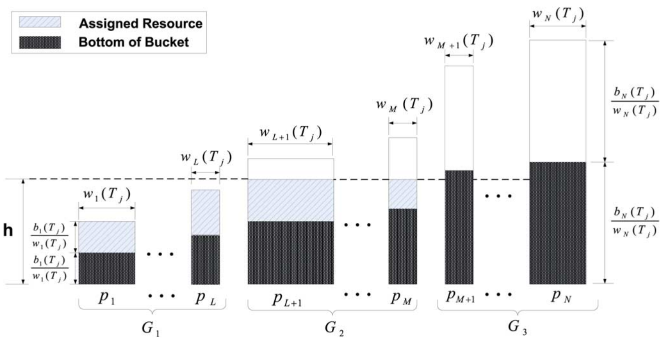


Fig. 4. Three bucket groups in the Resource Distribution Algorithm.

C. 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:

- The 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 the 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 the Internet clients are given, we present a practical RDA based on the so-called water filling algorithms [21], [22]. 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. 4. Each bucket has a bottom thickness $b_i(T_j)/w_i(T_j)$, and accordingly its total height amounts to $2b_i(T_j)/w_i(T_j)$. 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 $G_1 = \{P_1, \ldots, P_L\}$, $G_2 = \{P_{L+1}, \ldots, P_M\}$ and $G_3 = \{P_{M+1}, \ldots, P_N\}$, $1 \le L \le M \le N$. 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. 4 illustrates the general distribution result of the RDA.

The RDA can be expressed by the pseudo-code in Algorithm 2, which consists of three routines. Routine 1 and Routine 2 pick out the clients in group G_1 and group G_3 respectively, and Routine 3 calculates the final resource distribution results. 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

Algorithm 2 Resource Distribution Algorithm (RDA)

```
1: Input: $\mu$, $b_i$ and $w_i$, $\forall i \in I$ for the current time slot.
2: Output: Resource distribution results $x_k, \forall k \in I$.
3: Init: $low = 2b_1/w_1$, $high = b_N/w_N$, $b_0 = 0$, and sort all clients in ascending $b_i/w_i$ order, denoted by $\{b_1/w_1, b_2/w_2, \ldots, b_N/w_N\}$.
4: Routine 1: $/*\ pick\ out\ all\ clients\ in\ G_1 */$
5: for$k = 1 \to N$ do
6:   $\alpha = \sum_{i=1}^{i=k} b_i$;
7:   $j = k + 1$;
8:   while $low > b_j/w_j$ do
9:     $\alpha += (low * w_j - b_j)$;
10:    $j++$;
11:  if $\alpha \le \mu$ then
12:    $L = k$; $/*\ assigned\ P_k\ to\ G_1 */$
13:    $low = 2b_{k+1}/w_{k+1}$, $\alpha = 0$;
14:  else $\{\alpha > \mu\}$
15:    $\mu -= \sum_{i=0}^{i=k-1} b_i$;
16:    $\mu' = \mu$, exit for;
17: Routine 2: $/*\ pick\ out\ all\ clients\ in\ G_3 */$
18: for $k = N \to L + 1$ do
19:   if $high \ge 2b_{L+1}/w_{L+1}$ then
20:     $high = b_{k-1}/w_{k-1}$; $/*\ assigned\ P_k\ to\ G_3 */$
21:   else if $high < 2b_{L+1}/w_{L+1}$ then
22:     $\beta = \sum_{j=L+1}^{k} (high\ w_j - b_j)$;
23:     if $\beta \ge \mu$ then
24:       $high = b_{k-1}/w_{k-1}$; $/*\ assigned\ P_k\ to\ G_3 */$
25:     else $\{\beta < \mu\}$
26:       $M = k$;
27:       exit for;
28: Routine 3: $/*\ calculate\ final\ distribution\ results */$
29: $h = (b_M/w_M) + (\mu' - \sum_{i=L+1}^{M-1}((b_M/w_M) * w_i - b_i))/\sum_{i=L+1}^{M} w_i$; $/*\ final\ height\ in\ G_2 */$
30: $x_k(T_j) = \begin{cases} b_k(T_j) & \forall k \in [1, L]; \\ w_k(T_j) * h - b_k(T_j) & \forall k \in [L+1, M]; \\ 0 & \forall k \in [M+1, N]. \end{cases}$
```

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 taller bucket; otherwise the RDA records 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 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 shorter bucket; otherwise the RDA ends. Because Routine 1 and Routine 2 have picked out all the members of groups G_1 and G_3, the remaining buckets would be automatically assigned to group G_2. Thus, Routine 3 first calculates the final height h in group G_2 and then makes the final distribution.

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., $x_i(T_j) = b_i(T_j), \forall i \in I$.

Remark 6: The classic water-filling algorithm, as well as the lottery scheduling, had been introduced and utilized in different systems and areas: the mechanism for P2P networks in [23] is designed based on the same water-filling algorithm, and accordingly exhibits some similar system properties as ours. However, our algorithm design principles, algorithm implementations, context-driven design objectives, system performance gains and the framework applicable areas are all distinct from the previous research work.

D. Theoretical Analysis and Discussion

In the previous subsections, 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, a bidding strategy needs to be independently determined by the individual Internet client. In this subsection, we demonstrate that the Internet clients are motivated to actively share their actual Key Context Knowledge and moderately compete with a theoretical analysis. In addition, we also prove that the distribution results of the framework always maximize a particular form of the social welfare function.

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 the 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 the 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 UCS or the UIS, and meanwhile adopt a moderate bidding strategy to compete for the limited resource.

Proof: As mentioned earlier, in general, all the 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 UIS 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 UIS and not starving, the best policy is not to hide the UIS but to quickly update it to the resource owner. As a result, when its UCS 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 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 the Internet client 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 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, the proposed resource distribution framework effectively motivates context sharing and moderate competition among Internet clients. ■

Lemma 3: Under the proposed framework with the given WUA (Algorithm 1) and RDA (Algorithm 2), the distribution results $X = \{x_i(T_j) : \forall i \in I\}$ solves the following optimization problem:

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

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 (Algorithm 1) and RDA (Algorithm 2), the resource distribution results maximize a specific 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 $x_i(T_j)/b_i(T_j)$, 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 $\prod_{i=1}^{N}((x_i(T_j)/b_i(T_j))+1)^{w_i(T_j)}$ to simply describe the social welfare in time slot T_j, which considers the satisfactions from both the resource owner and all the 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 (1). Lemma 3 has proven that the resource distribution result X in any given time slot solves optimization problem (1). Hence, the resource distribution results always maximize the selected social welfare function in all time slots. ■

Remark 7: Many other functions, which consider satisfactions of the resource owner and Internet clients, can also be used to model the social welfare. The given distribution results of the proposed framework may not be their optimal solution.

V. 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 the excessive competition 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 UCS and the UIS, 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:

$$b_i(T_j) = \max\{1,\ \lceil w_i(T_j)\rceil\} \tag{2}$$

where $\lceil\cdot\rceil$ is the ceiling function. Accordingly, the control rules implemented on the client side actively adjust the number of parallel HTTP connections that the Web browser sends to the Web server. Given that $x_i^r(T_{j-1})$ is the number of established HTTP persistent connections between the Web browser and the Web server over the previous time slot T_{j-1}, the following control rules can be implemented on the Web browser of client P_i based on the bidding strategy (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 $b_i(T_j) \le x_i^r(T_{j-1})$, 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 $x_i(T_j) \le b_i(T_j), \forall i \in I$, and thus it is unnecessary to consider the case $\beta_i(T_j) < 0$ in the above control rules.

Remark 8: 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 (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 UCS and the UIS, which follows a similar state transition model given in [4]: the UCS duration is exponentially distributed with the mean value of 20 seconds, and the UIS duration is exponentially distributed with the mean value of 62.5 seconds. During the UCS period, each Web client makes sequential HTTP requests following a homogeneous Poisson process with a rate of 30 requests per minute. In the UIS period, each Web client stops generating HTTP requests and keeps silent.

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 ms. 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

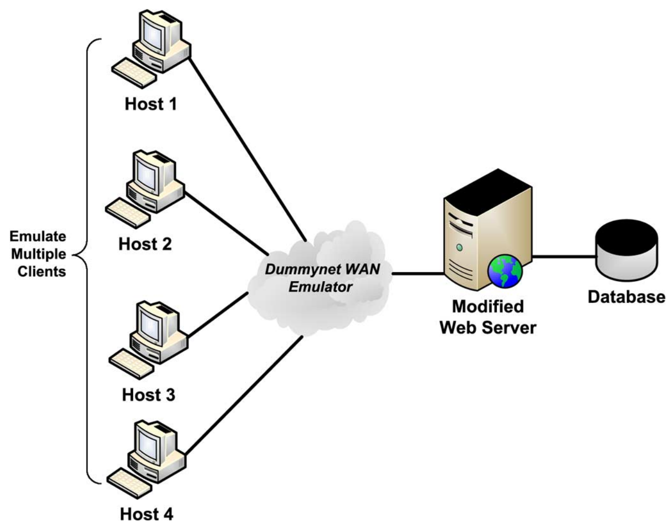


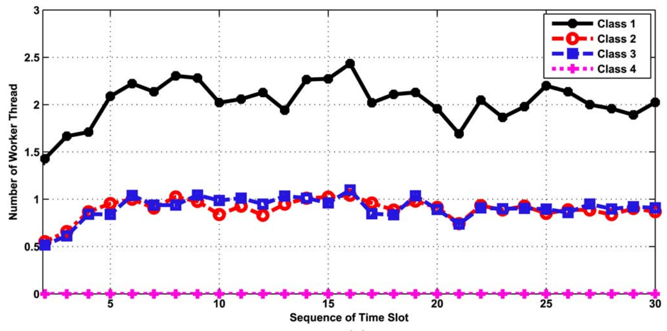
Fig. 5. Network topology in the experiment.

$t_i^{UIS}(\tau)$ to avoid a zero denominator when calculating the duration ratio $q_i(\tau)$.

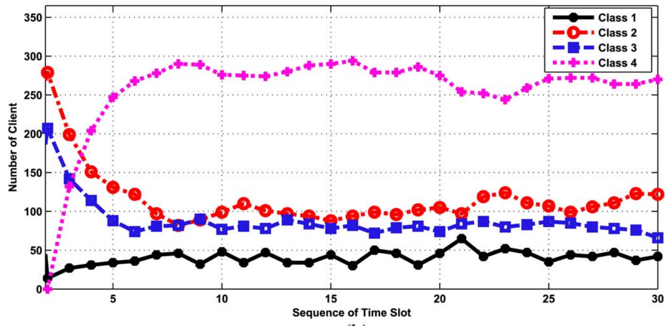
Our experimental hardware setup involves five hosts equipped with Duo Intel T7300 2.00 GHz processors and 2-GB RAM. One host runs the modified Apache HTTP Server, and others run the HTTP request generator to act as multiple Web clients. We adopt Dummynet [24], a widely used tool to enforce queue delay and packet loss, to emulate operating in a wide area network (WAN) environment. We enable its delay function and set it to 50 ms for both direction of each link. Thus, the round trip time (RTT) is around 150 ms, which includes the database access time. The experiment topology is depicted in Fig. 5. 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 (QoE-Aware Service Differentiation): We first consider 500 clients competing for the 256 worker threads. All clients share their actual KCI including the negative UIS, and adopt the moderate bidding strategy (2). Fig. 6(a) illustrates the average number of worker threads finally assigned to the individual client in the four classes categorized by the WUA. Fig. 6(b) illustrates the number of the clients in each class. Fig. 7 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 almost half of the clients waiting in the pending connection queue or being simply blocked. Under the proposed resource distribution framework, Fig. 6(a) shows that the Web clients who are currently in the UCS and hold high proportion of the UIS duration over the previous two time slots, i.e., the members of class C_1, obtain around two worker threads on average from the Web server. The Web clients who actively transit from the UCS to the UIS in the previous two time slots, i.e., the members of class C_2, can also obtain one worker threads on average. Meanwhile, the clients who are currently in the UCS but hold a low proportion of the UIS duration, i.e., the members of class C_3, also obtain nearly one worker threads, since the same number of tickets are assigned to classes C_2 and C_3 in



(a)



(b)

Fig. 6. Service differentiation under the resource distribution framework. (a) Average number of worker threads allocated to individual client in different classes; (b) number of the clients in the four classes.

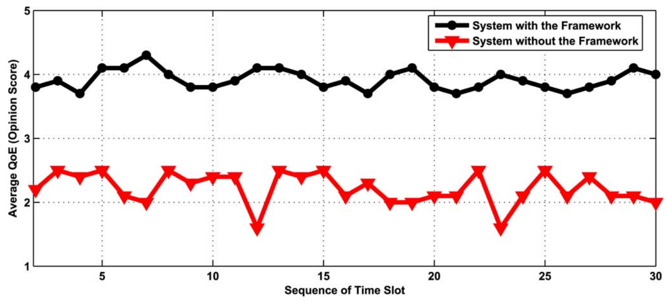


Fig. 7. A comparison of the average end-user's QoE on Web browsing.

the WUA. The Web clients who have stayed in the UIS for the previous two 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 [25], [26] have systematically investigated the quantitative relationship between the QoE and the Web page download time. Shaikh *et al.* [25] shows that the exponential relationship gives the best correlation result: $\text{QoE} = 4.836 * \exp(-0.15T)$, where T is the 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 UCS. Otherwise, the QoE would be always 0 no matter how much resource is allocated to the clients in the UIS. Hence, in each time slot, we simply compute the average download time of all the Web clients in the UCS, 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. 7 depicts the average end-user's QoE in both cases: we clearly see that the

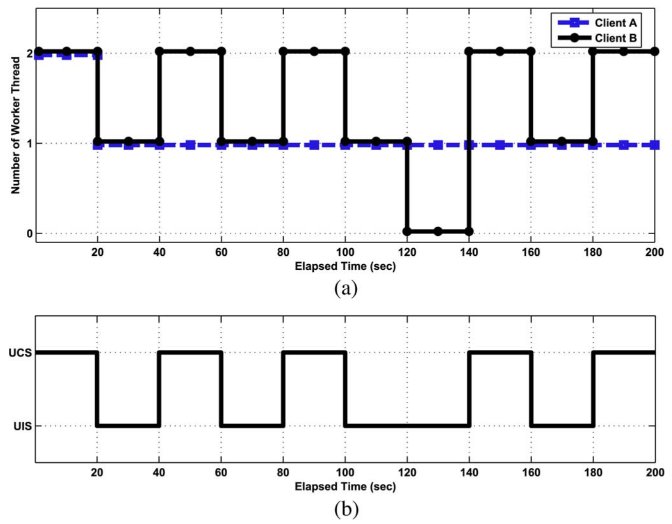


Fig. 8. A comparison between the honest client and the dishonest client. (a) Amount of the resource allocated to client A and client B; (b) KCI transitions of client A and client B.

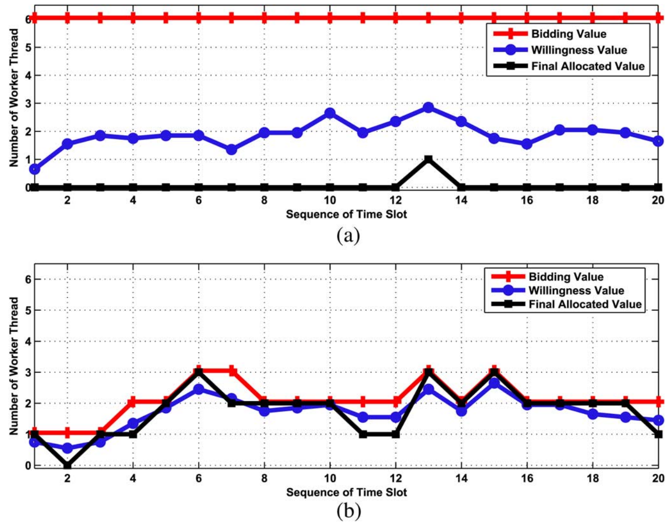


Fig. 9. A comparison between the aggressive client and the moderate client. (a) Client A (aggressive); (b) Client B (moderate).

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 a large queueing delay for the starving clients. Moreover, the Web page transmission time can also be reduced when multiple worker threads, e.g., two 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 QoE-aware 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 the time slots as shown in Fig. 8(b). Client A purposely never updates its UIS to the resource owner but fraudulently informs the resource owner the UCS. Client B honestly updates its KCI transitions to the resource owner. Both of them adopt the moderate bidding strategy (2). Fig. 8(a) demonstrates the final resource distribution results of the two clients.

As shown in Fig. 8, during the 1st and the 2nd time slots, both client A and client B are allocated two worker threads, because the WUA groups the new clients into class C_1. Both client A and client B transit from the UCS to the UIS before 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 one worker thread over the next two time slots. Then both clients switch back to the UCS before the 5th time slot comes, and client B also timely provides updates to the resource owner. Because client B shares its UIS 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 two worker threads during its UCS 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 UIS 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 UCS, which is due to client B actively sharing its negative UIS with the resource owner. Hence, the framework effectively encourages the clients to provide their actual KCI.

Note that during the 13th and the 14th 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 two 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 similar willingness values from the resource owner. Client A adopts an aggressive policy that keeps sending six parallel HTTP persistent connection requests to the Web server, i.e., bidding six worker threads. Client B adopts the given moderate bidding strategy (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. 9(a) and (b), respectively.

As shown in Fig. 9(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, the 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. 9(b). From the final distribution results of the two comparable clients, we see that the framework guarantees the moderate

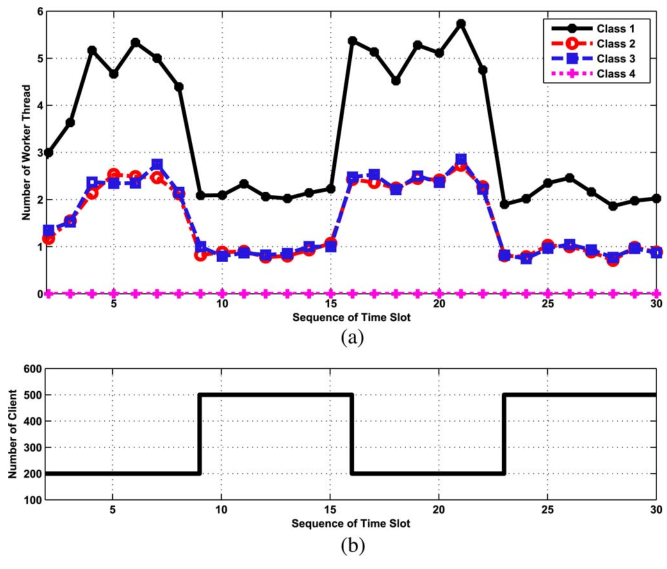


Fig. 10. Framework adaptivity in terms of the total client number. (a) Average number of worker threads allocated to individual client in different classes; (b) fluctuations in the total client number.

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 the 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. 10(b), only 200 clients compete during the first eight time slots. From the 9th time slot onwards, the total client number dramatically increases to 500 and holds for the subsequent eight time slots. From the 16th time slot onwards, the total client number drops back to 200 for another eight time slots and then increases to 500 again.

Fig. 10(a) illustrates the system performance under such dynamic conditions. We see that the framework can still effectively provide the QoE-aware service differentiation: when the total number of clients is 200, the individual client in class C_1 receives around five worker threads on average, and the individual client in classes C_2 or C_3 obtains more than two 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.

Remark 9: The additional stress tests shows that the dedicated time slots in the proposed framework have little negative impact on the system performance, given that time constraints are imposed on the Initialization Period and the resource owner re-allocates resources in a timely manner.

Remark 10: The threshold parameter θ used in our Web system is determined through the following process: First, we select 0.8 as the expected ratio of the client number in class C_1 to class C_3, namely 4/9 of the UCS clients are expected to be assigned into class C_1. Second, given the exponential distribution of the UCS duration with the mean value 20 seconds, we obtain that the probability of a client staying in the UCS shorter than 16 seconds is 44.9% (approximately equal to 4/9). Third, we have 4.0 as the initial value of θ by using $16/(T * \tau - 16)$, where the time slot length T and the parameter τ are 10 seconds and 2 seconds respectively in our Web system. Fourth, we further fine-tune θ after the system has been deployed and adjust θ to 3.0, as it achieves the expected UCS client ratio and high average QoE Opinion Score.

VI. Conclusion

Motivated by the availability of the Internet client's context information, we present a novel resource distribution framework that provides QoE-aware service differentiation, and meanwhile incentivizes context-sharing and moderate competition. By leveraging on the selfish but rational nature of Internet clients, the proposed framework efficiently allocates limited resources to the real starving end-users. The framework consists of a three-step workflow as well as the required WUA and RDA algorithms. The resource distribution process is modeled as a non-cooperative game, which guides all clients towards the unique system Nash equilibrium. We further prove that under the framework, the best policy for any Internet clients is to provide their actual KCI and self-enforce moderate competition policies. Moreover, we show that the final resource distribution results maximize a particular form of social welfare function. As an illustrative example, the proposed framework is implemented on a World Wide Web system, and the experimental results confirm that all of the original design goals are achieved.

These contributions lay a solid foundation for future work:

- Besides the Web system and the streaming media system, the proposed resource framework can explore its applications in other Internet systems and services and manage new types of limited resources.
- With rapid advancements in context-aware computing, more significant and applicable KCI of Internet clients would be captured or deduced. Accordingly, the framework can be further developed to utilize new KCI to manage underlying resources.
- Methods to describe and establish a generic relationship between the end-user's QoE and the amount of allocated resource require further study.

Our work reveals the fact that proper utilization of the valuable Internet client's context information can improve different Internet system's performance and enhance the corresponding end-user's QoE.

Appendix A
Proof of Lemma 1, 2 and 3

Lemma 1: Under the proposed resource distribution 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 $\sum_{i=1}^{N} w_i(T_j) = \mu$, i.e., the sum of the assigned willingness values equals to the total amount of resource. Hence, we have

$$w_c(T_j) = \frac{w_c(T_j)}{\sum\limits_{i=1}^{N} w_i(T_j)} * \mu. \tag{3}$$

In STEP 3, the given RDA classifies all clients into three groups $G_1 = \{P_1, \ldots, P_L\}, G_2 = \{P_{L+1}, \ldots, P_M\}$ and $G_3 = \{P_{M+1}, \ldots, P_N\}$, where $1 \le L \le M \le N$, and guarantees that all the members in group G_2 reach the same ***final height*** h. Consider P_L and P_M are the last member of groups G_1 and G_2 respectively, we have

$$\begin{cases} \frac{2b_L}{w_L} \le h < \frac{2b_{L+1}}{w_{L+1}} \\ \frac{2b_M}{w_M} \le 2h < \frac{2b_{M+1}}{w_{M+1}} \end{cases} \tag{4}$$

where the time expression T_j can be omitted within any individual time slot. Consider all the clients of G_1 receiving their bidding amount and (4), the amount of the resource assigned to group G_1, denoted by μ_1, satisfies

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

Similarly, 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 the clients satisfies

$$\mu_1 + \mu_2 + \mu_3 < \sum_{i=1}^{M} \frac{h}{2} * w_i. \tag{5}$$

Because only the clients in group G_1 receive their bidding amount of resource, we need to prove that the given client P_c must be assigned to group G_1 of the RDA. Apparently, there are two possible cases that P_c is not in group G_1:

1) Client P_c is assigned to group G_2: consider $b_c = w_c$ as well as (3) and (4), we have $h < 2\mu / \sum_{i=1}^{N} w_i$. Together with (5), we get

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

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

2) Client P_c is assigned to group G_3: consider $b_c = w_c$ as well as (3) and (4), we have $h < \mu / \sum_{i=1}^{N} w_i$. Together with (5), we get

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

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

Therefore, any client, who bids the given willingness value, can only be assigned to group G_1, 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 strategy profile B^*, we have

$$\sum_{c=1}^{N} b_c = \sum_{c=1}^{N} x_c = \mu \tag{6}$$

where the time slot expression T_j is omitted. Eqn. (6) shows that the resource is just used up and all clients are assigned to group G_1. 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 initial bidding strategy. Therefore, the strategy profile B^* is one pure-strategy Nash equilibrium of the competition game in time slot T_j.

Next, we 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 $\tilde{B}$ and the result $\tilde{X}$ must satisfy the condition

$$\begin{cases} \tilde{b}_c \ge b_c^* = w_c & \forall c \in I; \\ \tilde{x}_c = x_c^* = w_c & \forall c \in I. \end{cases} \tag{7}$$

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 $\tilde{b}_c = w_c$. Lemma 1 guarantees the new result $\tilde{x}_c = w_c$.

Eqn. (7) indicates 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. 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, e.g. P_1, can also improve its payoff from x_1^* to $x_1^* + \delta$ by unilaterally increasing its bidding value from b_1^* to $b_1^* + \delta$, given δ satisfying

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

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

$$\frac{x_1^* + \tilde{b}_1}{w_1} = \frac{x_2^* + \tilde{b}_2}{w_2} = \cdots = \frac{x_N^* + \tilde{b}_N}{w_N} = h' \quad (8)$$

where h' is the final height in group G_2. Equation (8) 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 the third possible case.

Because no pure-strategy Nash equilibrium exists in all possible cases, the strategy profile $B^*(T_j)$ is the unique pure-strategy Nash equilibrium within any individual time slot $T_j, j \in [1, 2, \ldots, +\infty]$. ■

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:

$$\max \prod_{i=1}^{N} \left(\frac{x_i(T_j)}{b_i(T_j)} + 1 \right)^{w_i(T_j)}$$
$$\textit{subject to} \quad 0 \le x_i(T_j) \le b_i(T_j), \forall i \in I,$$
$$\sum_{i=1}^{N} x_i(T_j) \le \mu \quad (9)$$

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

$$\min \quad -\sum_{i=1}^{N} \log(x_i + b_i)^{w_i}$$
$$\text{subject to} \quad \sum_{i=1}^{N} x_i - \mu \le 0,$$
$$x_i - b_i \le 0, \forall i \in I,$$
$$-x_i \le 0, \forall i \in I,$$

where 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. Hence, the Karush-Kuhn-Tucker (KKT) conditions are necessary and sufficient conditions. To prove X solves the original optimization problem, if and only if X satisfies the following KKT conditions:

$$\begin{cases} \sum_{i=1}^{N} x_i - \mu \le 0; \\ x_i - b_i \le 0, \quad i = 1, 2, \ldots, N; \\ -x_i \le 0, \quad i = 1, 2, \ldots, N; \\ \lambda_i \ge 0, \quad i = 0, 1, 2, \ldots, 2N; \\ \lambda_0 * \left(\sum_{i=1}^{N} x_i - \mu \right) = 0; \\ \lambda_i * (x_i - b_i) = 0, \quad i = 1, 2, \ldots, N; \\ \lambda_{i+N} * (-x_i) = 0, \quad i = 1, 2, \ldots, N; \\ \nabla \left(-\sum_{i=1}^{N} \log(x_i + b_i)^{w_i} \right) + \lambda_0 \nabla \left(\sum_{i=1}^{N} x_i - \mu \right) \\ + \sum_{i=1}^{N} \lambda_i \nabla (x_i - b_i) + \sum_{i=1}^{N} \lambda_{i+N} \nabla(-x_i) = 0, \end{cases}$$

where $\lambda_i, \forall i \in I$ are the Lagrange multipliers. For any Internet client, i.e., $\forall i \in I$, the KKT conditions can be further simplified to the following equivalent conditions:

$$\begin{cases} \left\{ \sum_{i=1}^{N} x_i - \mu < 0 \ \& \ \lambda_0 = 0 \right\} \ or \ \left\{ \sum_{i=1}^{N} x_i - \mu = 0 \ \& \ \lambda_0 \ge 0 \right\}; \\ \{x_i - b_i < 0 \ \& \ \lambda_i = 0\} \ or \ \{x_i - b_i = 0 \ \& \ \lambda_i \ge 0\}; \\ \{x_i > 0 \ \& \ \lambda_{i+N} = 0\} \ or \ \{x_i = 0 \ \& \ \lambda_{i+N} \ge 0\}; \\ \frac{-w_i}{x_i + b_i} + \lambda_0 + \lambda_i - \lambda_{i+N} = 0; \\ 0 \le x_i \le b_i. \end{cases} \quad (10)$$

As long as the above conditions (10) are satisfied, the resource distribution result X solves the original optimization problem. We consider two possible cases regarding the sum of bidding values.

1) $\sum_{i=1}^{N} b_i > \mu$: in this case, the framework in STEP 3 executes the given RDA, which ensures that $\sum_{i=1}^{N} x_i = \mu$. Since all clients are divided into three groups, i.e., "moderate" group G_1, "normal" group G_2 and "aggressive" group G_3, we consider the most general situation that all three groups 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, i.e., $h = (x_i + b_i)/w_i$, where $L + 1 \le i \le M$.

The "moderate" group G_1 satisfies $x_i = b_i$, and accordingly the conditions (10) require $\lambda_0 \ge 0$, $\lambda_i \ge 0$ and $\lambda_{i+N} = 0$, where $1 \le i \le L$.

The "normal" group G_2 satisfies $0 < x_i < b_i$, and accordingly the conditions (10) require $\lambda_0 \ge 0$, $\lambda_i = 0$ and $\lambda_{i+N} = 0$, where $L + 1 \le i \le M$.

The "aggressive" group G_3 satisfies $x_i = 0$, and accordingly the conditions (10) require $\lambda_0 \ge 0$, $\lambda_i = 0$ and $\lambda_{i+N} \ge 0$, where $M + 1 \le i \le N$.

Consider all the above requirements together, we have

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

The above solution satisfies the conditions (10) for any client. Therefore, the resource distribution result X solves the initial optimization problem.

2) $\sum_{i=1}^{N} b_i \le \mu$: in this case, the resource is enough for all the clients. As indicated earlier, the resource owner does not need

to execute the given RDA, but simply satisfies each client, i.e., $x_i = b_i, \forall i \in I$. Let

$$\begin{cases} \lambda_i = \frac{w_i}{2b_i}, & \forall i \in I; \\ \lambda_{i+N} = \lambda_0 = 0, & \forall i \in I. \end{cases}$$

The above solution satisfies the conditions (10) for any client in this case. Hence, the distribution result X also solves the initial optimization problem. ■

REFERENCES

[1] M. Baldauf, S. Dustdar, and F. Rosenberg, "A survey on context-aware systems," *Int. J. Ad Hoc and Ubiquitous Computing*, vol. 2, no. 4, pp. 263–277, 2007.

[2] H. L. Truong and S. Dustdar, "A survey on context-aware web service systems," *Int. J. Web Information Systems*, vol. 5, no. 1, pp. 5–31, 2009.

[3] X. Cheng and J. Liu, "NetTube: Exploring social networks for peer-to-peer short video sharing," in *Proc. IEEE INFOCOM*, Rio de Janeiro, Brazil, Apr. 2009.

[4] 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.

[5] S. Alcock and R. Nelson, "Application flow control in YouTube video streaming," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 2, pp. 25–30, 2011.

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

[7] M. Weiser, "The computer for the twenty-first century," *Scientific American*, Sep. 2002.

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

[9] H. Chen, "An intelligent broker architecture for pervasive context-aware systems," Ph.D. dissertation, Univ. Maryland, College Park, MD, USA, 2004.

[10] 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.

[11] I. Matsumura *et al.*, "Situated web service: Context-aware approach to high-speed web service communication," in *Proc. IEEE Conf. Web Services*, Chicago, IL, USA, Sep. 2006.

[12] C. D. Wang, T. Li, and L. C. Feng, "Context-aware environment-role-based access control model for web services," in *Proc. IEEE Conf. Multimedia and Ubiquitous Engineering*, Busan, Korea, Apr. 2008.

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

[14] 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.

[15] G. Owen*, Game Theory*, 3rd ed. San Diego, CA, USA: Academic, 1995.

[16] K. Akkarajitsakul, E. Hossain, D. Niyato, and D. I. Kim, "Game theoretic approaches for multiple access in wireless networks: A survey," *IEEE Commun. Surveys Tutorials*, vol. 13, no. 3, pp. 372–395, 2011.

[17] Y. Lu, M. Motani, and W. C. Wong, "When ambient intelligence meets the internet: User module framework and its applications," *Computer Networks*, vol. 56, no. 6, pp. 1763–1781, 2012.

[18] N. Mitra and Y. Lafon, Simple Object Access Protocol (SOAP), 2007 [Online]. Available: http://www.w3c.org/TR/soap

[19] A. Wollrath, R. Riggs, and J. Waldo, "A distributed object model for the Java system," Sun Microsystems, Tech. Rep., 2009.

[20] C. A. Waldspurger, "Lottery and stride scheduling: Flexible proportional-share resource management," Ph.D. dissertation, MIT, Cambridge, MA, USA, 1995.

[21] D. Bertsekas and R. Gallager*, Data Networks (2nd Edition)*. Englewood Cliffs, NJ, USA: Prentice Hall, 1992.

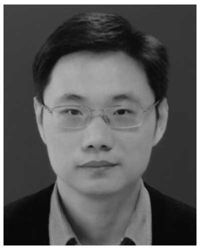
[22] J. L. Boudec, "Rate adaptation, congestion control and fairness: A tutorial," 2008 [Online]. Available: http://icapeople.epfl.ch/leboudec

[23] 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 Trans. Netw.*, vol. 14, no. 5, pp. 978–991, 2006.

[24] *Dummynet* [Online]. Available: http://info.iet.unipi.it/~luigi/dummynet/

[25] J. Shaikh, M. Fiedler, and D. Collange, "Quality of experience from user and network perspectives," *Ann. Telecommun.*, vol. 65, pp. 47–57, 2010.

[26] *Estimating End-To-End Performance in IP Networks for Data Applications*, Rec. G. 1030, ITU-T, 2005.



Yu Lu (M'11) received the Ph.D. degree in computer engineering from National University of Singapore (NUS) in 2012.

He is currently a research scientist at the Institute for Infocomm Research (I2R), A*STAR, Singapore. Previously, he was a software developer and project manager with several industrial companies and research labs. His recent research interests include architecture and protocol design for Internet, data analytics and big data, ubiquitous computing and mobile sensing.

Dr. Lu received the NGS fellowship for his Ph.D. study from NUS Graduate School for Integrative Sciences and Engineering.

Mehul Motani (M'00) received the B.E. degree from Cooper Union, New York, NY, USA, the M.S. degree from Syracuse University, Syracuse, NY, USA, and the Ph.D. degree from Cornell University, Ithaca, NY, USA, all in electrical and computer engineering.

He is currently an Associate Professor in the Electrical and Computer Engineering Department, National University of Singapore (NUS). He was a Visiting Fellow at Princeton University, Princeton, NJ, USA. He was a Research Scientist at the Institute for Infocomm Research, Singapore, for three years, and a Systems Engineer at Lockheed Martin, Syracuse, NY, USA, for over four years. His research interests are broadly in the area of wireless networks. Recently, he has been working on research problems which sit at the boundary of information theory, networking, and communications, with applications to mobile computing, underwater communications, sustainable development and societal networks.

Dr. Motani received the Intel Foundation Fellowship for his Ph.D. research, the NUS Faculty of Engineering Innovative Teaching Award, and the NUS Faculty of Engineering Teaching Honours List Award. He actively participates in the IEEE and the Association for Computing Machinery (ACM), and has served as Secretary of the IEEE Information Theory Society Board of Governors. He has been an Associate Editor for the IEEE TRANSACTIONS ON INFORMATION THEORY and an Editor for the IEEE TRANSACTIONS ON COMMUNICATIONS. He has also served on the organizing and technical program committees of several IEEE and ACM conferences.

Wai-Choong Wong (M'77–SM'93) received the B.Sc. (first class honors) and Ph.D. degrees in electronic and electrical engineering from Loughborough University, Loughborough, U.K.

He is a Professor in the Department of Electrical and Computer Engineering, National University of Singapore (NUS). He is currently Deputy Director at the Interactive and Digital Media Institute (IDMI) in NUS. He was previously Executive Director of the Institute for Infocomm Research (I2R) from November 2002 to November 2006. Since joining NUS in 1983, he served in various positions in department, faculty and university levels, including Head of the Department of Electrical and Computer Engineering from January 2008 to October 2009, Director of the NUS Computer Centre from July 2000 to November 2002, and Director of the Centre for Instructional Technology from January 1998 to June 2000. Prior to joining NUS in 1983, he was a Member of Technical Staff at AT&T Bell Laboratories, Crawford Hill Lab, from 1980 to 1983. His research interests include wireless networks and systems, multimedia networks, and source-matched transmission techniques with over 200 publications and four patents in these areas. He is a coauthor of the book *Source-Matched Mobile Communication* (IEEE Press, 1995).

Dr. Wong received the IEEE Marconi Premium Award in 1989, the NUS Teaching Award in 1989, the IEEE Millennium Award in 2000, the e-innovator Awards in 2000, and the Best Paper Award at the IEEE International Conference on Multimedia and Expo (ICME) in 2006.