# A QoS-Aware Transcoding Proxy Using On-demand Data Broadcasting

Jiun-Long Huang[†], Ming-Syan Chen[†] and Hao-Ping Hung[‡]
[†]Department of Electrical Engineering
[‡]Graduate Institute of Communication Engineering
National Taiwan University
Taipei, Taiwan, ROC
E-mail: jlhuang@arbor.ee.ntu.edu.tw, mschen@cc.ee.ntu.edu.tw, ropine@arbor.ee.ntu.edu.tw

*Abstract*— **The high diversity in the capabilities of various mobile de-vices such as display capabilities and computation power makes the design of mobile information systems more challenging. A transcoding proxy is placed between a client and an information server to coordinate the mismatch be-tween what the server provides and what the client prefers. However, most research works in transcoding proxies in mo-bile computing environments are under the traditional client-server architecture and do not employ the data broadcast technique which is has been deemed a promising technique to design a power conservation, high scalable and high band-width utilization. In addition, the issue of QoS provision is also not addressed. In view of this, we design in this paper a QoS-aware transcoding proxy by utilizing the on-demand broadcasting technique. We first propose a QoS-aware transcoding proxy architecture, abbreviated as QTP, and model it as a queueing network. By analyzing the queueing network, several theoretical results are derived. We then propose a version decision policy and a service admission control scheme to provide QoS in QTP. The derived results are used to guide the execution of the proposed version decision policy and service admission control scheme to achieve the given QoS requirement. To measure the performance of QTP, several experiments are conducted. Experimental results show that the proposed scheme is more scalable than traditional client-server systems. In addition, the proposed scheme is able to effectively control the system load to attain the desired QoS.**

## I. Introduction

The advance in wireless communication enables users to ac-cess information systems anytime, anywhere, via various mobile devices such as notebooks, tablet PCs, personal digital assistants (PDAs) and GPRS-enabled cellular phones. Service providers are establishing a number of mobile services including weather forecasting, stock information dissemination, location-dependent query, route guidance and so on. To provide such services, researchers have encountered and are endeavoring to overcome challenges in various research areas including mobile data management[4], wireless network infrastructure, location-dependent data management [19], pervasive computing, and so on.

In a pervasive computing environment [14], due to the constraints resulting from power-limited mobile devices and low-bandwidth wireless networks, designing a power conservation, high scalability and high bandwidth utilization mobile information system becomes an important research issue, and hence attracts a significant amount of research attention. In addition, the high diversity in the capabilities of various mobile devices such as display capabilities (e.g., screen size, color depth and supported data formats) and computation power makes the design of mobile information systems more challenging. This diversity also results in an increasing demand on the capability of *context awareness* [11] for mobile information systems.

*Content adaptation*, which is an important technique to realize context awareness, emerges to remedy the problem resulting from the said diversity by offering the different mobile users suitable *versions* of the same object according to the capabilities of the mobile devices, the traffic of the networks and the users' preferences [12]. *Transcoding*, which transforms a data object from one version into another, is recognized as a promising technique to realize content adaptation [12]. A proxy capable of transcoding (referred to as a transcoding proxy) is placed between a client and an information server to coordinate the mismatch between what the server provides and what the client prefers. Since proxy-based approaches are transparent to the content providers and users, this kind of approaches is able to simplify the design of servers and clients, and as a result, attracts much research attention, including cache replacement schemes [7][8], system architectures [10][15] and proxy collaboration [7].

In recent years, data broadcast [2][3] has been employed as an important technique to design a power conservation, high scalability and high bandwidth utilization mobile information system. However, most research works in transcoding proxies in mobile computing environments are under the traditional client-server architecture and do not employ the data broadcast technique. Hence, the transcoding proxies are not scalable and the network bandwidth is not well utilized. In addition, most prior studies do not consider the issue of quality of service (abbreviated as QoS) which is crucial in a mobile computing environment.

In view of this, we design in this paper a scalable and QoS-aware transcoding proxy by utilizing the on-demand broadcasting technique. Explicitly, we first propose a QoS-aware transcoding proxy architecture, abbreviated QTP, and model it as a queueing network with three queues. By analyzing the queueing network, several theoretical results are derived to formulate the average waiting time of each queue. We

0-7803-8356-7/04/$20.00 (C) 2004 IEEE

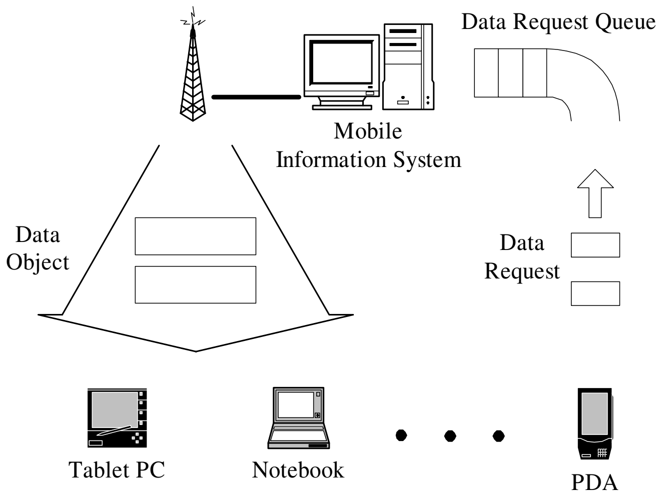


Fig. 1. An example on-demand broadcasting system

then devise scheme ODB-QoS to provide QoS in QTP where ODB-QoS stands for "On-demand Data Broadcasting with QoS". Scheme ODB-QoS is an online, iterative and adaptive algorithm comprising

1) a version decision policy to determine the suitable version for each data request according to the users' device profiles and the network state, and
2) a service admission control scheme to determine whether to grant a service registration or a service handoff according to the network state.

In each iteration, scheme ODB-QoS evaluates the average waiting time of each queue according to the derived results, determines the state of each queue according to the corresponding obtained average waiting time, and configures the behavior of the version decision policy and the service admission control scheme according to the states of these queues to attain the desired QoS. To measure the performance of QTP, several experiments are conducted. Experimental results show that the proposed approach is more scalable than traditional client-server systems. In addition, the proposed system is able to achieve the system administrators' QoS requirements by the devised version decision policy and the service admission control scheme. To the best of our knowledge, there is no prior research on the design of transcoding proxies employing data broadcast. This feature distinguishes this paper from others.

The rest of this paper is organized as follows. The description of on-demand data broadcasting and the proposed transcoding proxy architecture, QTP, are given in Section II. An analytical model and a transcoding model are devised in Section III. Section IV describes the proposed scheme, and the performance evaluation is shown in Section V. Finally, Section VI concludes this paper.

## II. Preliminaries

### A. On-demand Data Broadcasting

Figure 1 shows an example on-demand broadcasting system. In an on-demand data broadcasting system, a server maintains a data request queue and serves these requests according to the employed scheduling algorithm. When requiring one data item, a mobile client sends a data request to the server. After receiving a data request, the server first checks whether there exists another data request in the data request queue with the same required data object. If yes, the new-coming data request is *merged* into the existing data request. This phenomenon is called *request merge*. Data requests with the same requested data object can be safely merged since one transmission of the data object in a broadcast channel is able to satisfy all merged data requests. Therefore, the higher the occurrence probability of request merge is, the more efficient the system is. Otherwise, the new-coming data request is *inserted* into the data request queue.

A scheduling algorithm is used to prioritize all data requests in the data request queue, and the server will serve these data requests according to their priorities. To serve a data request, the system retrieves the required data object from the corresponding data server, and then broadcasts this object to all its clients via a dedicated and shared broadcast channel. As a result, the on-demand broadcast system is more scalable and can obtain higher network utilization than traditional client-server architecture.

### B. System Architecture

Figure 2 shows the proposed architecture of QTP. In a cellular network architecture, the whole service area of a mobile environment is divided into a number of cells. Two dedicated channels, a control channel and a broadcast channel, are provided in each cell. A control channel is used to transmit control messages such as registration messages, data requests, acknowledgements, and so on. On the other hand, a broadcast channel is used by the transcoding proxy to disseminate data objects to its clients. In according to the locations of these components, QTP comprises the following two types of components: front-end and back-end.

A front-end which comprises a service manager and a scheduler is allocated to each cell. These two components are described below.

- *Service Manager:* A service manager is in charge of all service-related operations such as service registration, service termination, service admission control and so on. Each service manager owns a profile database storing the profiles of users using the service and the profiles of these users' devices. Several standards such as CC/PP (stands for Composite Capabilities/Preferences Profile) [17] and UAProf (stands for User Agent Profile) [18] have been proposed to describe the capabilities of devices and users' preferences.
- *Scheduler:* A scheduler is a software component which handles the data requests of its corresponding cell. After receiving a user's data request, the scheduler will first determine a suitable version for this data request according to the user's device profile and the network state. Then, the scheduler will check whether the received data request can be merged into an existing data request in the data request queue. Different from the traditional on-demand broadcasting architecture showing in Section II-A, request merge occurs only when there exists another

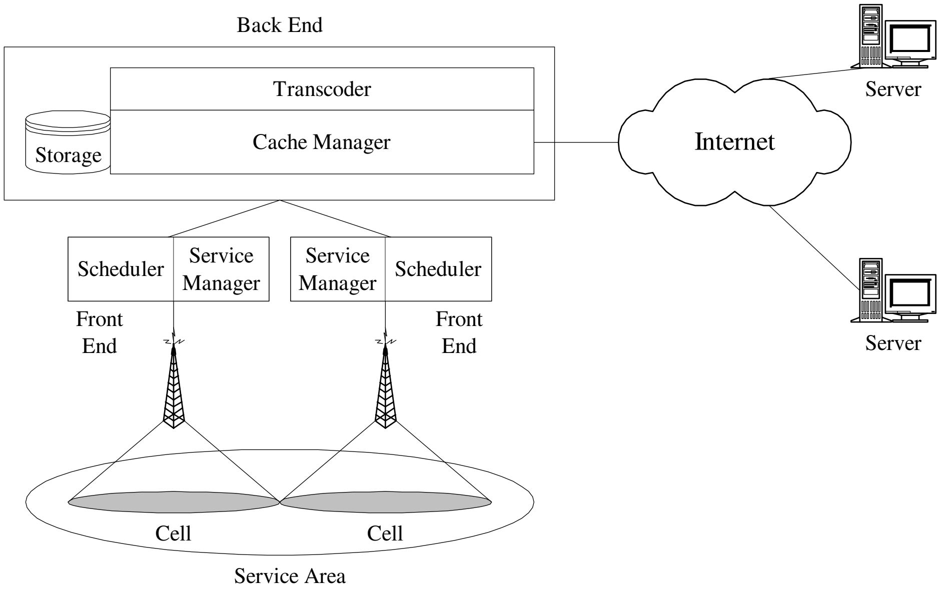


Fig. 2. The architecture of QTP

data request in the data request queue asking for the same version of the same required data object of the received data request. Otherwise, the scheduler will insert the received data request into the data request queue.
In addition, a scheduling algorithm is employed to determine the service order of these data requests in the data request queue. While serving a data request, the scheduler will send this request to the cache manager and the cache manager will respond with the content of the required data object. The scheduler then broadcasts the returned data object via the downlink channel, then serves the next data request in the data request queue.

A back-end which comprises a cache manager and a transcoder behaves like a traditional transcoding proxy.

- *Cache Manager:* After receiving a data request from a scheduler, the cache manager is responsible for returning the required version of the required data object to the scheduler. Suppose that the cache manager receives a data request of the $j$-th version of data object $D(i)$. If the $j$-th version of $D_i$ is cached, the cache manager will return the cached data object to the scheduler immediately. If the $j$-th version of $D_i$ is not cached, the cache manager will check whether there exists another version of $D_i$ which can be transcoded into the $j$-th version of $D_i$. If yes, the cache manager will ask the transcoder to generate the $j$-th version of $D_i$. Otherwise, the cache manager will request the original version of the requested data object from the data server, ask the transcoder to transform the returned data object into the required version, and then respond with the transcoded data object to the scheduler.
- *Transcoder:* A transcoder is in charge of the transformation of data objects among different versions according to the transformation requests of the cache manager.

Since the design of the back-end is similar to the systems proposed in some prior works [7][8][10][15], we focus in this paper on the design of the front-end to provide scalable and QoS-aware transcoding proxy services.

### C. Signalling Procedures

Before using the transcoding proxy, a mobile user should register the service in advance by sending a registration message via a control channel. After the transcoding proxy receives the registration message, a service admission control scheme is activated to determine whether to grant the service registration. If yes, the mobile device will send the device profile to the proxy, and the proxy will record the user profile and device profile in its profile database. Otherwise, the service registration is blocked. The rate that a service registration is blocked is called the *service blocking rate* (abbreviated as SBR).

After the service registration is granted, the mobile user can issue data requests to the corresponding transcoding proxy by the control channel. When receiving a data request, the transcoding proxy first determines the suitable version of the requested data object by a version decision policy, and returns an acknowledgement message containing the decided version information via the control channel to the mobile user. Then, the transcoding proxy will return the decided version of the required data object via the corresponding broadcast channel as soon as possible. After receiving the acknowledgement message, the mobile device will tune to the broadcast channel to wait for the appearance of the decided version of the requested data object. When the mobile user decides not to use the transcoding proxy service, the mobile device will send a de-registration message to terminate the service.

Since a mobile user is able to freely move around these cells, a service handoff will occur. A service admission control scheme is executed to determine whether the service handoff is granted. If yes, the mobile user can use the service as usual. If not, the system will force mobile user to terminate the service (we say that the call is dropped). Since a service admission control scheme is employed, a service handoff may be rejected.

| Symbol | Description |
|---|---|
| $P_i$ | $i$-th device profile |
| $D_j(k)$ | $k$-th version of data item $D_j$ |
| $N_{User}$ | Number of users in the cell |
| $\lambda_{Ctrl.}$ | Aggregate request rate in the cell |
| $\mu_{Ctrl.}$ | Service rate of the control channel |
| $\mu_{Sche.}$ | Service rate of the cache |
| $\mu_{BCast.}$ | Service rate of the broadcast channel |
| $\rho_{Sche.}$ | Standard deviation of the service time of the cache |
| $B_{Ctrl.}$ | Bandwidth of the control channel |
| $B_{BCast}$ | Bandwidth of the broadcast channel |

TABLE I
DESCRIPTION OF SYMBOLS

The rate that a service handoff is forced to terminate is called the *service dropping rate* (abbreviated as SDR).

## III. ANALYTICAL AND TRANSCODING MODELS

### A. Analytical Model

In this subsection, we derive the worst case of the average access time of QTP, and use the derived results to guide the version decision policy and the server admission control scheme proposed in Section IV. To facilitate the following discussion, we make the following assumptions.

1) The employed scheduling scheme of the scheduler is FCFS (standing for first come, first serve).
2) No request merge occurs in the data request queue of the scheduler.
3) One transmission of a data object in the broadcast channel is received by exactly one client.
4) The messages of registration, de-registration and handoff are negligible.

Assumptions 2 and 3 occur when the users' interests are highly diverse, and hence the effect of on-demand broadcast is eliminated. We make these two assumptions since we focus on the worst case of the transcoding proxy. Assumption 4 is made since we focus on the the situation that the number of data requests is much higher than the number of control messages (i.e., registration, de-registration, handoff and service termination). These assumptions will be relaxed in our simulation model described in Section V. For better readability, a list of used symbols is shown in Table I.

We model QTP as a queueing network as shown in Figure 3. Queue 2 is a physical queue which is located in the scheduler. On the contrary, Queue 1 and Queue 3 are logical queues which are only used to model the control and broadcast channels in order to derive the average waiting time of a data request on the control and broadcast channels, respectively. Suppose that the data requests submitted by a mobile user $i$ in service follow a Poisson process with rate $\lambda_i$, and $N_{User}$ is the number of mobile users in service in the cell. To facilitate the following discussion, we number the mobile users in the cell as user 1, 2, $\cdots$, $N_{User}$. Due to the characteristic of the Poisson process, the aggregate data requests of all mobile users in the cell follow a Poisson process with rate $\lambda_{Ctrl.} = \sum_{i=1}^{N_{User}} \lambda_i$. Denote the sizes of data requests and request acknowledgements as $s_{Ctrl.}$ and $s_{Ack.}$, respectively.

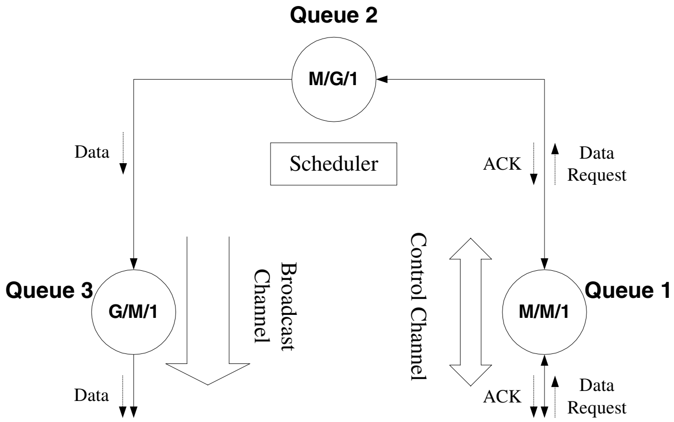


Fig. 3. The analytical model of the proposed transcoding proxy

We also let $B_{Ctrl.}$ be the bandwidth of the control channel, and let the waiting time of the control channel for a data request (denoted as $W_{Ctrl.}$) be the time interval between the user sending a data request and the user receiving the acknowledge. Then, we have the following lemma.

**Lemma 1:** The average waiting time of the control channel can be formulated as below.

$$W_{Ctrl.} = \frac{1}{\frac{B_{Ctrl.}}{s_{Ctrl.}+s_{Ack.}} - \lambda_{Ctrl.}}$$

Let the waiting time of the scheduler for a data request (denoted as $W_{Sche.}$) be the time interval from the arrival of the data request from the scheduler's perspective to the time that the requested data object has been obtained by the scheduler. Note that the service time of a cache manager is affected by several factors such as cached status of required data objects, the employed replacement scheme, the characteristic of the input jobs, and so on. The service time of the cache manager cannot be modeled by a particular mathematical distribution. Therefore, we model the average service time of the cache manager as an arbitrary distribution with mean $\frac{1}{\mu_{Sche.}}$ and variance $\sigma^2_{Sche.}$. Suppose that $\rho_{Sche.} = \frac{\lambda_{Ctrl.}}{\mu_{Sche.}}$ is the load of the scheduler. We then have the following lemma.

**Lemma 2:** The average waiting time of the scheduler is

$$W_{Sche.} = \frac{1}{\mu_{Sche.}} + \frac{\frac{\rho_{Sche.}}{\mu_{Sche.}} + \lambda_{Ctrl.}.\sigma^2_{Sche.}}{2(1-\rho_{Sche.})}.$$

Let the waiting time of the broadcast channel for a data request be the time interval between the time that the requested data object has been obtained by the scheduler and the time that the user has received it. Then, we have the following lemma.

**Lemma 3:** The average waiting time of the broadcast channel can be formulated as

$$W_{BCast} = \frac{1}{\mu_{BCast}(1-r_0)},$$

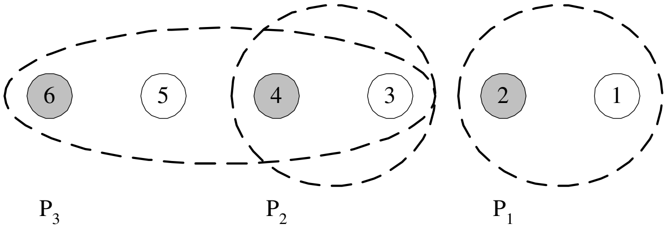


Fig. 4. Example device profiles

where $r_0$ is the root of the following equation with value larger than zero and less than one.

$$z = A^*[\mu_{BCast}(1-z)]$$

Finally, the average waiting time of the whole system (denoted as $W_{Sys.}$) is equal to the summation of the average waiting time of the control channel, the scheduler and the broadcast channel. Then, with Lemmas (1), (2) and (3), $W_{Sys.}$ can be formulated as

$$W_{Sys.} \quad = \quad W_{Ctrl.} + W_{Sche.} + W_{BCast} \qquad (1)$$

### B. Transcoding Model

Suppose that the mobile devices are classified into several categories based on the their capabilities, and the capabilities of each category are described by one device profile. Let $P_i$ be the $i$-th device profile. Without loss of generality, we order the device profiles according to their capabilities in ascendent order. That is, the capability of $P_i$ is better than that of $P_j$ when $i > j$. We also let $D_i(j)$ be the $j$-th version of data object $D_i$. Again, we order all versions of a data object according to their quality in ascendent order, which means that the quality of $D_i(j)$ is better than that of $D_i(k)$ when $j > k$. For each data objects, we assume that the data size of a version with higher quality is larger than that of another version with lower quality.

To facilitate the following discussion, the concept of viewable version set is defined below.

**Definition 1:** A *viewable version set* of a device profile $P_i$ and a data object $D_j$ (denoted as $VVS(i,j)$) is a set of versions of $D_j$ which are able to be displayed by mobile devices with profile $P_i$.
Then, we have the following example.

**Example 1:** Consider the example shown in Figure 4. Mobile devices are classified into three categories: notebook, PDA and smart phone, and their capabilities are described in device profiles $P_3$, $P_2$ and $P_1$, respectively. In addition, there are six versions of data object $D_j$. $VVS(3,j)$, $VVS(2,j)$ and $VVS(1,j)$ are $\{3,4,5,6\}$, $\{3,4\}$ and $\{1,2\}$, respectively. We have $VVS(2,j) \subset VVS(3,j)$ since devices with profile $P_3$ (e.g., notebooks) are capable of displaying all versions of $D_j$ viewable by devices with profile $P_2$ (e.g., PDAs). On the other hand, we have $VVS(3,j) \bigcap VVS(1,j) = \phi$ and $VVS(2,j) \bigcap VVS(1,j) = \phi$ since devices with profile $P_1$ (e.g., smart phone) employ special data formats (e.g., WML and WBMP) that are not supported by devices with profile $P_2$ and $P_3$.

Let the function $BEST(i,j) = k$ (respectively, $WORST(i,j) = k$) represent that the best (respectively, worst) viewed version of data object $D_j$ for a mobile device with device profile $P_i$ is version $k$. In practice, we have $BEST(i,j) \geq BEST(l,j)$ and $WORST(i,j) \geq WORST(l,j)$ when $i > l$. We also have $BEST(i,j) = \max\{VVS(i,j)\}$ and $WORST(i,j) = \min\{VVS(i,j)\}$.

**Example 2:** Consider the example shown in Figure 4. The best viewable versions of $P_3$, $P_2$ and $P_1$ are $D_j(6)$, $D_j(4)$ and $D_j(2)$, respectively. As a result, we have $BEST(3,j) = 6$, $BEST(2,j) = 4$ and $BEST(1,j) = 2$. In addition, we also have $WORST(3,j) = 3$, $WORST(2,j) = 3$ and $WORST(1,j) = 1$.

When a user registers the service, the user's mobile device will transmit the identifications the user and the corresponding device profile to the system. Suppose that the device profile of the mobile device is $P_i$. Then, when the mobile user requests $D_j$, the system will return a suitable version of $D_j$, say the $k$-th version of $D_j$ where $k \in VVS(i,j)$, according to the result of the underlying version decision policy.

## IV. Design of Scheme ODB-QoS

This section shows the design of the proposed scheme ODB-QoS (standing for On-demand Data Broadcasting with QoS). An overview of scheme ODB-QoS is given in Section IV-A. The determination of the system state is given in Section IV-B. Finally, the proposed version decision policy and admission control scheme of scheme ODB-QoS are described in Section IV-C and IV-D, respectively.

### A. Overview

We take the average access time as the QoS metric. Before executing scheme ODB-QoS, system administrators specify a QoS requirement by setting two thresholds of average access time, $W_1$ and $W_2$ where $W_1 < W_2$. The meanings of the two thresholds are as follows. The users are guaranteed to receive the best viewable versions of requested data objects when the average waiting time is smaller than $W_1$. On the other hand, scheme ODB-QoS will use its best effort to prevent the average waiting time from being larger than $W_2$.

Scheme ODB-QoS is an online, iterative and adaptive algorithm which comprises a version decision policy and a service admission control scheme. The flowchart of scheme ODB-QoS is shown in Figure 5. Scheme ODB-QoS is executed periodically, and the following three steps are executed in each iteration. First, scheme ODB-QoS measures the average waiting time of each queue according to the derived results in Section III. Then, in state determination step, scheme ODB-QoS measures the load of each queue based on the obtained average waiting time, and determines the current state of each queue accordingly. Finally, scheme ODB-QoS configures the version decision policy and the service admission control scheme according to the state of each queue. The details of scheme ODB-QoS are described in the following subsections.

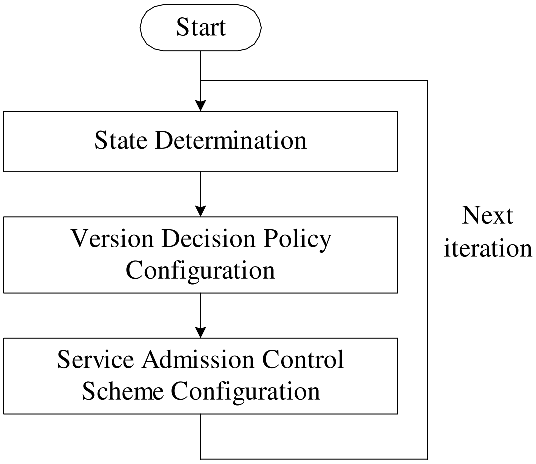

Start
State Determination
Version Decision Policy Configuration
Service Admission Control Scheme Configuration
Next iteration

Fig. 5. The flowchart of scheme ODB-QoS

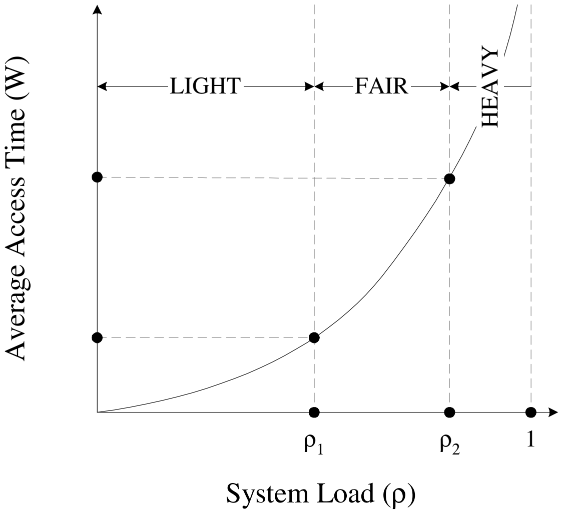


Fig. 6. The relationship between load and average access time of a queue

### B. State Determination

Three positive factors, $\gamma_1$, $\gamma_2$ and $\gamma_3$ where $\gamma_1+\gamma_2+\gamma_3 = 1$, are also defined to determine the values of $\rho_1^{Ctrl.}$, $\rho_2^{Ctrl.}$, $\rho_1^{Sche.}$, $\rho_2^{Sche.}$, $\rho_1^{BCast}$ and $\rho_2^{BCast}$. The values of $\rho_1^{Ctrl.}$ and $\rho_2^{Ctrl.}$ are first determined so that the average waiting time of the control channel is equal to $W_{Ctrl.} = \gamma_1 \times W_1$ and $W_{Ctrl.} = \gamma_1 \times W_2$, respectively. The values of $\rho_1^{Sche.}$ and $\rho_2^{Sche.}$ are then determined so that the average waiting time of the cache is equal to $\gamma_2 \times W_1$ and $\gamma_2 \times W_2$, respectively. Analogously, the values of $\rho_1^{BCast}$ and $\rho_2^{BCast}$ are determined so that the average waiting time of the cache is equal to $\gamma_3 \times W_1$ and $\gamma_3 \times W_2$, respectively.

The values of $\gamma_1$, $\gamma_2$ and $\gamma_3$ are determined adaptively and automatically. When the system starts up, $\gamma_1$, $\gamma_2$ and $\gamma_3$ are initialized to be $\frac{1}{3}$. In each execution, they are determined as follows: $\gamma_1 = \frac{W_{Ctrl.}}{W_{Sys.}}$, $\gamma_2 = \frac{W_{Cache.}}{W_{Sys.}}$ and $\gamma_3 = \frac{W_{BCast.}}{W_{Sys.}} = 1-\gamma_1-\gamma_2$. Note that in scheme ODB-QoS, only the QoS requirement (i.e., $W_1$ and $W_2$) are required to be specified by system administrators.

### C. Version Decision Policy

Figure 6 shows the relationship between the average waiting time and the load of a queue. We can observe that when the the load is larger than or equal to one, the system is not stable since the average waiting time does not converge and will approach to infinite. In addition, when the load is smaller than one, the average waiting time increases as the load increases, and the increment will increase drastically when the load approaches one.

With the above observations, the rationale of our scheduling algorithm is *to keep the system loads of the scheduler (i.e., Queue 2 in Figure 3) and broadcast channel (i.e., Queue 3 in Figure 3) smaller than one at the cost of the quality of requested data objects*. As a consequence, when the load of the scheduler or the broadcast channel is high, for each data request, the system will return the version of worse quality than the best viewed version. The strategy has the following two effects:

1) Decrease the average waiting time of the broadcast channel ($\frac{1}{\mu_{BCast}}$) since the data size of a data object with lower quality is usually smaller than that of the same data object with higher quality. The load of the broadcast channel ($\rho_{BCast}$) is hence reduced.
2) Increase the occurrence probability of request merge. Consider the device profiles shown in Figure 4, and two data requests of $D_j$ for device profiles $P_2$ and $P_3$, respectively. These two data requests will not be merged when the load of the scheduler or the broadcast channel is light. However, when the load is high, the system will decide to return the third version of $D_j$, and hence, these two data requests can be merged. The arrival rates of the input processes of the cache and the broadcast channel them decrease. As a result, this strategy will reduce the load of the cache ($\rho_{Sche.}$) and broadcast channel ($\rho_{BCast}$).

Two thresholds, $\rho_1^{Sche.}$ and $\rho_2^{Sche.}$ (respectively, $\rho_1^{BCast}$ and $\rho_2^{BCast}$), are specified first and they divide the load of the scheduler (respectively, the broadcast channel) into three states: LIGHT, FAIR and HEAVY. Figure 7 shows the state transition diagram of the scheduler. The state transition scenarios are as follows. When the previous state is LIGHT, the current state will transit to FAIR if $\rho_{Sche.} > (1+\alpha)\times\rho_1^{Sche.}$. Otherwise, the current state will still be LIGHT. When the previous state is FAIR, the current state will transit to LIGHT if $\rho_{Sche.} < (1-\alpha)\times\rho_1^{Sche.}$. If $\rho_{Sche.} > (1+\alpha)\times\rho_2^{Sche.}$, the current state will transit to HEAVY. Otherwise, the current state will still be FAIR. When the previous state is HEAVY, the current state will transit to FAIR if $\rho_{Sche.} < (1-\alpha)\times\rho_2^{Sche.}$. Otherwise, the current state will still be HEAVY. The factor $\alpha$, where $0<\alpha<1$, is set to avoid state oscillation. We assume that $(1+\alpha)\times\rho_2^{Sche.} < 1$ without loss of generality. The state transition diagram and transition scenario the broadcast channel are as shown in Figure 7 by substituting $\rho_1^{BCast}$ and $\rho_2^{BCast}$ for $\rho_1^{Sche.}$ and $\rho_2^{Sche.}$, respectively. The determination of $\rho_1^{Sche.}$, $\rho_2^{Sche.}$, $\rho_1^{BCast}$ and $\rho_2^{BCast}$ are described in Section IV-B.

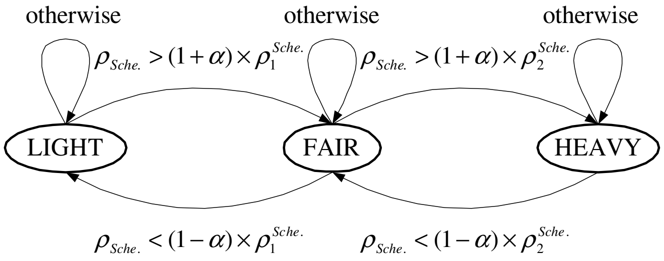

otherwise
otherwise
otherwise
$\rho_{Sche.} > (1+\alpha)\times\rho_1^{Sche.}$
$\rho_{Sche.} > (1+\alpha)\times\rho_2^{Sche.}$
LIGHT
FAIR
HEAVY
$\rho_{Sche.} < (1-\alpha)\times\rho_1^{Sche.}$
$\rho_{Sche.} < (1-\alpha)\times\rho_2^{Sche.}$

Fig. 7. State transition diagram

We also define the *aggregated state* of the scheduler and the broadcast channel as follows. The aggregated state is LIGHT when the loads of the scheduler and the broadcast channel are both LIGHT. The aggregated state is HEAVY when at least one load of the cache and broadcast channel is HEAVY. Otherwise, the aggregated state is FAIR. For each new-coming data request, the scheduler will decide a suitable version, fill a version information into the data request according to the aggregated state, and insert it into the data request queue. The scheduler will also inform the mobile client of the decided version by replying an acknowledge message. Formally, the version decision policy is described below.

- **LIGHT:** The scheduler operates in the traditional on-demand broadcast mode when the aggregated state is LIGHT. Hence, the system guarantees that each client will receive the best viewed versions of data objects being requested. That is, the system will return the $BEST(i,j)$-th version of $D_j$ when a user requests $D_j$ by a mobile device belonging to device profile $P_i$.
- **FAIR:** In FAIR state, the quality of received data object may be degraded. Let $degradation$ and $maxDegradation$ indicate the current and maximal degrees of degradation, respectively. The value of $maxDegradation$ is obtained by

  $$\max_{\forall P_k, D_j}\{BEST(k,j) - WORST(k,j)\}.$$

  When receiving a data request of $D_j$ from a mobile device belonging to device profile $P_i$, the system will return the $(BEST(i,j)-degradation)$-th version of $D_j$. When the current aggregated state is FAIR, the previous aggregated state is also taken into consideration. When the previous aggregated state is LIGHT, $degradation$ is set to be one. When the previous aggregated state is HEAVY, $degradation$ is set to be $maxDegradation-1$. Otherwise, when the previous aggregated state is also FAIR, the load of the scheduler is considered. The value of $degradation$ increases by one when $\rho_{cur}^{Sche.} > (1+\beta)\times\rho_{pre}^{Sche.}$. Similarly, the value of $degradation$ decreases by one when $\rho_{cur}^{Sche.} < (1-\beta)\times\rho_{pre}^{Sche.}$. Otherwise, the value of $degradation$ remains the same. The constant $\beta$ is used to avoid the oscillation of $degradation$.
- **HEAVY:** When the aggregated state is HEAVY, the system will force each client to receive the worst viewed versions of data objects which it requests. That is, the system will return the $WORST(i,j)$-th version of $D_j$ when a user requests $D_j$ by a mobile device belonging to device profile $P_i$.

As a consequence, the algorithmic form of the version decision procedure is as below.

```
Procedure VersionDecision($P_i$, $D_j$)
Input: A user requests $D_j$ by a mobile device belonging to device profile $P_i$.
Output: A version of $D_j$.
Let $preState$ and $curState$ ← be the previous/current state of the scheduler, respectively
Let $\rho_{pre}^{Sche.}$ and $\rho_{cur}^{Sche.}$ ← be the previous/current load of the scheduler state of the scheduler, respectively
$maxDegradation \leftarrow \max_{\forall P_k}\{BEST(k,j) - WORST(k,j)\}$
if ($curState$=LIGHT) then
    return $BEST(i,j)$ /* The system will return the best viewable version to the user */
else if ($curState$=HEAVY) then
    return $WORST(i,j)$ /* The system will return the worst viewable version to the user */
else /* $curState$=FAIR */
    if ($preState$=LIGHT) then
        $degradation \leftarrow 1$
    else if ($preState$=HEAVY) then
        $degradation \leftarrow maxDegradation - 1$
    else /* $preState$=FAIR */
        if ($\rho_{cur}^{Sche.} > (1+\beta)\times\rho_{pre}^{Sche.}$) then
            $degradation \leftarrow \min\{degradation+1, maxDegradation\}$
        else if ($\rho_{cur}^{Sche.} < (1-\beta)\times\rho_{pre}^{Sche.}$) then
            $degradation \leftarrow \max\{degradation-1, 0\}$
        end if
    end if
    return $BEST(i,j) - degradation$
end if
```

### D. Service Admission Control Scheme

A service admission control scheme is employed in each service manager to determine whether to grant a service registration or a service handoff by considering the number users in service, the network status, and so on. Analogously, the rationale of our service admission control scheme is *to keep the system load of the control channel (i.e., Queue 1 in Figure 3) smaller than one at the cost of SBR and SDR.* As shown in Section III-A, the incoming rate of data requests is in proportion to the number of users using the service in the cell. As a result, the service admission control scheme should keep the number of users in service under a reasonable level. To achieve this, two thresholds, $\rho_1^{Ctrl.}$ and $\rho_2^{Ctrl.}$ where $\rho_1^{Ctrl.} < \rho_2^{Ctrl.} < 1$, are specified first and they divide the load of the control channel into three states: LIGHT, FAIR and HEAVY. The state transition diagram and transition scenario of the service manager are shown in Figure 7 by substituting $\rho_1^{Ctrl.}$ and $\rho_2^{Ctrl.}$ for $\rho_1^{Sche.}$ and $\rho_2^{Sche.}$, respectively. The determination of $\rho_1^{Ctrl.}$ and $\rho_2^{Ctrl.}$ are described in Section IV-B.

Although the proposed version decision policy can reduce the loads of the scheduler and the broadcast channel, the effect of the proposed version decision policy is limited since it depends on several factors such as the locality of data requests, the cache size and so on. As a consequence, in addition to the load of the control channel, the service admission control scheme should also takes the loads of the scheduler and the broadcast channels into consideration. Note that SBR is sacrificed first since mobile users can tolerate a service registration being blocked rather than a service handoff being forced to terminate (i.e., dropped). The proposed admission control scheme is as below.

- A service handoff is dropped when the state of the control channel is HEAVY or when $degradation = maxDegradation$.
- A service registration is blocked when the state of the control channel is FAIR or when

$$degradation \geq \frac{maxDegradation}{2}.$$

The algorithmic form of the revised service admission control scheme is as below.

Procedure ServiceAdmission
**Input:** A service registration or a service handoff.
**Output:** Decision of the incoming service registration or service handoff.

```
$curState \leftarrow$ current state of the control channel
if (the input is a service handoff) then
    if ($curState$=HEAVY or
    $degradation = maxDegradation$) then
        return REJECT
    else
        return GRANT
    end if
else /* the input is a service registration */
    if ($curState$=FAIR or
    $degradation \geq= \frac{maxDegradation}{2}$) then
        return REJECT
    else
        return GRANT
    end if
end if
```

## V. Performance Evaluation

To evaluate the performance of the scheme ODB-QoS, we build an event-driven simulator with SIM [5]. Scheme ODB-QoS is executed periodically with period two minutes and the simulation is run for 12 hours. Scheme CS (standing for traditional Client-Server) and scheme ODB (standing for On-Demand Broadcasting) are also implemented for comparison purposes. The average waiting time is employed as the performance measurement for each scheme. In addition, SBR and SDR are taken as the measurement of the cost of scheme ODB-QoS. The simulator is coded in C++ and run in a PC with Pentium III 400 MHz CPU and 256 MBytes RAM.

### *A. Simulation Model*

| Parameter | Value |
|---|---|
| Data object number | 4000 |
| Data object sizes | Lognormal dist. (mean 18 KB) |
| Data access probabilities | Zipf dist. with parameter 1.1 |
| Cache replacement scheme | AE |
| Cache capacity | $0.01 \times \sum$ object size |
| Object fetch delay | Exponential dist. with $\mu = 2.3$ |
| Transcoding rate | 30 KB/sec |
| Client number | 1000 |
| Cell residence time | Exp. dist. with $\mu = 40$ minutes |
| Cell holding time | Exp. dist. with $\mu = 15$ minutes |
| Cell establishing time | Exp. dist. with $\mu =$ one hour |

TABLE II
Default system parameters

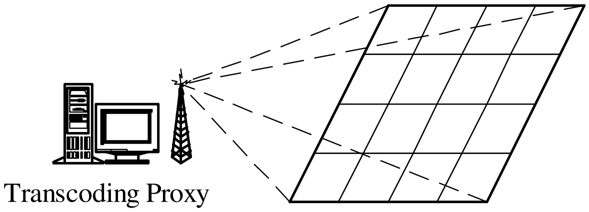


Fig. 8. The simulation topology

Similar to [16], we set the cell topology as a 4×4 cells wrapped-around mesh topology shown in Figure 8. We take LWF (standing for Longest Wait First) as the underlying scheduling algorithm. Scheme AE [8] is employed as the cache replacement policy since it outperforms the other replacement policies for transcoding proxies. Each cell provides one control channel and one download channel with network bandwidth 10 KByte/sec and 100 KByte/sec, respectively. Analogously to [8], we assume that there are 4000 data objects and the sizes follow a lognormal distribution with a mean of 15 KBytes. The size of a control message (e.g., data request message and acknowledgement message) is set to be 1 KByte. The access probability of data objects follows a Zipf distribution, which is widely adopted as a model for real Web traces [1][6]. The parameter of the Zipf distribution is set to be 1.1 with a reference to the analyses of real Web traces [6][13]. Since small objects are much more frequently accessed than large ones [9], we assume that there is a negative correlation between the object size and its access probability. The default capacity of the cache is set to be $0.01 \times \sum$ object size and the fetch delays of data objects follow an exponential distribution with mean 2 seconds [8]. The values of $W_1$ and $W_2$ (i.e., the QoS requirement) are set to be two seconds and five seconds, respectively.

In the client model, as in [7] and [8], we assume that the mobile clients can be classified into five device profiles, and

| Profile | Viewable version set |
|---|---|
| $P_1$ | $\{2, 1\}$ |
| $P_2$ | $\{4, 3, 2, 1\}$ |
| $P_3$ | $\{6, 5\}$ |
| $P_4$ | $\{8, 7, 6, 5\}$ |
| $P_5$ | $\{10, 9, 8, 7, 6, 5\}$ |

TABLE III
Device profiles and viewable version sets

the distribution of these five device profiles of mobile clients is modeled as a device vector of $\langle 15\%, 20\%, 30\%, 20\%, 15\% \rangle$. Without loss of generality, we also assume that all objects could be transcoded into ten versions, and the sizes of the ten versions (from version one to version ten) are assumed to be 10%, 20%, 30%, $\cdots$ and 100% of the original object sizes [8]. The viewable version set for each device profile is shown in Table III. By a reference to [8], we assume that a more detailed version can be transcoded into a less detailed one and the transcoding delay is determined as the quotient of the object size to the transcoding rate. The transcoding rate is set to be 30 KBytes/sec [7]. The number of users in the network is set to be 1000. The cell residence time, service holding time and service establishing time for each user are set to be exponential distributions with means of 50 minutes, 10 minutes and one hour, respectively. We also assume that the data requests for each user follow a Possion process with parameter $\frac{1}{\lambda} = 60$.

### B. The Effects of the Number of Users

Figure 9 shows the experimental results with the number of users varied. The number of users is set from 400 to 1400. From Figure 9a, we observe that when the number of users is small (400 in this experiment), the system load is light and the average waiting times of all schemes are close. When the number of users increases, the average waiting time of scheme CS and scheme ODB also increases. In addition, the increment of the average waiting time of scheme CS and ODB increases as the number of users increases, especially when the number of users is larger than 1200. Since a large number of users implies the high arrival frequencies of data requests, the system load becomes heavy when the number of users is large, and hence, the average waiting time increases drastically. When the number of users is 1400, the average waiting time of scheme CS does not converge as the time advances since the system load is larger than one. This situation again agrees with the observation in Section IV-C. The average waiting time reduction of scheme ODB over scheme CS increases from 47.11% to 74.2% as the number of users increases from 400 to 1400. Scheme ODB is more scalable than scheme CS due to the effect of request merge and the employment of data broadcast.

Next, consider scheme ODB-QoS. When the number of users is small (400 in this experiment), scheme ODB-QoS and scheme ODB have similar behavior. This can be explained by the reason that when the average waiting time of scheme ODB-QoS is smaller than $W_1$, scheme ODB-QoS is degenerated to scheme ODB and guarantees that each user will receive the best viewable versions of the requested data objects. When the number of users increases to 1000, some service registrations are blocked since the average waiting time cannot be satisfied with the QoS requirements (i.e., in the interval $(W_1, W_2)$). Similarly, some service handoffs are dropped when the number of users is larger than 1200. We observe that scheme ODB-QoS is able to keep the average waiting time satisfying the QoS requirement by controlling the quality of received data objects and the number of users in service even when the offered system load is heavy.

### C. The Effects of Skewness of Access Probabilities

We investigate the effect of varied skewness of access probabilities in average waiting time, SBR and SDR. The degree of skewness is measured by the value of the Zipf parameter which is set from 1 to 1.4. The larger the Zipf parameter is, the higher the degree of skewness is. As shown in Figure 10a, the average waiting time of all schemes increases as the value of Zipf parameters decreases. It is because that the degree of request locality is high when the access frequencies is highly skewed (i.e., high Zipf parameter). Therefore, with the same cache size, the cache hit ratio is high and is able to effectively reduce the average access time. Moreover, scheme ODB can further reduce the average waiting time since the probability of request merge increases. We also observe that the increment of the average waiting time of scheme CS and ODB increases drastically when the value of Zipf parameter decreases (i.e., one in this experiment). The reason is that the effect of cache decreases as the degree of skewness decreases. Hence, the system load becomes heavy when the degree of skewness is low, and therefore, the increment of average waiting time increases. This result conforms the observation in Section IV-C. The average waiting time reduction of scheme ODB over scheme CS ranges from 36.9% to 65%.

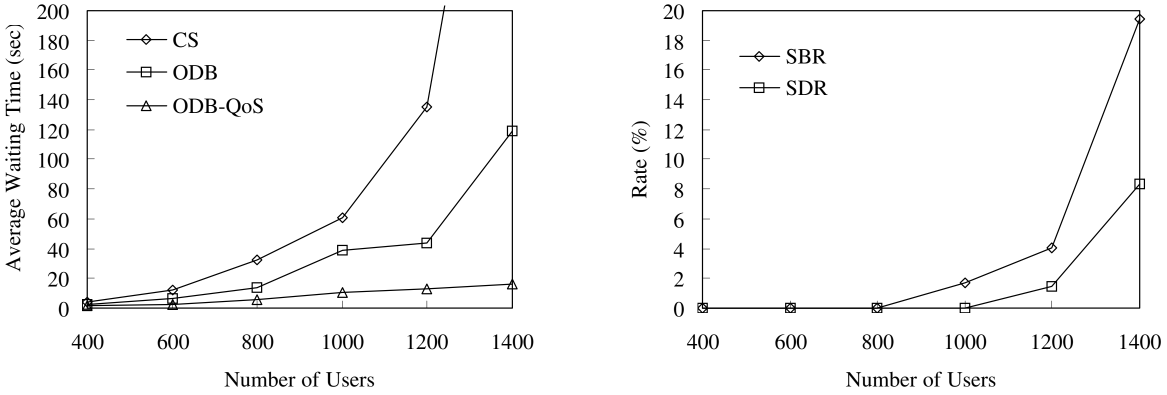
Figure 10b shows the produced SBR and SDR of scheme ODB-QoS with the value of Zipf parameter varied. We observe that when the skewness of access frequencies is high (Zipf parameter=1.4 in this experiment), scheme ODB-QoS is degenerated to scheme ODB since the average waiting time of scheme ODB-QoS is smaller than $W_1$. When the Zipf parameter is 1.2, some service registrations are blocked (SBR$>$ 0) since the average waiting time without blocking service registration cannot be satisfied with the QoS requirement. SBR is sacrificed first since mobile user can tolerate a service registration being blocked rather than a service handoff being forced to be dropped. In addition, when Zipf parameter is smaller than 1.1, some service handoffs are also dropped since the system load is too heavy.

## VI. Conclusion

We explored in this paper the effect of on-demand broadcasting technique in the design of a QoS-aware transcoding proxy. We first proposed a QoS-aware transcoding proxy architecture, QTP, and modeled it as a queueing network. By analyzing the queueing network, several theoretical results were derived to formulate the system average waiting time. We then proposed a version decision policy and a service admission control scheme to provide QoS in QTP. The derived results were used to guide the execution of the proposed version decision policy and service admission control scheme to fulfill the given QoS requirement. To measure the performance of QTP, several experiments were conducted. Experimental results show that the proposed approach is more scalable than traditional client-server systems. In addition, the proposed approach can effectively achieve the desired QoS.

## Acknowledgement

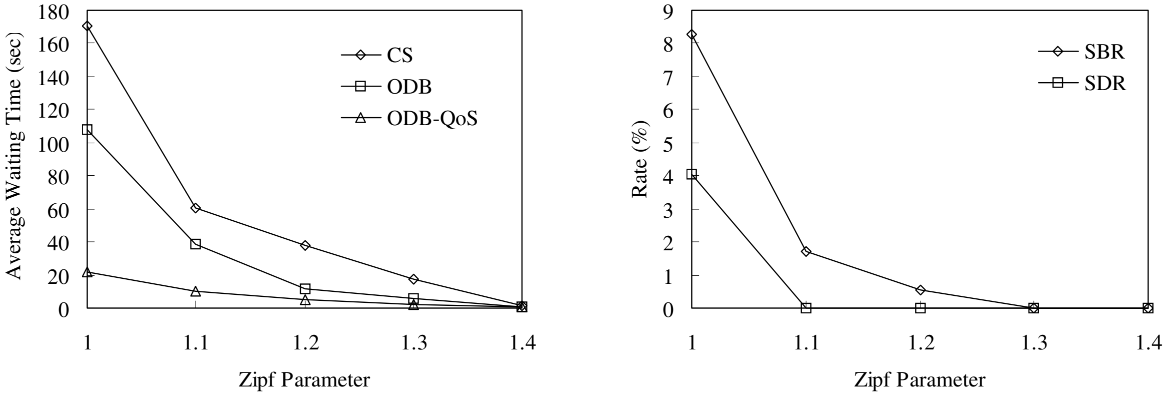
The authors are supported in part by the Ministry of Education Project No. 89-E -FA06-2-4, and the National Science



(a) Average Waiting Time

(b) SBR/SDR

Fig. 9. The effects of the number of users



(a) Average Waiting Time

(b) SBR/SDR

Fig. 10. The effects of the Zipf parameters

Council Project No. NSC 92-2213-E-002-001 and NSC 92-2213-E-002-010, Taiwan, Republic of China.

## References

[1] C. Aggarwal, J. L. Wolf, and P. S. Yu. Caching on the World Wide Web. *IEEE Transactions on Knowledge and Data Engineering*, 11(1):94–107, 1999.

[2] M. Agrawal, A. Manjhi, N. Bansal, and S. Seshan. Improving Web Performance in Broadcast-Unicast Networks. In *Proceedings of the IEEE INFOCOM Conference*, March-April 2003.

[3] D. Aksoy and M. J. Franklin. Scheduling for Large-Scale On-Demand Data Broadcasting. In *Proceedings of IEEE INFOCOM Conference*, pages 651–659, March 1998.

[4] D. Barbara. Mobile Computing and Databases - A Survey. *IEEE Transactions on Knowledge and Database Engineering*, 11(1):108–117, January/February 1999.

[5] D. Bolier and A. Eliëns. SIM: a C++ library for Discrete Event Simulation. *http://www.cs.vu.nl/~eliens/sim/*, October 1995.

[6] L. Breslau, P. Cao, G. Phillips, and S. Shenker. Web Caching and Zipf-like Distributions: Evidence and Implications. In *Proceedings of the IEEE INFOCOM Conference*, March 1999.

[7] V. Cardellini, P. S. Yu, and Y.-W. Huang. Collaborative Proxy System for Distributed Web Content Transcoding. In *Proceedings of the 9th ACM International Conference on Information and Knowledge Management*, November 2000.

[8] C.-Y. Chang and M.-S. Chen. Exploring Aggregate Effect with Weighted Transcoding Graphs for Efficient Cache Replacement in Transcoding Proxies. In *Proceedings of the 18th IEEE International Conference on Data Engineering*, Feburary 2002.

[9] S. Glassman. A Caching Relay for the World Wide Web. *Computer Networks and ISDN Systems*, 27, 1994.

[10] R. Han, P. Bhagwat, R. Lamaire, T. Mummert, V. Perret, and J. Rubas. Dynamic Adaptation in an Image Transcoding Proxy for Mobile Web Browsing. *IEEE Personal Communications*, 5(6), December 1998.

[11] A. Harter, A. Hopper, P. Steggles, A. Ward, and P. Webster. The Anatomy of a Context-Aware Application. In *Proceedings of the 5th ACM/IEEE International Conference on Mobile Computing and Networking*, pages 59–68, August 1999.

[12] W. Y. Lum and F. C. M. Lau. A Context-Aware Decision Engine for Content Adaptation. *IEEE Pervasive Computing*, 1(3), July-September 2002.

[13] V. Padmanabhan and L. Qiu. The Content and Access Dynamics of a Busy Web Site: Findings and Implications. In *Proceedings of the IEEE SIGCOMM Conference*, pages 293–304, August-September 2000.

[14] M. Satyanarayanan. Pervasive Computing: Vision and Challenges. *IEEE Personal Communications*, 8(4):10–17, August 2001.

[15] J. R. Smith, R. Mohan, and C.-S. Li. Content-based Transcoding on Images in the Internet. In *Proceedings of IEEE International Conference on Image Processing*, October 1998.

[16] C.-C. Tseng, G.-C. Lee, R.-S. Liu, and T.-P. Wang. HMRSVP: A Hierarchical Mobile RSVP Protocol. *ACM Wireless Networks*, 9(2):95–102, 2003.

[17] W3C. http://www.w3.org/Mobile/CCPP/.

[18] WAP Forum. http://www.wapforum.org/.