

# Mobility Aware Server Selection for Mobile Streaming Multimedia Content Distribution Networks

Muhammad Mukarram Bin Tariq, Ravi Jain, Toshiro Kawahara  
DoCoMo Communications Laboratories USA, Inc.  
181 Metro Drive, Suite 300. San Jose, CA 95110.  
Email: {tariq, jain, kawahara}@docomolabs-usa.com

*Abstract—*

*We propose a Content Delivery Network (CDN) with servers arranged hierarchically in multiple tiers. Lower-tier servers are topologically closer to the clients, and hence can deliver better QoS in terms of end-to-end delay and jitter. On the other hand, higher-tier servers have a larger coverage area and hence their clients incur fewer server handoffs. We present a server selection scheme that reduces the number of server handoffs while meeting differentiated QoS requirement for each client. The scheme dynamically estimates the client's residence time and uses a simple algorithm to assign clients to the appropriate tier. The scheme also caters for traditional server selection criteria, such as the expected QoS from the servers, bandwidth consumption, and the server load. We show through simulations that the scheme can achieve up to 15% reduction in handoffs at the cost of minimal increases in delay and jitter while ensuring that clients of different QoS classes experience different delays.*

## I. INTRODUCTION AND OVERVIEW

It is expected that multimedia and data traffic will surpass the traditional voice traffic in mobile networks by the year 2005 [18]. High quality streaming multimedia content is likely to form a significant portion of this traffic. It is therefore important that large mobile networks find ways to efficiently manage client traffic and data.

Content distribution networks (CDN) have proven effective for managing content-based traffic for large numbers of clients in the Internet. CDN consist of surrogate servers that replicate the content of the origin servers and serve it to the clients. CDN employ server selection and request redirection methods for selecting an appropriate (surrogate) server and redirecting the client's request to that server. CDN reduce load on both the network and origin server by localizing the traffic and providing many alternate sources of content. Iyengar et al., [8] provide an overview of existing CDN technologies.

Although there has been much work on the Internet-wide CDN, CDN for mobile networks have received little attention so far. Mobile CDN proposals, such as [4], target static WAP or HTML content for mobile clients, while only [16][17] consider streaming multimedia content distribution for mobile networks. This lack of attention has largely been because of the limited Internet access capabilities of most mobile terminals of recent past. However, this is changing quickly with deployment and acceptability of 3G services [11] that allow mobile terminals to access Internet and other data services at speeds comparable to traditional wired access. In this paper we consider server selection algorithms for streaming multimedia content distribution in networks with mobile users.

The large size of most multimedia content, long-lived nature of typical multimedia sessions, client mobility and the capricious nature of wireless communication medium, put together, present an interesting challenge. A CDN for mobile networks must address all of these issues. [16] has shown that QoS, in terms of delay and jitter, can be significantly improved by changing the server as the client moves so that the client continues to receive content from a nearby server; this process is referred to as *server handoff*.

But server handoff itself can have adverse effects. It can disrupt the streaming from the server, causing glitches at the client. Moreover, changing the server can be an expensive process for the network. Before a session can be handed over to a new server, sufficient content must be pre-fetched (or placed) at the server to ensure smooth delivery [15]. Random and frequent client movements can cause significant signaling and content placement overhead. One approach to the problem of stream disruption is to mitigate it by sufficient buffering at client equipment at the cost of increased playback delay. Another approach is using make-before-break handoffs. But actually

reducing number of handoffs is not trivial. [16] proposes simply delaying the server handoff process to reduce the number of handoffs and resulting overhead. In this paper we present a more sophisticated server selection scheme.

Our scheme reduces the number of server handoffs for mobile clients by using client mobility information and selecting a server with which the client can remain associated for an extended period of time, thus reducing the need for server handoffs. We also cater for traditional server selection criteria such as expected QoS (in terms of delay and jitter) from the server, traffic localization, and server load. The basic trade-off that we explore is how to maintain QoS service differentiation for clients while reducing the network cost due to server handoffs.

The rest of this paper is organized as follows. Section II describes our mobility-based server-selection scheme. Sections III and IV are dedicated to simulation and results. In section V we discuss related work, and conclude the paper in section VI.

## II. MOBILITY BASED SERVER SELECTION

Our approach is to use the client’s mobility information as an additional metric in the server selection process, along with traditional metrics such as client-server proximity, expected QoS, content availability and server load. This allows us to select a server that will remain suitable for content delivery to a client for an extended period of time and thus eliminating the need for frequent server change.

### 1) Layout of Servers in Content Distribution Network

We consider a content distribution network with a relatively large number of servers arranged in a logical tree structure, with servers closest to the access network belonging to the lowest tier (see Figure 1). This hierarchical server layout allows us to maximize the traffic localization and obtain desired trade-off between the number of server handoffs and the QoS to the clients. These advantages become further apparent in proceeding sections.

Each server has a coverage-area defined as the subnets and cells in the underlying network (each subnet comprises one or more cells). Generally, each subnet (and cell) is in the coverage area of

the closest server at any given tier, but multiple servers may serve subnets at coverage-area boundaries. We use the term *server zone* to refer to the coverage area of lowest tier servers. Servers at higher tiers have larger coverage areas and every subnet is covered by multiple tiers.

As in [16], a server handoff is performed whenever a client moves from the coverage area of one server to another. However, with the multi-tier arrangement of servers, there is an opportunity to trade-off QoS with the number of server handoffs by choosing the tier of the new server. Data delivered from the lower tier servers will suffer less delay (and probably less jitter) than that from the higher tiers but at the same time will result in more server handoffs and may increase the overhead for the network.

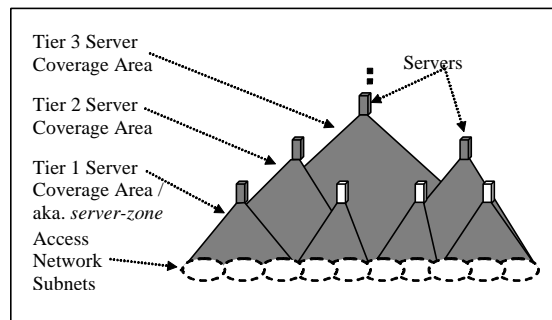


Figure 1: Tiered Layout of Servers in CDN

We assume that clients have different categories of QoS requirements in terms of delay and jitter. We further assume that each of these categories of QoS can be roughly characterized as the property of a particular server tier. For example, we can assume that highest QoS class, with strict delay and jitter requirement (QoS class 1) maps to the lowest tier server. A fixed mapping of this type is, however, not necessary. Other QoS metrics such as available bandwidth, response-time, and media quality may be used for server classification. Several dynamic QoS probing and measurement methods such as in [5][10] can be used to dynamically classify servers based on different QoS metrics.

We assume that there is a Request Routing function (RR) in each server zone (recall a server zone is the coverage area of lowest-tier servers). We recommend that this function be a proxy-based RR that may be co-located with lowest-tier servers. The proxy-based approach provides greater control and flexibility but other RR

techniques may also be used [1]. As the client moves from one server zone to another, it is assigned a new serving RR. The new RR selects servers for any new sessions that the client may request and also initiates and coordinates server handoffs for the client's existing sessions if needed. We assume that the RR has sufficient knowledge about the CDN topology and the client mobility information to select appropriate servers and route requests correctly. We describe the information collection and server selection process in the following.

## 2) *Considerations in Mobility Based Server Selection*

We must consider the following in mobility based server selection.

1. Using lower tier servers reduces delay and jitter, but increases the number of server handoffs. Thus high QoS users must be served from lower tiers and high mobility users must be served from higher tier servers. But there will be users that have high mobility and high QoS requirements at the same time. We must find way to assign server to clients with such conflicting characteristics.
2. If a disproportionate number of clients in a region have similar mobility characteristics, they may overload a certain tier. Therefore load balancing among tiers must be performed. For example, during commute hours many clients may have high mobility. Similarly, in a region spanning high-speed freeways, the average user mobility would be higher than other regions. Reduction in handoffs requires assigning higher tier servers. But if the server load is not considered, higher tier servers may get overloaded.
3. Client requests belong to various differentiated QoS classes. We must maintain a desired differentiation in terms of the mean QoS for sessions of each QoS class.

## 3) *Measurement of Mobility Rate and Server Residence Time Estimation*

Our scheme uses the client's expected residence time in a server's coverage area to select an appropriate server. The *residence time* is defined as the time between client entering the coverage area of a server and leaving this coverage area.

Each RR maintains the residence time value  $r_i$  for every client  $i$  in its server zone. This value is computed by the clients themselves as a running average of the residence time over their  $k$  most recent movements, and reported to the serving RR upon each movement. This movement can be a movement across cell, subnet or other lower layer region boundary, but information based on movements across subnet boundaries can be easily collected and reported using protocols such as Mobile IP [6]. Each RR also maintains the average subnet residence time  $\bar{r}$  over all the clients in its server zone. In addition RR also maintains the mean server residence time  $\bar{R}_t$  over all servers in each tier  $t$ .

After each successful handoff, the new RR send a server handoff notification message to the old RR. This message includes the client identifier, session identifier and the identifiers of the new and the old servers involved in the handoff. Using the information contained in this message, the old RR determines the server residence time as the difference of the time at which the client was originally assigned to the old server and the time of server handoff. Information about the time of original assignment can either be contained in the notification message, or can be assumed to be maintained by the old RR since it is in path of session control signaling traffic. The server tier can be determined using the server identifier. The old RR updates the mean server residence time for the corresponding tier after each handoff notification message.

Using this set of information, the RR can compute the estimated residence  $E_{i,t}$  of a client  $i$  with a server at tier  $t$ .

$$E_{i,t} = \frac{r_i \bar{R}_t}{\bar{r}} \quad (1)$$

In a refinement to mobility estimation, RR maintains the average subnet residence time  $\bar{r}_s$  and server tier residence time  $\bar{R}_{t,s}$  separately for clients in each subnet  $s$  in its coverage-area and uses these values for estimating residence time of a client in subnet  $s$ . This improves the accuracy of estimation because the mobility pattern of a client is likely to be similar to the clients in its close geographical proximity. Expected residence time of a client  $i$  in subnet  $s$  for a server in tier  $t$  with this refinement is given as:

$$E_{i,t} = \frac{r_i \bar{R}_{t,s}}{\bar{r}_s} \quad (2)$$

In the following text, we refer to equation (1) and equation (2) as *lower granularity residence time estimation* and *higher granularity residence time estimation* respectively.

Several mobility prediction schemes have been proposed (e.g., [2], see [3] for a survey), which can be used by our algorithm with minor modifications. These schemes will likely provide better accuracy but only at the expense of extensive state maintenance and large overhead. We will discuss existing prediction schemes in section V.

#### 4) Server Load and QoS Information Collection

Each server periodically sends information to all the RR in its coverage area about the number of sessions that it can accept from the RR. Servers observe the load from each RR (i.e., the number of sessions emanating from the RR's server-zone) and allocate its remaining capacity to each RR proportionately to the load observed from that RR; we call this the *server load allowance*. The sum of load allowance reported to an RR by all servers at tier  $t$  is denoted as  $L_t$ . With a high density of servers the number of RR in coverage area of each server is likely to be low, thus dispersal of such information is likely to be feasible.

Each RR obtains information about the nominal QoS obtained from each tier. This can be done using QoS probing techniques, such as in [5][10], or QoS feedback information, such as through RTCP [14] or by other means. For the following discussion we assume delay as the primary QoS metric, but other QoS metrics can be used in a similar manner with our algorithm. The nominal delay information for a tier  $t$  is represented as  $D_t$ . Each RR also maintains the number of client requests of each QoS class  $q$  that it sends to tier  $t$  as  $N_{q,t}$ .

#### 5) Server Selection Algorithm

Listing 1 shows the pseudocode for the server selection process. When an RR needs to select a server for request for a new session of QoS class  $q$  from a client  $i$ , it starts by selecting the

highest possible server tier  $t$  whose reported nominal delay  $D_t$  meets or exceeds the requirements of QoS class  $q$ . If the request is for a server handoff of an existing session then the RR sets  $t$  to the tier of the currently assigned server.

The RR then uses a heuristic (lines 3-7 of Listing 1) to determine whether the client should be assigned to tier  $t+1$ . It finds the load allowance  $L_{t+1}$  for tier  $t+1$ , and determines whether the client  $i$  is among the fastest clients. The reasoning behind the intuition here is that since the load allowance is limited and only a limited number of clients can be served from higher tiers, the reduction in number of server handoffs can be maximized if only the fastest clients are served from the higher tiers.

Instead of relying on any absolute value for residence time or speed that would qualify a client for service from higher tiers, we simply use the client's relative mobility with respect to the other clients in its region and see if the client is fast enough.

The heuristic used to determine whether a client  $i$  is "fast enough" is based on the assumption that the residence time distributions of the clients in the recent past are similar to those of the immediate future. The RR sums the load due to all the clients whose residence times are less than the client  $i$  and sees whether it is less than the load allowance  $L_{t+1}$ ; if so, the client  $i$  is a candidate to be moved to tier  $t+1$ .

The RR then checks whether moving the client  $i$  to tier  $t+1$  will maintain the desired separation among the adjacent QoS classes. The comparison is performed using the weighted mean QoS of client sessions of each QoS class (line 27 in Listing 1). If this QoS separation is maintained, the client is assigned to a server at tier  $t+1$ .

If either of above conditions are not satisfied, the RR checks whether the client should be moved to the tier  $t-1$  (lines 8-17. of Listing 1). For this the load allowance of tier  $t-1$  must be positive. Additionally it must ensure that moving the client to a lower tier would not increase the number of handoffs. This is only possible if the client is relatively slow moving. To check whether the client qualifies, we test whether client's estimated residence time for the lower

tier,  $E_{i,t-1}$ , exceeds the average residence time of current tier  $t$  (line 11 in Listing 1). Either lower granularity residence time estimation or higher granularity residence time estimation may be used for  $E_{i,t-1}$ .

Note that if the residence time is under estimated, moving a client to lower tiers will likely increase the number of server handoffs. To study the effect of this we have introduced two modes, *lazy* and *eager*, in our algorithm. In the eager mode the RR takes the risk and selects tier

$t-1$  if it the client's estimated residence time with tier  $t-1$  exceeds the average residence time. In lazy mode, however, the RR acts more cautiously; it additionally checks whether there are indications that the higher tier servers are starting to get overloaded, in particular when the load allowance reported by higher tier servers falls below a certain threshold (lines 34-36 in Listing 1).

We will compare the results of these several possibilities through simulations in next section.

### Listing 1: Mobility Based Server Selection Algorithm

```

1  proc selectServer (i: Client, q: QoS, t: serverTier, mode: String)
2      selectedTier := t;
3       $L_{t+1} := \text{findLoadAllowance}(t+1)$ 
4      if clientInFastest ( $L_{t+1}$ , i)
5          if (qoSSeparationMaintained(q))
6              selectedTier := t+1;
7          endif
8      elseif(mode = "eager" ||
9          (mode = "lazy" & LoadAllowanceLow ( $L_{t+1}$ , t+1))
10          $L_{t-1} = \text{findLoadAllowance}(t-1)$ 
11         if ( $L_{t-1} > 0$ )
12             if ( $E_{(i,t-1)} > \bar{R}_t$ )
13                 if (qoSSeparationMaintained(q))
14                     selectedTier := t-1;
15                 endif
16             endif
17         endif
18         serverIdentifier := findServer(selectedTier)
19         return serverIdentifier;
20 endproc

21 proc qoSSeparationMaintained (q: QoSClass): bool
22     return (meanDelay(q+1)-meanDelay(q) >  $d_{q,q+1}$ ) &
23         (meanDelay(q)-meanDelay(q-1) >  $d_{q-1,q}$ )
24     %%  $d$  is desired delay separation of QoS Classes
25 endproc

26 proc meanDelay (q: QoSClass): QoS
27      $T := \text{number of server tiers that cover server-zone of the RR}$ 
28      $\bar{D}_q := \frac{\sum_{t=1}^T N_{q,t} D_t}{\sum_{t=1}^T N_{q,t}}$            %% weighted mean QoS of class q
29     return  $\bar{D}_q$ 

```

```

29 endproc
30 proc clientInFastest (Allowance L, Client i): bool
31   C := total number of clients of RR
32   return  $\sum_{j:r_j < r_i}^C U_j < L;$       %%Uj is number of sessions of client j
33 endproc
34 proc LoadAllowanceLow (L: Allowance, t: serverTier): bool
35   return (L < minThreshold * MaxAllowance(t));
36 endproc

```

---

### III. SIMULATION SETUP

We have simulated behavior of our algorithm. The following subsections describe the system layout, the simulation scenarios and the results obtained.

#### 1) Mobility Simulation

To get realistic mobility information we have built a custom mobility simulator. This tool simulates movement of mobile clients in a digitized geographical map comprising roads, streets and railways. For this paper, we have used a map of San Francisco Bay Area and its surroundings, covering 3575 sq. miles.

Using this tool, we can specify populations at different locations in the map. The simulator chooses the origin and destination points of the mobile clients such that the number of points chosen from an area is proportional to its population. Thus any desired overall flow of clients in the network can be specified in using population distributions and different movement patterns can be effectively simulated just by specifying a different population distribution.

Once the origin and destination points are chosen, the simulator uses Dijkstra's shortest path algorithm to compute the path for the hosts. The simulator is capable of using different metrics for computing shortest path. These include distance, estimated travel time, or a preference of type of transport. Presently we consider clients with distance and time based short paths only and in equal proportions.

The simulator supports four mediums of transport, on-foot, car, train and city bus. On-foot mode is used over very short distances, less than 0.25 miles, and it does not necessarily abide by the road directions and conditions. Cars and buses

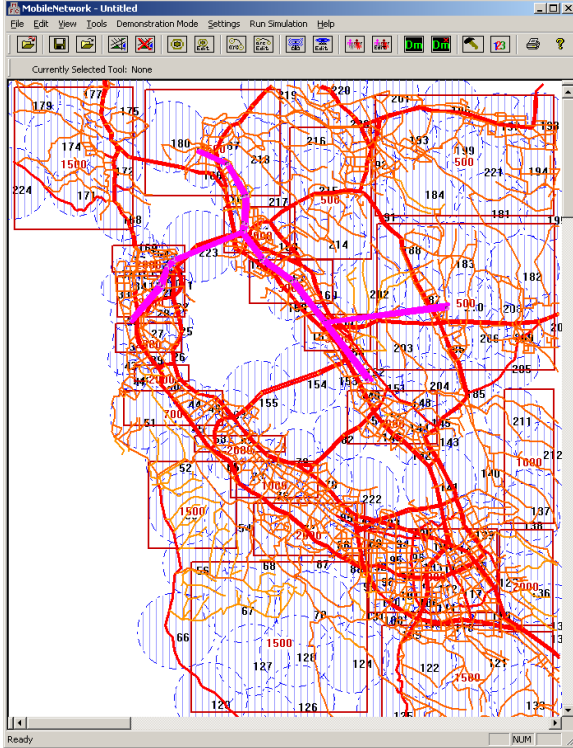
follow the roads directions and conditions. But while clients in cars move asynchronously, the clients on buses must travel together according to route and schedules of buses. Same is the case with trains except that they have specialized tracks. The simulator allows specifying capacity and schedules for buses and trains. In the current simulator, each car is occupied by just one client.

Once the path is calculated the client host travels along this path and its speed is varied according to speed limits and congestion conditions on the roads and highways and by schedules of trains and buses; hosts slow down in congested segments and wait on designated stations for the next bus or train to arrive. Clients on roads can re-route in response to traffic congestion by calculating a new path that avoids congested segments.

The geographic map is overlaid with 187 cells of varying radii. The radius of the cell varies between 0.2 miles to 5 miles depending on population distribution and terrain; this is in line with typical 3G network planning[22]. The radio network is further overlaid with 59 network layer (IP) subnets. Each subnet has a router that serves 2 to 4 radio cells.

We have simulated behavior of movement of 2500 clients. We log events as these clients move from one cell to another. These events are then converted into network subnet level movement events and can directly be consumed by our server selection algorithm.

Figure 2 shows a screen shot of the mobility simulator showing bay area map, radio network base-stations covering the area and population distributions (the rectangular boxes).



**Figure 2: Mobility Simulator**

## 2) CDN Layout

### a. Topology

In the simulation scenario the tiered CDN topology comprises 34 surrogate servers arranged in 3 tiers, thus  $t \in \{1, 2, 3\}$ . We have 21 servers in tier 1 and thus 21 server zones, each containing an RR. There are 8 servers in tier 2 and 4 in tier 3. The CDN has 4 origin servers that place content on the surrogate servers if it is already not there. The model also contains background traffic servers to create configurable amount of background traffic to load the network. We have modeled the server selection algorithm and the network using a combination of Matlab [9] and OPNET [12]

### b. Network delay

For our current simulation, we assume that when there is no load on the network, there is 80ms end-to-end delay from a client in the access network to the nearest tier 1 server. Similarly, there is 160ms end-to-end delay from nearest tier 2 servers and 240ms delay from nearest tier 3 servers, thus  $D_1 = 80\text{ms}$ ,  $D_2 = 160\text{ms}$  and  $D_3 = 240\text{ms}$ . In general, any real-time delay estimation process, such as in [5] [9], can be used

to obtain these values, but for simplicity we do not simulate these processes here. We also assume the desired delay separation between QoS classes is 20 ms, thus  $d_{1,2} = d_{2,3} = 20\text{ms}$ .

### c. Server characteristics

The behavior of surrogate servers is characterized by:

1. The time to process a session request; for the current simulation it is 100ms.
2. The time to process mobility notification message (e.g., binding update message of Mobile IP [6]); currently it is fixed at 10ms.
3. The maximum media packet delivery rate, current value is 10000 per second.
4. The cache hit-ratio i.e., the probability of finding the content locally. In case of a cache miss, the server sends a content placement request to the origin server to retrieve the content.

Origin servers have same characteristics as surrogate servers, except that the content is always assumed to be present. Surrogate servers act as clients to origin servers. The data-rate of content placement is higher than the actual streaming data-rate from the surrogate server to the client; in our simulation it is 1Mbps.

Servers process all requests (session and mobility notification) on first-come first-served basis. Because of high packet delivery rate, the number of sessions that can be served from a server is limited by the network capacity.

### d. Client characteristics

We assume that client requests belong to one of the three QoS classes, high (1), medium (2) and low (3). We assume that the tier 1 servers are appropriate for high QoS class. Tier 2 servers are appropriate for medium QoS class and tier 3 servers are appropriate for low QoS class.

The number of client requests in the three QoS classes is proportional to number of servers in three tiers, thus the ratio is 21:8:4. This ensures that if there is no mobility based server selection (simulation scenario 5, explained in next subsection), the clients are evenly distributed across the tiers. This is important for obtaining a fair assessment of benefits of our scheme. In our present simulation we assume that data rate for all

three QoS classes is 64kbps and 20 packets per-second and end-to-end delay is the main QoS determinant. We assume that each client has only one active session at a time, thus  $U_j$  is 1.

### 3) Simulation Scenarios

We have simulated 5 scenarios.

1. Lazy server selection mode using higher granularity residence time estimate (using equation 2).
2. Eager server selection mode using higher granularity residence time estimate (using equation 2).
3. Lazy server selection mode using lower granularity residence time estimate (using equation 1).
4. Eager server selection mode using lower granularity residence time estimate (using equation 1).
5. Non-adaptive server selection. In this scenario we do not attempt to reduce the number of handoffs. A client request is always sent to the

highest tier that can meet or exceed the QoS requirements.

We have simulated each scenario for different simulation parameters. These are explained in next section. We use the scenario 5 as a baseline for comparison.

### 4) Simulation Parameters

#### a. Session duration

The session durations vary from 75 seconds to 1500 seconds. Each client waits for a random time from 0 and 20 seconds before the start of the first session and between subsequent sessions.

#### b. Server capacity

We have also obtained results for different server capacities. In particular we have simulated the scenarios for server capacities varying between 50 to 300 simultaneous sessions.

#### c. Load allowance threshold for lazy mode

For the lazy mode the minimum load allowance threshold is 10% of maximum reported load allowance.

**Table 1: Summary of Simulation Parameters**

Parameter	Values/Range
<b>Mobility Related Parameters</b>	
Number of users	2500
Simulation area	3575 sq. miles
Simulation duration	7200 seconds
Number of base stations and subnets	187 base stations, 59 subnets
<b>Server Selection Parameters</b>	
$t$ : tier identifier	{1,2,3}
$q$ : qos class identifier	{1,2,3}
$D_t$ : delay to the tier $t$ in millisecond, $t \in \{1,2,3\}$ .	{{80, 160, 240}, {80, 120, 160}}
$d_{q,q+1}$ : desired delay separation between qos class $q$ and $q+1$ .	$d_{1,2} = d_{2,3} = 20\text{ms}$
Total servers	34
Servers at each tier $t \in \{1,2,3\}$ .	{21, 8, 4}
Session duration (seconds)	{50, 100, 200, 500, 1000, 1500}
Server capacity (number of simultaneous sessions)	{50, 75, 100, 200, 300}
Server overload threshold for Lazy mode.	10% of maximum reported load allowance.
<b>Content Distribution Parameters</b>	
Server's session request processing time	100ms
Server's new address binding update processing time	10ms
Data-rate per user	64kbps, 20 packets per second.
Maximum server packet delivery rate (processing limited)	10000 packets per second

#### d. Network delay

We have also performed simulations for different network delays to confirm the independence of algorithm from absolute QoS values.

Table 1 summarizes different simulation parameters.

### IV. RESULTS

#### 1) Reduction in Number of Handoffs

Figure 3 shows the reduction in the number of server handoffs as compared to the non-adaptive server selection scenario. Detailed results for  $\{D_1=80\text{ms}, D_2=160\text{ms and } D_3=240\text{ms}\}$ ,  $\{d_{1,2} = d_{2,3} = 20\text{ms}\}$  are included here. Later in the section we will summarize how varying network delays impacts our server selection algorithm.

##### a. Impact of session duration

Both lazy and eager approaches achieve better performance for longer session durations, because the algorithm gets more time to adapt to behavior of the mobile client. In fact, for very small sessions, there is a slight increase in number of handoffs, because the algorithm does not get a chance to correct a wrong estimate before the session is ended.

##### b. Impact of server capacity

Reduction in number of handoffs is similar in both lazy and eager approach for smaller server capacity, since the minimum load allowance threshold is reached more quickly. However, as server capacity increases, lazy approach provides significant improvement in reduction of number of handoffs. We found that for eager approach the reduction in number of handoffs was often limited by the desired delay separation whereas for lazy approach the limiting factor is the load allowance of higher tiers. The reduction in number of handovers in figure 3c and 3d for longer duration and low load allowance is not well behaved. This is because with lower load allowance, the criterion for fastest users is strict, and an error is more likely.

##### c. Impact of granularity of residence time estimation

Both lazy and eager approaches give better performance when subnet-specific mobility

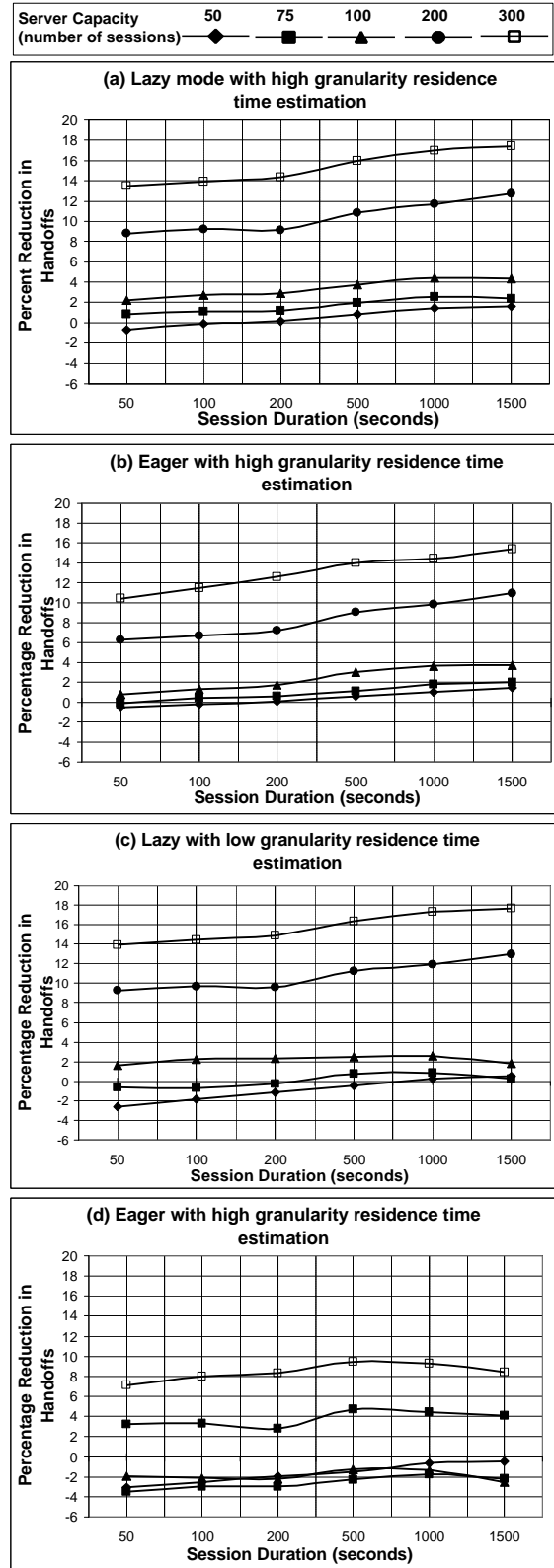


Figure 3: Percentage reduction in server handoffs for various scenarios

information is used for residence time estimation, as these estimates are more accurate than estimates made using server-zone wide information.

#### d. Impact of network delay

We have performed simulations for different values of  $D_t$  and  $d_{q,q+1}$ , and we found that the performance of the algorithm is largely independent of absolute value of QoS metric ( $D_t$ ) and depends more on the desired separation  $d_{q,q+1}$ . For example we kept the desired separation at the same level, that is,  $d_{1,2} = d_{2,3} = 20\text{ms}$  but reduced the server delays such that  $D_1 = 80\text{ms}$ ,  $D_2 = 120\text{ms}$  and  $D_3 = 160\text{ms}$ .

We observed that the performance of our scheme does not degrade significantly. The mean deterioration in reduction in number of handoffs is 0.9%. This is in comparison with the results in Figure 3 where  $D_1 = 80\text{ms}$ ,  $D_2 = 160\text{ms}$  and  $D_3 = 240\text{ms}$ . The higher deterioration (maximum deterioration observed is 2.5%, which still corresponds to 15% reduction in number of handoffs) is only for longer session durations and higher server load allowances, but since we already have significantly larger reduction in number of handoffs for these parameters, this deterioration is not worrisome.

We also observed that for eager mode, lower network delay actually slightly improves the reduction in the number of handoffs. Because in this case the delay separation limit is reached more quickly that stops the algorithm for making further erroneous server assignment.

#### 2) Impact on Delay and Jitter

Figure 4 shows the impact on the mean end-to-end delay for the three QoS classes. Both lazy and eager approaches maintain delay separation among the QoS classes; however, the overall delay for each QoS class is more with lazy approach. Observe that as the load allowance increases, the delay separation between the different QoS classes tends to decrease in all cases, because with lower server capacity, the load allowance diminishes more quickly, resulting in fewer transitions from one tier to another and thus maintenance of higher delay separation.

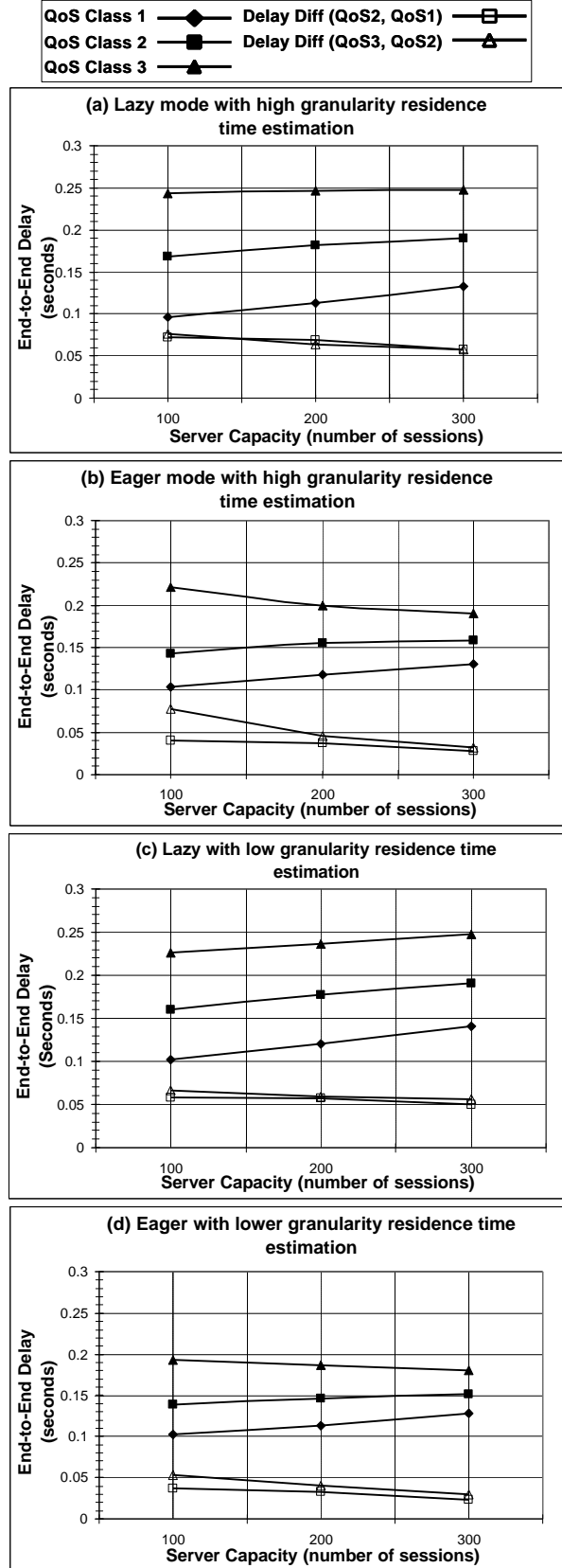


Figure 4: Impact on End-to-End Delay

We have also obtained results for end-to-end jitter; these are omitted here but are quite similar in nature.

### 3) *Comparing Lazy and Eager approaches*

Although we are able to achieve significant reduction in number of handoffs for each of the four simulation scenarios, the results highlight several comparative strengths and weaknesses of the two approaches.

Lazy mode achieves more reduction in number of handoffs than the eager mode (comparing figure 3a and 3c with 3b and 3d). It is also less susceptible to errors in residence time estimation (comparing figure 3c and 3d). This is because a client moved to lower tier as a result of under estimated residence time will require more server handoffs. Lazy mode moves fewer clients to lower tiers thus reducing the cases where a wrong client is assigned to a lower tier server.

Assigning users to higher tiers also has some disadvantages. Comparing figures 4a and 4c with figures 4b and 4d respectively we can see that there is that clients incur overall higher delays with lazy mode compared with the eager mode.

At the network level, assigning clients to higher tiers translates to lesser traffic localization and higher utilization of network resources. This can be a significant factor if network bandwidth is more expensive than placing servers at lower tiers.

## V. RELATED WORK

There has been a great deal of work on web performance enhancements. For CDN, several content placement, server placement, and server selection techniques have been proposed, including schemes geared towards load balancing among local and distributed servers, improving client's perceived QoS such as [5], and reducing network cost through intelligent server placement. Several request routing schemes [1] are used in conjunction with the server selection schemes. A comprehensive overview of various techniques used for enhancing web performance can be found in [8].

Yoshimura et al., [17] have proposed a multimedia CDN architecture. In their work, a centralized portal and content location manager selects an appropriate server for the client. The server selection is based on information about the

client's current location, server load and network conditions. Although they have proposed server handoff in response to client movement under certain conditions, the server selection process does not attempt to select a server that would reduce the probability of need for server handoff.

In [16] we show that for long-lasting streaming multimedia sessions, the delay and jitter can be significantly improved by server-handoff as clients move. But in [16] the server selection was based on client location and content availability. No explicit effort is made at the time of server selection to select a server that suits the mobility profile of the client.

The notion of server handoffs to improve performance for mobile users has been proposed in [20], but not applied to multimedia or CDN systems.

To the best of our knowledge, there is no work that uses mobility information for server selection in CDN. In estimating residence time and dealing with mobility information, our focus has been to enable server selection in a fully distributed manner. Additionally we try to minimize the station information that be maintained for the purpose of tracking client movement or making prediction, because maintaining such information for a large number of clients can consume significant resources. Mobility information gathering, as it is currently defined in our scheme, can be easily collected using existing protocols such as Mobile IP [6] binding update signaling messages. It is likely that mobility prediction algorithms, such as used for resource reservation and call admission control [19] in mobile networks can be used.

## VI. CONCLUSIONS

We have presented a mobility aware server selection scheme that can significantly reduce the number of server-handoffs necessary for streaming multimedia content delivery via a high density CDN. In our server selection algorithm we use mobility information about the clientele along with traditional server selection criteria such as expected QoS from the server, traffic localization, and server load. The effectiveness of our server-selection algorithm depends on the accuracy of the clients residence time estimate. While the estimation schemes that we have proposed are decentralized, stateless and simple,

they are not very accurate. Nonetheless, our simulations show that our server selection algorithm can reduce the server handoffs by up to 10-15% at the expense of small increase in mean end-to-end delay, while maintaining delay separation among different QoS classes. More elaborate mobility information prediction method can improve the performance of our algorithm at the expense of increased complexity.

#### REFERENCES

- [1] A. Barbir, B. Cain, R. Nair, O. Spatscheck. "Known CN Request-Routing Mechanisms." IETF Internet Draft, work in progress. Apr 2003.
- [2] C. Cheng, R. Jain and E. van den Berg, Location Prediction Algorithms for Mobile Wireless Systems, in Handbook of Wireless Internet, M. Illyas and B. Furht (eds.), CRC Press, Dec. 2002.
- [3] I.R. Chen and N. Verma, "Simulation study of a class of autonomous host-centric mobility prediction algorithms for cellular and ad hoc networks." in proc. of 36th Annual Simulation Symposium, Orlando, FL, USA, Mar 2003.
- [4] "Cisco's Mobile CDN" <http://www.cisco.com/warp/public/784/packet/apr02/p49-cover.html>.
- [5] Z. Fei, S. Bhattacharjee, E. Zegura, M. Ammar. "A Novel Server Selection Technique for Improving the Response Time of a Replicated Service." in proc. of Infocomm 98, Apr 1998. San Francisco, California.
- [6] D. Johnson, C. Perkins, J. Arrko. "Mobility Support in IPv6." IETF Internet Draft, work in progress. May 2003.
- [7] Intelligent Content Distribution Service, AT&T. <http://www.research.att.com/news/2002/September/ICDS.html>
- [8] A. Iyengar, E. Nahum, A Shaikh, R. Tewari. "Enhancing Web Performance." in proc. of 2002 IFIP World Computer Congress. Aug 2002. Montreal, Canada.
- [9] The Mathwork Inc. <http://www.mathworks.com/>
- [10] J. Matta, A. Takeshita. "End-to-End Voice Over IP Quality of Service Estimation Through Router Queuing Delay Monitoring." in the proceedings of IEEE Globecom2002, Nov 17-22, 2002. Taipei, Taiwan.
- [11] NTT DoCoMo. FOMA Subscriber Growth. <http://www.nttdocomo.com/>
- [12] OPNET Technologies. <http://www.opnet.com>
- [13] M. Rabinovich, A. Aggarwal. "Radar: A Scalable Architecture for Global Web Hosting Service." in the proc of The 8th Int. World Wide Web Conf, May 1999.
- [14] H. Schulzrinne et al. "RTP: A Transport Protocol for Real-Time Applications." IETF RFC-1889. January 1996.
- [15] S. Sen, J. Rexford, D. Towsley. "Proxy Prefix Caching for Multimedia Streams", in proc. of IEEE INFOCOM, Apr 1999.
- [16] M. Tariq, A. Takeshita. "Management of Cacheable Streaming Multimedia Content in Networks with Mobile Hosts." in the proceedings of IEEE Globecom2002, Nov 17-22, 2002. Taipei, Taiwan.
- [17] T. Yoshimura, Y. Yonemoto, T. Ohya, M. Etoh, S. Wee. "Mobile Streaming Media CDN Enabled by Dynamic SMIL." in the proc. of WWW2002, May 7-11, 2002, Honolulu, Hawaii, USA.
- [18] H. Yumiba, K. Imai, and M. Yabusaki, "IP-Based IMT Network Platform". IEEE Personal Communications, Oct 2001, pp 18-23.
- [19] T. Zhang, E. Berg, J. Chennikara, P. Agrawal, J. Chen, T. Kodama. "Local Predictive Resource Reservation for Handoff in Multimedia Wireless IP Networks." IEEE Journal on Selected Areas in Communications, Vol 19, No. 10, Oct 2001, pp 1931-1941.
- [20] R. Jain and N. Krishnakumar, Network support for personal information services to PCS users, IEEE Networks for Personal Communications (NPC), Jan. 1994.
- [21] A. K. Elmagarmid, J. Jing, and O. Bukhres. An efficient and reliable reservation algorithm for mobile transactions. Proceedings of the 4th International Conference on Information and Knowledge Management (CIKM'95).
- [22] Nokia Network Planning White Paper: [http://www.nokia.com/downloads/aboutnokia/press/pdf/MWR\\_Planning\\_A4.pdf](http://www.nokia.com/downloads/aboutnokia/press/pdf/MWR_Planning_A4.pdf)