

# Performance Implications of Different Adaptation Mechanisms for Network Content Delivery

Xiaodong Fu and Vijay Karamcheti  
Department of Computer Science  
New York University  
New York, NY. 10003

## Abstract

*Because of heterogeneous and dynamic changing network environments, content delivery across the network requires system support for coping with different network conditions in order to provide satisfactory user experiences. Despite the existence of many adaptation frameworks, the question that which adaptation approach performs the best under what network configurations still remains unanswered. The performance implication of different adaptation approaches (end-point, proxy-based and path-based approaches) has not been studied yet. This paper aims to address this shortcoming by conducting a series simulation-based experiments to compare performance among these adaptation approaches under different network configurations. The experiment results show that there are well-defined network environments under which each of these approaches delivers its best performance, and among them, the path-based approach, which uses the entire communication path to do adaptation, provides the best and the most robust performance under different network configurations, and for different types of servers and clients.*

## 1 Introduction

Recent advances in network technology have made it possible to deliver real-time rich media content across the Internet. However, despite increasing bandwidth, the Internet still remains a best effort platform, and exhibits considerable heterogeneity (on links and end devices) and highly dynamic resource availability for individual network paths, which is a major cause for the unsatisfactory performance perceived by end users.

To cope with the above problems, a widely used approach in current-day network services is to provide differentiated service for different kinds of (last-hop) network links and end-devices. For example, most media sites usually supply several *versions* of content for different client groups that use different connection options. These versions usually have different requirements on bandwidth and client computation capacity. Although this approach

can improve user experience to some extent, the view that clients can be placed into a few fixed classes with constant characteristics is incompatible with the fact that availability of network resources can change over small timescales.

More promising approaches to address this problem rely on a general framework that provides system support enabling applications to *adapt* to different network conditions. Examples of such systems include Rover [7], Odyssey [9], ActiveProxy [2], Conductor [11], Active Names [10], Ninja [6], CANS [5] and Scout [8], to name a few. A common property of these systems is that the data in transmission can be processed on the fly with various operators (in some cases, routed via) to match different network conditions. Some of the systems, such as CANS [5], also provide built-in support for automatic and dynamic modification of these operators. Compared with the static nature of the differentiated services solution, these systems because of their flexibility can deliver better user experience.

Despite these common benefits, the general frameworks described above are quite different from each other. An important difference has to do with the location in the network where adaptation functionality is introduced. With regards to this issue, the systems above can be grouped into three categories: *end-point* approaches (e.g., Rover [7], Odyssey [9]) where only client and server nodes are involved in adaptation; *proxy-based* approaches (e.g., Ninja [6], Active Service [1]) where a proxy node, usually close to the client side, is used to process data, possibly with the cooperation of client nodes; and *path-based* approaches (e.g., Active Names [10], Conductor [11], CANS [5], and Scout [8]), where adaptation can happen at arbitrary intermediate locations in the network in addition to the server, client, and proxy sites as above.

A natural question to ask is whether these three categories of approaches bring any special advantages or disadvantages. Unfortunately, although previous work has shown that the use of such adaptation systems does result in better performance and user experience as compared to situations when no adaptation is provided, the perfor-

mance implication of these different approaches has not really been studied in any detail. In particular, the answers to the following questions are not readily available: How is the performance that is achievable constrained by the location of adaptation? Under which network conditions is one kind of approach preferred over another? Does the additional complexity of path-based approaches, which distributes computation resources across the network and requires distributed control over several locations in the network, introduce performance penalty? Answering these questions is important, not only for understanding the performance of existing adaptation frameworks, but also for building appropriate solutions as new devices and networks are deployed.

The study presented in this paper aims to answer these questions. We extensively simulate the behavior of different adaptation approaches in the context of a large scale network topology, characterizing in detail the performance and resulting client behaviors. The study looks at how different approaches fare for different assumptions about server computation capacity, client capacity and connectivity options, and request distribution between clients and servers. A key aspect of our methodology is the use of approach-neutral mechanisms for constructing communication paths (i.e. operators and their mapping to network nodes) and managing network resources, which attempt to make a fair comparison by maximizing the performance of each approach given its individual constraints of where network adaptation can happen.

The key finding is that there are well-defined network environments under which each of the approaches delivers its best performance. More importantly, each approach, with one exception – the path-based approach, also exhibits performance shortcoming in certain environments. Among these approaches, the path-based approach provides the best and most robust performance under most network configurations.

The rest of this paper is organized as follows. In Section 2, we describe the simulation scenario used in this study. Section 3 describes path creation and resource management strategies that can be uniformly applied to all three adaptation approaches. The results from the simulation study are presented and analyzed in Section 4. We conclude in Section 5.

## 2 Methodology and Simulation Scenario

In order to study the performance of different adaptation approaches under different network conditions, we adopt a simulation-based methodology. Using a detailed simulator modeling a typical large-scale network (Figure 1) where multiple concurrently-active clients download media content from server sites. We provide an overview of our simulation scenario of interest below, deferring a detailed de-

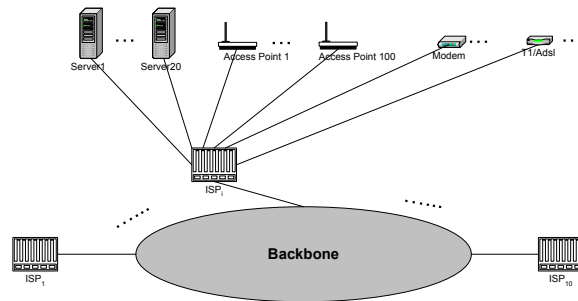


Figure 1: Experiment Network Topology

scription of the specific parameters to Section 4.

**The simulated network** contains multiple ISP regions, each of which is modeled as a gateway/proxy node providing a connection to the Internet backbone. The server and client nodes in the network have different computation capacity and are attached to one of these ISP nodes using various connectivity options.

**Application behavior.** The simulation models users connecting to server nodes from client nodes to download and display streaming media content. To display the received content appropriately, the throughput of a download path is required to be in some specific range (i.e. a certain number of frames per second). When the available bandwidth is not sufficient to meet the requirement, several operators can be used to reduce bandwidth consumption (e.g. by filtering the content). The most important performance metrics here is the aggregate time (across all sessions) when the throughput of the path is in the required range.

## 3 Approach-Neutral Strategies for Path Creation and Resource Management

Since the focus of this study is on the performance achievable by different adaptation approaches, a key challenge is to ensure that the comparison between approaches is as fair as possible while allowing each of these approaches to reach its best potential performance. This entails ruling out the following factors that are not inherent to the adaptation approach being evaluated, but which can nevertheless impact performance:

1. How should one manage shared network resources?
2. Given an allocation of network resources for a particular connection, how should one select operators and map them to network nodes?

Decisions on these two factors directly affect the performance achievable by an approach (and for reasons that have very little to do with the study’s central issue of adaptation location), what is required in each case are

strategies that permit each adaptation approach to achieve its best possible performance.

In the rest of this section, we present strategies that meet the above objective.

For the operator selection problem, in this study we used an algorithm proposed in our previous works in CANS. Given the allocated resources along a path, the algorithm can be uniformly applied to those adaptation approaches to construct a sequence of operators and mapped them to network resources so that the performance of end applications can be optimized. A detailed description of this algorithm appears in [3].

For the resource management, there are two main issues. The first question is how to allocate network resources among those continually adapting communication paths. The second question, primarily for path-based infrastructures, is how to distribute computation resources across the network so that it can achieve its best possible performance. For end-point or proxy-based approaches, the distribution of computation resources is trivial, i.e. either totally resides on server nodes or on proxy nodes.

### 3.1 Resource Sharing among Multiple Paths

To efficiently partition network resources among multiple paths while not introducing bias into the comparison among those adaptation approaches, we first need to understand how an adaptive path behaves under a shared network environment. Figure 2(a) shows the state transitions such a path goes through during its lifetime (the start and finish states are omitted to simplify the presentation). If the resources allocated to the path is sufficient for meeting its performance requirements, a path is deemed to be in *In-Range* state, i.e., its performance is in the desired range. When some of its resource shares change, either it continues to meet its performance requirements or not. In the latter case, there are two possibilities depending on whether or not the path can manage to reconfigure itself to go back into the *InRange* state. If it can, it enters a state called *Adaptation* until reconfiguration is complete. If not, it enters the *OutOfRange* state, from which it can transition to the other states only when the path's allocated resource shares have been raised. We call these three states *stable* to distinguish them from transition periods between these states.

From Figure 2(a), one can also observe that there are three different types of resource shares that can be associated with a path during its lifetime. The first one is the share value used in planning for a new configuration. In general, the greater the value, the better the generated plan will be. The second is the upper bound that the path is allowed to use, i.e. the allocated share. The last one is the actually used by the path at current time. Taking these three types of resource shares into consideration, to improve both individual path performance and throughput of

the whole network, an ideal scheme for allocating resource shares among multiple paths should be able to 1) maximize the value of the share value for planning purpose to produce good quality paths; 2) minimize the difference between the allocated and what is actually used to avoid resource waste. This is the basic idea of our scheme, which employs the following strategies: 1) when a planning is needed, a path is allowed to ask for increasing its allocated shares on network resources; 2) whenever a path enter a stable state, it is required to release unused resources. The state diagram of an individual path using this scheme is shown in Figure 2(b). The *Transition* state is for the path to send *allocation* requests to all resources along the path.

To realize this scheme, we need to specify how to (1) calculate the share for a path's request; and (2) adjust shares for existing paths.

**Allocation.** The allocation step maximizes the likelihood that an appropriate plan for a path can in fact be constructed. When the resource is underutilized, allocation requests result in a predefined MAX amount of resources being allocated. Information about this amount can either be provided by the path or specified by the resource. Note that since paths return unused resources, allocating a large share for planning purposes does not negatively impact resource availability for future paths. When the resource is oversubscribed (i.e., fewer than MAX resources are available), the allocation strategy used in this study basically partitions the resource fairly among different paths.<sup>1</sup>

**Adjustment.** The adjustment step is required in two situations: (1) when there are insufficient resources available to satisfy requests for a new share or increases to already allocated shares, and (2) when an allocated share is released.

For the first situation, a set of existing paths need to be selected and their shares will be reduced in order to accumulate a large-enough share for the requester. The allocation step above is for determining how large this share needs to be; the adjustment step decides which paths to take away resources from. Several different heuristic schemes can be employed to guide the latter process. Our scheme picks victims in decreasing order of the allocated shares, affecting paths that have a larger share of the resource. The selection is further restricted only to paths in the *InRange* state. The intuition here is twofold. First, such paths are more amenable to reconfiguration for staying in the desired range, as opposed to the paths in the *Transition* or *OutOfRange* states. Second, if resources are overcommitted, it is usually more acceptable to reject new connections than push existing paths into the *OutOfRange* state.

The adjustment in the second situation is simpler: when

---

<sup>1</sup>Allocation strategies for different types of applications are investigated in our another study.

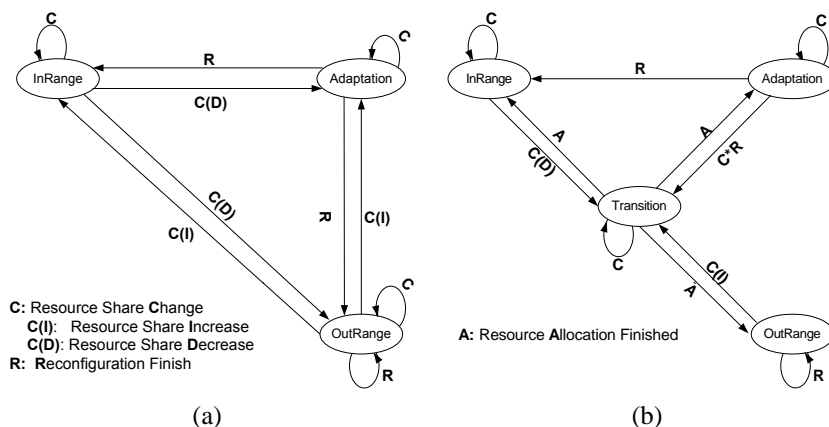


Figure 2: (a) General state transitions for an individual Path (b) In our scheme.

a share of the resource is released, it is used to increase the allocation of paths in the OutRange state up to the pre-configured maximum.

### 3.2 Resource Distribution Between Server and ISP Nodes

This question of how to distribute computation resources among server sites and ISP-level proxies is of interest primarily for path-based approaches, where different distributions of a same amount of total computation capacity among the network nodes can result in very different performance. Given the study’s objective, we need to distribute the computation resource in a way that path-based approaches can perform their best.

Our strategy is motivated by the observation that although path-based approaches can in general deploy operators on any node along a network path, usable nodes in practice are most likely a small set of strategic nodes such as ISP and gateway nodes. These kinds of nodes are also typically subject to administrative agreements between a higher-level network domain (e.g., the ISP) and a lower-level one (e.g., the server). Therefore, one can view the computation distribution problem as one of rearranging computation resources in a hierarchical network graph. Specifically, given a fixed *computation budget*, initially assumed allocated to nodes of a lower-level domain (servers in our case), the problem becomes one of moving a portion of the budget to nodes in a higher-level domain (ISP nodes in our case) to provide better overall performance by providing resource sharing across the network: overloaded servers see improved performance by taking advantage of spare resources at underloaded servers aggregated into a shared resource pool at a higher-level node in the network graph. Our strategy achieves this *without compromising* the performance of the domain from which computation resources are moved out.

The reassignment problem is illustrated in Figure 3.

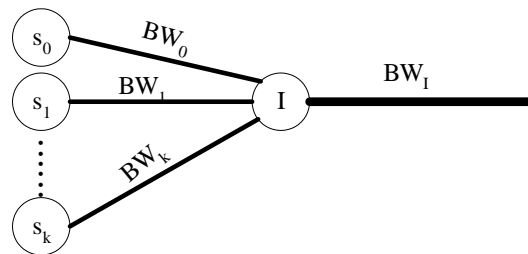


Figure 3: Hierarchical arrangement of servers and ISP nodes.

Each server  $s_i$  has a computation budget  $C_i$  (number of operations per time unit), and is connected to the ISP node ( $I$ ) via a link with bandwidth  $BW_i$ . The ISP node in turn is connected to the Internet backbone with a link of bandwidth  $BW_I$ . We use the terms *server link* and *ISP link* to distinguish between the two types of links above. The problem is one of determining what portion of  $C_i$  ought to be moved from  $s_i$  to  $I$ .

We only present the main intuition behind our algorithm here, more detailed description can be found in our technical report [4]. Given a load distribution for the various servers and a model of the computation and bandwidth resource utilization of client connections, the algorithm compares the aggregate number of connections that can be sustained on the servers assuming the server link becomes the bottleneck with that sustainable when the ISP link becomes the bottleneck. Note that because typically  $\sum_i BW_i \geq BW_I$  (as otherwise the bandwidth in the higher-level network domain would remain underutilized), there must exist some servers for which the ISP link becomes a bottleneck prior to the corresponding server link. For these servers, it is possible to move the unused portion of the computation budget to the ISP node. Our algorithm is used to identify these servers and calculate how many

resources can be moved out without compromising their performance.

Our description above considered a two level hierarchy (i.e., between servers and ISPs). However, this strategy can be easily extended to a hierarchically organized network domain with multiple levels ([4]).

## 4 Performance Implications of Adaptation Approaches

To study the performance implications of end-point, proxy-based, and path-based adaptation approaches, we simulate their behaviors on a large scale network in the context of a representative workload, which models clients downloading streaming real-time media content from server sites. Our simulation study investigates how these approaches perform under different loads, for different types of servers or clients, and for different client connectivity options. We start by describing the simulation settings, and then present the detailed results of these investigations.

### 4.1 Simulation Settings

**Application Performance Requirements.** In our simulation, every client downloads continuous JPEG image frames (with an average size of 4K bytes) from a server site. In order to display the received content appropriately, the throughput of a download path is required to be in the range of 10 to 16 frames per seconds,<sup>2</sup> and within this range higher data quality is preferred.

Possible operators that can be used with these paths include an *image-filter* and an *image-resizer*, which reduce bandwidth consumption by degrading image quality or reducing image size respectively. Both operators support ten different configurations; in each case the  $n^{\text{th}}$  configuration reducing image quality or size by a factor of  $n/10$ . Details about the computation overhead and compression ratio values for each operator configuration are omitted here for brevity.

The reason we choose to use this application in this study is because that we have used this application extensively in our previous experiments and have comprehensive knowledge of its performance characteristics. Besides, the communication paths of this application is reasonably complicated for different combinations of configurations and possible mapping.

**Network Characteristics.** The topology of our simulated network was shown earlier in Figure 1. For the results reported here, the network is assumed to comprise ten ISPs. Each ISP is connected to the Internet backbone via an OC48 (2.488Gbps) link and includes 20 media servers,

100 public IEEE802.11b (6.0Mbps<sup>3</sup>) access points, and a number of client sites.

Connectivity options for clients include T3 (44.73Mbps), T1 (1.544Mbps), ADSL (1.5Mbps), Dialup (56Kbps), and IEEE802.11b connections (via the public access points). The T3, T1 and ADSL links are assumed to have sufficient bandwidth for the media application, while Dialup connections are incapable of meeting throughput requirements without the use of compression operators. For wireless connections, available bandwidth is dependent on the load of the access point and may sometimes necessitate compression operators along the path.

At each ISP, we model the arrival of clients as a Poisson process; the arrival rate of clients is a parameter that can be adjusted to achieve different load levels. Once initiated, the duration of a download session is assumed exponentially distributed with an average of 1 minute.

Media servers within each ISP fall into one of two configurations. One fourth of the servers are categorized as *large sites*, with high-bandwidth connections to the ISP node (via an OC12 link operating at 622Mbps) and a computation budget uniformly distributed between 100 to 200 units.<sup>4</sup> The remainder three fourth of the servers are categorized as *small sites*, with lower-bandwidth connections to the ISP node (an OC3 link operating at 155Mbps) and a smaller computation budget uniformly distributed between 10 and 100 units.

**Adaptation Approaches.** Our studies considered five different adaptation approaches: the end-point approach, the proxy approach, an approach that use servers in addition to proxies (labeled as *server+proxy*), the path-based approach, and a path-based approach without reconfiguration support (labeled as *path-reconfig*). The last approach clarifies the benefits of dynamic adaptation; communication paths in this approach can adapt to different network conditions only at path-creation time. As mentioned earlier, the first four approaches represent different constraints on where adaptation is allowed. For the *end-point* approach, only the server node and the client node of a communication path can be involved in adaptation. The *proxy* approach is allowed to use client nodes and client-side ISP nodes. The *server+proxy* approach represents an intermediate point, which, in addition to nodes used by the proxy approach, can also use server nodes for adaptation. Finally, *path* approach can use all four nodes along a communication path.

To make a fair comparison between these approaches, our studies used the same total computation resource bud-

<sup>2</sup>This means the media player must display at least 10 frames per second for a satisfactory user experience, but can not handle more than 16 frames in one second.

<sup>3</sup>We assume a 60% bandwidth utilization of an IEEE802.11b network.

<sup>4</sup>One unit is normalized as a computer with a Pentium III 1GHZ processor and 256MByte 800MHz RDRAM.

get in each case.<sup>5</sup> In the end-point approach, all resources reside on server sites. For the proxy approach, all resources on server sites are aggregated on the ISP nodes they attach to. For the server+proxy approach and the path approach, a portion of the computation budget of every server site is moved to its ISP node using the strategy described in Section 3.2. The redistribution assumes that requests from clients are uniformly distributed among all server sites. Our study also examines situations where this assumption does not hold, providing insights into how performance is affected by inaccuracies in client traffic models.

**Performance Metrics.** Our simulations characterize two major performance metrics. The first measures the aggregate time of all paths when the throughput of a path is in the desired range. We refer to this as the *InRange* time, i.e., the time where paths stay in the *InRange* state of the state diagram shown in Figure 2. Another reasonable metric is the aggregate *InRange* time that is further weighted by data quality of the communication paths. Since we observed a similar trend between the weighted and unweighted metrics in our study, we report only on results for the unweighted metric below.

The second performance metric is the total number of connection failures due to insufficient resources. Connection failures result from admission control, which actively rejects any incoming connection request if the initial planning cannot produce a communication path that meets the performance requirements.

In addition to aggregate data for the whole network, we also collected data for different types of servers and clients to further examine how different adaptation approaches perform from the perspective of individual servers or clients. In particular, we report on data for server sites that have the maximum or minimal computation budget, and for clients that use different connectivity options.

Since the goal of our study is to investigate the best possible performance of each of the adaptation approaches, strategies that can be uniformly applied to improve performance for all approaches (such as support for multicast and caching) are not taken into account in the experiments.

Path reconfiguration and its overhead in our study is modeled after a general protocol proposed in the context of the CANS infrastructure [5].

In the rest of this section, we first report on the performance achieved by different adaptation approaches with client traffic uniformly distributed among server sites for a particular client connectivity profile. We then separately examine how performance is affected by non-uniform traffic distribution, and when the client connectivity profile is

<sup>5</sup>The computation budget refers only to resources available for path transcoding and compression operations. Sufficient resources are assumed available on the server and proxy nodes for data retrieval from disk and forwarding through the protocol stacks.

changed. In each case, we simulate the network for 4 minutes, recording data only for sessions that are started within the last 2 minutes, i.e. after the network reaches a stable state (recall the average length of a session is 1 minute). The measurement ends at the 4 minute mark.

## 4.2 Performance under Uniform Load Distribution

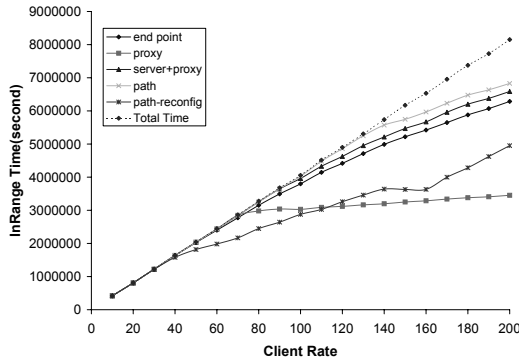
This configuration uniformly distributes client requests among all server sites, varying client arrival rates at each ISP from 10 to 200 clients per second. These rates correspond to 6000 to 120,000 active paths simultaneously existing in the network. The client connectivity profile is fixed as follows: 25% use links with sufficient bandwidth (T1/T3/ADSL), 25% use Dialup, and the remaining 50% use wireless connections. We examine the impact of changes from this profile later in Section 4.4.

The performance achieved by different adaptation approaches for this configuration is shown in Figure 4. Plots (a) and (b) respectively show values for the *InRange* time and number of rejected connections, aggregated over all servers. Plots (c) and (d) show the *InRange* time for the server with the maximum computation budget (199 units), and for that with the minimum budget (10 units). Figures 4(e) and (f) show the *InRange* time (normalized with respect to the total session time) seen by clients who use T1/T3/ADSL links and those that use dialup/wireless connections.

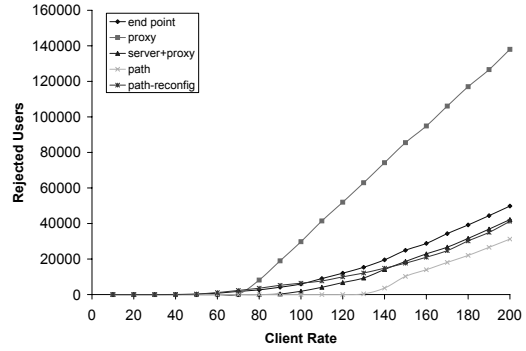
**Analysis of Aggregate Performance.** From Figures 4(a) and (b), it can be observed that all four adaptation approaches that provide reconfiguration support perform very well when the network is lightly loaded. However, after the load increases to some level (client rate=80 in Figure 4(a)), the performance of the proxy approach is the first to reach saturation. This can be explained by bandwidth waste in the network core because adaptation using the proxy-based approach can only occur on the node before the last hop.

Compared with the proxy approach, the end-point approach performs better (with higher *InRange* Time and fewer connection failures), especially after the “saturation” point of the proxy approach. This is expected because the end-point approach uses server sites to do image filtering or resizing, and does not waste bandwidth on the network links. However, it can also be observed from the Figure 4(b) that the end-point approach starts to reject connections earliest among all approaches, even when the network is lightly loaded. These rejections mainly come from clients that use weaker links such as Dialup to access small sites with limited computation capacity.

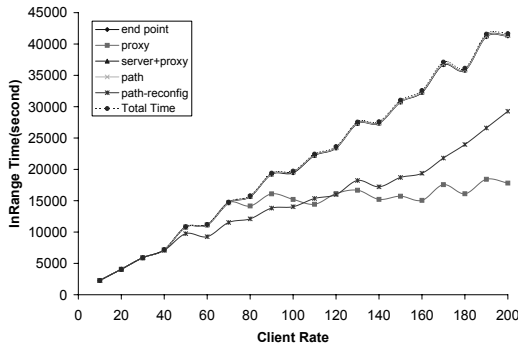
Figures 4(a) and (b) also show that the path-based approach provides the best performance at all load levels. From our result, the *InRange* time of the path-based ap-



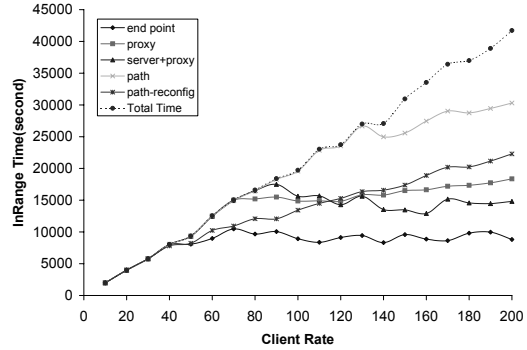
(a) Aggregate InRange Time



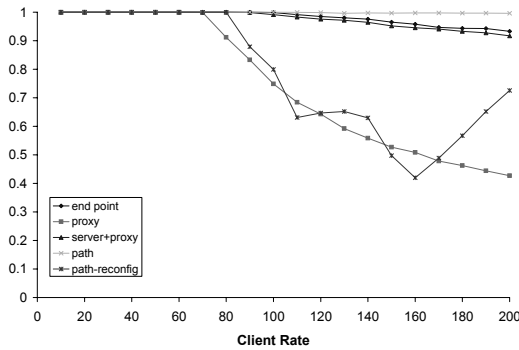
(b) Aggregate Connection Failures



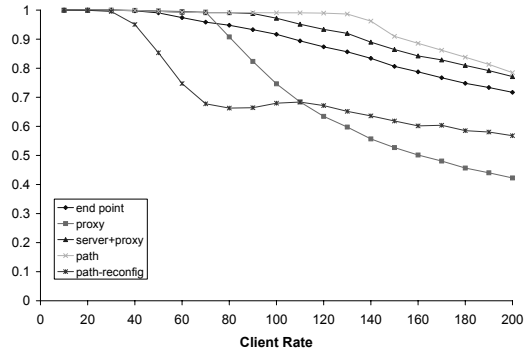
(c) InRange time for Server with Max. Budget



(d) InRange time for Server with Min. Budget



(e) Normalized InRange Time for T3/T1/ADSL Clients



(f) Normalized InRange Time for Dialup/Wireless Clients

Figure 4: Performance under Uniform Load Distribution.

proach is up to 12% and 97% higher than that of the end-point approach and the proxy approach respectively. The number of connection failures of the path-based approach is also much lower. For example, for a client rate of 200 connections/second, the end-point approach rejects 59% more connections and the proxy approach rejects about 343% more connections than the path approach. The reason for this behavior is because of the flexibility of the path-based approach that any resources along a path can be used to do adaptation: on one hand, similar to the end-point approach, the path-based approach can utilize up-

stream nodes along a communication path to ensure that network bandwidth is not wasted; and on the other, similar to the proxy approach, the path-based approach can set up shared resource pools across the network, permitting overloaded servers to benefit from spare computation resources elsewhere.

The performance of the server+proxy approach falls between the path-based approach and the end-point approach, which verifies that allowing adaptation to happen on even one more node in the middle of the communication path can improve overall performance.

**Performance of Different Server Sites** Comparing between Figures 4(c) and (d) allows us to draw conclusions about how the different adaptation approaches perform from the perspective of connections targeting servers with higher or lower computation budgets. The results indicate that the path-based approach performs as well as the end-point approach for the largest server, and performs the best for the smallest server. This can again be explained by the flexibility brought by resource sharing and being able to use any nodes along a path to do adaptation.

**Performance of Different Clients** Figures 4(e) and (f) show the performance of the adaptation approaches from the perspective of different client classes, i.e., clients connected to the network with sufficient bandwidth versus those that use weak connections. We can observe that while the proxy approach exhibits more or less uniform behavior, the end-point approach demonstrates considerable preference for clients with better connectivities over others. The path approach, in addition to providing the best performance, uniformly supports different classes of clients until one runs out of computation resources beyond a certain load level. At this point, all approaches end up rejecting more clients with weak connectivity because they require more computation than what is available along the paths.

**Performance Impact of Dynamic Reconfiguration** The plots in Figure 4 also show that there is a considerable performance penalty incurred for disallowing reconfiguration after the path has been created. This validates the need for a *reactive* mechanism to cope with dynamic changes. In general, different types of paths may have different requirements on network resources (e.g. some of them may require more bandwidth while others may need more computation). As load changes, it is necessary to adjust allocated shares of existing paths in order to accept more connections.<sup>6</sup> Without reconfiguration, adjustments for one path may end up pushing other paths out of the required range, and thereby negatively impact overall performance.

### 4.3 Performance under Non-Uniform Load Distribution

This configuration examines how different adaptation approaches perform when connection requests from clients are directly non-uniformly towards servers. Similar to load patterns observed on the Internet, we assume a “hot-spot” model, where a small number (20%) of servers (the hot-spots) receive most (80%) of the requests from clients. We further ensure that the average load of large sites (i.e., sites with an OC12 link with computation budget uniformly dis-

<sup>6</sup>One can argue that using reservations may eliminate the need for dynamic adjustments, but such approaches usually have poor throughput as load dynamically changes

tributed in the range [100,200]) is about 4 times the average load of small sites (i.e., sites with an OC3 link with computation budget uniformly distributed in the range of [10,100]).

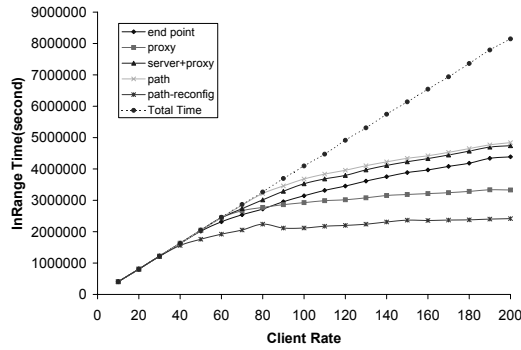
Figures 5(a)–(d) show the performance achieved by the different approaches. There are several observations that one can make here. First, focusing on aggregate performance, we see that the overall ranking of performance among these adaptation approaches remains the same as in the uniform distribution case. However, the total In-Range time is noticeably lower than the values we saw in Section 4.2. This is expected because the overloaded hot-spot servers cause increased connection failures due to limitation on server links. Second, the path-based approach maintains the best performance all the way, which validates the path-based approaches is suitable for unbalanced/unexpected load situation. Third, the relative performance of the path-reconfig approach is worse than seen earlier. This verifies our intuition that such an approach performs poorly for unbalanced load: due to the absence of reconfiguration, existing paths cannot be adjusted so they take advantage of surplus resources in lightly loaded regions of the network.

Looking at the performance from the perspective of servers with the maximum and minimum computation budgets (Figure 5(c) and (d)), it can be observed that the path-based approach outperforms all other approaches. This again verifies the benefit of resource sharing in the network: overloaded sites can always take advantage of spare computation resources elsewhere. This is true even for sites that have plenty of computation resources.

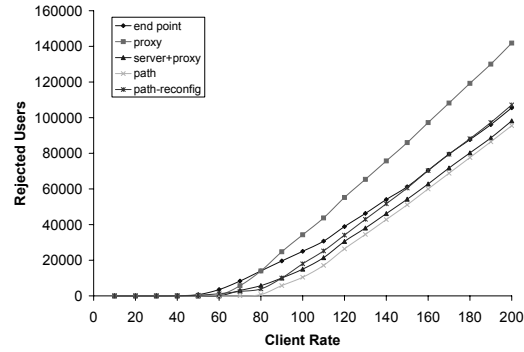
A more interesting point with this set of results is that they correspond to the same resource distribution between server and ISP nodes as in Section 4.2, namely one that *assumes a uniform load distribution*. This is relevant because load distributions at run-time are likely to be different from that considered when deciding about how to provision resources in the network. Our results show that the path-based approach still performs very well even with an inaccurate knowledge of load distribution. This robustness mainly comes from the shared resource pools across the whole network that act like “buffers”, absorbing any negative impact because of the unexpected load.

### 4.4 Performance under Different Client Connectivity Profiles

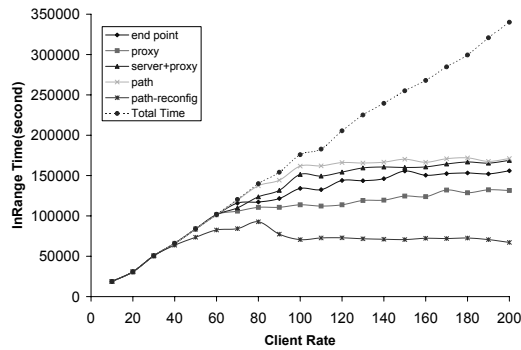
In this configuration, we examined how the different adaptation approaches perform when different fractions of clients use different connectivity options. The simulations run with the same settings as in Section 4.3 with only two differences: the client arrival rate was fixed at 100 users per second, and we varied the percentage of clients that use weak connections (dialup or wireless) from 0 to 100 per-



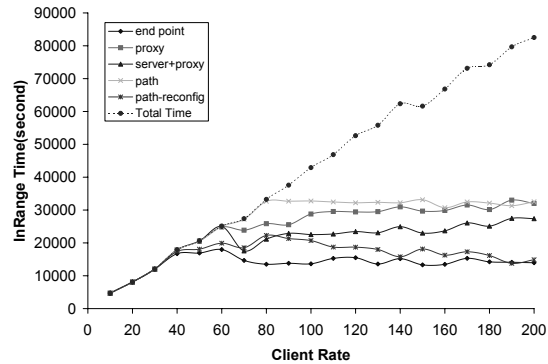
(a) Aggregate InRange Time



(b) Aggregate Connection Failures



(c) InRange time for Server with Max. Budget



(d) InRange time for Server with Min. Budget

Figure 5: Performance under Non-Uniform Load Distribution.

cent (the ratio between numbers of clients that use dialup and wireless connections was maintained at 1:2).

Figure 6 shows the performance results. One can observe that among the four approaches with reconfiguration support, the end-point approach, because of the lack of resource sharing, is the only one that exhibits decreasing performance as more clients use weak connections while the other three approaches achieve relatively stable performance across different configurations.

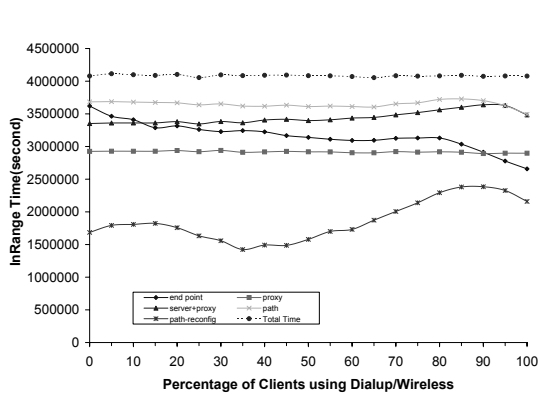
It can also be seen that the path-reconfig approach performs relatively better when the client connectivity profile is more uniform. This can be explained as follows: as more paths exhibit similar behavior (i.e., have similar resource requirements), there is lower likelihood that an existing path will get pushed out of its required performance range by the arrival of a new connection. Stated differently, the more heterogeneous the environment, the larger the need for dynamic reconfiguration.

#### 4.5 Summary of Simulation Results

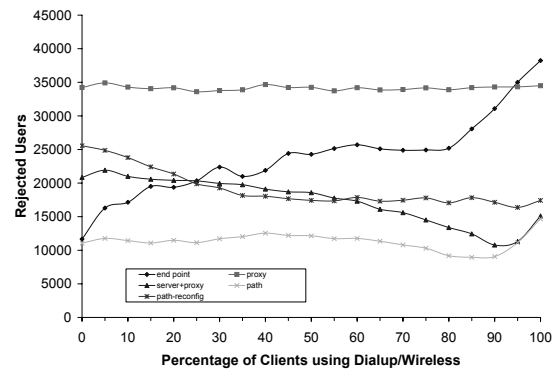
The main results from our study are summarized below:

1. Dynamic reconfiguration is important for the performance of both individual paths and the whole network.

2. The end-point approach usually works well with server sites that have a large amount of computation resources and for clients that connect to the network with relatively high bandwidth links. However, due to the lack of resource sharing, servers that have limited computation capacity or clients that use weak connections usually suffer from poor performance.
3. The proxy approach provides limited resource sharing at proxy sites so that it works well with small server sites or clients that have weak connectivity. However, constraining the adaptation to only occur at the edge of the network can cause considerable resource wastage in the network.
4. For the path-based approach, shared resource pools are set up across the whole network; adaptation can be conducted anywhere along a network path as long as there are network resources available. This provides the most flexibility for overloaded network portions to benefit from spare computation resources elsewhere. Distributing resources across network does not cause visible performance penalty with effective resource management strategies. In summary, this approach provides the best and the most robust performance un-



(a) Aggregate InRange Time



(b) Aggregate Connection Failures

Figure 6: Performance under Different Client Connectivity Profiles.

der different network configurations.

## 5 Conclusions

In this paper, we have investigated the performance implication of different adaptation approaches that have different constraints on adaptation location: the end-point approach, the proxy approach and the path-based approach. By conducting a series of simulation-based experiments using different network configurations we have shown that there are well-defined network environments under which each of the adaptation approaches delivers its best performance. More importantly, each approach, with one exception – the path-based approach, also exhibits performance shortcomings in certain environments. The path-based approach ends up delivering the best and most robust performance under different network configurations, and for different types of servers and clients because of its ability to carry out adaptation along the entire path and pool resources across the while network. Thus, despite their somewhat increased complexity, path-based approaches appear the most promising for delivering high performance in environments comprising diverse network resources and where additionally the characteristics of these resources are subject to constant change.

## References

- [1] E. Amir, S. McCanne, and R. Katz. An Active Service Framework and its Application to Real-time Multimedia Transcoding. In *Proc. of the SIGCOMM'98*, August 1998.
- [2] A. Fox, S. Gribble, Y. Chawathe, and E. A. Brewer. Adapting to Network and Client Variation Using Infrastructural Proxies: Lessons and Prespectives. *IEEE Personal Communication*, August 1998.
- [3] X. Fu and V. Karamcheti. Planning for network-aware paths. In *Proc. of the 4th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS'03)*, November 2003.
- [4] X. Fu and V. Karamcheti. Why path-based adaptation? performance implications of different adaptation mechanisms for network content delivery. Technical Report TR2003-843, Computer Science Department, New York University, July 2003.
- [5] X. Fu, W. Shi, A. Akkerman, and V. Karamcheti. CANS:Composable, Adaptive Network Services Infrastructure. In *Proc. of the 3rd USENIX Symposium on Internet Technologies and Systems (USITS)*, March 2001.
- [6] S. D. Gribble and et al. The Ninja Architecture for Robust Internet-Scale Systems and Services. *Special Issue of IEEE Computer Networks on Pervasive Computing*, 2000.
- [7] A. D. Joseph, J. A. Tauber, and M. F. Kasshoek. Mobile Computing with the Rover Toolkit. *IEEE Transaction on Computers:Special Issue on Mobile Computing*, 46(3), March 1997.
- [8] A. Nakao, L. Peterson, and A. Bavier. Constructing End-to-End Paths for Playing Media Objects. In *Proc. of the OpenArch'2001*, March 2001.
- [9] Brian D. Noble. *Mobile Data Access*. PhD thesis, School of Computer Science, Carnegie Mellon University, 1998.
- [10] A. Vahdat, M. Dahlin, T. Anderson, and A. Aggarwal. Active names: Flexible location and transport of wide-area resources. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems (USITS)*, October 1999.
- [11] M. Yavis, A. Wang, A. Rudenko, P. Reiher, and G. J. Popek. Conductor: Distributed Adaptation for complex Networks. In *Proc. of the Seventh Workshop on Hot Topics in Operating Systems*, March 1999.