

# Location Selection for Active Services

Roger Karrer and Thomas R. Gross

Laboratory for Software Technology  
Departement Informatik  
ETH Zürich  
CH 8092 Zürich  
Switzerland

## Abstract

*Active services are application-specified programs that are executed inside the network. The location where the active service is executed plays an important role. The dynamic behavior of networks requires that the selection of the most suitable location to instantiate a service is done at run time. To dynamically place an active service, information about the network (topology, bandwidth) and application (type of the service) is necessary.*

*This paper describes a method to dynamically search for available active service locations in the Internet. To be deployed in the current Internet, a solution is required to (i) scale well to large networks, (ii) to demand as little changes to the Internet as possible, especially not at lower network layers. Finally, the solution must be flexible and customizable to take application requirements into account.*

*The proposed solution makes use of the routing path between two endsystems. Active service locations that are located close to the routing path are then found via DNS queries.*

*The evaluation shows that the application pays an overhead at startup time. For applications that can tolerate a startup delay, we show with two experiments using a video application that the quality of the application can be increased by dynamic placement of active services.*

## 1 Introduction

Active services are application-specific programs that are executed on special-purpose nodes within a network [2]. Several types of active services, e.g., adaptive and multicast services, have been presented for different types of applications in previous work. Fox et al. [6] describe an adaptive (transcoding) proxy for images. Hemy et al. [9]

describe an intelligent frame-dropping filter (service) that adapts a video stream. Fontana [5] implements a network-based video multicast service.

These projects focus on the usage of the active services and evaluate their work with respect to the application quality. They are able to show the benefit of the active services under some conditions. One of these conditions is the correct placement of the service. Fox et al. [6] state that the adaptive filter must be placed before a bottleneck and not after. Similarly, a multicast filter must be placed where two data streams separate. However, none of these projects has addressed the problem of correctly placing a service in a real network. They assume that they know the location of the bottleneck and they assume that the bottleneck stays at the same place. The latter assumption may be true for connections that include wireless or modem links, but it is not true in general.

To place an active service, information about the network and the application is needed. The network information includes available locations where active services can be instantiated (assuming that not all nodes in a network offer this possibility) and information about the available resources (e.g., the available bandwidth between such nodes). Application knowledge is needed because the type of service and the requirements of a particular application determine the correct place of a service. The location strategy for a service of an adaptive type is different from that of a multicast services. And while the available bandwidth may be sufficient for one movie, another movie may require the placement of an adaptive service before a bottleneck.

Learning about active service locations and selecting the most suitable location is far from easy, unless a service management system is available for active services. Previous efforts in defining such systems such as Jini [11] or SLP [8], have focused on local area networks or have not yet been deployed widely. These architectures are hard to deploy in large and dynamic networks, such as the Inter-

net: they often scale badly to a large number of nodes, and fluctuations in the resource availability make it difficult to keep the resource information accurate. Nevertheless, the Internet is an excellent target for the deployment of active services. In contrast to local area networks, where the resource availability is often sufficient, the Internet is often *the* bottleneck in an application. Any solution to location selection in the Internet must therefore be (i) scalable, i.e. the needed information must be retrieved quickly even if there is a large number of nodes involved, and (ii) practical, i.e. the solution should only use information and tools that are currently available. If such a solution was available, more information about the usage of active services could be gained. Without such a solution, the benefit of active services cannot be assessed.

The paper is organized as follows: Section 2 presents the problem of locating services in more detail. It introduces the most important definitions, our network and application model and the assumptions we make. In Section 3, our path-based approach for finding service locations and selecting the most suited location is presented. The solution is evaluated in Section 5. After a discussion of related work, we conclude in Section 7.

## 2 Locating Services

Before an application can use active services, it must search for available locations on which active services can be instantiated, and it must select the most suited location if several locations are available.

The problem of locating active service locations in local area networks has been addressed and several protocols have resulted from these efforts, e.g., SLP [8] or Jini [11]. Jini manages not only services, but an application can also define search criteria. This feature could be used to select the most suitable service location. However, these solutions are not well suited for large networks, mainly because they use multicast or broadcast mechanisms, and these mechanisms do not work well in the Internet.

The selection of the most suited active service location depends on parameters that are purely application specific. Different service types require different strategies to place the filter: a multicast filter is placed where two data stream diverge, whereas the placement of an adaptive filter depends on the bottleneck of a connection. In addition, some parameters depend on constraints that are only known at run-time. The placement of a multicast filter depends on the actual distribution tree. The placement of an adaptive filter may depend on the bandwidth requirements of the application. Different videos, e.g., have different bandwidth requirements. While the network may satisfy the requirements of one movie, it may not be sufficient for another. In the first case, no adaptive service is needed at all. The prob-

lem of the location selection is therefore to take application- and network-parameters into account and select a service location at run time.

The process of location selection is transparent to the application. Lessons learnt from Remos [4] show that it is possible and desirable to clearly separate networks and applications. First, it is much easier to integrate new applications if the application does not have to deal with the details of networks. To integrate Web applications, e.g., a system that supports the location selection should offer an interface that accepts a URL and returns a new URL that contains one or more active service locations. Second, no difference should be visible to the application whether it works on a LAN or a WAN, although the mechanisms that search for available services may be very different.

The paper concentrates on the location selection when an application starts up. The idea is similar to server selection: before the application transmits the data, it checks for available service locations, instantiates them and only then starts the transmission. However, the idea of location selection is not limited to this scenario. If the application runs longer than just a few seconds, it is possible that the resource availability changes. The application may then be forced to change the service location. If, e.g., the bottleneck of a connection shifts, a new location for an active service must be searched. One such scenario for multimedia applications is a *dynamic handoff of streams* [10]. The application is forced to switch from a low-bandwidth connection to a new one that offers better performance. A change at run-time may also be necessary for multicast applications: when a new client joins an exiting transmission, a new multicast service may be instantiated.

One of the challenges is to present a solution to select active service locations in the Internet although there exist no specialized nodes that support active services. We therefore make several assumptions and simplifications; they have to be revisited once active service nodes become widely available.

First, we model active service locations by endsystems. Because it is unlikely that there will be a time when all nodes in the Internet offer active services, we require that active service nodes can be distinguished from other nodes. We assume that active service locations have a special host name or a particular IP.

Second, we assume that these nodes are located close to the routers, but not on the routers. By closeness we mean that the connection between the active service host and the router provides high bandwidth and low latency compared to the performance of the connections to the Internet. There are several reasons for not putting active services on the router. First, it would require a change in the Internet architecture. It is much easier to hook in new nodes than changing the existing routers. Second, active services may use up

resources that are then no longer available for routing. The routing performance is one of the important factors of the Internet performance and must not be reduced by the active services. Finally, several active service nodes may be attached to a single router to increase the capacity for active services. It is easier to attach such a cluster to the router than having multiple routers. Having multiple routers that may be traversed in sequence may slow down the transmission as well. Following the definition of [14], we call such a cluster a *cloud*.

### 3 Location Selection

#### 3.1 Path Discovery

The original design philosophy of the Internet was to clearly separate the endsystems from the network. A network and its task, the data transport, should be hidden from the application. Therefore, when an application starts up, its view of the network is that of a black box, as depicted in Figure 1(a). This figure shows two clients that want to download data from a server. Such a scenario could be a teleteaching session or a video multicast session.

The first problem in the location selection is to get information about the available active service locations in the network. In a local area network, it is possible that some system is available that hosts this information. Extending such a system to manage services in the whole Internet is likely to fail for scalability reasons.

Scanning the whole Internet for available locations is unrealistic. First, because it takes too long for the scan, and second, the time to select the most suited location is likely to scale with the number of locations that are available. For these two reasons the search space must be limited to that part of a network that is of interest to the application.

As indicated in Figure 1(a), the application is most interested in that part of the network that lies between the application components (server, clients). Nodes that lie "behind" the server or the client are not of interest.

One way to span the search space for active service locations between two endsystems is to follow the routing path. The routing path can be detected from the endsystems and therefore provides a practical solution. It also has other advantages. The routing path is usually the shortest connection between two endsystems in terms of the number of hops. It is therefore guaranteed to be a scalable solution (the average path length is almost independent of the size of the Internet), and the information can be retrieved quickly.

However, there are two drawbacks that must be taken into account. First, applications are usually interested in high-bandwidth or low-latency connections. These application-specific metrics are not considered in the packet routing. Savage et al. [15] show that alternate paths can be

found that often provide better performance than the routing path. One way to take application-specific metrics into account would be to construct alternate paths as outlined by Savage et al: an alternate path is constructed by concatenating two paths. However, in contrast to this approach, since there exist active service nodes, the node that connects the two paths can be placed inside the network. These services could also provide support for application-specific metrics, such as bandwidth measurements.

The second problem is that metrics such as bandwidth and latency tend to vary. It is therefore hard to find the optimal path. In the search for active service locations a tradeoff must be found between the search time and the quality of location that is finally selected. If the application runs for a long time, a longer search time may be considered.

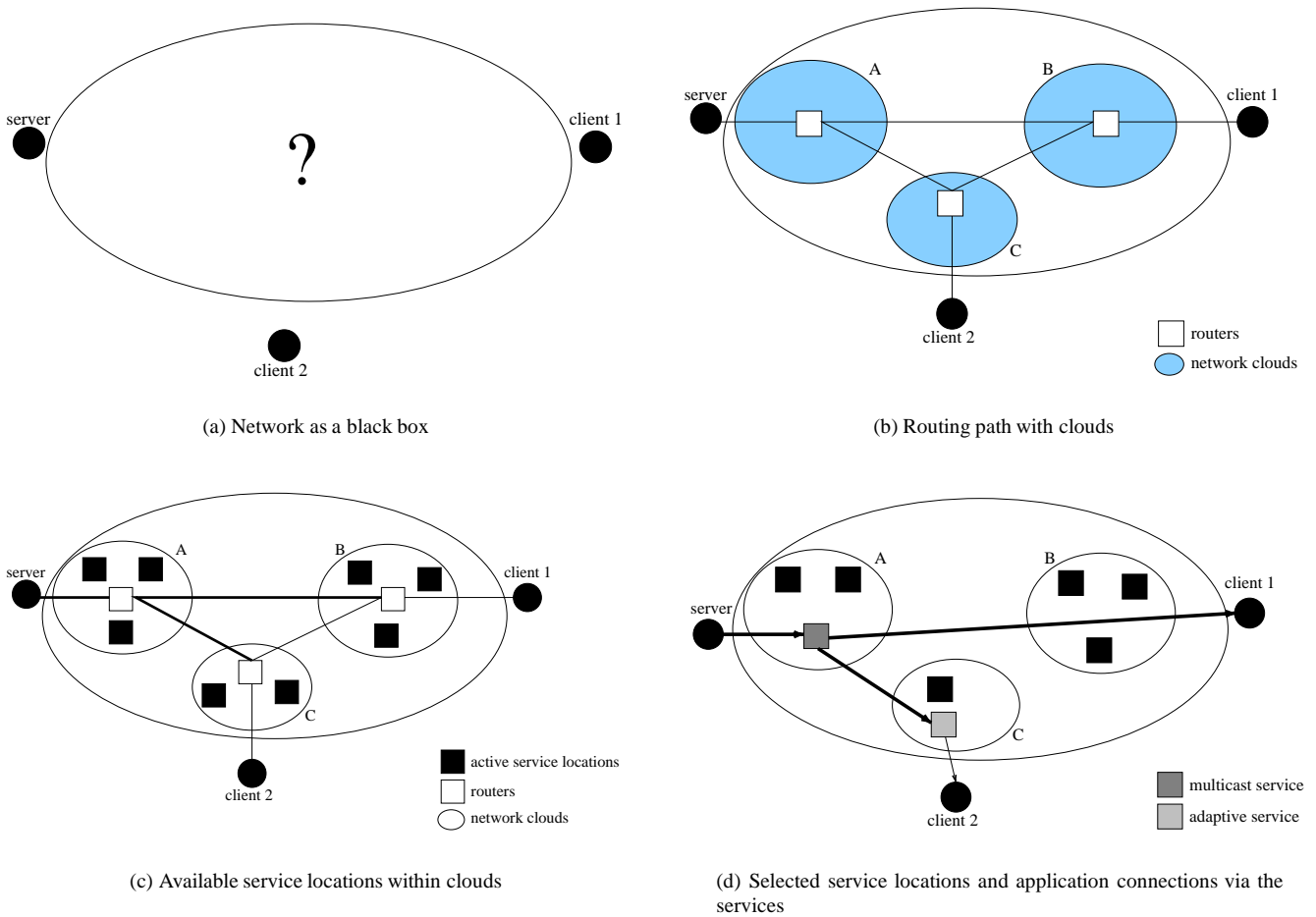
Figure 1(b) sketches the situation after the path discovery. For reason of simplicity, not all routers are shown. One observation is that no active service locations are available after the path discovery because we have required that the active service locations must not be located on routers. The lack of nodes that offer active services must be addressed in a next step.

#### 3.2 Service Location Discovery

From the path nodes, the application must find the active service locations that are close to the routers, i.e., that belong to the same cloud.

Clouds can be implemented in various ways. To implement the clouds, we make use of the Domain Name System (DNS). Although the DNS is a purely organizational collection of hosts and is not based on performance metrics, the hosts within a domain usually have a better connectivity than to the Internet.

The use of the DNS system has several advantages. First, the DNS provides all functionality needed to find active service locations close to the path. For each router, the corresponding cloud must be found. DNS allows to query the domain of a network node by its node name or the IP. Similarly, all nodes that belong to a certain domain can be listed as well. Because we require that the active service nodes can be recognized by the hostname or the IP, these special nodes can easily be filtered out from the DNS reply. Second, because the definition of a cloud does not rely on dynamic parameters, no dynamic measurements of the closeness are necessary and the networks are less loaded. In contrast to the paths, which vary over time and for each application, the location of the active service locations within a network do not change often. A management system for active services within a LAN is therefore more feasible than for the Internet. Another advantage is that the DNS system is set up hierarchically. These hierarchies allow the definition of different levels of closeness. When no active service location



**Figure 1. The four phases of the location selection.**

can be found close to a router, the next hierarchy level can be considered. In addition, the hierarchy limits the number of nodes that can be found with a single query. The available locations can be found faster, and the final selection of the most suited active service location is faster if the number of nodes is kept smaller.

In Figure 1(c), the situation after the service location discovery is shown. Active service locations that are available close to the routers are visible. The number of these locations may vary from cloud to cloud.

Figure 2 sketches the algorithm of the path and the service location discovery. After the path discovery, every router is replaced by the corresponding domain. Nodes without an entry in the DNS are removed. Then, each domain is queried for the nodes in its domain and the nodes that provide active services are extracted. Finally, the domain is replaced by the active service nodes. The result of the service location discovery is a graph-like structure.

```

path = trace(source, destination);
for each hop in path {
  domain = DNS.map(hop);
  if (domain==null) path.remove(hop);
  else path.replace(hop, domain);
}
for each domain in path {
  nodelist = DNS.list_nodes(domain);
  services = filter(nodelist);
  path.replace(domain, services);
}

```

**Figure 2. Algorithm for resource discovery.**

### 3.3 Service Selection

The previous steps in the resource selection are independent of the application. The selection of the most suited location to instantiate an active service depends on application-specific information. To select a node, the graph is first traversed to find the shortest path between two peers. The metric that is used to label the links depends on the application requirements. Bandwidth is probably the most often used metric, but the error rate for wireless connections or jitter for multimedia applications may also be used.

Depending on the metric and the implementation of the previous phases, not all information may be available. Our implementation, e.g., only measures the latency of the links directly. To label the links with the available bandwidth, additional measurements are necessary. The measurements must be done only at a cloud level, but not for every single active service node, because the cloud definition implies that all nodes within a cloud have a similar bandwidth range, compared to the inter-cloud connections. This simplification reduces the load on the network and the nodes, is faster than measuring the bandwidth to every node and allows applications that share the same path to share the bandwidth information.

After the labeling of the links, a shortest path algorithm is used to find the path that best matches the requirements of the application. The service locations are then typically chosen on these paths. Another application-specific parameter, namely type of the service, plays an important role for the placement. To place an adaptive filter, the bottleneck of the path is identified and the service is instantiated at the closest location before the bottleneck. Theoretically, any node before the bottleneck could be used, but the location that is closest to the bottleneck has two advantages: first, adaptive filters are often feedback based. The feedback loop is smaller if the filter is close to the bottleneck. This placement increases the responsiveness of the filter. Second, if the data is multicasted or cached (at setup time or also later), the data stream is transmitted at the highest quality up to the bottleneck. If the multicast service is instantiated before the bottleneck, it receives the best available quality. To place a multicast service, the shortest paths are traversed from the server towards the clients until the two streams separate. For efficiency reason, the selection of the service location can be combined with the graph traversal for the shortest path.

Figure 1(d) shows the result after the placement of the services. As indicated by the thin line, the resource measurements have shown that the connection between cloud C and client 2 has a low quality. The graph, formed after the location discovery, has been traversed from the server to each client and the shortest paths have been calculated. Obviously, the shortest path to client 1 goes via clouds A

and B, to client 2 via A and C. While searching the shortest paths, cloud A is identified to be the location for a multicast service. If more than one location is available, additional criteria may be taken into account, such as the load on the node. In addition to the multicast service, an adaptive service must be placed at cloud C to adapt to the slow link from cloud C to client 2.

## 4 Implementation

The proposed algorithms have been implemented as follows: the path discovery and the service location discovery have been implemented as an extension to the Remos system [4]. Remos consists of collectors that are responsible for gathering information about the network topology and the available resources. Different collectors are available for local area networks (using SNMP) and wide area networks (using traceroute and nettest). The information is merged into a graph-like structure by a modeler. Remos has been extended to map routers to clouds and to replace clouds by the active service locations.

A small framework on top of Remos is responsible for the selection of the most suited service location. The framework contains the graph traversal algorithms as well as selectors for adaptive and multicast service locations. Additional selectors can easily be implemented for other service location strategies.

The collaboration of this framework and Remos has been implemented slightly different from the described algorithm for efficiency reason. After the path discovery and the mapping of the routers to the domains, not all domains are immediately searched for active service locations. Then, in contrast to the description, a first selection is performed at the cloud level, i.e., the resources are measured between the clouds, the shortest path is calculated and the cloud where the service should be placed is determined. Only this cloud is then queried for available service locations. This change in the algorithm improves the performance in two ways. First, the shortest path algorithm is faster because the number of nodes (the clouds) is smaller than the number of active service locations. Second, not all clouds are queried for the available service locations.

Finally, two existing applications have been extended to use active services: a video application that consists of a multicast service and an adaptive frame-dropping filter [9, 5] and an image application that contains an image filter, similar to the filter presented by Fox et al. [6]. The only extension to the applications are the call to the framework in which the URL and the desired metric for the evaluation are passed. After the evaluation, the system returned a new URL that contained the additional service locations.

Client	Path			DNS mapping			Graph eval time(s)
	time (s)	hops	domains	nodes	time	service locations	
ethz.ch	0.03	1	1	28710	0.9	80	0.12
epfl.ch	0.28	8	3	14219	1.22	5	0.06
cs.cmu.edu	1.53	13	6	5658	0.68	30	0.10
uc.pt	2.25	16	5	487	0.87	2	0.03
uva.es	3.56	13	5	n.a.	n.a.	2	0.07
ufmg.br	3.55	15	6	n.a.	n.a.	3	0.08

**Table 1. The overhead of Internet measurements.**

## 5 Evaluation

### 5.1 Overhead

In a first step we try to quantify the overhead of the location selection. One dimension of overhead is the amount of resources that is needed for the measurements. This overhead is described in the evaluation of Remos [4].

The overhead for the application is the time needed to select the location. This time depends on various parameters that depend on a set of parameters: the path length, the number of clouds, the number of active service nodes within the clouds, and the application requirements. The goal of these measurements is to provide only a rough idea of the overhead of a particular application. Nevertheless it should show for which classes of applications a server selection is suited.

To measure the overhead, the video application is used [5]. A video server is located at ETH. 6 clients are placed at different locations (ethz.ch, epfl.ch, cmu.edu, uc.pt, uva.es, ufmg.br). Each site downloads the same video in turn and the length of the different steps is measured. Table 3.3 shows the time needed for the different steps. The path discovery is influenced by the path length. The average path length in the Internet is about 15 nodes, so that an overhead of 1 to 2 seconds can be expected. The time for the DNS query (list all hosts in a domain and filter out the service locations) is also in the order of 1 second. The number of service locations indicates the number of active service locations whose host name corresponds to the search criteria defined for this experiment. Finally, the graph evaluation is an order of magnitude smaller than the rest of the times. One number that is not shown in the table is the time needed for the bandwidth measurement. To measure the bandwidth, Remos stresses the network by sending data as fast as possible over the network. The measurement time can be configured by the application, and the accuracy of the measurement correlates with the measurement interval. For the experiments, we used an interval of 10 seconds for each measurement. These measurements use up most of the time unless the measurement time shortened. However, a shorter measurement interval may lead to less accurate results.

Summing up the contributions, an application is likely to be delayed for several seconds for selecting the active service location. The time to instantiate the service is not yet included in this time. Selecting active service locations is therefore certainly not usable for applications that are latency bound and have a short data transmission time. In contrast, the delay may be better supported by video applications that have a longer run time and that transmit more data.

The overhead of the location selection can be considerably reduced if the network provides the resource information that is needed by the application. First, the time for the DNS mapping can be reduced if the information can be retrieved from a particular cloud. Because this information is likely to remain stable for a longer time period, it can easily be stored. Second, a cloud may periodically measure the resources to other clouds. Instead of measuring the bandwidth on its own, the location selection system could retrieve the bandwidth of each link with a single query. The delay could be reduced considerably. An additional advantage is that the resource data can be shared by different applications. This sharing also reduces the load on the network that is imposed by multiple applications measuring the available resources. Remos is also extended to periodically measure the resources, store the measurements for direct application usage and also calculate a prediction of future resource usage. Other systems are currently being developed by different groups.

### 5.2 A simple video example

Although the overhead of the location selection is not negligible, this experiment shows how the application can benefit from the location selection. The video application [5] is used for the experiment and set up as follows: two video clients, located at CMU download and play a video in real time from a server located at ETH Zurich. Active service locations are only available at ETH Zurich and CMU. A multicast and an adaptive filter are available as active services. Although the experiment is run over the real Internet, the bandwidth of two links is emulated. First, the link to client 2 at CMU is limited to 0.4 MBps. This value lies be-

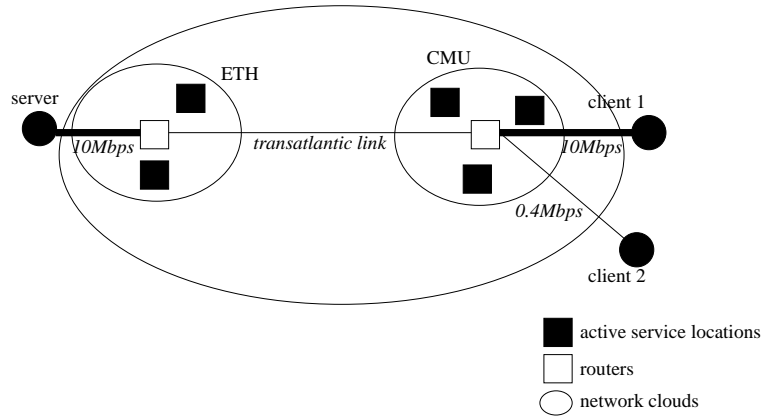


Figure 3. Network topology for the simple video example.

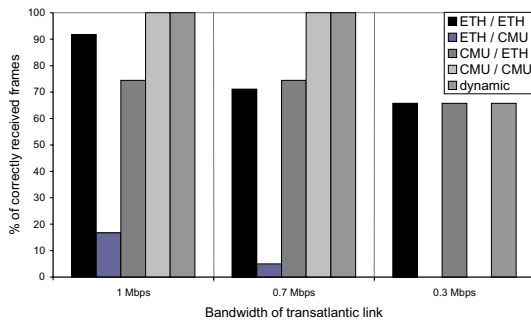


Figure 4. Percentage of correctly received frames at client 1

low the bandwidth required to play the movie in full quality. The data stream must therefore be adapted for this client. Second, the transatlantic link is emulated with a constant bandwidth to create reproducible results. The used values of 1 Mbps, 0.7 Mbps and 0.3 Mbps have been measured at three different times during the day [1]. The denote different levels of bandwidth usage during a day. All local links have a bandwidth of 10 Mbps. The video requires an average of 0.6 Mbps. For bandwidths that are below this level an adaptive filter should be used.

At the client, the numbers of correctly received frames is measured. Hemy et al. [9] argue that this metric corresponds to the quality of the received video stream. For each bandwidth value of the transatlantic link, the proposed algorithm selects the location to instantiate the service. The video quality is then compared to all combinations of filter placement.

Figure 4 shows the number of correctly received frames at client 1, depending on the location of the services and the available bandwidth of the transatlantic link.

Experiment 1 shows the situation when the data flows first through a multicast filter and then each stream through an adaptive filter. Both filters are located at ETH. Because the data stream is multicast before the transatlantic link, the two data streams compete for the available bandwidth. The result is that several frames have to be dropped for client 1.

The situation is worse if the multicast client is placed at ETH and the adaptive filters at CMU (experiment 2). The multicast filter splits the data stream before the bottleneck and thereby congests it. The adaptive filter is placed after the bottleneck. The effect is that packets have to be dropped over the transatlantic link. The packet loss leads to incomplete frames (each frame is split up into several packets for the transmission), and incomplete frames cannot be played. The result is that the number of correctly received frames is very low for client 1.

In the third experiment, the adaptive filter is placed at ETH and the multicast filter at CMU. If the transatlantic link is the bottleneck (0.3 Mbps), this setup obtains the maximal number of frames at the client. For higher bandwidths, however, this setup is dominated by client 2. The multicast filter at CMU is not combined with an adaptive filter. Because the number of incoming frames must match the number of outgoing frames at the multicast filter, the slower connection also slows down transatlantic link. The adaptive filter at ETH sees only an available bandwidth of 0.4 Mbps. The number of correctly received frames at client 1 is therefore below the maximum. This number could only be obtained if an additional adaptive filter would adapt the stream between CMU and client 2.

In experiment 4, all filters are placed at CMU. No adaptation is performed over the shared transatlantic link. When enough bandwidth is available, no adaptation is requested, and both clients receive the full number of frames. However, once the bandwidth drops below the threshold of 0.6 Mbps, many packets are dropped and no frame is received correctly.

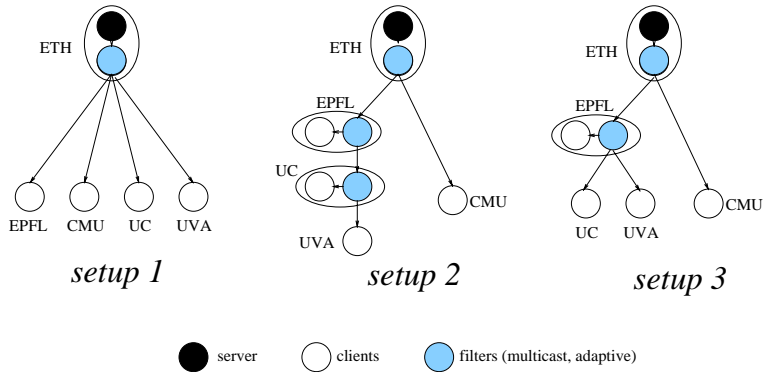


Figure 5. Network topology for the complex video experiment.

The last experiment shows the results of the dynamic location selection. The algorithm places the multicast filter always at CMU. The transatlantic link can therefore be shared by both clients. The placement of the adaptive filter depends on the bottleneck. If the transatlantic link is the bottleneck (0.3 Mbps), the adaptive filter is placed at ETH. For the other values of the transatlantic link, the adaptive filter is placed at CMU to adapt the stream for client 2.

These experiments show that the dynamic selection is capable of finding the best location. A static configuration of the services, which does not imply the time overhead for searching service locations, may obtain the same results, but its benefit changes in time, e.g., at different times during a day. Similarly, a user may select the most suitable location without the time overhead. For such a simple scenario, the user may well guess the best location. However, if more clouds are involved, if the bottleneck is not so easily guessable, or if the multicast tree is more complex, a user may no longer be capable of guessing the right location.

### 5.3 A complex video example

For a second experiment, a more complex setup is used. The video stream is sent from the ETH server to clients at CMU, EPFL, UVA (University of Valladolid, Spain) and UC (University of Coimbra, Portugal). The video requires an average bandwidth of 0.6 Mbps to be transmitted without packet loss.

Each of these domains offers nodes on which active services can be instantiated. We compare three setups for active service locations in this experiment.

The first setup, as illustrated on the left side of Figure 5, is a simple multicast scenario where the server at ETH already splits up the data streams. Each stream is adapted to the available bandwidth. The first data row in Figure 6 shows the percentage of correctly received frames for this experiment. The bandwidth limits the percentage of received frames for the clients at UC and UVA whereas the

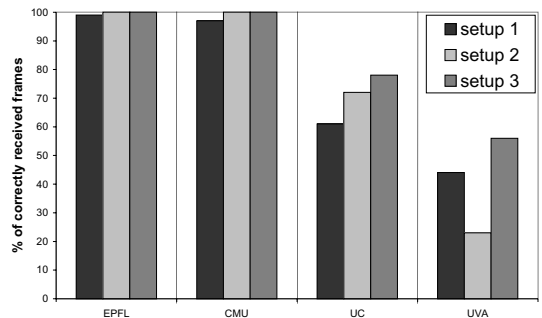


Figure 6. Percentage of correctly received frames.

clients at EPFL and CMU receive almost all frames. The small number of missing frames can be explained by the load on the server node. All services (multicast filter, adaptive filter) run on this node and the sum of all tasks loads the server up to its limits. Especially the two slower connections seem to suffer from this load.

The second setup, as shown in the center of Figure 5, could be defined by a user who has some knowledge about the network topology and performance. The bandwidth between ETH and EPFL is usually an order of magnitude higher than the bandwidth between ETH and the other servers. The two Iberian sites (UVA, UC) have a low bandwidth connection to ETH. Because of the geographical closeness of the two sites, a user may assume that the bandwidth between the two sites is higher than to other sites. Therefore, the data stream can be forwarded from one site (e.g., UC) to the other (UVA). Finally, to shed the load, the CMU client is attached to ETH whereas the Iberian sites are attached to EPFL. In contrast to the first experiment, the results in Figure 6 show that the clients at EPFL and CMU receive all frames. No overload has been detected for this experiment, so that UC also receives a large number of

frames. However, the client at UVA suffers a considerable amount of lost frames. An analysis of the network paths shows that there is in fact no direct path between the two sites. The data sent from UC to UVA travels about half the way back to EPFL before it takes a different route. Sending the data to UC and back again over the same path obviously increases the load on a bottleneck link and leads to a high loss rate at UVA.

Finally, the image on the right of Figure 5 shows the setup chosen by the dynamic service selection.

Location selection can also be used while an application is running. When a handoff is triggered in a video application [10], the application can perform a location selection to identify locations on alternate paths. If the old connection is still sending data during a handoff, the overhead of the location selection can even be hidden.

## 6 Related work

The Remos REsource MOnitoring System [4] provides an infrastructure for obtaining topology and bandwidth information of a network. This information can be used for network-aware applications, e.g., to adapt to the available bandwidth. Remos is well suited for endsystem-based applications. As a matter of fact, our implementation is an extension to Remos. Miller and Steenkiste [12] give a good overview of related work in the area of topology discovery.

Chae et al. [3] describe a method to query and synthesize network information, e.g., to identify a bottleneck in a network. In their model, each node is assumed to store local resource information, which can be queried by an application. A query is scattered into the network, the results are gathered by the different nodes and already processed there (e.g., the minimum bandwidth of two links can already be determined in the network). Their system is targeted at active networks; they require changes in routers (storing and providing resource information). In contrast, our goal is to have a discovery phase that is based only on the information the Internet provides, so that our proposal can be deployed in the current Internet. In addition, it is not clear how well the method proposed by Chae et al. scales to large networks. Such a method may scale well to the size of a LAN and might be integrated into our system to efficiently retrieve the bandwidth within a LAN.

Obraczka et al. [13] describe *topology-d*, service that informs the application about the topology and the performance of a network. Network-aware application use this information, e.g., to select one server out of a pool of available servers. The tools used for *topology-d* are similar to ours. However, their work is targeted at applications that need end-to-end information. No discovery of active service locations is therefore needed.

Content distribution services, such as Akamai or Digital Island, also make use of the Domain Name System (DNS) to dynamically redirect requests to appropriate content servers or proxies. They use the closeness of the client to the name server to find the closest server nearby. Skaikh et al. [16] provide in their analysis several reasons why this use of DNS can lead to an inefficient server selection. In contrast to these content distribution services, the service selection algorithm described here uses only the information provided by the DNS server and not the location of the server itself.

The ALAN project [7] describes an infrastructure that permits applications to dynamically deploy active services at the application layer. This infrastructure allows applications to place ALAN-supported services in the network according to their needs. The infrastructure is required to be scalable and customizable by applications. In contrast to our system, their approach is to build a large distributed database of all nodes that support active services. For scalability reasons, the database is built hierarchically. The database may not only store the locations of the active services but also include bandwidth and latency information. It is not clear, however, how accurate this dynamic information will be and how long it takes to find a given active service location.

## 7 Conclusions

This paper describes a method to search for and dynamically select available active service locations in the Internet. The presented method is practicable because it can be deployed in the current Internet; there are especially no requirements to change or enhance routers. By limiting the search for active service locations to the routing path, the method even scales to large networks such as the Internet. Finally, the presented system can be customized and extended for different groups of applications. A consequence of this feature is that the system can take application-specific information, such as the service type or a specific metric into account. A metric that is relevant, e.g., for multi-media applications, is the bandwidth of a connection.

This functionality comes at a price: An evaluation of the overhead of the location selection shows that the time needed to measure the available bandwidth takes up the most time. This time could be shortened if the network could provide information about the available resources. There exist a (growing) number of such monitoring systems, so we are optimistic that in the future, this aspect of our system will be improved.

A consequence of this overhead is a delayed start of sending the data stream. If an application can tolerate such a wait, then our evaluation with a video application demon-

strates the benefits that can be obtained from dynamic selection of an active service location. As the Internet grows in size and complexity, it becomes less and less practical for a user to perform service placement by hand. Even sophisticated users cannot keep up with the number and frequency of changes. So although a user may be able to guess the correct service placement in a simple scenario, for complex scenarios this task is hardly manageable. We therefore anticipate that service location selection will remain a challenging problem.

## References

- [1] Switch traffic statistics. [www.switch.ch/lan/stat](http://www.switch.ch/lan/stat).
- [2] E. Amir, S. McCanne, and R. Katz. An active service framework and its application to real-time multimedia transcoding. In *Proceedings of ACM SIGCOMM '98*, pages 178–189, Vancouver, BC, Canada, August 1998.
- [3] Y. Chae, S. Merugu, E. Zegura, and S. Bhattacharjee. Exposing the network: support for topology-sensitive applications. In *Proceedings of IEEE OPENARCH 2000*, pages 65–74, Tel Aviv, Isr, March 2000.
- [4] P. Dinda, T. Gross, R. Karrer, B. Lowekamp, N. Miller, P. Steenkiste, and D. Sutherland. The architecture of the Remos system. In *Proceedings of HPDC-10*, San Francisco, CA, August 2001.
- [5] R. Fontana. Video multicast. Master's thesis, Lab for Software Technology, ETH Zurich, 2000.
- [6] A. Fox, S. Gribble, E. Brewer, and E. Amir. Adapting to network and client variability via on-demand dynamic distillation. In *Proceedings 7th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-VII)*, pages 160–173, Cambridge, MA, October 1996.
- [7] A. Gosh, M. Fry, and J. Crowcroft. An architecture for application layer routing. In *Intl. Workshop for Active Networks (IWAN 2000)*, Tokyo, Japan, October 2000.
- [8] E. Guttman, C. Perkins, J. Veizades, and M. Day. Service location protocol, version 2. RFC 2608, June 1999.
- [9] M. Hemy, P. Steenkiste, and T. Gross. Evaluation of adaptive filtering of MPEG system streams in IP networks. In *IEEE International Conference on Multimedia and Expo 2000 (IDME 2000)*, pages 1313–1317, New York, 2000.
- [10] R. Karrer and T. Gross. Dynamic handoff of multimedia streams. In *Proceedings of NOSSDAV '01*, Port Jefferson, NY, June 2001.
- [11] Sun Microsystems. The Jini homepage.
- [12] N. Miller and P. Steenkiste. Collecting network status information for network-aware applications. In *Proceedings of IEEE Infocom 2000*, pages 641–650, Tel Aviv, Isr, March 2000.
- [13] K. Obraczka and G. Gheorghiu. The performance of a service for network-aware applications. In *ACM Sigmetrics SPDT*, 1998.
- [14] V. Paxson, G. Almes, J. Mahdavi, and M. Mathis. A framework for IP performance metrics. Request for Comment 2330, May 1998.
- [15] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson. The end-to-end effects of Internet path selection. In *Proceedings of ACM SIGCOMM '99*, pages 289–299, Boston, Massachusetts, August 1999.
- [16] A. Shaikh, R. Tewari, and M. Agrawal. On the effectiveness of DNS-based server selection. In *Proceedings of IEEE Infocom 2001*, Anchorage, Alaska, April 2001.