Two-Phase Multi-criteria Server Selection for Lightweight Video Distribution Systems

Octavian Catrina, Eugen Borcoci, Piotr Krawiec

Two-Phase Multi-criteria Server Selection for Lightweight Video Distribution Systems

Octavian Catrina[1], Eugen Borcoci[1], and Piotr Krawiec[2]

[1] University Politehnica of Bucharest, Romania
[2] Warsaw University of Technology and National Inst. of Telecommunications, Poland

Abstract. Video streaming services need a server selection algorithm that allocates efficiently network and server resources. Solving this optimization problem requires information about current resources. A video streaming system that relies entirely on the service provider for this task needs an expensive monitoring infrastructure. In this paper, we consider a two-phase approach that reduces the monitoring requirements by involving the clients in the selection process: the provider recommends several servers based on limited information about the system's resources, and the clients make the final decision, using information obtained by interacting with these servers. We implemented these selection methods in a simulator and compared their performance. The results show that the two-phase selection is effective, improving substantially the performance of lightweight service providers, with limited monitoring capabilities.

Keywords: Content networks, Video streaming, Server selection, Multi-criteria decision algorithms

1 Introduction

An essential task of a video streaming service provider is to select a suitable content server. This task can be formalized as a multi-criteria optimization problem, that takes into account various static and dynamic attributes of the system's components. The goal is to allocate efficiently the network and server resources, while providing a suitable Quality of Experience (QoE) to the service users.

These problems are NP-complete, in general, but practical server selection algorithms are available. As these algorithms need information about network and server resources, a service provider has to deploy a complex monitoring infrastructure in order to use them. Lightweight service providers, without monitoring capabilities, can only make suboptimal decisions.

In this paper, we consider a more flexible approach, that splits the selection process between the service provider and the clients. The provider recommends a short list of content servers, using limited information (e.g., the lengths of the paths between servers and clients). The client refines the selection using additional information, obtained by interacting directly with these servers. The provider and the clients can use various combinations of algorithms, corresponding to different capabilities to obtain information about the system, and different

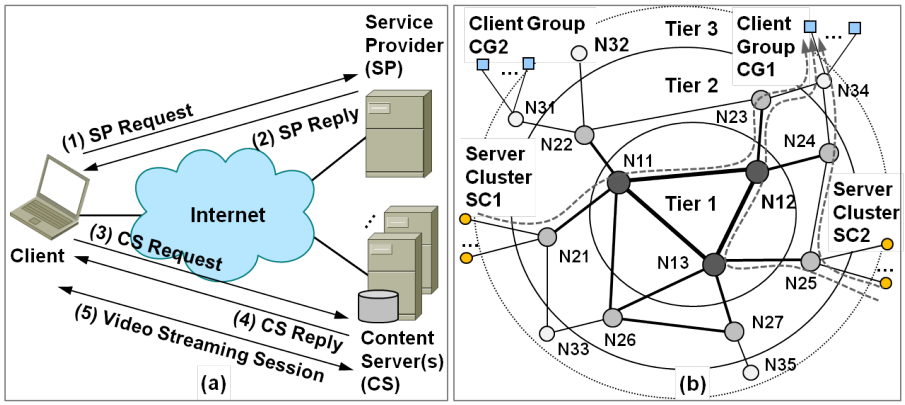

Fig. 1. Video streaming model: (a) Actors and interactions. (b) Infrastructure.

tradeoffs between performance and complexity. In particular, this approach can improve the performance for lightweight service providers.

The paper has the following structure. Section 2 introduces a generic video streaming model and Section 3 describes the server selection methods. We implemented these methods in a simulator, presented in Section 4. The results of the simulations are discussed in Section 5, followed by a summary in Section 6.

2 Video Streaming Framework

We introduce in the following the generic model of online video streaming used in this paper. Figure 1 shows the main actors, the interactions between them, and the infrastructure used to deliver the service. We separate the management of the service, assigned to a service provider (SP) entity, from the delivery of the video content, performed by a collection of content servers (CS). An essential function of the SP is to select a content server when a client initiates a session.

We assume over-the-top delivery, using video streaming over HTTP or a similar technology [**?**]. The video content is stored on a collection of servers deployed in the Internet, each video file being available from several servers. These servers usually operate in a Content Delivery Network (CDN) [**?**]. We expect efficient CDN operation, with server placement and content replication adapted to the video streaming traffic. The video streaming system can include a CDN, or use the services of one or more CDN providers [**?**].

A video streaming session starts with the following interactions (Figure 1a):

- The client sends a request to the SP, specifying the desired video content.
- The SP runs a server selection algorithm and replies with a list of servers that could deliver the requested video content (or rejects the request). Table 1 lists the information used by the SP in the server selection process.
- In a single-phase selection process, the client simply asks one of the servers indicated by the provider to deliver the desired video content.

Table 1. Information used by service providers for content server selection.

Static or quasi-static information	Dynamic information
Lists of content servers and video files.	Available capacity of the content servers.
Mapping of video content to servers and of clients and servers to network nodes.	Available capacity of the paths between servers and clients.
Length of server-client network paths.	Admission control for servers and paths.

– In a two-phase selection process, the client chooses a server from the provider's list by running its own selection algorithm, with additional information.

For example, if the service is implemented using the DASH standard [**?**], the provider can answer with a Media Presentation Description (MPD) file, containing information about the encoding of the video content (including available resolutions and data rates) and the URLs of the servers that can deliver it.

We assume that the main bottlenecks of the transport network are the links between the autonomous systems (AS) of the network service providers. Therefore, the network is modeled as a collection of interconnected nodes, corresponding to an AS-level, multi-tier network topology, similar to the Internet topology (Figure 1b). The clients and the content servers are attached to these network nodes. The Border Gateway Protocol (BGP), responsible for inter-domain routing in the Internet, determines a single best path between ASes, based on network service provider policies and path length. Load balancing on multiple paths would help to allocate more efficiently the network resources to the video streaming sessions, and could be achieved using application-level overlays. We consider, therefore, this additional capability for some of the server selection algorithms.

3 Content Server Selection

Multi-Criteria Decision Algorithm (MCDA). Selection of a server and a path in content delivery systems can be formulated as a multi-criteria optimization problem. These problems are NP-complete in general, and are solved in practice using heuristics methods. Practical and efficient methods have been proposed for server and path selection based on Multi-Criteria Decision Algorithms (MCDA) [**?**]. The variant of MCDA used in this paper is described below:

– The algorithm determines a set of candidate solutions, $S = \{S_i\}_{i\in 1,\ldots,n}$, from information about the request and the delivery system. A candidate solution is a vector of decision variables, $S_i = (v_{i,j})_{j\in 1,\ldots,m}$, representing characteristics of a server and a path that could deliver the requested content.
– For each candidate solution S_i and variable $v_{i,j}$, the algorithm calculates the component achievement function $R_{i,j} = \frac{r_{i,j}-v_{i,j}}{r_{i,j}-a_{i,j}}$, where $v_{i,j}$ is the value of the variable, while $r_{i,j}$ and $a_{i,j}$ are the reservation level and aspiration level for this variable, respectively (the algorithm can be extended with more complex functions). We use two decision variables: the available (unused) capacity of the server and the available capacity (unused bandwidth) of the

path. For these variables, the reservation level is a lower bound for suitable solutions, and the aspiration level is an upper bound beyond which the preference of the solutions is similar. We set $a_{i,j}$ to the maximum capacity of the server and of the path, respectively, and $r_{i,j}$ to the amount of server capacity and path capacity consumed by a session, respectively.
- The algorithm calculates the rank of each solution, $R_i = \min_j\{R_{i,j}\}$, and then determines the index of the best solution, $\text{argmin}_i\{R_i\}$.

Single-phase selection. The single-phase selection relies entirely on the service provider to find suitable servers for the clients' sessions. In its simplest variant, the provider answers a client's request by indicating a single server [**?**]. We consider a more general variant, in which the provider answers with an ordered list of servers. The client can start the session by connecting to one of these servers and use the others as back-up [**?**]. By delivering a list of servers, the provider enables the client to improve the initial server selection and/or use certain adaptive streaming techniques [**?**]). The selection proceeds as follows:

- The service provider maintains a resource database with the information about network and content servers that is necessary for server selection. The database contains static information and some of the dynamic information listed in Table 1, depending on the algorithm being used.
- When a service request arrives, the provider determines an ordered list of suitable servers and delivers it to the client. The selection is based on information received in the service request (e.g., client and video ids) and information available in the resource database. If the provider does not find any suitable server the request is rejected.
  Candidate solutions consist of a server that has a copy of the requested video file and a path from that server to the client. The selection algorithm takes as input a set of candidate solutions that (optionally) satisfy additional requirements. The provider can use a range of server selection algorithms, that offer different tradeoffs between the complexity of the system and the optimality of the solution (e.g., different requirements for the information collected by the provider about network and servers). The algorithms used in our simulation experiments are listed in Table 2.
- If the request is accepted, the client starts the video session by connecting to the first server in the list. In case of failure, the client can try the next servers, without having to ask the provider to recommend another server.

This approach simplifies the functionality of the client. On the other hand, the service provider has, essentially, two main options: algorithms that use only static information and offer poor performance and algorithms that can achieve much better performance, but require an expensive monitoring infrastructure.

Two-phase selection. We consider now a more flexible, two-phase approach, that involves the clients in the selection process:

- The provider's resource database contains only the information about network and servers that is necessary for a preliminary selection: static and

Table 2. Server selection algorithms used by the service provider.

Algorithm	Input	Output (list of servers).
Random servers	Set of feasible solutions.	Randomly chosen servers; shortest or random path.
Closest servers	Set of feasible solutions. Length of the paths from servers to client.	Servers with the shortest paths to the client.
MCDA	Set of feasible solutions. Available capacity of the servers and the paths.	Servers in the solutions with the highest MCDA rank.

quasi-static information (Table 1) and, possibly, dynamic information that is easier to collect (e.g., server load).
- When a request arrives, the service provider makes a preliminary server selection and delivers to the client a short, ordered list of recommended servers. The procedure is essentially the same as for single-phase selection, using algorithms that match the limited information collected by the provider: e.g., random servers or closest servers.
- If the request is accepted, the client performs the second phase of the selection, choosing the server that will be used by the video streaming session. The input of the client's selection algorithm is the list of servers received from the provider and additional information obtained by the client:
  1. Local information: Client capabilities, processor and link load.
  2. Current state of the recommended servers: Reports from the servers, indicating if they can handle the request and/or the available capacity.
  3. Current state of the paths between the servers and the client: Throughput of the connection with each server.

  The client can use MCDA based on available server and path capacity, or choose the least loaded server, depending on the information it can obtain.

Admission control. The video streaming sessions have to ensure continuous playout. This requires timely data delivery, and hence a certain lower bound for the end-to-end data rate. An important issue, therefore, is to add some form of admission control to the server selection process. This can be achieved by removing the solutions with fully loaded servers or paths from the set of candidate solutions given as input to the selection algorithms.

Admission control is simpler for content servers. The available capacity can be estimated by a local monitoring application and reported by the servers to the provider or the clients. Moreover, a fully loaded server can reject additional requests. This offers basic admission control, independent of the selection process. However, it is more difficult to determine if the path from a server to a client can handle additional sessions. The service providers need for this purpose a network monitoring infrastructure that measures the throughput between network nodes. The clients can measure the throughput of the connections with the recommended servers. These end-to-end measurements are more relevant, but increase the delay and overhead of session establishment. Also, they may fail to detect initial congestion and avoid the QoE degradation of current sessions.

4 Simulation Software

We analyzed the performance of the server selection methods described in Section 3 by simulation. The simulation software developed for this purpose is written in C++ and uses the discrete event simulator OMNeT++ [**?**] as basic simulation engine. Figure 2 shows the main components of the simulation software.

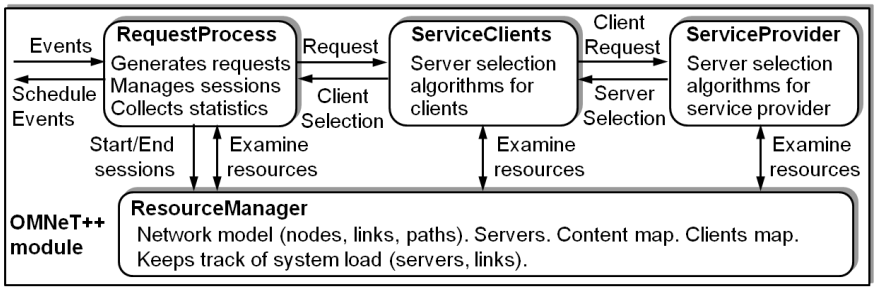

Fig. 2. High-level architecture of the simulation software.

The class ResourceManager provides a resource database with information about the entire system infrastructure: network topology (nodes and links, link capacity and load), network paths, list of content servers (server capacity and load), mapping of video content to servers, mapping of clients and servers to network nodes. The simulations are driven by the RequestProcess, which generates the service requests, manages the video streaming sessions, and collects statistics. When a session starts or ends, the RequestProcess notifies the ResourceManager, which updates accordingly the load of the server and the load of the links on the path from the server to the client.

The server selection process is implemented by the classes ServiceProvider and ServiceClients, and obtains information about system resources from the ResourceManager. The class ServiceProvider handles the clients' service requests and implements the server selection algorithms run by the service provider for single-phase and two-phase selection (Table 2). The class ServiceClients implements the server selection algorithms run by the clients in two-phase selection. The network model uses random topologies generated using the tool aSHIIP [**?**] and the network nodes are the autonomous systems (AS) of the network service providers (Figure 1b). The topologies have a multi-tier structure similar to the Internet. We assume that this structure corresponds to the usual business relations between network service providers: peering relations between nodes in the same tier, and customer-provider relations between nodes in different tiers, with the customer in the lower tier. The paths used in the simulations are similar to the BGP paths, according to the relations between network providers [**?**].

The clients and the content servers are attached to network nodes. Servers are deployed all over the network, in well-connected nodes, close to clients, taking into account the number of links of the node and their bandwidth.

The resources consumed by the video streaming sessions are measured using a metric for server capacity and a metric for link capacity. To simplify the analysis, we assume that a session consumes 1 unit of server capacity and 1 unit of link capacity. The system is provisioned by assigning a fixed maximum capacity to servers and links, which is available for video streaming. The servers have the same capacity and are deployed in clusters; better connected nodes have larger clusters. The network is provisioned by assigning capacity to links according to its hierarchical structure, with higher capacity in upper tiers.

The main performance metric used in the analysis of the server selection process is the success rate, defined as the ratio between the number of successful service requests and the total number of service requests. A request is successful if the system has sufficient server and network resources to deliver the requested video content to the client, with the desired quality, for its entire duration.

Failures can occur in the server selection phase or during the video streaming session. If admission control is available, a request is rejected when the selection process does not find a server and a path with sufficient unused capacity; the load of the system remains unchanged. Otherwise, a new session is created, which may overload the server and/or some links on the path from server to client. The RequestProcess monitors the current sessions and updates their state. If a session has been successful so far and does not have sufficient resources any more, its state changes from success to fail.

5 Performance Evaluation

We discuss in this section the results of simulations with single-phase and two-phase selection, for different selection algorithms. The experiments use a network with 1000 nodes, structured into 4 tiers, and (up to) 3 paths for any pair of nodes. The network is provisioned so that the number of concurrent sessions can reach the total capacity of the servers, 61500 sessions, for a suitable server selection. Service requests are generated at random time intervals with exponential distribution. We measure the session success rate as a function of the request rate.

Analysis of single-phase selection. Figure 3 shows the success rate for single-phase selection and the configurations in Table 3. Service providers with network monitoring capability can use MCDA and admission control for servers and paths. The success rate of MCDA is equal to 1 for request rates up to 150 requests/sec and about 60000 concurrent sessions, close to the total server capacity (the mean session duration is 400 seconds). The performance drops drastically if the providers have only static information and select random servers or closest servers, without admission control. The degradation is due to sessions rejected by fully loaded servers and, especially, congestion of network links.

Congested sessions may be aborted due to poor QoE. The freed resources can then be used to start other sessions, which may be successful (e.g., their paths do not include congested links). Figure 3 shows the simulations results when the congested sessions are aborted and when they are allowed to continue.

Table 3. Configurations used for server selection by the service provider.

Name	Server and path selection algorithm	Admission control (AC)
RandRand	Random servers and paths	Server reject
RandShrt	Random servers, shortest paths	Server reject
NearSrv	Servers with shortest paths to client	Server reject
MCDA	MCDA for available server and path capacity	AC for servers and paths

Important performance differences appear for randomly chosen servers and/or paths, because the allocation of network resources is less efficient.

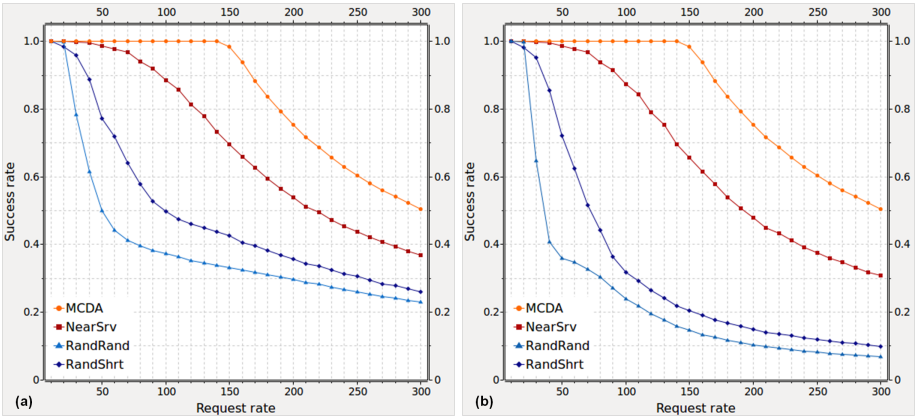

Fig. 3. Single-phase selection: (a) with session abort; (b) without session abort.

Intuitively, the performance should improve when the load is distributed on multiple network paths. We observe, however, that using the shortest path is better than the random choice of a path (e.g., RandShrt versus RandRand). In a multi-tier network with valley-free paths, the shortest path may avoid the core of the network, while the other paths are more likely to cross it (e.g., the paths between SC2 and CG1 in Figure 1). Thus, the random choice of the path concentrates the traffic in the network core. Therefore, if multiple paths are available, the choice should take into account the path load (e.g., using MCDA).

For similar reasons, the performance improves when the provider selects the servers with the shortest paths to the client (NearSrv), instead of choosing them randomly (e.g., RandRand, RandShrt): the sessions use less network resources and the paths are more likely to avoid the core. However, the performance of NearSrv selection is very sensitive to server placement.

Analysis of two-phase selection. In the two-phase selection process, the provider and the clients can use various combinations of algorithms. Our main goal is to see the effects of this approach on the performance of the video streaming

services offered by lightweight providers. We assume, therefore, that the provider has global, static information about the entire system, and the clients can obtain dynamic information for a particular request and several candidate servers.

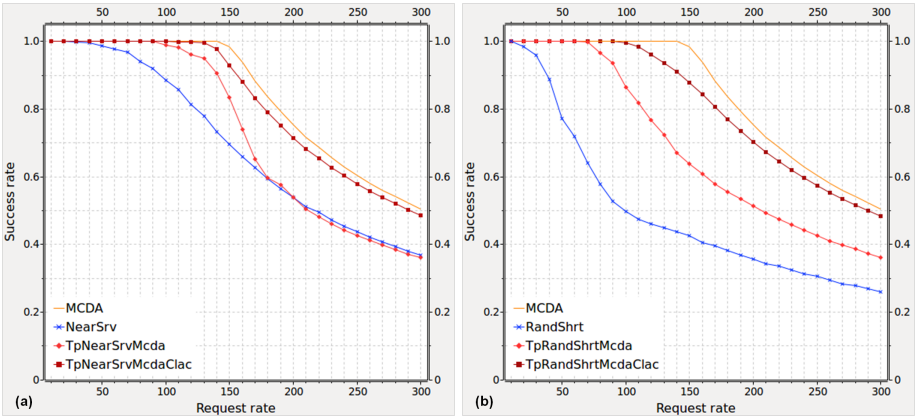

Fig. 4. Two-phase versus single-phase selection for: (a) NearSrv; (b) RandShrt.

Figure 4 shows the simulation results for the following configurations:

- TpNearSrvMcda/TpRandShrtMcda: Provider: NearSrv/RandShrt. Client: MCDA for available server and path capacity. Congested sessions are aborted.
- TpNearSrvMcdaClac/TpRandShrtMcdaClac: Similar to the previous configurations, with client admission control for servers and paths.
- The provider sends a list of 3 recommended servers, out of 8 candidates.
- The results for single-phase selection using NearSrv, RandShrt, and MCDA are added to Figure 4 to facilitate the performance comparison.

The two-phase selection clearly improves the success rate with respect to single-phase selection when the provider chooses the recommended servers using NearSrv or RandShrt and the clients use MCDA. NearSrv can achieve better performance than RandShrt, for a suitable placement of the servers in the network, but the performance varies significantly depending on server placement.

However, the two-phase selection cannot reach the performance of the single-phase selection using MCDA (with admission control): the provider runs MCDA with the entire set of candidate solutions, while the client has a short list of servers, chosen without information about system load. The size of the list is a trade-off between system performance and the delay and overhead of the interactions between client and servers, that collect additional information for the final selection. The results suggest that a short list of 3 servers is sufficient.

The performance improves significantly when the clients add admission control to MCDA, becoming close to the performance of a provider that is able to run MCDA itself. The improvement is more important when the provider uses RandShrt, which is more vulnerable to network congestion.

6 Conclusions

A service provider that offers over-the-top video streaming needs an expensive monitoring infrastructure for efficient server selection. Lightweight providers, without such capabilities, can only use simple selection methods, based on static information. Simulation results show a large performance gap between these two approaches. We explored, therefore, a two-phase selection process, that aims at improving the performance for lightweight providers, by involving the clients.

We studied the performance of two-phase selection by simulation. The results show that the approach is effective, improving substantially the success rate. In the ideal case, when the clients can reliably apply admission control, the performance of the two-phase selection becomes close to the performance of a provider that runs MCDA using a network monitoring infrastructure.

Future work will focus on implementing the two-phase selection, using, e.g., DASH [**?**]. The critical issue is to enable the clients to obtain the information needed for the final selection with minimum delay and overhead, and sufficient accuracy. Ideally, this information could be collected during the initial buffering phase. The DASH standard offers several features that could be exploited for this purpose, e.g., chunks of video content can be downloaded from several servers.

Acknowledgments. This work was partially supported by the Research Project DISEDAN, No.3-CHIST-ERA C3N, 2014-2015.