# QoS Dynamic Routing in Content Delivery Networks

Krzysztof Walkowiak

Chair of Systems and Computer Networks, Faculty of Electronics, Wroclaw University of Technology, Wybrzeze Wyspianskiego 27, 50-370 Wroclaw, Poland
tel. (+48)713202877, fax. (+48)713202902
Krzysztof.Walkowiak@pwr.wroc.pl

**Abstract.** Recently, much research in quality of service (QoS) routing has focused on the unicast communication technique. However, Content Delivery Network (CDN) approach becomes popular because CDN enables effective and inexpensive improvement of Internet service quality. Therefore, we analyze in this paper a network processing two kinds of demands: content demands to CDN servers and standard unicast demands. Since MPLS defines effective mechanism for traffic engineering, we assume that the CDN is located in MPLS network. To examine the QoS performance of CDN-enabled MPLS network we propose new constraint-based algorithms for CDN server selection and evaluate relative performance of these algorithms and the most effective existing QoS unicast routing algorithms in terms of the demand rejection ratio.

**Keywords:** CDN, dynamic routing, QoS, server selection

## 1 Introduction

Most of interest in context of Quality of Service (QoS) dynamic routing has been focused on the unicast traffic [1-2], [5], [7], [14], [17]. This is obvious, since unicast routing between a specified pair on network nodes is relatively well understood in best effort networks [4]. However, various techniques of network caching have gained much attention in recent years. In this work we focus on one of the most efficient caching approach – Content Delivery Network (CDN).

CDN is an interesting and robust approach to enhance the Internet quality. CDN uses many servers offering the same content replicated in various locations. User-perceived latency and the other QoS parameters (e.g. network reliability) can be easily and inexpensively improved by various techniques of Web content caching. Every replicated system must deal with: deciding on placement of replicas and distributing requests to object replicas. In this work we focus on the second problem. At present, conventional CDNs support only best-effort services. However, due to a big competition on the telecommunication market, Internet service providers will need to provide also QoS services to attract more clients [19]. We assume that the CDN is located in MPLS (Multiprotocol Label Switching) network. The MPLS approach proposed by the Internet Engineering Task Force (IETF) in [12] is a networking technique that enables traffic engineering (TE) for carrier networks. We consider a CDN system with traffic engineering capabilities provided by the MPLS. Note that the traffic of

CDN is also referred to as anycast traffic [4], [18]. The novelty of this work is that we analyze QoS performance of a network that can accept two kinds of demands: content demands to CDN servers and standard unicast demands. Only few previous works on this subject uses such approach [4]. There has not been, however, any study we are aware of that examines QoS dynamic routing of both unicast and anycast traffic in MPLS network.

The main goal of this work is to examine performance of various server selection algorithms and unicast dynamic routing algorithms in CDN network using the MPLS environment. Due to limited size of the paper, we don't present and discuss other issues related dynamic routing in CDN like: signalling protocols, client demand prediction, QoS network architecture, content distribution.

The remainder of the paper is structured as follows. Section 2 describes issues of CDN and MPLS. In Section 3, we present algorithms for content server selection. Section 4 includes results of simulations. We conclude in Section 5.

## 2 CDN-enabled MPLS Network

Content Delivery Network is defined as mechanisms to deliver a range of content to end users on behalf of origin Web servers. The original information is offloaded from source sites to other content servers located in different locations in the network. For each request, the CDN tries to find the closest server offering the requested Web page [8]. CDN delivers the content from the origin server to the replicas that are much closer to end-users. The set of content stored in CDNs servers is selected carefully. Thus, the CDNs' servers can approach the hit ratio of 100%. It means that almost all request to replicated servers are satisfied [9].

Conventional CDN systems routes user requests to the nearest server using a redirection methods. Since redirection mechanisms don't directly select a route to the server, they are not effective in terms of network resources usage. Hence, some requests are rejected because the CDN redirects user to a server, which can be strongly loaded or links leading to this server are congested. Therefore, in this work we assume that the CDN consisting of content servers is located in MPLS network that provides traffic engineering capabilities. We call such a network CDN-enabled MPLS network. However, also other architectures can be used for provision of QoS and TE capabilities in CDN. For more details refer to [4], [18].

A user in the CDN-enabled MPLS network can generate two kinds of requests:

- **Unicast** is a standard MPLS unicast demand defined by a following triple: ingress router (source node), egress router (destination node) and the amount of required bandwidth requirement. A constraint-based dynamic routing algorithm is used to calculate a path for a unicast demand. If a feasible path doesn't exist, the demand is rejected.
- **Content** is a demand to the Content Delivery Network. It is defined by a triple: ingress router – node in which a CDN client is located, bandwidth requirement of upstream LSP (from the client to the server) and bandwidth requirement of downstream LSP (from the server to the client). When a content demand is issued, first QoS-based server selection algorithm is used to pick a content server.

If none of CDN servers is reachable, the demand is rejected. Otherwise, two LSPs are established: upstream (from ingress node to server) and downstream in the opposite direction. A constraint-based dynamic routing algorithm is applied for calculation of both LSPs.

For simplicity, we assume that CDN servers provide the same long-lived content related to services like: distance learning, e-books, software distribution, archives of electronic entertainment (MP3 files, movies), FTP. Examples of unicast demands are: Voice over IP, teleconferences, exchanging of files, VPN, less popular WWW servers.

The CDN-enabled MPLS can deal with both kinds of demands in order to satisfy selected QoS constraints. It is worth mentioning that most of previous work on dynamic routing and CDNs consider only one of the two kinds of demands presented above. In our work we are interested in the combination of these two trends. It is much more realistic case, since usually existing network carry various kinds of traffic in the same links using the same routing algorithms.

One of the most important issues of CDN is the mechanism used for requests redirection. Transparent replication assumes redirecting a client's request for a document to one of the physical replicas. In [8] it has been presented the most popular practical and theoretical approaches of requests redirection: client multiplexing, IP multiplexing, DNS indirection, HTTP redirection and anycast, peer-to-peer routing. Generally, CDN routes user requests to the nearest or lowest-load server to reduce network latency. The effectiveness of a server selection strategy depends on its ability to take into account the metrics that contribute to the improvement of the QoS perceived by the users. Therefore, in our work we assume that when a user issues a new request to a CDN server, the special constraint-based server selection algorithm is applied. Such an algorithm uses the traffic engineering information on network saturation and selects the best content server in terms of network resources usage. We apply the explicit routing mode of MPLS [12]. Thus, the server selection can be transparent to end user, because the selection algorithm is located in the ingress node of MPLS network that is responsible also for the path selection. Similar approach referred to as source route option is mentioned in [18] in the context of anycast in Ipv6 networks. We assume that the following link state information applied for server selection either is flooded by the routing protocol or known administratively: total flow on link, link capacity, location of content servers and network topology. Three first items require extensions considering the traffic engineering provided by extended version of OSPF [6], [13].

## 3 Algorithms for Content Server Selection

In this section we present 3 algorithms that can be used for selection of content server when a content demand arrives in the network. The CDN-enabled MPLS network is modeled as $(G,c)$, where $G=(V,A)$ is a directed graph with $n$ vertices representing routers or switches and $m$ arcs representing links, $c: A \rightarrow R^{+}$ is a function that defines capacities of the arcs. We denote by $o : A \rightarrow V$ and $d : A \rightarrow V$ functions defining the origin and destination node of each arc.

To mathematically represent the problem we introduce the following notations

| | |
|---|---|
| $f_a$ | Represents the total flow on arc *a*, it is calculated as a sum of all LSP's bandwidth requirements that uses arc *a*. |
| $c_a$ | Capacity of arc *a*. |
| $r_a = c_a - f_a$ | Residual capacity of arc *a*. |
| $g_v = \sum_{i:d(i)=v} f_i$ | Flow of node *v* - aggregate flow of incoming arcs of *v*. |
| $e_v = \sum_{i:d(i)=v} c_i$ | Capacity of node *v* - aggregate capacity of incoming arcs of *v*. |
| $h_v = e_v - g_v$ | Residual capacity of node *v*. |
| $Q_d^{\text{up}}$ | Upstream bandwidth requirement of demand *d*. |
| $Q_d^{\text{down}}$ | Downstream bandwidth requirement of demand *d*. |
| $o_d$ | Origin (ingress) node of content demand *d*. |

We say that a content server is *located* in node *v* when it is connected to node *v* using a link of very high capacity. Consequently, the connection between content server and network node (e.g. router) cannot become a bottleneck. Furthermore, we make an assumption that every server can serve a large number of demands. Thus, we don't limit the amount of service that can be provided at any server. According to [9], it is a reasonable assumption, since increasing the number of content servers is much more difficult than increasing the capacity of a server. The number of servers is frequently given a priori due to cost and administrative reasons, while the capacity constraint can be overcome by adding more machines and storage space. Consequently, the only limitation of the content server workload is capacity of links leading the network node in which the server is located. However, server selection algorithms presented below can be easily modified to include also constraints on server workload or storage capacity.

In the server selection algorithms we use the *feasible* network approach applied in the context of QoS unicast routing algorithm [1], [5], [7], [14], [17]. The feasible network for a new demand consists of all routers and links, for which residual capacity exceeds bandwidth requirement of the new demand. More precisely, we consider bandwidth requirements of both paths: upstream and downstream associated with a particular anycast demand and use for calculation of feasible network the bigger value of these two bandwidth requirements. Thus, routing in the feasible network guarantees that acceptance of a new anycast demand will not violate the capacity constraint. Moreover, only content servers reachable in the feasible network can be taken into account in the selection what facilitates the process. Such servers are called *available*. Let $B_d$ denote the set of servers that are available for demand *d*. Note that the terms server and server node are used interchangeably and they mean the node in which the server is located. Let $Q_d = \max(Q_d^{\text{up}}, Q_d^{\text{down}})$. To find the set $B_d$ we do the following algorithm

1. **Prune the network.** Remove from the network each link for which $r_a \le Q_d$ .

2. **Server checking.** Assign metric 1 to each link. For each node $v$ with content server apply the shortest path algorithm (e.g. Dijkstra) to check if a path between $o_d$ and $v$ exists. If such a path exists add the node server $v$ to the set $B_d$.

Now we present three QoS server selection algorithms for a content demand $d$.

*Hop Number Server (HNS) Algorithm*

1. **Find available servers.** Calculate set $B_d$ of available servers for demand $d$. If set $B_d$ is empty, reject the demand and stop the algorithm.
2. **Find the server cost.** In the feasible network for demand $d$ assign metric 1 to each link. For each server node $v$ included in set $B_d$ find the shortest path between $v$ and $o_d$. Let $L(v)$ denote length of the path.
3. **Server selection.** Select server from the set $B_d$ with the lowest value of $L(v)$. If two or more servers have the same minimal value of $L$, choose server for which residual capacity $h_v$ of node $v$ is the largest.

*Hop Number Widest Server (HNWS) Algorithm*

1. **Find available servers.** Calculate the set $B_d$ of available servers for demand $d$. If set $B_d$ is empty, reject the demand and stop the algorithm.
2. **Find the server cost.** In the feasible network for demand $d$ assign metric 1 to each link. For each server node $v$ included in set $B_d$ find the shortest path between $v$ and $o_d$. Let $L(v)$ denote length of the path.
3. **Server selection.** Select server $v$ from the set $B_d$ with the lowest value of $(L(v)/h_v)$.

*Residual Capacity Server (RCS) Algorithm*

1. **Find available servers.** Calculate the set $B_d$ of available servers for demand $d$. If set $B_d$ is empty, reject the demand and stop the algorithm.
2. **Server selection.** Select server from the set $B_d$ with the largest value of residual capacity $h_v$ of server node $v$.

In all algorithms we take into account only available servers. Server selection is done according to two metrics: hop distance to the server and the residual capacity of server node. The former metric ensures selection of the nearest (in terms of the hop number) server. However, this can lead to the congestion of network links leading to the server node. Therefore, the latter function is applied to balance the content demands among various servers in the network.

Computational cost of all three algorithms proposed above is $O(mnr)$, where $r$ denotes the number of content servers in the network. The major effort is to find shortest paths for all content servers. Therefore, complexity $O(mn)$ of shortest path algorithm is multiplied by $r$. However, in some cases the number of servers is a constant and it does not depend on the network size (defined by the number of nodes). Consequently, under such assumption complexity of server selection algorithms can be reduced to $O(mn)$ – the same as many dynamic online routing algorithms, e.g. Minimum Hop Algorithm (MHA), Constraint Shortest Path First (CSPF) [2], second stage of Dynamic Online Routing Algorithm (DORA) [14] and Least Interference Optimization Algorithm (LIOA) [1]. In contrast, Minimum Interference Routing Algorithm (MIRA) [7] has complexity of $(O(n^5)+O(m^2))$, the first stage of DORA has complexity of $O(n^3m^2)$.

Sometimes server selection algorithm may seem to overlap with the routing algorithm. However, one of our major assumptions is that the consider network supports and applies existing traffic engineering online routing algorithms (e.g. MHA, CSPF

[2], DORA [14], LIOA [1]) and we do not have to change the routing protocol. Consequently, the same routing algorithm can be used for unicast and anycast demands. Therefore, we develop special algorithms for CDN server selection that cooperating together with unicast routing algorithms enable establishing of anycast demands. Nonetheless, it is possible to develop a mixed algorithm that can jointly select a server and find a path to the server. Such an approach leads to the situation, in which two various routing algorithms are indispensable in the network: one for unicast demands and one for anycast demands.

In this paper we focus on online, dynamic routing of demands. However, also many static routing optimization problems are widely discussed in the literature, e.g. [3], [10]. In [15-16] static versions of CDN network design problems are considered – heuristics and an exact algorithm based on the branch-and-cut approach are proposed.

## 4 Results

We now describe our simulation setup and scenarios. We conducted simulation experiments to evaluate the server selection algorithms: HNS, HNWS, RCS and four dynamic routing algorithms: MHA, CSPF [2], DORA using the bandwidth proportion parameter BWP=0.5 [14] and LIOA using the calibration parameter $\alpha$=0.5 [1]. It makes 12 combinations of both types of algorithms. As the performance indicator we apply the demand rejection ratio. The objective of conducting performance evaluation is twofold. First, we want to evaluate various algorithms in terms of rejection ratio for various simulation scenarios. Next, we plan to examine the influence of number and location of CDN servers on the demand rejection.

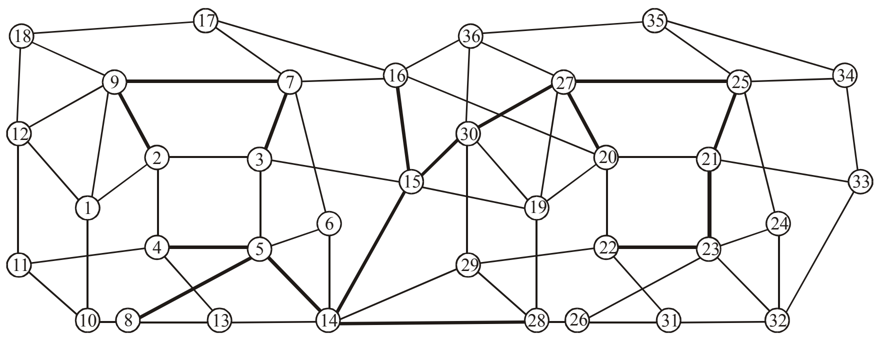

**Fig. 1.** Network topology used in the experiments

The network on which we conduct our experiment consists of 36 nodes and 144 directed links (Fig. 1). Each network node can be used as an ingress and egress node. The bold lines represent links of size 96 units while other lines are links of size 48 units. The link capacities were chosen to model capacity ratio of OC-48 circuits. During simulations, the link capacities were scaled by a factor 100 to enable establishing of thousands of LSPs. We consider static demands resembling long-lived MPLS tunnels that once established, they stay in the network for a long time. The same network was analyzed in [16].

In our experiments we conducted simulations for 12 combinations of content servers location (Table 1). The number of content servers is from 1 to 4. Since we want to examine how the number of content servers has an effect on demand rejection, we consider three cases (1A, 1B and 1C), in which there is only one server. Actually, we cannot call such scenario a CDN. Therefore, these three cases were not considered in the phase of algorithms comparison.

**Table 1.** Simulation scenarios of server number and server location

| Case | Number of servers | Location of servers | Case | Number of servers | Location of servers |
|---|---|---|---|---|---|
| 1A | 1 | 14 | 3A | 3 | 9, 14, 27 |
| 1B | 1 | 15 | 3B | 3 | 5, 23, 30 |
| 1C | 1 | 30 | 3C | 3 | 9, 15, 25 |
| 2A | 2 | 14, 30 | 4A | 4 | 5, 9, 23, 27 |
| 2B | 2 | 5, 27 | 4B | 4 | 5, 7, 25, 30 |
| 2C | 2 | 9, 25 | 4C | 4 | 9, 14, 23, 30 |

In the simulation we use the content demand ratio (CDR) parameter that is defined as the ratio of the content demands number to the number of all demands. For instance, CDR=0.3 means that 30% of all demands are addressed to content servers. In order to make performance evaluation for various scenarios, we generated randomly 10 sets of demands for each of the following five values of CDR: 0.1, 0.3, 0.5, 0.7 and 0.9. It gives 50 sets of demands in total. Each set consisting of 50000 demands was tested for each of 12 server location cases. It means that we have made 50x12=600 runs of the program that simulates combinations of server selection and routing algorithms. We assume that the requested bandwidth of unicast and content demand is randomly distributed between 1 to 10 units of bandwidth. However, in the context of content demand this is the downstream bandwidth requirement of the path from the server to the user. The upstream path in the opposite direction requires only 1 bandwidth unit. It is due to the asymmetric character of traffic in the Internet.

In the simulation we used the following methodology. For a content demand we first apply the server selection algorithm. If the algorithm cannot select a server, the demand is rejected. Otherwise, the dynamic routing algorithm is applied to find two LSPs between the client node and the server node and reserved bandwidth is updated on all links along both paths. A unicast demand is processed in a standard way, i.e. the path between end nodes of the demand is calculated by a routing algorithm. The demand is rejected if the algorithm cannot yield a feasible path. Otherwise, reserved bandwidth is updated on all links along the path. Note that for both kinds of demand we apply the same routing algorithm. We repeat the same procedure for all 12 combinations of server selection and routing algorithms.

We do not model the real-time aspects of signaling protocols. We model these processes as functioning perfectly and we focus only on the purely capacity-related issues. After each individual demand placement, the available capacity is updated and the next demand in the sequence gets its chance, and so on until all demand have had a chance to be processed. Outcome of each experiment is the limiting outcome as it depends purely on basic routing and capacity considerations – which are repeatable by others. On the contrary, any attempt to also model the signaling dynamics is probably not repeatable as it is so utterly dependent on exact implementation details and simulation timing. Random timings and delays in signaling interactions with capacity seizure effects, protocol delays, and network timing effects will ultimately determine the result.

**Table 2.** Percentage of rejected demands aggregated over various scenarios

| | Value of CDR parameter | | | | | | Number of servers | | |
|---|---|---|---|---|---|---|---|---|---|
| Algorithms | All | 0.1 | 0.3 | 0.5 | 0.7 | 0.9 | 2 | 3 | 4 |
| HNS_DORA | 32.95 | 27.93 | 25.79 | 29.29 | 35.37 | 46.38 | 41.81 | 32.29 | 24.76 |
| HNS_LIOA | **30.43** | **27.49** | **23.17** | **25.02** | **31.90** | **44.57** | **40.13** | 29.64 | **21.52** |
| HNS_CSPF | 31.62 | 27.54 | 23.88 | 27.10 | 34.00 | 45.57 | 40.78 | 31.02 | 23.05 |
| HNS_MHA | 33.22 | 30.01 | 27.01 | 28.86 | 34.66 | 45.55 | 41.91 | 32.86 | 24.88 |
| HWNS_DORA | 33.08 | 27.98 | 25.99 | 29.47 | 35.50 | 46.45 | 41.89 | 32.38 | 24.97 |
| HWNS_LIOA | 30.44 | 27.49 | 23.22 | **25.02** | **31.90** | **44.57** | 40.14 | **29.63** | 21.55 |
| HWNS_CSPF | 31.62 | 27.51 | 23.90 | 27.11 | 34.00 | 45.58 | 40.79 | 31.01 | 23.06 |
| HWNS_MHA | 33.23 | 30.01 | 27.06 | 28.87 | 34.66 | 45.55 | 41.91 | 32.88 | 24.90 |
| RCS_DORA | 39.27 | 30.22 | 32.52 | 37.82 | 43.88 | 51.92 | 45.42 | 38.68 | 33.73 |
| RCS_LIOA | 36.58 | 29.53 | 28.69 | 34.13 | 41.29 | 49.24 | 43.46 | 35.50 | 30.69 |
| RCS_CSPF | 37.48 | 29.53 | 29.80 | 35.39 | 42.08 | 50.62 | 44.17 | 36.64 | 31.64 |
| RCS_MHA | 38.44 | 31.93 | 32.45 | 36.61 | 41.84 | 49.39 | 45.16 | 37.86 | 32.31 |

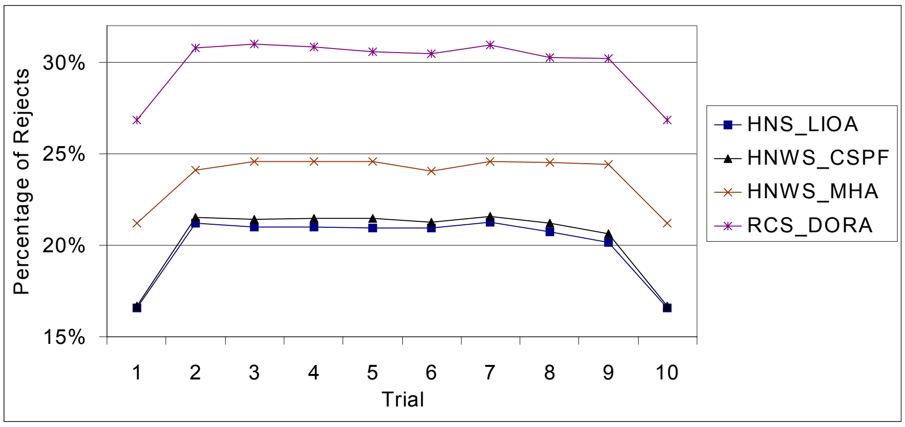

**Fig. 2.** Percentage of rejected demands versus trial number for case 4B and CDR=0.3

Table 2 shows the percentage of rejected demands for aggregated results. We present results aggregated for all tested values of CDR (second column) and individual results for each of CDR value (columns 3-7). Also the percentage of rejected demands obtained for different number of content servers placed in the network aggregated over all value of CDR is reported (columns 8-10). The best result (lowest rejection ratio) for each category is typed bold. The best performance is obtained for server selection algorithm HNS and routing algorithm LIOA. RCS algorithm yields higher rejection ration than two other server selection algorithm. Good performance of LIOA confirms the results presented in [1].

Figure 2 shows the percentage of rejected demands for different algorithms obtained for the case 4B with CDR=0.3. The general trend on Figure 2 is similar to that presented in Table 2. The comparison of various algorithms performance shows that combination of HNS and LIOA provide the best performance of demand rejects. Therefore, in the remainder of the section we present results of these two algorithms in order to examine the influence of the number and location of CDN servers on demand rejection ratio.

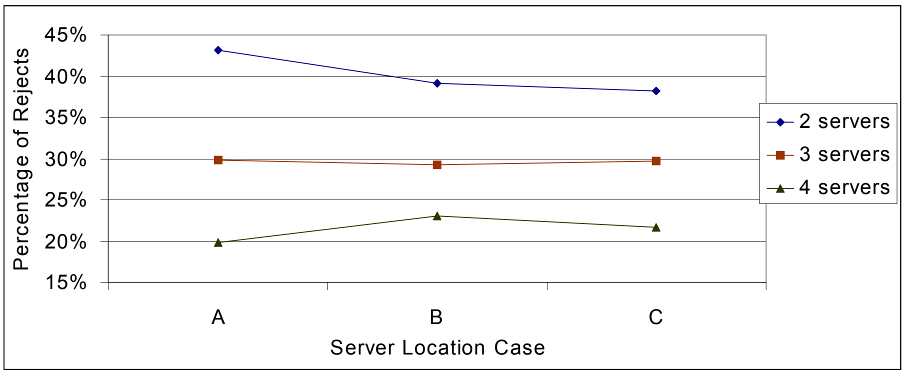

**Fig. 3.** Percentage of rejects for various simulation scenarios of location and number of servers

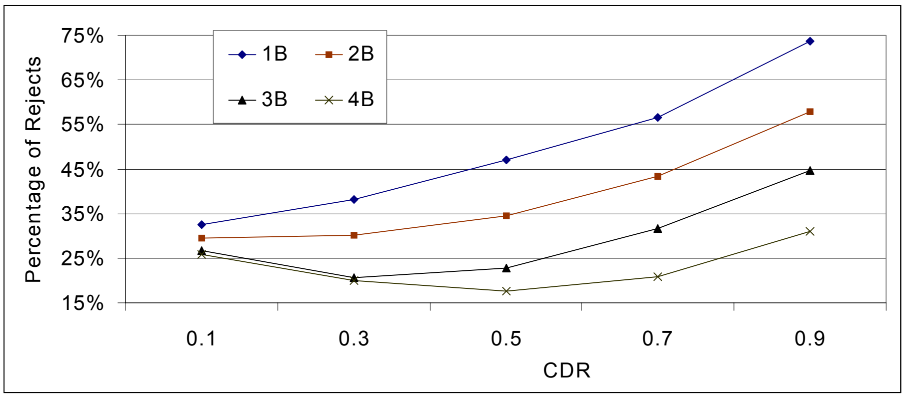

**Fig. 4.** Percentage of rejects as a function of CDR for cases 1B, 2B, 3B and 4B

Figure 3 shows the percentage of rejects for various simulation scenarios in terms of the number of content servers and server location. Each point in the figure represents the result aggregated over all tested 50 demands patterns. As one can see, the performance depends on the server selection. The largest deviation is observed for 2 servers case. Obviously, when the number of servers increases, the rejection ratio goes down.

Figure 4 depicts the demand rejection as a function of CDR for scenarios 1B, 2B, 3B and 4B. If the number of servers is 1 or 2, the percentage of rejected demands grows with increasing of CDR. However, when the number of servers is 3 or 4, curves are different. Initially, when CDR is below 0.3 for 3 servers and 0.5 for 4 servers, the percentage of rejects decreases. It follows from the fact that there are more content servers in the network, which can accept content demands. Consequently, LSPs associated with content demands are shorter and leave more open capacity for other demands. However, when the ratio of content demands grows more, the limitation on capacity of links leading to servers surpasses the effect of higher number of content servers. Thus, relatively more content demands are rejected due to capacity constraint on links leading to server nodes. Similar trend can be observed on Figure 5 presenting the rejection ratio of content and unicast demands aggregated for all cases with 4 content servers.

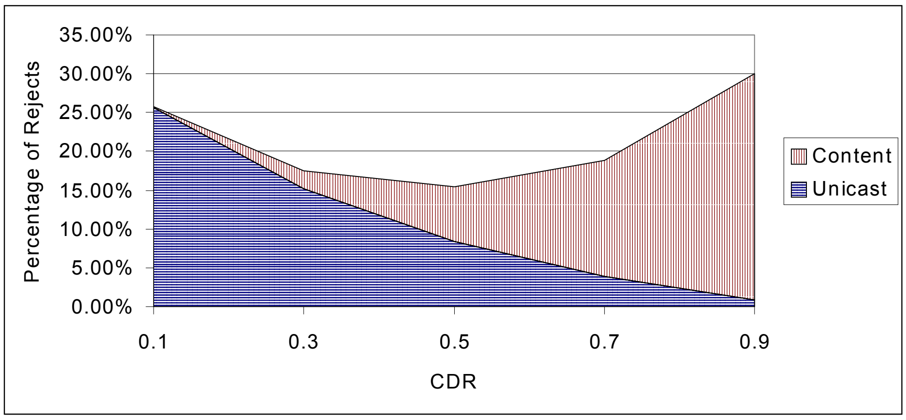

**Fig. 5.** Content vs. unicast demand rejection aggregated for 4 servers scenario

Figure 6 shows the percentage of rejected demands as a function of demand number for cases 1B, 2B, 3B and 4B with CDR=0.5. We can watch that increasing the number of content servers reduces the number of rejects. For one server case after processing of 15000 demands the network starts to reject demands. If there are 4 content servers, about 37000 demands are accepted until demand rejection occurs.

## 5 Conclusion

In this paper we have examined a network that can accept two kinds of demands: content demands to CDN servers and standard unicast demands. We have proposed three algorithms for constraint-based selection of content servers. These algorithms either find the best server in terms of traffic engineering or reject the CDN request.

One of the main advantages of our approach is that for routing of content demand the traditional unicast QoS dynamic routing algorithm can be applied. The only difference, comparing to unicast demands, is that two LSPs are established between client node and server node. We have studied the relative performance of four robust unicast routing algorithms cooperating with tree server selection algorithms. The lowest number of rejected demands provides the combination of HNS and LIOA algorithms. Our analysis of results suggests that both number and location of content servers contribute significantly to overall network reliability in terms of the rejection ratio. Important factor that has en effect on the number of rejected demands is the proportion of content traffic.

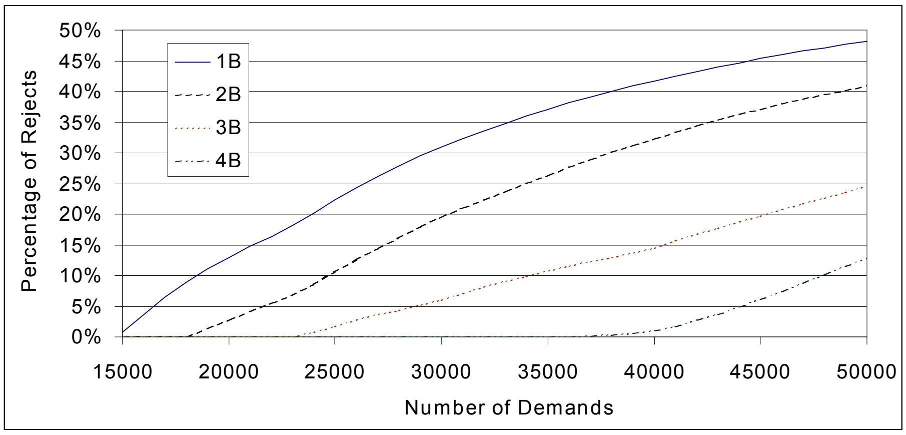

**Fig. 6.** Percentage of rejects as a function of demand number for cases 1B, 2B, 3B and 4B

As an ongoing work, we want to explore the performance of server selection algorithms in a CDN-enabled MPLS network from the perspective of network survivability. We plan to simulate content and unicast demand rerouting for various failure scenarios using different restoration methods.

**Acknowledgements.** This work was supported by a research project of the Polish State Committee for Scientific Research carried out in years 2005-2007.

## References

1. Bagula, B., Botha, M., Krzesinski, A.: Online Traffic Engineering: The Least Interference Optimization Algorithm. To appear in the IEEE ICC 2004 (2004)
2. Crawley, E., Nair, R., Jajagopalan, B., Sandick, H.: A Framework for QoS-based Routing in the Internet. RFC2386 (1998)
3. Gola, M., Kasprzak, A.: Exact and Approximate Algorithms for Two–Criteria Topological Design Problem of WAN with Budget and Delay Constraints. Lectures Notes in Computer Science, LNCS 3045 (2004), 611-620
4. Hao, F., Zegura, E., Ammar, M.: QoS routing for anycast communications: motivation and an architecture for DiffServ networks. IEEE Communication Magazine, 6 (2002), 48-56

5. Kar, K., Kodialam, M., Lakshman, T.: Minimum interference routing of bandwidth guaranteed tunnels with MPLS traffic engineering applications. IEEE JSAC, 12 (2000), 2566-2579
6. Katz, D., Kompella, K., Yeung, D.: Traffic Engineering (TE) Extensions to OSPF Version 2. RFC 3630 (2003)
7. Kodialam, M., Lakshman, T.: Minimum Interference Routing with Applications to MPLS Traffic Engineering. In Proceedings of INFOCOM (2000), 884-893
8. Krishnamurthy, B., Wills, C., Zhang, Y.: On the Use and Performance of Content Delivery Networks, in Proceedings of ACM SigComm Internet Measurement Workshop, (2001)
9. Peng, G.: CDN: Content Distribution Network. Technical Report, sunysb.edu/tr/rpe13.ps.gz, (2003)
10. Pióro, M., Medhi, D.: Routing, Flow, and Capacity Design in Communication and Computer Networks. Morgan Kaufman Publishers (2004)
11. Qiu, L., Padmanabhan, V., Voelker, G.: On the Placement of Web Server Replicas, in Proceedings of IEEE Infocom (2001), 1587-1596
12. Rosen, E., Viswanathan, A., Callon, R.: Multiprotocol Label Switching Architecture. RFC 3031 (2001)
13. Smit, H., Li, T.: ISIS Extensions for Traffic Engineering. RFC 3784 (2004)
14. Szeto, W., Boutaba, R., Iraqi, Y.: Dynamic Online Routing Algorithm for MPLS Traffic Engineering. Lectures Notes in Computer Science, LNCS 2345 (2002), 936-946
15. Walkowiak, K.: Some approaches to solve a Web replica location problem in MPLS networks, in Internet Technologies, Applications and Societal Impact, Kluwer (2002), 61-72,
16. Walkowiak, K.: An exact algorithm for design of content delivery networks in MPLS environment. Journal of Telecommunications and Information Technology 2 (2004), 13-22
17. Walkowiak, K.: Survivable Online Routing for MPLS Traffic Engineering. Lectures Notes in Computer Science, LNCS 3266 (2004), 288-297
18. Weber, S., Cheng, L.: A Survey of Anycast in IPV6 Networks. IEEE Comm. Magazine, 1 (2004), 127-132
19. Yamada, H. *et al.*: QoS Control by Traffic Engineering in Content Delivery Networks. Fujitsu Sci. Tech. J., 2 (2003) 244-254