RC 22450 (Log# W0205-052) (05/06/2002)
Computer Science/Mathematics

IBM Research Report

Using Mobility Support for Request-Routing in IPv6 CDNs

A. Acharya, A. Shaikh

IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, New York 10598

Using Mobility Support for Request-Routing in IPv6 CDNs

Arup Acharya   Anees Shaikh

IBM T.J. Watson Research Center
Hawthorne, NY 10532

Abstract

Content distribution networks have become widespread in the current IPv4-based Internet, and request-routing in CDNs today are primarily based on DNS mechanisms. This paper considers request-routing in CDNs as and when IPv6 networks begin to be deployed. Unlike IPv4, IPv6 includes explicit support for host mobility. That is, the functionality and mechanisms of a separate protocol, such as mobile IP (in IPv4), is part of the IPv6 protocol definition. In this paper, we demonstrate that packet routing to and from mobile hosts has a strong conceptual similarity with wide-area request routing in CDNs. We then present schemes where different variations of request routing in IPv6 based CDNs are all accomplished via the same IPv6 mechanisms that are used to support mobility. These schemes will eliminate the need for a separate infrastructure, like the DNS, to perform request-routing in IPv6 CDNs.

1 Introduction

Growing reliance on the Web to deliver business-critical content at high levels of performance and reliability, and at reduced costs, has led to the advent and acceptance of content distribution networks (CDNs) as part of the Internet infrastructure. These networks are collections of servers or caches that deliver content to users on behalf of content providers. The intent of a CDN is to serve content to a client from a CDN server such that the response-time performance is improved over contacting the origin server directly.

The overall performance of a content distribution network (CDN) is largely determined by its ability to direct client requests to the most appropriate server. An inefficient request-routing system or poor server selection decision can defeat the purpose of the CDN, namely to improve client response time over accessing the origin server. Request-routing in CDNs consists of routing a client's TCP connection (HTTP request) to a "proximal" server where proximity may be defined in terms of number of network hops, or more broadly, as a server that provides the best service (e.g. least loaded server). This, therefore, may be viewed as primarily a routing function. Request-routing in most commercial CDNs is accomplished with the Domain Name System (DNS) by extending the directory service of the DNS, while remaining transparent to the client protocol stack.

In this paper, we propose the use of mobility mechanisms provided in IPv6 as an alternate means for request-routing in IPv6-based CDNs. We argue that IPv6 provides sufficient flexibility with regard to how packets are routed such that request-routing can be accomplished entirely within the IPv6 networking layer. This is made possible in IPv6, but not in IPv4, primarily because IPv6 incorporates support for host mobility and optimized packet routing to a mobile node. Our approach removes the need for additional protocols and applications, or other network services such as DNS to perform request-routing.

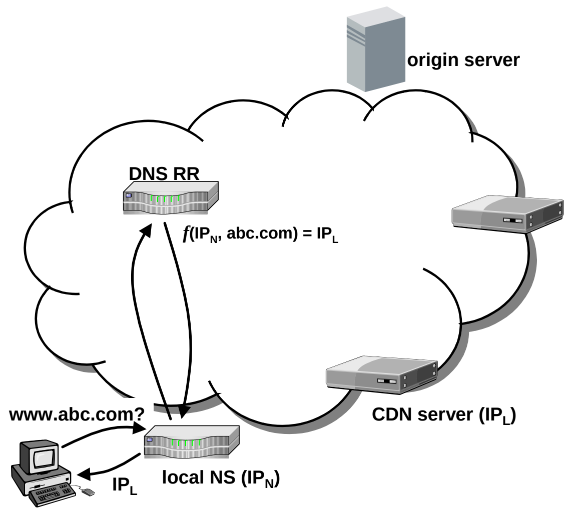


Figure 1: DNS-based request-routing

In the next section, we outline the operation of DNS-based request-routing in IPv4 CDNs, along with some of the issues that arise with the DNS approach. Section 3 describes the mobility support in IPv6, pointing out differences with mobile IPv4. In Section 4 we describe our proposal for leveraging mobility support in IPv6 for request-routing. Section 5 discusses some additional issues such as how to implement content internetworking, and how to capitalize on the large address space offered by IPv6.

2 Request-Routing in CDNs

Clients typically access content from CDN servers by first contacting a request router. The request router makes a server selection decision and returns a server assignment to the client. The client then retrieves content from the specified CDN server. CDN request routers usually rely on a combination of static and dynamic information when choosing the best server, including such metrics as server and network conditions, or client proximity to the candidate servers. Several currently used request-routing techniques are described in [1].

DNS-based request routing has emerged as the *de facto* standard server selection mechanism used by most commercial content distribution service providers (CDSPs) and many equipment vendors. In this approach, clients are directed to an appropriate CDN server during the name resolution phase of Web access. As shown in Figure 1, a specialized DNS-based request router receives name resolution requests, determines the location of the client based on its local nameserver address, and returns the address of a nearby CDN server or a referral to another nameserver. The answer is typically cached at the client's local nameserver for only a short time so that the request router can adapt quickly to changes in network or server load. This is achieved by setting the associated time-to-live (TTL) field in the answer to a very small value (e.g., 20 seconds).

DNS-based request routing may be implemented with either full- or partial-site content delivery [2]. In full-site delivery, the content provider delegates authority for its domain to the CDSP or modifies its own DNS servers to return a referral (CNAME record) to the CDSPs DNS servers. In this way, all requests for `www.abc.com`, for example, are resolved to a CDN server which then delivers all of the content. With partial-site delivery, the content provider modifies its content so that links to specific objects have hostnames in a domain for which the CDSP is authoritative. For example, links to `http://www.abc.com/image.gif` are changed to `http://cdsp.net/abc.com/image.gif`. In this way, the client retrieves the base HTML page from the origin server but retrieves embedded images from CDN servers to improve performance.

DNS-based server selection is appealing due to its simplicity and generality. It requires no change to existing protocols or client software, and it works for any IP-based application. DNS request-routing also simplifies content internetworking (CDI), which allows different CDNs to share resources in order to increase their reach and scale beyond what they could provide individually [3]. If the partnering CDNs both use DNS-based request-routing, then directing a client to a neighboring CDN is simply achieved by returning a referral (e.g., a CNAME record) to the target CDN's DNS request router.

Despite these advantages, DNS-based request-routing does have several fundamental drawbacks, some of which have been recently studied and evaluated [4, 5, 1]. One problem is that request-routing is done on the granularity of DNS domains, rather than per-object, thus limiting the ability to make object-specific server selection decisions. A second problem is that requests usually come to the DNS server not from clients, but from their local nameservers. Hence, the CDN server is chosen based on the local nameserver address instead of the client, which may lead to poor decisions if clients and their local nameservers are not proximal. Also, if the local nameserver serves a very large client population (e.g., in dial-up ISPs), it will potentially return the same name lookup result to many clients, resulting in an an unexpectedly high load at the CDN server. As mentioned earlier, DNS request routers return answers with small TTLs to facilitate fine-grained load balancing. This may actually increase Web access latency, however, because clients must contact the DNS server more frequently to refresh the name-to-address mapping. The DNS was not designed for such high traffic volume, relying instead on widespread caching of information to reduce load on DNS servers. Furthermore, using the TTL to control load-balancing is unreliable, as client applications (e.g., Web browsers) often apply their own arbitrary caching policies.

3 Mobility Support in IPv6

Any IPv6 host has support for mobility. This has three key implications:

- An IPv6 host has the necessary mechanisms in place to enable it to become mobile, i.e. it can change its attachment point to the network and yet retain network connectivity and preserve any ongoing transport-layer connections.
- An IPv6 node can operate as a *home agent* for a mobile node, i.e. packets that are addressed to the mobile node are trapped by its home agent and re-routed to the mobile's node current location. The mobile node informs its home agent of its movements through *location updates*.
- A host (*correspondent node* (CN)) that initiates a connection to a mobile node can participate in route optimization such that packets to the mobile node are forwarded on a direct path to the mobile node, instead of being re-routed in a triangular fashion via the mobile node's home agent.

In standard IPv4, none of the above features are supported. A key design point of Mobile IP [6] was to support host mobility in IPv4 networks without mandating changes to every existing IPv4 node. Instead, host mobility is supported by running mobile IP at a mobile node and a home agent. Consequently, mobile IP supports the first two features above, but not the third, i.e. connections to a mobile host would be forced to use triangular routing, since packets addressed to the mobile node would be routed by default to its home agent, which would then tunnel the packets to the mobile node. While route optimization mechanisms do exist in Mobile IP [7], this requires changes to hosts that are neither mobile hosts nor home agents, and are therefore not mandated by Mobile IP. Hence, in practice such nodes will not support any of the mechanisms of mobile IP or route optimization.

In IPv6, mobility was recognized to be one of the base requirements for a new design and, therefore, the IPv6 packet header was designed to enable addressing to and from mobile hosts. The IPv6 packet header contains a destination header option, which allows additional addresses to be passed between endpoints of a transport connection in a manner that is completely transparent to intermediate routers.

4 IPv6-based Request Routing

Mobility support in IPv6 enables a correspondent node to reach a mobile node even if it does not know the mobile node's current location. Moreover, this redirection is handled completely by IP, transparent to the transport layer protocol (e.g., TCP or UDP). Our key observation is that these mobile routing mechanisms can be similarly used to direct clients to an appropriate server among a set of distributed replicas.

By exploiting the mobility support in IPv6, we can perform CDN request-routing without relying on legacy network infrastructure such as DNS, and overcome some of the problems with DNS-based server selection. To relate CDN request-routing to IPv6 mobility, consider the request router as the *home agent*, the CDN server as the *mobile node*, and the client analogous to the *correspondent node* [8]. One difference between our proposal and IPv6 mobility is that the request router does not receive any equivalent of a "location update", telling it which CDN server should be chosen. Instead, the request router must make its decision based on its own information regarding client location and CDN server load.

The IPv6-based request-routing process is illustrated in Figure 2 and explained below in detail.

1. The client begins, as before, by querying the DNS for the Web site `www.abc.com` (Figure 2(a)). The authoritative DNS server for `abc.com` is not a request router – it simply returns the (virtual) address of a request router (IP_V) for the requested domain.

2. The client proceeds to establish a connection to the request router by sending a TCP `SYN` packet destined for IP_V (Figure 2(b)). The request router applies its server selection function, which may consider the client's IP address (IP_C), along with IP_V and any other criteria such as server load.

3. The request router selects the CDN server with address IP_L, and sends the `SYN` packet to the chosen server using an IP-in-IP tunnel (Figure 2(c)). The CDN server sees that the `SYN` packet source was IP_C and was destined for IP_V. It sends a `SYN/ACK` packet addressed to the client with IP_L in the IP source address field. In this packet, the CDN server includes an IPv6 *binding update* destination option indicating that

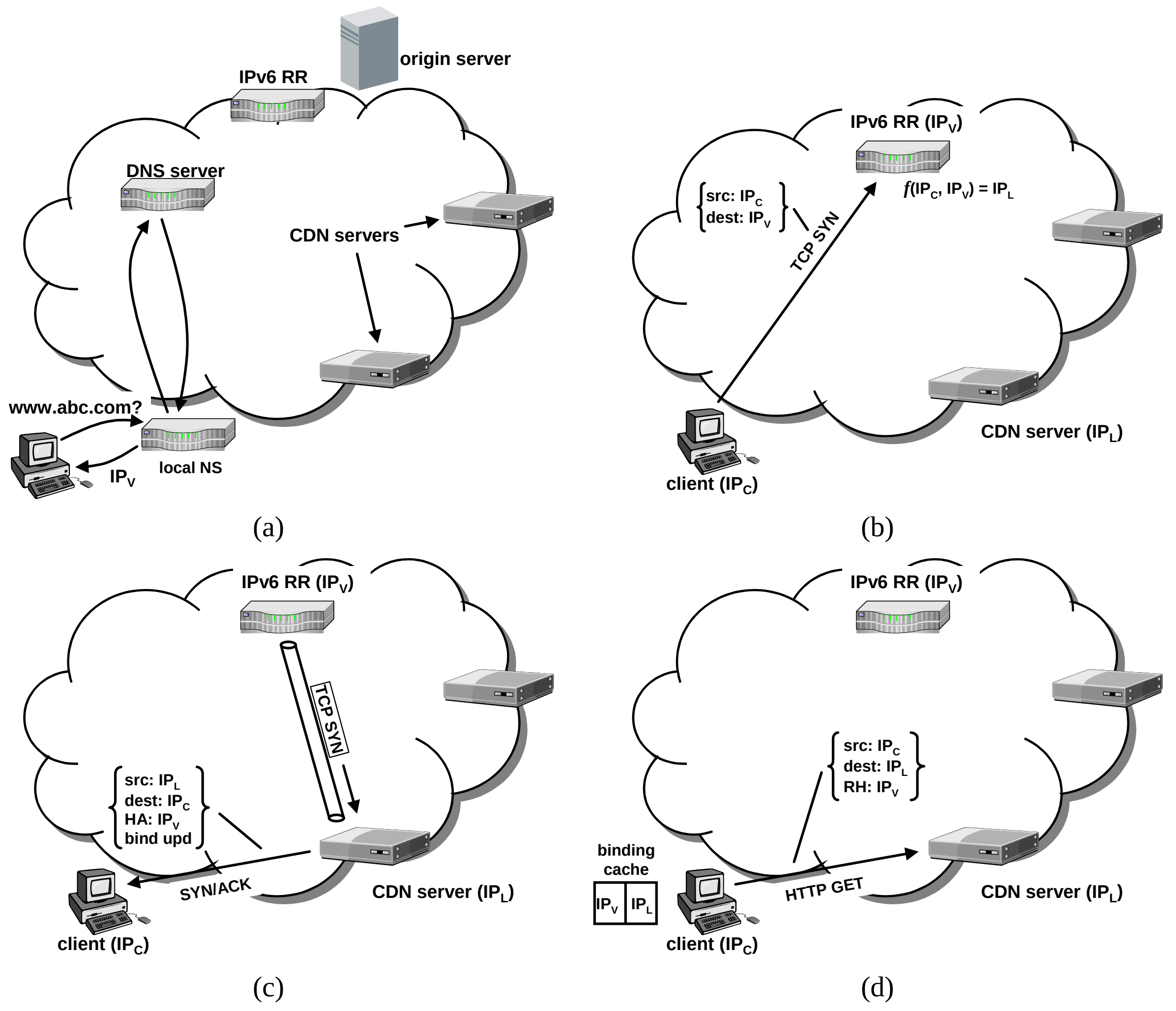
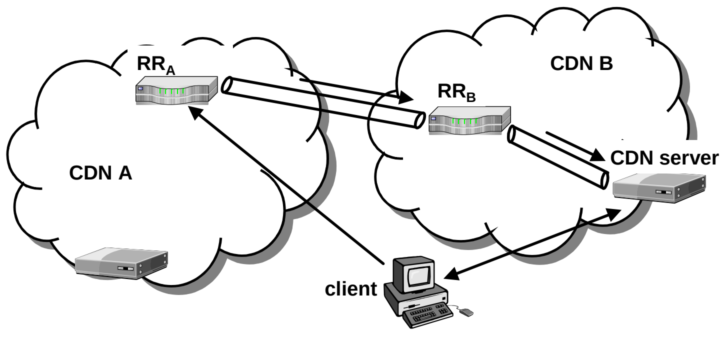


Figure 2: Routing client requests using IPv6

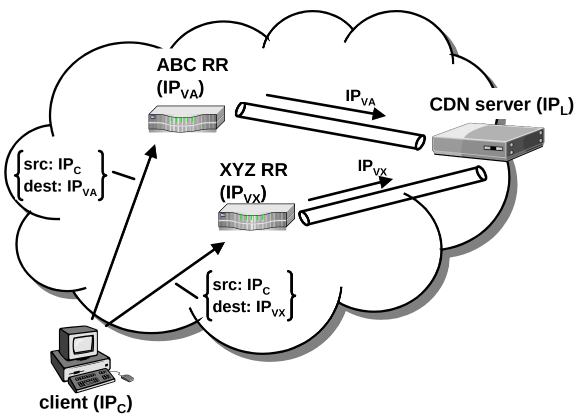
IP_V is the home address. This mobility header tells the IP layer at the client to replace the source address (i.e., IP_L) with the address in the home address field of the binding update (i.e., IP_V). The client TCP layer sees a packet with source address IP_V, thus allowing it to complete the connection.

4. The binding update mobility header also has a lifetime field which, if nonzero, has the effect of creating a *binding cache* entry at the client IP layer. In this case the binding update establishes a cache entry for address IP_V (Figure 2(d)). This entry causes the IP layer to (1) replace the destination address for packets sent to IP_V with IP_L, and (2) insert a *type 2 routing header* with IP_V in the home address field. Subsequent packets from the client (including the final ACK to complete the connection) will include this routing header as long as the binding cache entry is valid. Packets from the CDN server to the client will continue to carry a home address destination option set to IP_V, to enable the proper address replacement at the client.

Note that in step 2, the request router bases its decision on the actual client address, not the local nameserver, thus solving one of the main problems with DNS-based request-routing. This approach also addresses problems with other request-routing methods [1]. For example, the request router does not introduce connection establish-



(a)



(b)

Figure 3: Extending IPv6-based request-routing

ment overhead like application-layer redirection. Also, route optimization is part of the protocol, thus obviating the triangular routing problem in transport-layer redirection (and mobile IPv4).

The lifetime field in the binding update can be used by the CDN server to control how long it is used by the client. For example, if the load on the server suddenly increases, it can close the connection with a client and send a binding update with lifetime zero to delete the binding cache entry at the client. The client will send its next connection request to the request router for a new server assignment without requiring an additional DNS lookup. The binding cache lifetime serves much the same function as the TTL in DNS request-routing, without the possibility of being easily subverted by client-side applications.

5 Additional request-routing issues

The scheme described in Section 4 is sufficient for basic CDN request-routing. There are, however, other issues that arise when additional request-routing functions are considered. In this section we briefly discuss some of these issues.

Content internetworking: To support CDI, the request router must be able to direct client requests into neighboring CDNs for further request-routing within the target CDN. The IPv6-based scheme can support CDI by extending the initial packet tunneling into the target CDN as shown in Figure 3(a). Basically, the first connection establishment packet is tunneled twice – first, from the initial request router to the request router in the target CDN, and second, from the target request router to an appropriate CDN server. After the request is tunneled to the target CDN server, the direct connection establishment with the client proceeds as before. Note that this additional tunneling step is transparent to the client.

Request-routing granularity: One of the key benefits of using IPv6 is its large address space. Our proposal can take advantage of this feature to make finer-grained server selection decisions without requiring additional DNS lookups. For example, if all customers are assigned one virtual IP address, then the request router must be able to direct the request to any CDN server, since it would not be able to differentiate between requests for different customers. By assigning one virtual IP address per CDN customer, the CDSP can control the allocation of different customers to CDN servers. The request router uses the destination IP address to choose a CDN server that is assigned to that customer. If each customer is assigned multiple virtual IP addresses, the request router could choose a CDN server based on additional criteria, such as the type of content being requested. For example, `www.abc.com` could be resolved to IP_{V1}, while resolving `streaming.abc.com` may return IP_{V2}. In contrast to DNS-based request-routing, these name resolution results could be cached for a relatively long time since the request router uses IP_{V1} and IP_{V2} to choose an appropriate CDN server. With partial site delivery, these content-specific addresses could be part of the rewritten URLs, thus removing the need for any DNS lookups.

Handling multiple customers: An additional consideration in our proposal is the requirement for CDN servers to accept (re-routed) connections for all customer addresses. That is, each CDN server may have multiple "home addresses," where each home address corresponds to the (virtual) IP address associated with a CDN customer. In Figure 3(b), for example, we show a CDN server accepting connections via two request routers, one for customer `abc.com` and the other for customer `xyz.com`. Note that when only a subset of CDN servers is assigned to a particular CDN customer only those CDN servers need to accept the virtual IP address associated with that customer as one of their home addresses. Servers outside of the subset do not need to be programmed to recoginize that address.

Request router configuration: The CDSP may maintain a separate request router for each customer to avoid the case where a failure affects multiple customers, or may use a single (logical) request-router for all of its customers. In the latter case, the common request router may have a better view of the loading on CDN servers since it handles requests for all customers. At the same time, however, the addresses assigned to different customers may need to be assigned with a common prefix so that the request-router cluster can advertise a single prefix (i.e., through address aggregation) for routing in the wide area.

Handling cache misses: When a request is routed to a CDN server, it is possible that the server does not have the requested content in its cache. The question arises of how the CDN server contacts the origin server to fetch the content. This information may be configured at each CDN server handling a particular customer, though this approach can introduce excessive management overhead in a large CDN. An alternative is to use the request router itself to route the fetch request to the origin server on a cache miss. The request router policies can be configured so that requests originating from CDN servers are routed to the appropriate origin server. For example, when the request router sees a request from IP_L to IP_V, it knows to send the request to the origin server corresponding to IP_V (which has been programmed at the request router).

6 Conclusion

In this paper, we observed and demonstrated how support for host mobility in IPv6 may be used for request-routing in content distribution networks. The schemes we describe remove the need to overload other network infrastructure such as the DNS, and also solve many of the problems with existing request-routing mechanisms. Since host mobility support is built into the IPv6 protocol, these techniques will work for any IPv6-compliant host, without specialized request-routing support at the transport- or application-layer. We point out that IPv6 mobility has been submitted to the Internet Engineering Steering Group (IESG) as a proposed standard, but there is ongoing discussion regarding some parts of the protocol, particularly as they relate to security. None of the proposed security-related modifications would alter the basic operation of request-routing, however.

We also note that IPv6 mobility support can play a useful role in scenarios other than CDN request-routing, where there is a need to translate a virtual IP address to a physical IP address during the process of connection establishment. One application of this mechanism is virtualization support in iSCSI (SCSI-over-IP) where physical storage blocks may be moved between different storage devices without explicitly informing the client; the client simply accesses the disk with a virtual IP address which is translated to the physical disk's IP address (where the requested storage blocks reside) during the process of TCP connection establishment. We intend to explore this scenario in more detail as part of future work.

References

[1] A. Barbir *et al.*, "Known CN request-routing mechanisms." Internet Draft (draft-ietf-cdi-known-request-routing-00.txt), February 2002.

[2] B. Krishnamurthy, C. Wills, and Y. Zhang, "On the use and performance of content distribution networks," in *Proceedings of ACM SIGCOMM Internet Measurement Workshop*, November 2001.

[3] M. Day, B. Cain, G. Tomlinson, and P. Rzewski, "A model for content internetworking (CDI)." Internet Draft (draft-ietf-cdi-model-02.txt), May 2002.

[4] A. Shaikh, R. Tewari, and M. Agrawal, "On the effectiveness of DNS-based server selection," in *Proceedings of IEEE INFOCOM*, (Anchorage, AK), April 2001.

[5] Z. M. Mao, C. D. Cranor, F. Douglis, M. Rabinovich, O. Spatscheck, and J. Wang, "A precise and efficient evaluation of the proximity between web clients and their local DNS servers," in *Proceedings of USENIX Annual Technical Conference*, June 2002.

[6] C. E. Perkins (ed.), "IP mobility support for IPv4." Internet Request for Comments (RFC 3220), January 2002.

[7] C. Perkins and D. B. Johnson, "Route optimization in mobile IP." Internet Draft (draft-ietf-mobileip-optim-11.txt), September 2001.

[8] D. B. Johnson and C. Perkins, "Mobility support in IPv6." Internet Draft (draft-ietf-mobileip-ipv6-16.txt), March 2002.