RACIER – A Hierarchical Approach for Content Internetworking Through Interdomain Routing

by

Xiaojuan Cai

M.E., Southeast University, China, 1995

B.E., Southeast University, China, 1992

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE STUDIES

(Department of Computer Science)

we accept this thesis as conforming
to the required standard

The University of British Columbia

September 2002

In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of Computer Science

The University of British Columbia
Vancouver, Canada

Date Oct. 20, 2002

Abstract

Nowadays, Internet traffic is mainly comprised of the distribution of static and media rich content. While ad hoc mechanisms may extend the reach of content, Content Internetworking (CI) attempts to foster the interoperability of distinct, independent Content Networks (CNs). Unfortunately, known CI content routing approaches have scaling problems and partly duplicate existing network layer functions. In this thesis, we propose a new Routing Architecture for Content IntERnetworking (RACIER). By extending existing IP networking infrastructures to incorporate content intelligence, an Internet built on the new routing architecture can be both IP and content aware. The key features of the architecture are hierarchy and parallel IP-address based routing and name based routing, which lend themselves well to scalability and performance, while maintaining compatibility with traditional Internet architecture.

The Border Gateway Protocol (BGP) is extended to support the content internetworking in our approach. We implemented the system prototype based on the Multithread Routing Toolkit (MRT) and conducted preliminary experiments to verify our system architecture.

Contents

List of Tables

List of Figures

Acknowledgements

First of all, I would like to express my gratitude to my supervisor, Dr. Son Vuong, for his essential guidance, inspiration and suggestion. Without him, this thesis would have been impossible. I am grateful to Dr. Alan Wagner for being my second reader and giving me useful comments for improvement.

The financial support from the Computer Science Department at UBC in the forms of TA and RA as well as the partial support from the CITR Network of Centers of Excellence for my involvement in the SCE project are gratefully acknowledged.

I would also like to thank all my family members for their consistent support during my graduate study here.

Thanks also go to my project partner Ming Gan and all other friends at UBC.

XIAOJUAN CAI

The University of British Columbia
September 2002

Chapter 1

Introduction

1.1 Motivation

The past several years have seen the explosive growth of the Internet, especially the World Wide Web. The primary use of the Internet is content delivery, such as web pages, images, and increasingly, audio and video streams. Some measurements indicate that 70 to 80 percent of Internet traffic is HTTP traffic (and most of the rest of the traffic is for routing and DNS) [11]. That is, almost all of the traffic in the wide area is delivery of content, and ancillary traffic to locate it and determine how to deliver it. Today, millions of clients are accessing thousands of web sites on a daily basis, with the top 20 web sites supplying about 10 percent of the total content [11]. As the scale and use of the Internet increase, web content providers find it increasingly difficult to serve all users wishing to access their content with an appropriately low response time, especially in the face of unexpectedly high loads (often called "flash crowds"). This is the case despite the massive additions in bandwidth across the whole Internet [1]. Content Networks (*CN*s) have become a popular means of addressing this problem; they typically deploy multiple surrogates

in the Internet, allowing them to offload the management of these surrogates from individual content providers. This offers economies of scale and greater ability to respond to high loads. *CN*s also try to decrease latency for individual clients by satisfying client requests from nearby surrogates. With these benefits, CNs are of increasing importance to the overall architecture of the Web.

Despite all the advantages, *CN*s have their own limitations. It is extremely capital-intensive and operationally complex to achieve the scale necessary to be present at the edge of every network or region of the world. Even where they have content replicas close to a particular network, demand may outstrip capacity from time to time or failures may occur. Moreover, each CN also has its own physical scope and content capacity limitation. Clearly, the ability to foster interoperability of independent *CN*s through open standards to extend the reach of content can offer additional flexibility and expand the market for content networking more rapidly. This falls into the task category of "Content Internetworking" (*CI*), which aims at improving scale, fault tolerance and performance for individual *CN*s. Content Internetworking is an emerging technology that has promising features to improve Internet access experience.

IETF working standards of *CI* mainly consist of the internetworking of three subsystems: *Content Distribution* [6], *Content Request-Routing* [5] and *Accounting* [7]. Our research focus is on the Request-Routing. The *Content Request-Routing* system redirects client requests for content to an appropriate server, selecting from the origin server and surrogate servers holding that content. The choice is made according to which one is expected to best serve the client in terms of latency attributes, such as network proximity, server health and load, content availability, bandwidth usage and congestion. While *CN* internal request routing techniques

have been explored and developed extensively in the content networking industry, *Content Request Routing* in *CI* still remains a compelling technology for research and is the focus of our efforts in this thesis.

Although the IETF CDI work group specializes in *CI*, and other research groups also work on similar topics, most of the existing work focuses on addressing the problem at the application layer. We believe addressing the problem on the application layer constitutes a duplication of network layer routing infrastructure and thus violates the principle of clean layering architecture. Some other approaches [11,12] delegate content routing to the network layer and adopt a pure name based routing scheme as the only routing mechanism on the Internet. This is not practical from the deployment viewpoint and is difficult to scale.

This thesis focuses on a new approach to solve the request routing internetworking problem in the *CI*, aiming at overcoming most of the defects of the existing methods.

1.2 Thesis Contributions

A novel hybrid Routing Architecture for Content IntERnetworking (*RACIER*) is proposed in this thesis that combines the strengths of the previous approaches while overcoming their disadvantages. In our approach, content internetworking is delegated to the network layer to fully exploit existing routing infrastructures. *RACIER* seamlessly supports both IP-address and name-based routing; these two mechanisms function in parallel on the Internet.

We adopt a hierarchical system architecture to achieve scalability, which consequently enables a hierarchical request resolution process. Given the fact that a *CN* is normally under the management of a single set of policies and operated by a

single internet service provider (ISP), it is reasonable to assume that most *CNs* are confined to the boundary of one *Autonomous System* (AS). In the few cases where a *CN* encompasses multiple ASs, it could have multiple gateways residing on different ASs, serving the purpose of interconnection. We therefore build an internetworking architecture within the framework of AS interconnections.

Our Architecture accommodates both IP and content routing, and is a superset of BGP4 [8]. This makes it compatible with the existing IP routing architecture and infrastructure and is easy to deploy. The dual addressing mechanism of *Name+*IP we adopt in the routing process utilizes the features of existing network functionality efficiently and greatly reduces the round trip time for content request resolution. This hybrid routing architecture along with hierarchical routing and resolution scheme make request routing work effectively, which greatly reduces the size of the content route table for individual *CN*s.

In our approach we choose a distributed routing scheme over a centralized one, in which original content servers are treated equally as a replicated server. Thus, we don't have an authoritative request routing system as in [5], and thus avoid the performance and scalability bottleneck.

Furthermore, in our design, only when a client requests content that is not replicated across the globe, and whose *origin* server is located in a remote AS, will the request be redirected to a normal DNS [9, 10] resolution chain. According to statistics [11], a relatively small number of web sites comprise a large proportion of the traffic. Discriminating between the handling of replicated content and non-replicated content further reduces the size of route tables and allows for finer-grained content routing. Our approach significantly reduces the response time for accessing replicated content, especially for the most popular content sites, the routes to which

are likely to be cached on local *CIGs*. Furthermore, our experiment shows that our approach only incurs negligible round trip time overhead for accessing non-replicated content, as compared to normal DNS resolution performance. Our approach thus lends itself well to performance and scalability, while maintaining compatibility with the traditional Internet architecture.

To prevent the overloading of a single "best" server, we also explore and embed a global load-balancing scheme, by supplying multiple good routes to a piece of content simultaneously, and by using a weighted random selection algorithm to improve the overall load distribution.

A complete suite of protocols for our content internetworking architecture have been developed, including the following: *Content Border Gateway Protocol* (CBGP), *Content Gateway Interconnection Protocol* (CGIP), *Content Gateway Border Protocol* (CGBP), encompassing inter-AS, inter-*CN* and *CN*-AS communication, respectively. We have also implemented a system prototype based on a routing simulation toolkit called *Multi-threaded Routing Toolkit* (MRT) [14], and conducted experiments for system evaluation and analysis.

1.3 Thesis Organization

The thesis comprises of 6 chapters.

In Chapter 2, the background information is introduced which includes the definition and composition of *CN*s, the proposed draft for the solution of CI from IETF CDI group, and brief introduction to BGP. Chapter 3 describes our hierarchical system architecture and the content request resolution process. A detailed design about the interdomain content routing for the content internetworking and routing decision process is provided in Chapter 4. Chapter 5 gives the prototype

implementation and experiment setup with performance evaluation and analysis. Chapter 6 concludes the thesis and discusses future work.

Chapter 2

Background and Related Work

2.1 Content Networks and Proprietary Solutions

We introduce the origin of CNs, the composition of a CN and a few proprietary CN solutions in this section.

2.1.1 Traditional Content Access Accelerators

The past several years have seen the evolution of technologies centered around "content." Protocols, appliances, and entire markets have been created exclusively for the location, download, and usage tracking of content. Some sample technologies in this area have included web caching proxies, content management tools, intelligent "web switches", and advanced log analysis tools.

These technologies typically play a role of solving the "content delivery problem". Abstractly, the goal in solving this problem is to arrange a rendezvous between a content source at an origin server and a content sink at a viewer's user agent. In a trivial case, the rendezvous mechanism is that every user agent sends every request directly to the origin server named in the host part of the URL identifying

the content [1].

As the audience for the content source grows, so do the demands on the origin server. To achieve better performance, the apparent single logical server may in fact be implemented as a large "farm" of server machines behind a switch. Both caching proxies (acting on behalf the clients' side) and reverse caching proxies (acting on behalf of the server side) can be deployed between the client and server, so that requests can be satisfied by some cache instead of by the original server. In the caching approach, an ISP can also deploy regional parent caches to form a hierarchy that further aggregates user requests and responses.

Both server farms and hierarchical caching are useful techniques, but have limits. Server farms can improve the scalability of the origin server. However, since the multiple servers and other elements are typically deployed near the origin server, they do little to improve performance problems that are due to network congestion. Caching proxies can improve performance problems due to network congestion (since they are situated near the clients), but they cache objects based on client demand. Caching based on client demand performs poorly if the requests for a given object, while numerous in aggregate, are spread thinly among many different caching proxies [1]. Also, caches don't provide enough control over what data is actually served by them.

Thus, a content provider with a popular content source may find that it has to invest in large server farms, load balancing, and high-bandwidth connections to keep up with demand. Even with these investments, the user's experience may still be relatively poor due to congestion in the network as a whole. To address these limitations, CDN (Content Delivery Network) or *CN* (Content Network) [1], which is a special type of network regarding content delivery, has been deployed in

increasing numbers in recent years.

2.1.2 Content Networks (CNs)

A *CN* essentially spreads server-farm-like configurations out to network locations more typically occupied by caching proxies. A *CN* has multiple replicas of each content item being hosted. A request from a browser for a single content item is directed to a "good" replica, where "good" usually means that the item is served to the client quickly compared to the time it would take to fetch it from the origin server, with appropriate integrity and consistency. A *CN* typically incorporates dynamic information about network conditions and load on the replicas, directing requests to a good replica and to balance the load.

A *CN* has some combination of a content-delivery infrastructure, a request-routing infrastructure, a distribution infrastructure, and an accounting infrastructure. The content-delivery infrastructure consists of a set of "surrogate" servers [22] that deliver copies of content to sets of users. The request-routing infrastructure consists of mechanisms that move a client toward a rendezvous with a surrogate. Many request-routing systems route users to the surrogate that is physically "closest" to the requesting user, or to the "least loaded" surrogate. However, the only requirement of the request-routing system is that it route users to a surrogate that can serve the requested content [1]. The distribution infrastructure consists of mechanisms that move content from the origin server to the surrogates. Finally, the accounting infrastructure tracks and collects data on request-routing, distribution, and delivery functions within the *CN* [1].

Figure 2-1 is a diagram depicting a simple *CN* as described above:

Compared to using servers and surrogates in a single data center, a *CN* is

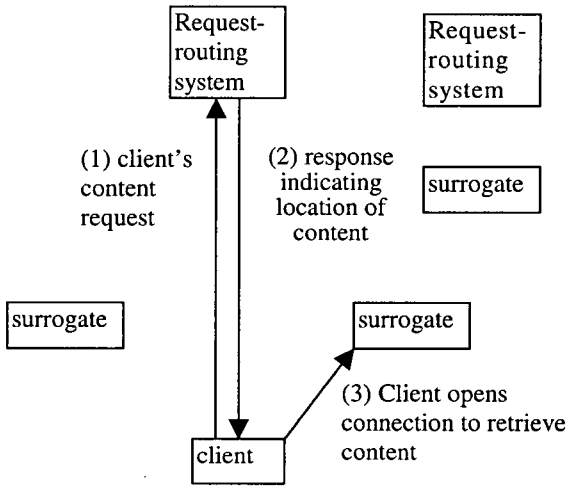


Figure 2.1: Structure of a *CN*

a relatively complex system encompassing multiple points of presence, in locations that may be geographically distant. Operating a *CN* is not easy for a content provider, since a content provider needs to focus its resources on developing high-value content, not on managing network infrastructure. Instead, in a more typical arrangement, a network service provider builds and operates a *CN*, including the request-routers, the surrogates, and the content distributors. This service provider establishes (business) relationships with content publishers and acts on behalf of their origin sites to provide a distributed delivery system. The value of that *CN* to a content provider is a combination of its scale (the increased aggregate infrastructure size) and its reach (the increased diversity of content locations).

Some *CN*s create their network across other providers' physical network, and these *CN* providers must be cautious of the quality of the facilities they use. Because they don't control the networks, or data centers that host their equipment, they must be diligent in evaluating quality and effectiveness. If they are unable to negotiate their requirements, they need to move elsewhere, which can be difficult

and costly. So, over time, pure playCDNproviders, such as Akamai, are likely to be acquired by a larger service provider that wants to enhance its services suite.

2.1.3 Sample Proprietary Solutions for CN

Most *CN*s constructed today do not rely on overly complex or intricate technologies. Akamai's service was primarily based on homegrown applications and technologies, while other providers' services were built using many standard products from companies such as Cisco, Nortel, and Inktomi. We give a brief introduction about solutions provided by Cisco and Akamai in the following.

- Cisco Distributed Director

Cisco's Distributed Director (DD) is its main product to support *CN*. DD performs load distribution and content routing in a manner that accounts for relative client-to-server topological proximities ("distances") to determine the "best" server [28]. It uses the Director Response Protocol (DRP) to gather the route table information from Cisco routers that are DRP enabled [28].

DD acts as the primary DNS caching name server for a specific host name or subdomain. It redirects a name lookup from the main site to a replica site closer to the requesting client address, making "network intelligent" load distribution decisions without considering the server side metric [28].

With this product, clients have to incur the response time penalty of accessing this main site DD before being directed to a closer site.

- Akamai Solution

Akamai uses its proprietary technology called FreeFlow to deliver content. Figure 2-2 illustrates a typical user interaction with a FreeFlow-enabled website.

First, the user's browser sends a request for a web page to the site. In response, the web site returns the appropriate HTML code as usual, the only difference being that the enclosed embedded object URLs have been modified to point to the Akamai network. As a result, the browser next requests and obtains media-rich embedded objects from an optimally located Akamai server, instead of from the home site [20].

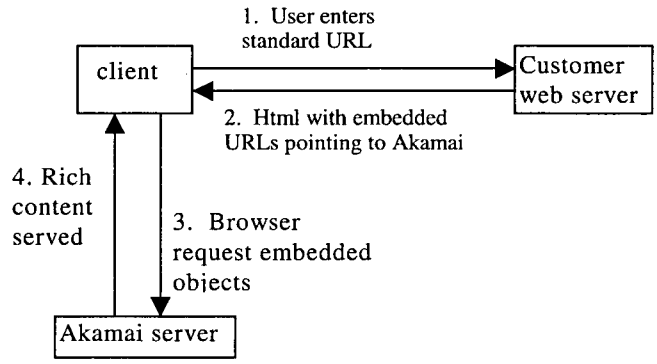


Figure 2.2: Typical user interaction with an Akamaized web site

The FreeFlow DNS system makes fast delivery of the requested content by resolving each **.g.akamai.net* server name, which represents the customer content server, to the IP address of the Akamai server that delivers the requested content to the user most quickly. FreeFlow DNS is implemented as a 2-level hierarchy of DNS Web servers: high-level *.akamai.net* servers (HLDNS) and low-level *.g.akamai.net* servers (LLDNS). Each HLDNS server is responsible for directing each query it receives to an LLDNS server that is close to the requesting client. The LLDNS servers perform the final resolution of IP name to server address, directing each client to the Akamai server best located to serve the client's requests. FreeFlow DNS continuously monitors network conditions and the status of each server [20].

2.2 Internet Draft for Content Internetworking

We introduce the proposed solution for CI from IETF CDI group in this section.

2.2.1 Motivation of CN Internetworking

There are limits to how large the scale and reach of any one network can be. The increase in either scale or reach is ultimately limited by the cost of equipment, the space available for deploying equipment, and/or the demand for that scale/reach of infrastructure. Sometimes a particular audience is tied to a single service provider or a small set of providers by constraints of technology, economics, or law. Other times, a network provider may be able to manage surrogates and a distribution system, but may have no direct relationship with content providers. Such a provider strives for a means of affiliating his delivery and distribution infrastructure with other parties who have content to distribute. The proliferation of content networks and content networking capabilities brings increasing interest in interconnecting content networks so as to provide larger scale and/or reach to each participant than they could otherwise achieve.

The following two subsections introduce the internetworking schema developed by the current Internet working draft [5]. Instead of introducing all three subsystems' internetworking, including distribution and accounting, we focus on introducing the request routing internetworking.

2.2.2 Request Routing (RR) Internetworking

Request routing internetworking is the interconnection of two or more request routing systems which increases the number of reachable surrogates. In order for a publisher's content to be delivered by multiple *CN*s, each Content Network request

routing system is federated under the Universal Resource Identifier (URI) name space of the publisher object. This federation is accomplished by first delegating the authority of the publisher URI name space to an authoritative request routing system. This authoritative request routing system subsequently splices each interconnected (it is called "peeing" in the IETF draft) content network request routing system into this URI name space, and transitively delegates URI name space authority to them for their participation in request routing. Figure 2-3 is a diagram showing the request routing (RR) internetworking system architecture. There could be multiple levels of inter-CN interconnection beyond what is shown in the sample architecture of Figure 2-3.

The request routing internetworking system is hierarchical in nature. There exists exactly one request routing tree for each publisher URI. The authoritative request routing system is the root of the request-routing tree. There may be only one authoritative request routing system for a URI request routing tree. Subordinate to the authoritative request routing systems are the request routing systems of the first level peering *CN*s. There may exist recursive subordinate request routing systems of additional peering *CN* levels [4].

2.2.3 Request-Routing

The actual "routing" of a client request is through RR *CIGs*. The authoritative request routing *CIG* , which is a globally centralized node, receives the client requests and forwards them to an appropriate peering *CN*.

This process of Inter-*CN* request-routing may occur multiple times in a recursive manner between request routing *CIGs* until the request routing system arrives at an appropriate *CN* to deliver the content.

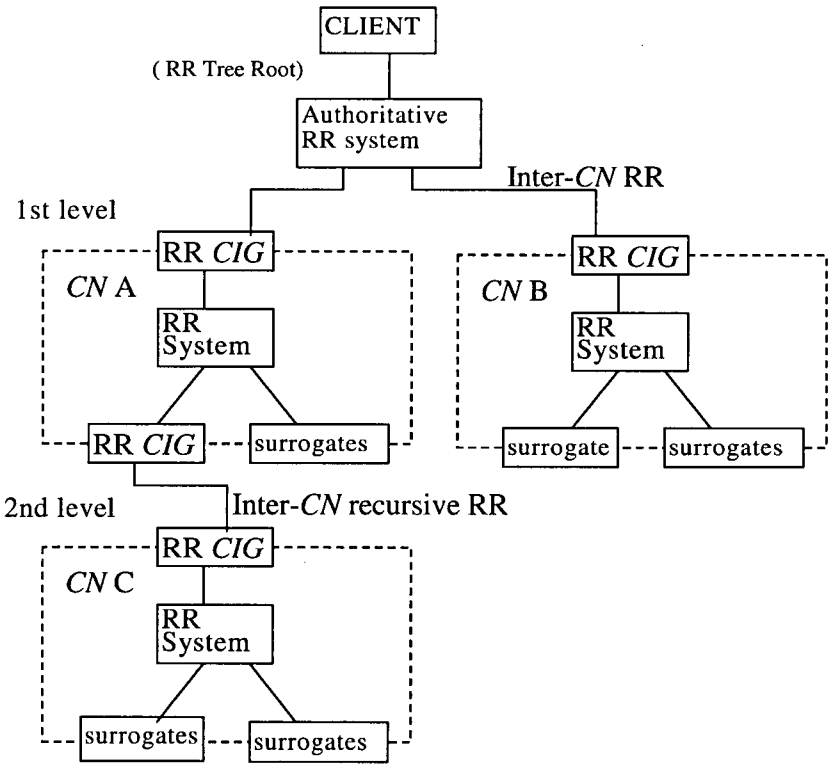


Figure 2.3: Request routing internetworking system architecture

In this schema, there may exist multiple levels of off-path (path means the physical path to the content) redirections between *CN*s before a name lookup is finally solved, and this adds a fairly high cost to the content access process. This is further exacerbated by the necessity of going to the globally centralized authoritative request routing system first, which is a performance bottleneck. In addition, advertisements about aspects of topology, geography and performance of a single *CN* to other *CN*s, required in [5], does not help much in making the global request routing decision due to the lack of knowledge of the global topology. The feature of content networks being overlay networks inherently makes decision processes very complex [5].

2.3 A Glance of BGP

The Border Gateway Protocol (BGP) is an inter-Autonomous System (AS) routing protocol. The primary function of a BGP speaking system is to exchange network reachability information with other BGP systems. This network reachability information includes information on the list of ASs that the reachability information traverses. For this reason, BGP is called a path vector protocol. The network reachability information is sufficient to construct a graph of AS connectivity, from which routing loops may be pruned and some policy decisions at the AS level could be enforced [8].

BGP4 is the newest version of BGP and it provides a new set of mechanisms for supporting classless interdomain routing by advertising an IP prefix with arbitary length which is less than 32. BGP4 also introduces mechanisms which allow the aggregation of routes, including aggregation of AS paths.

BGP uses TCP as its transport protocol (port 179). This ensures that all

transport reliability, such as retransmission, is taken care of by TCP and does not need to be implemented in BGP itself.

Two BGP routers form a transport protocol connection with each other. These routers are called neighbors or peers. Peer routers exchange multiple messages to open and confirm connection parameters, such as BGP version. In a case of disagreement between peers, notification errors are sent, and the peer connection does not get established. After the connection is established, all candidate BGP routes are exchanged initially, and later, incremental updates are sent as network information changes.

Routes are advertised between a pair of BGP routers in UPDATE messages. The UPDATE message contains, among other things, a list of ¡length, prefix¿ tuples that indicate the list of destinations reachable via each system. The UPDATE message also contains the path attributes, which include information such as the degree of preference for a particular route [22]. Path attributes fall into four separate categories [8]:

1. Well-known mandatory.
2. Well-known discretionary.
3. Optional transitive.
4. Optional non-transitive.

Well-known attributes must be recognized by all BGP implementations. Some of these attributes are mandatory and must be included in every UPDATE message. Others are discretionary and may or may not be sent in a particular UPDATE message.

All well-known attributes must be passed along (after proper updating, if necessary) to other BGP peers.

In addition to well-known attributes, each path may contain one or more optional attributes. It is not required or expected that all BGP implementations support all optional attributes. The handling of an unrecognized optional attribute is determined by the setting of the Transitive bit in the attribute flags octet. Paths with unrecognized transitive optional attributes should be accepted. If a path with unrecognized transitive optional attribute is accepted and passed along to other BGP peers, then the unrecognized transitive optional attribute of that path must be passed along with the path to other BGP peers [8].

Important well-known attributes include the following:

- AS_PATH: This attribute identifies the autonomous systems through which routing information carried in this UPDATE message has passed. The components of this list can be AS_SETs or AS_SEQUENCEs. AS_SETs is an unordered set of ASs a route in the UPDATE message has traversed, while AS_SEQUENCEs is an ordered set of ASs a route in the UPDATE message has traversed.

- NEXT_HOP: This path attribute defines the IP address of the border router that should be used as the next hop to the destinations listed in the UPDATE message.

- LOCAL_PREF: This attribute is a degree of preference given to a route determined by the local policy settings, which is used by routing decision process to compare with other routes to the same destination. The higher the number, the more favorable the route is.

- MULTI_EXIT_DISC: This attribute may be used on external (inter-AS) links to discriminate among multiple exit or entry points to the same neighboring

AS.

In case of information changes, such as a route becoming unreachable or a better path emerging, BGP informs its neighbors by withdrawing invalid routes and injecting new routing information. Withdrawn routes are part of the UPDATE message, with a specific value of the message type field.

If no routing changes occur, the routers exchange only KEEPALIVE packets. KEEPALIVE messages are sent periodically between BGP neighbors to ensure that the connection is kept alive.

A typical path selection process for a BGP router is similar as included in the following process: (this is derived from the strategy used by CISCO routers [24])

1. If the path specifies a next hop that is inaccessible, drop the update message.
2. If the weights are the same, prefer the path with the largest local preference.
3. If no route was originated, prefer the route that has the shortest AS_path.
4. If all paths have the same AS_path length, prefer the path with the lowest origin type (where Interior Gateway Protocol (IGP) is lower than Exterior Gateway Protocol (EGP)).
5. If the origin codes are the same, prefer the path with the lowest MULTI_EXIT_DISC attribute.
6. Prefer the path with the lowest IP address, as specified by the BGP router ID.

Chapter 3

Content Internetworking Architecture – the Big Picture

3.1 System Architecture

As we mentioned before, the IETF draft approach requires the world-wide clients of a site to incur the long round-trip time to go to the authoritative request routing system first, which in turn redirects the request to other *CN*s, an unknown number of times, until the content request is finally resolved. These long round-trip times are purely overhead, potentially far higher than the round-trip to the origin content server itself. This issue is fast becoming the dominant performance problem for clients as Internet data rates move to multiple gigabits, sometimes reducing the transfer time for content to insignificant lows. The name resolution process may also use congested portions of the network that the content delivery system is otherwise designed to avoid, and the same congested portion can be repeatedly used in the process since the peering *CN*s have no information about network conditions. Besides having high latency, the authoritative request RR is also a bottleneck for

scalability. Further selecting a good content surrogate usually requires the proximity measurement of a particular piece of content to existing/potential clients to make routing decisions. Content provider networks must either obtain routing information from routers near their servers, or else make direct network measurements, which imposes huge traffic increases on the whole network. Meanwhile, aggregate network information for scalability is still required, which duplicates the existing routing functions of the network.

Considering that the dominant traffic in the Internet is content access, we need to provide an infrastructure that serves this purpose effectively and efficiently. We keep the following points in mind when creating our design:

- In order to avoid a performance bottleneck of the whole system, we can not adopt a centralized scheme; we need to design it in a fully distributed way for scalability.

- We should make full use of the already available network level proximity information in IP routing because any content access follows the physical path decided by the network layer routing strategy anyway, and we can not impose the duplication of existing network functionalities, wasting Internet bandwidth.

- Our approach should be compatible with the existing Internet infrastructure for easy and step-by-step deployment.

- Our approach should be able not only to accelerate the content access to replicated content by selecting a good content server to serve the client's request, but also to avoid overloading a single selected server; thus, maintaining a global load balance.

Taking into consideration the nature of content routing and a clean layering architecture, and with the awareness of large scale routing using interdomain routing infrastructure, we propose an approach that delegates content internetworking from application layer to network layer. In our approach, the existing interdomain routing infrastructure is extended to accommodate content routing, which is name-based. That is, in our system, IP routing and content routing share routing-related information and function in parallel.

Our proposed content internetworking architecture integrates content routing with traditional IP routing in a hierarchical way as shown in Figure 3-1.

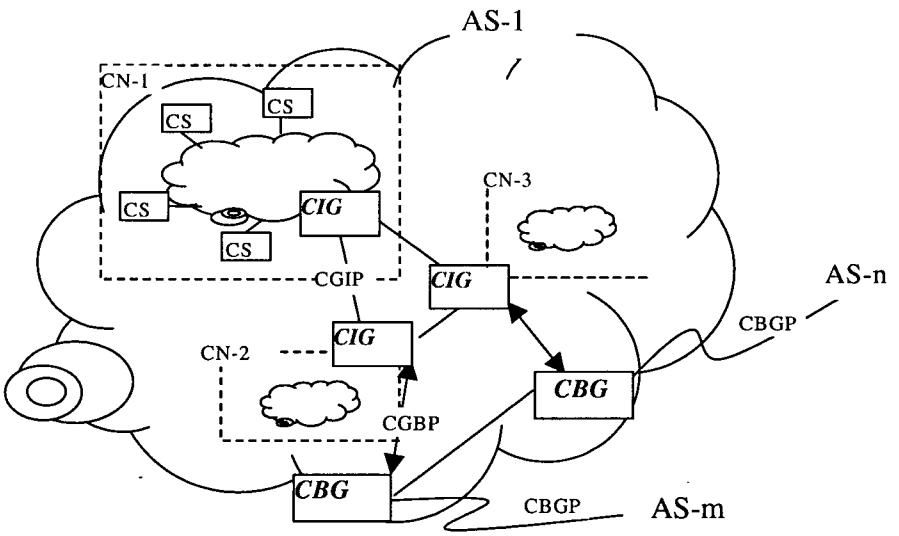


Figure 3.1: System architecture

CS – Content Server or Surrogate

For the interdomain connection, we inherit the mechanism of the current Internet which uses BGP. Our content internetworking infrastructure integrates with

BGP. The Border Gateway in our approach supports both IP and content routing. To distinguish itself from the original Border Gateway, which is only capable of IP routing, ours is called Content Border Gateway (CBG).

The system architecture consists of three levels, named from lowest to highest: the *CN* internal system level, the *CN* level, and the AS level.

As we mentioned before, in a typical *CN*, a single service provider operates the request-routers, the surrogates, and the content distributors. Therefore, these *CN*s usually have a large presence within a particular network, confined within the boundary of an administrative system. In a case where there exists some overlay *CN* encompassing multiple ASs, we can have one *CIG* of that *CN* in each AS to perform the task of content intnetworking. For the rest of this thesis, we assume every CN is contained in a single AS.

In a single AS, there may be multiple *CN*s. Of these, only *Content Interconnection Gateways* (*CIGs*) are visible, and *CN*s communicate via *CIGs* with the external world. Each *CIG* has knowledge of the contents residing on its local *CN*. Each CN has one or more *CIGs*. *CIGs* within one AS communicate using the *Content Gateway Interconnection Protocol* (CGIP) to maintain a consistent view of the content routing situation in a local AS. The *CIG* (the representative of all CIGs in the local AS) also communicate with a local CBG, to propagate knowledge of the replicated content residing on the local AS. *CIGs* learn routing information about content residing on foreign ASs directly from the *CBG* when necessary.

CBGs interconnect ASs through *Content Border Gateway Protocol* (CBGP), replacing the original BGP router in the traditional Internet routing architecture. The *CBGs* seamlessly accommodate IP-address and name-based routing, and is backward compatible with traditional BGP routers. A *CBG* acquires routing infor-

mation for all the replicated content residing on its local AS, through communication with the *CIG*. *CBG* keeps exchanging this knowledge and the corresponding updates with its peers; hence, each *CBG* has complete information about the routes to all replicated content throughout the Internet. To reduce the load of CIGs, the *CBG* does not propagate content routing information it learns from the Internet into the *CIG* under most circumstances. Occasional communication initiated by *CBG* to *CIG* is detailed in section 3.2.4.

To summarize, our proposed content internetworking architecture consists of the following set of protocols:

1. *Content Gateway Interconnection Protocol* (CGIP): dealing with interaction between *CIGs*.
2. *Content Gateway and Border gateway interconnection Protocol* (CGBP): dealing with interaction between *CIG* and CBG.
3. *Content Border Gateway Protocol* (CBGP): dealing with interaction between *CBGs*.

3.2 Subsystem Descriptions

In this section, we depict functions of each sub-system shown in Figure 3-1.

3.2.1 CN Internal Content Request-Routing System

A *CN*'s internal RR system is responsible for delivering client requests to the "nearest" surrogate located in that *CN* [3]. DNS and URL redirection based request routings are the two most commonly accepted *CN* internal content routing techniques. In general, content internetworking views *CNs* as a black boxes; allowing the

CN supplier the maximum flexibility in choosing its own internal content routing scheme.

3.2.2 Content Gateway Interconnection Protocol (CGIP)

Initially, each *CIG* has only the knowledge of routing status for content residing on its own *CN*, learned through routing updates from the *CN*'s internal request routing protocol. If there is more than one *CN* in an AS, a *CIG* can, by establishing peering relationships with neighboring *CIGs* in its local AS, exchange routing updates with others so that all *CIGs* in an AS share a consistent view of content routing states for all content residing on that AS. A *CIG* can make content routing decisions based on its local policies and a set of metrics exchanged with peers.

CIGs also need to communicate with *CBG*s, and since all *CIGs* in the same AS can synchronize with each other, and since all have global content information inside the local AS, it is not necessary for all the *CIGs* to communicate with all *CBGs*. System administrators choose only one *CIG* to represent all *CIGs* in that AS to communicate with a certain *CBG* for content routing exchanges in order to minimize the traffic load. The representative can be manually specified in the configuration. As a fault tolerance consideration, besides the primary representative, a secondary representative can be configured. However, this method is not flexible enough. A better mechanism lets *CIGs* communicate with each other to negotiate how to dynamically elect a *CIG* that best serves as a representative in terms of *CIG* load and network proximity to the CBG.

3.2.3 Content Gateway and Border Gateway Interconnection Protocol (CGBP)

In order to let the external world of an AS know about the content information inside a local AS, the *CIG* in the local AS needs to communicate with the *CBG*. We use CGBP to handle the communications between the *CIG* and the CBG. The *CIG* acts as a special peer of the *CBG* in its AS, and exchanges content routing updates with the *CBG* in an asymmetric way. The *CIG* propagates all routing updates to the *CBG*, however, the *CBG* does not flood the *CIG* with all the content routing information it learns that may never be useful for clients inside a *CN*. The few situations where the *CBG* takes the initiative to propagate routing updates to the *CIG* are discussed in the following paragraphs.

For content routing decisions in a single AS, usually, a good surrogate which exists in the local AS is considered the best candidate to serve a client's requests originating within that AS. However, there may be exceptional situations where a remote surrogate is better than local surrogates, for example, when there is an extremely heavy load on local surrogates. The *CBG* has the whole picture of both global and local replicated content routing information, therefore, it is the *CBG* that decides, for particular content with replicas in both the local and remote ASs, which surrogate is best. If a routing decision is made that the best surrogate has changed from a local surrogate to a remote surrogate, or from a remote to a local, the *CBG* advertises this route to the *CIG* representative, which propagates this update further to other *CIGs* within that AS. If this situation happens too often, then it is the local *CN* provider's responsibility to enhance its surrogates' capabilities.

Another situation that can cause reverse communication from the *CBG* to the *CIG* representative arises when the *CBG* finds there is an explicit content route

withdrawal from its route table. This explicit withdrawal indicates that a content server is not available for service for some reason, and this update must be passed to the *CIG* immediately.

3.2.4 Content Border Gateway Protocol (CBGP)

CBGP interconnects ASs for routing exchanges. CBGP is a superset of BGP. It supports BGP for traditional IP routing, as well as content routing for content internetworking purposes. While the address mechanism for the IP routing part of CBGP remains the same as in the original BGP, the content routing part adopts a dual addressing scheme: *Name+*IP.

Initially, each *CBG* has knowledge of content available only in its own AS, learned through CGBP routing updates from a *CIG* representative (see 3.2.4). By establishing peer relationships with neighboring ASs, it then propagates this knowledge and updates to its CBGP peers. When a *CBG* receives routing updates from a CBGP peer, if necessary, it updates its *Routing Information Base* (RIB) and further propagates routing updates to other CBGP peers. In this way, a *CBG* receives the knowledge of the content routing of the replicated content in other ASs.

In this thesis, unless specified, we use CBGP to refer only to the content routing part of the complete CBGP protocol. The details of CBGP are covered in Chapter 4.

3.2.5 Route Caching at CIG

In our architecture, fundamentally, a *CIG* has routing information only for content available in the local AS. However, the *CIG* can send a query to the *CBG* for routing information of particular content residing on foreign ASs to resolve client requests.

When considering a high similarity in access patterns in a group of users, one of the most commonly accepted mechanisms for improving performance is caching, which can be applied in this content routing scenario. Unlike most known web caching schemes that cache the content itself, our cache scheme caches routes for content. When a *CIG* receives a content response from a *CBG* with an IP address for a name request, the *CIG* caches the routes for future use. Thus, if a client requests content that has a replica only in a remote AS, and if the *CIG* finds a cached route for that content, the name request can be resolved in that *CIG* locally. Unless there is a cache-miss in the *CI*G, the *CBG* does not bother with name resolution, which greatly speeds up the resolution process. It reduces also the overall demand for the involvement of CBGs, which could give rise to a potential performance bottleneck. On the other hand, caching can introduce the possibility of "out-of-date" routes. An appropriately set TTL value may compensate for this shortage by trading off the extra volume of traffic for cache refreshing. From another point of view, the fact that those cached routes are still available for content service (i.e. those servers are not down) makes this problem appear less serious.

3.3 Content Request Resolution Process

Our hierarchical system architecture enables a hierarchical request resolution process, which gives the system good scalability. In addition, the hierarchical name resolution approach completely eliminates the globally centralized authoritative system from the resolution chain, which significantly improves service performance, system availability and reliability. Resolution happens at three levels, from lowest to highest: the *CN* internal RR System level, the *CN* level, and the AS level. Each level tries its best with the knowledge of content status it has to resolve name requests

submitted by clients or from lower levels. Only when a request cannot be resolved at a given level does the request escalate to the next level; in a case where even the highest level fails to resolve a request, it is redirected to the normal DNS chain for resolution.

A client's name request originating within a *CN* follows the resolution path as the steps described below:

1. *CN* Internal RR System Level - *CN* Internal Resolution

   If the client's request can be resolved by a *CN*'s internal RR system, which means the requested content has either a replica or an origin (depending on the *CN*'s internal RR scheme) located within the local *CN*, the resolution process stops. Otherwise, the request is forwarded by the internal RR system to the pre-configured *CIG*, and goes to 2.

2. *CIG* Level

   2.1 *CIG* Resolution

   If the request can be resolved by the *CIG* (usually *CIGs* replaces whatever DNS server originally exists in the *CN*), which means the requested content has either an origin or a replica copy located within its local AS, or if the *CIG* finds cached routes for the requested content, which is located only in foreign ASs, it responds with the address of the "nearest" server, using the load balancing method described in step 2.2.2. That can be either the origin server, or a surrogate server for the requested content; otherwise, it goes to step 2.2.

   2.2 Content Query from *CIG* and Content Response by *CBG*

2.2.1 The *CIG* sends a content query for the requested content to a connected *CBG* and waits for a response from the CBG. Then it goes to step 3.

2.2.2 Upon receiving a content response from a connected CBG, the *CIG* caches the response, and if the response contains one content server's address, the *CIG* responds to the *CN* internal request routing system with the address of that server. When there are multiple servers' addresses in the response, the *CIG* chooses one of the servers based on load balancing algorithms. This algorithm randomly chooses one of the servers, but it uses a weighted random number generator to select the servers with the probability corresponding to the metric of each server. If no route is available from the *CBG*, the *CIG* forwards the client request further to the normal DNS request resolution chain.

3. CBG Level Resolution

   Upon receiving a content query from a *CIG*, the *CBG* checks its *Routing Information Base* (RIB) for the requested content. If routes are found in RIB, meaning the requested content has replicas in foreign ASs, the *CBG* responds to the *CIG* with all the "best" routes stored. Otherwise, the *CBG* responds indicating that the requested content is not replicated in any foreign AS, and goes to 2.2.2

The discrimination in the handling of replicated content and non-replicated content is an important feature of our approach and an important strategy for achieving good scalability. The *CIG* has knowledge of non-replicated content but only propagates replicated content to the CBG. When a client requests content that is not replicated, if the content's origin server resides on the local AS, the request is resolved by the *CIG* directly; if the content's origin server resides on a remote AS,

and there is no cached route for that content, the request is finally redirected to the normal DNS chain for resolution.

Compared to traditional DNS resolution approaches, ours significantly reduces the response time for replicated content, which is the most widely accessed content on the Internet. However, there is a small penalty imposed on requests for content without replica and with the origin server located in remote ASs for which there are no cached routes. This extra overhead, however, is only one round trip from the *CIG* to the *CBG* located in the same AS, which is acceptable. Additionally, the reason a particular piece of content is not replicated across the Internet is very likely to be that it is unpopular, and therefore, seldom requested or accessed. Thus, the small additional overhead does not noticeably degrade overall service quality. Further, by populating the *CBG* with the routing states for only replicated content, we significantly reduce the size of its routing table, compared with the pure name based routing approach [11].

While *Content RR* within a *CN* is handled by the proprietary schemes of different vendors, routing processes for CI is our research focus in this thesis. In our hierarchical architecture, the CBGP deals with the interconnection of content in different ASs, while the CGIP deals with the interconnection of *CIGs* in the same AS. As the CGIP is described in other works [35], our discussion focuses only on CBGP in this thesis. We give its design details and discuss design issues in Chapter 4.

Chapter 4

Design of CBGP

4.1 Overview

The goal of content internetworking can be briefly described thus: by interconnecting the *CN*s around the world, we can find a good surrogate server globally that can serve client requests anywhere on the Internet with good quality. "good" can be measured in different metrics, such as server response time, server-client latency and throughput, server health and load, and so forth. Apparently, network proximity between clients and servers is an important factor, even the *only* factor considered in some solutions for the content route selection process. This factor is naturally a built-in feature of the physical path from server to client. The physical path information is well maintained by routers along the path, therefore, it is natural to have routers involved in the content routing process to provide real time network accessibility information. This network-integrated content internetworking approach saves the application layers from doing the proximity measurements, for example, probing with "ping", thereby significantly reducing network traffic. It can also solve the problem that arises when servers are behind a firewall or other Network Address

Translation (NAT) device, where probing is prohibited. The motivation behind the decision to delegate content internetworking from the pure application layer to the network layer is the desire to fully exploit the existing network layer's functions and resources. From the scalability point of view, the interdomain routing system using BGP is regarded as one of the most successful, and the largest, distributed systems, since it enables the working of the whole Internet. Our content internetworking shares similar feature with this.

Depending on the granularity of the replicated content, we may have first level content names such as "www.ubc.ca," or second level content names, such as "www.ubc.ca/student." While disk storage is not a scarce resource nowadays, most content names should only be located at the first level.

Content routing, which attempts to find a short route to a replicated piece of content, is very similar in nature to a normal IP anycast that tries to find the "shortest" route based on measurements such as network hops. In content routing, all replicas of a piece of content, for example www.ubc.ca, share the same *Anycast Content Name* (A*CN*) www.ubc.ca. Further, this A*CN* appears in the *RIB* representing a virtual content destination node on the network which is shown in Figure 4-1.

We route content in a way similar to IP routing, except that, instead of trying to find the best route to a destination represented by an IP prefix, the A*CN*, representing several content replicas, indicates the content destination address. This is discussed in detail later.

As shown in the general system architecture, we interconnect the content of *CN*s in different ASs through interdomain routing, and in particular, we modify the Border Gateway Protocol (BGP), which forms the CBGP, to enable the content

internetworking function. In implementation, the newest version of the BGP is used: BGP4.

4.2 Features of CBGP

We will now discuss the features of the CBGP design.

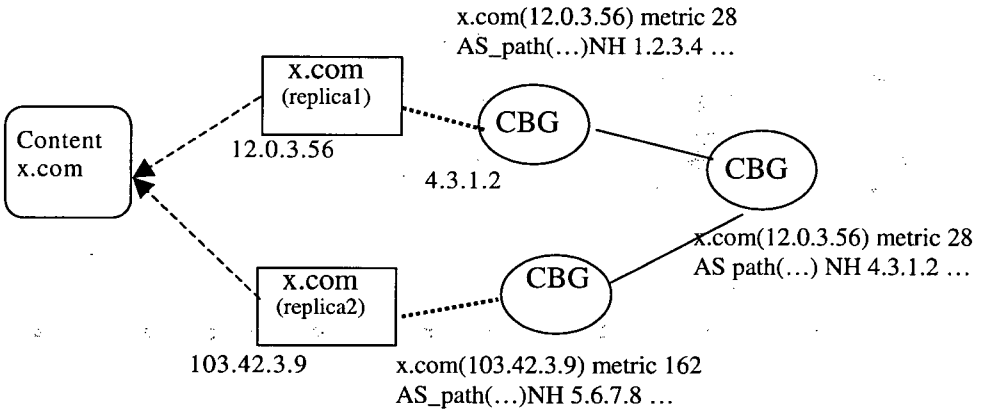
1. Name+IP addressing mechanism

   As content destination is represented by content name, theoretically, we can route content using purely name-based routing globally, just as suggested in [11]. That means the IP address in traditional IP routing is completely replaced by the ACN. However, this requires a change in the transport layer protocol, and correspondingly, all the client side networking software, since all network connection establishment and data transmission still uses the IP addresses instead of names currently on the Internet. Practically, we can not use pure name-based routing, but we rely on the routing process to find a good route to content by directly pointing at a surrogate address for access; hence we do not have a Next_Hop attribute for the content routing table, as the normal network routing table does. Therefore, we call our content RIB "content route table" instead of a "routing table".

   We adopt a combination of the Name and the IP addressing mechanisms to represent replicated content, which means a piece of content is represented by its name, plus the IP address of the server on which the content resides. For example, if content "x.com" has a replica on server 12.0.3.56, that specific piece of replicated content is identified as a tuple $\langle x.com, 12.0.3.56 \rangle$. When content represented as a tuple $\langle ACN, IP \rangle$ has changes in its metric, the updates are

sent in content routing update packets, and the *CBG* makes a decision by comparing the content routes.

Figure 4-1 illustrates an example for the CBGP routing process.



: virtual content destination node

: pointing to a virtual content destination node from a content server

: indirect connection passing through networks

: direct connection

Figure 4.1: CBGP routing process

2. Load Balance Considerations

The content route table of a *CBG* maintains the "best" route selected by the routing decision process to all replicated content by exchanging content routing information with CBGP peers. For load balancing and fault tolerance purposes, multiple routes to each piece of content may be stored in order of goodness from the "best" route, to the secondary route, to the third route, and so on. The number of spare routes the content routing system actually

keeps for a particular piece of content is an implementation trade-off between system flexibility and space overhead.

3. Server Side Metric Participating in the Routing

   As the routing process aims at locating a server that can serve content requests well, we need to accommodate server side metrics in overall routing decisions in addition to network conditions. Thus in our approach, the server side metric directly participates in content routing.

4. Fully Distributed

   Content internetworking through CBGP has the same nature as IP routing on the Internet: it is fully distributed without any centralized point.

5. Content Query Resolution Function

   As well as performing normal content updating processing, the *CBG* also functions as a content query resolution server by answering content queries from *CIG*(s).

4.3 Overview of Content Announcement and Withdrawal

We introduce the basic idea about how to modify BGP to accommodate content routing in this section.

4.3.1 Content Update Message Format

The CBGP shares many protocol features with the BGP, such as an open connection and periodic probing. Thus we integrate the CBGP with BGP for efficiency and easy deployment. The CBGP adopts similar message types as BGP4 – OPEN,

KEEPALIVE, UPDATE, and NOTIFICATION. CBGP's OPEN, KEEPALIVE and NOTIFICATION message formats are same to those of BGP4's. As the UPDATE message intrinsically carries content routing information, it embeds mechanisms specifically for content routing; this is done by extending the path attributes of BGP.

In BGP, an UPDATE message is used to advertise a single feasible route to a peer, or to withdraw multiple unfeasible routes from service. As name based routing differs from IP based routing in route attributes, we add an extra path attribute called "*CONTENT*" to represent content routing information.

Each path attribute in the UPDATE message is a triple ¡attribute type, attribute length, attribute value¿ of variable length. There is an unique attribute type code in attribute type to identify individual attribute. The "*CONTENT*" attribute has the following value:

- *Attribute Length*: specify the total length of this type of attribute

Attribute type: Type Code: 20; the attribute flag is set to define this attribute as optional transitive

- *Attribute value:*

- *Content Update Type*: 1 bit ("0" – content withdraw, "1" – content announcement)

- *Host Sub-addr*: with variable length, specify host part of the IP address of the content server. The actual length of this field can be calculated from the length of *Network Prefix* field in *Network Layer Reachibility Information* (NLRI) which is a key field in each UPDATE message.

- *Content Identifier Length*: specifies the length of the Content Identifier field

- *Content Identifier*: variable length, specifies the name of the content
- *Server-Side Metric*: specifies processing capability of the associated content server
- *Valid Time Period:* how long this route can be thought of as valid
- *Optional Field:* this is a field for future function extension

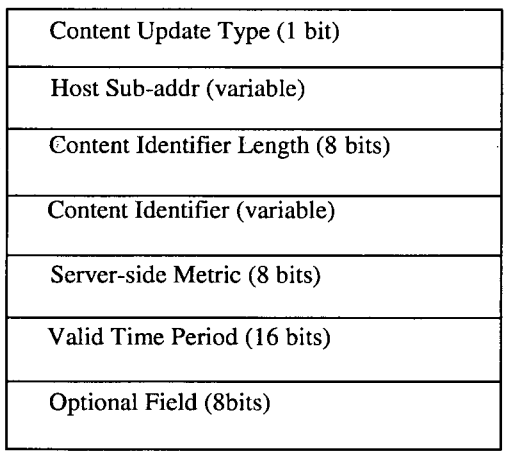


Figure 4.2: CONTENT attribute format

We use the field "Valid Time Period" mainly for considering the "health" of the server. The *CBG* has no way of knowing if a specific server is frequently down when the server itself keeps the relevant statistics. This information is propagated to *CIG* which in turn sends it to *CBG*. Therefore we keep a time period to represent the effective time of metrics and routes. Using 'Valid Time Period' instead of the absolute time of expiration accommodates the time difference of each system if they are not synchronized.

BGP is a path vector protocol as such if there exists a loop in the path, the update packet is discarded. Therefore, we do not need to make an extra effort to

prevent the loop of routing advertisement packets since we inherit the BGP routing mechanism.

In CBGP, one single UPDATE message can announce multiple content routes availability, and withdraw multiple previous content routes availability information, as long as the servers for the content share the same *Network Layer Reachability Information.* The announcement and withdrawal updates are not necessarily put in order in the UPDATE message. For content withdrawal, no *Server Side Metric* and *Valid Time Period* field are presented. If the *Content Identifier Length* is set to 0, all content on that content server are to be withdrawn, and no Content Identifier field is followed.

Here is a sample scenario where a *CBG* sends an update: the *CBG* propagates content announcements for 190.10.20.26 (with *yahoo.com*, server side metric 36), 190.10.20.5 (with *Disney.com*, server side metric 129), 190.10.20.108 (with *CNN.com*, server side metric 825), and makes content withdrawal for 190.10.20.16 (*google.com*), and all contents on 190.10.20.31. The three content advertisements and two withdrawals can be aggregated into one UPDATE message as follows:

Network Layer Reachability Information: 190.10.20.0/24
Attribute Type Code: 20
Attribute Length: 58 bytes
Attribute Value:
* *Content Update Type*: 1
IP Subnet: 26/8
Content Identifier Length: 9
Content Identifier: yahoo.com
Server Metric: 36
* *Content Update Type*: 1
IP Subnet: 5/8
Content Identifier Length: 10

Content Identifier: Disney.com
Server Load: 129
* *Content Update Type*: 1
IP Subnet: 108/8
Content Identifier Length: 7
Content Identifier: *CNN*.com
Server Load: 825
* *Content Update Type*: 0
IP Subnet: 16/8
Content Identifier Length: 10
Content Identifier: google.com
* *Content Update Type*: 0
IP Subnet: 31/8
Content Identifier Length: 0

4.3.2 Processing of the Content Update

When a *CBG* receives a content update message, its actions are as outlined below:

- Upon receiving a content announcement:

  if there is no existing content entry for the announced content in its RIB, it adds that content in the RIB with the associated advertised metric; otherwise,

  if there already exists a route for the announced content, the *CBG* compares (in a routing decision process covered in later sections) the newly updated route with existing routes. If the new route is better than any of the existing routes calculated by the routing decision process, the RIB is updated by replacing the most undesirable route with the announced route; otherwise, it ignores the content advertisement.

- Upon receiving a content withdrawal:

it deletes the related content entries associated with that specific server address from its RIB, and then recalculates the new routes to the content as well as sending the withdrawal to the peering *CIG* if the withdrawal affects the content route table.

- Upon receiving an IP route withdrawal that withdraws an IP prefix's availability:

  it deletes all related content entries associated with that IP prefix from its RIB, and then recalculates the new routes to the content, as well as sending the withdrawal to the *CIG* if the withdrawal affects the content route table.

The entries in the content route table take the format (Content_Name, $\langle IP_1$, Metric_1, $\text{Period}_1 \rangle, \langle IP_2$, Metric_2, $\text{Period}_2 \rangle \ldots \langle IP_n$, Metric_n, $\text{Period}_n \rangle$), where n is the number of best routes the *CBG* maintains and Period represents the valid time period the route has. When a change occurs in its RIB, the *CBG* propagates this update to its CBGP peers.

To achieve scalability and incorporate the diversity of different autonomous systems, BGP can not rely on accurate network proximity information when making routing decisions; instead, it uses predefined policies to calculate route preference. Generally, these policies reflect the wisdom of choosing the most suitable (though not necessarily always "best") route for the current AS. In addition, some recent studies [27,31] show the AS hop count of a path is a decent indicator of the path's proximity, reliability, and stability. To simplify things and avoid heavy modification of BGP, we decided to adopt the path preference calculation strategy of BGP and adopt a reasonable way to make comprehensive content routing decisions. This preliminary solution is described in section 4.4. For a more accurate solution, we

discuss another design using refined metrics in section 4.5.

4.4 A Preliminary Solution for Content Routing Decision

We give a preliminary method to calculate content route preference, and describe the basic routing decision process in this section.

4.4.1 Content Route Preference Calculation

The *CBG* may receive multiple routes for particular content via the CBGP peers' advertisements that originate from the same or from different content servers. These routes contain both network accessibility information and a server-side metric. When considering the network level measurement factor, we adopt a similar method as used in BGP4 for calculating the degree of route preference, in order to achieve compatibility and share computation processes. The difference is that instead of solely comparing routes to the destination of the same IP prefix, multiple IP addresses (to be accurate, multiple NLRIs) of those content servers are used to compare the routes, since the concept of "destination" now refers to a piece of content with many replicas. The BGP local policies also apply to CBGP, for example, whether the current AS is willing to be the transit for the content in certain other ASs.

For content routes originating from different content servers, if a specific route is superior to the others in terms of both network level path preference and the server side metric, routing decision can easily be made. But if there is such apparent advantage to both factors, comprehensive consideration of network level factors and the server side metric is needed.

If two routes have unbalanced advantages on either the network level or the server side, we define a function that takes the attributes of a given route as the arguments and return a value denoting the degree of preference for the content route. For the time being, we consider all the main attributes of a route which include the LOCAL_PREF, the AS_PATH and the CONTENT, and define the overall preference of a route as the following:

$$Pref = scale\ (\gamma \cdot Server\_Metric) + scale(\beta \cdot AS\_path) + \alpha \cdot Loc\_Pref$$

The value of the three route attributes LOCAL_PREF, AS_PATH length and Server_Metric could be measured on different scales, which gives them incomparable values. Therefore, we need to re-scale these values to the same magnitude to keep consistency in the comparison. We scale the AS path length and server side metric to the same order of the largest value of local preferences, since the router knows the value of local preferences for each of its interfaces.

The coefficients for each metric are currently defined as γ=0.5, as we consider the server side metric has almost the same weight as the network level, according to the measurement of [25], which states that the correlation between the ping round trip time and the HTTP request response time is 0.51. β=0.4, as the AS path length is an indicator of network level proximity in the BGP level. Finally α=0.1, as local preference is also a non-negligible factor in making routing decision in BGP. These values are only our initial settings with simple heuristics; a system administrator can adjust these parameters in response to real situations, including considering different set of coefficients for different categories of routes. We also need to perform preprocessing of the value of local preference, as the router usually

prefers the one with the largest value, which is opposite to AS path length and server side metric (the lower the value is, the more preferable the route is).

There are two parts of the AS_PATH attribute, AS_set (unordered set of ASs a route in the UPDATE message has traversed) and AS_sequence (ordered set of ASs a route in the UPDATE message has traversed considering the aggregation functions). We calculate the actual AS path length using the following formula:

$$AS\_Path\_Len = length(AS\_sequence) + log_2(length(AS\_set) - length(AS\_sequence))$$

Our rational behind this is that each time several routes originating from different ASs are aggregated into one route, the AS numbers for those independent routes are eliminated from the AS_SEQUENCE and instead, the local system which does the aggregation prepends its own AS number in the AS_SEQUENCE. All the AS numbers still appear in the AS_SET. The aggregation process is quite similar to (not exactly like, in the real Internet) a tree structure, so the logarithm *approximates* the height of the tree that the route traverses in addition to the current AS_SEQUENCE length.

We can also define a local policy with a re-configurable threshold T on the server side metric for filtering out those routes that have unacceptable server response times.

4.4.2 Content Routing Decision Process

The decision process followed by the CBGP to select a preferred route to a specific content from multiple ones is described below:

1. Filter out the unacceptable routes using threshold T

2. If a route update specifies a next hop that is inaccessible, the route is dropped

3. Calculate the preference of each content route using the above formula, the one with the lowest value wins

4. If a tie results from step 3, the one with the better server-side metric is selected (since the server-side metric is a more accurate measurement than network preference as calculated using the routing strategy of BGP)

In CBGP, route aggregation in the process of route updating is possible through integration with BGP4 and by sharing the IP prefix of content destination. We have already shown an example in the scenario depicted in section 4.3.1.

4.5 Design with a Refined Metric for Network Conditions

As believed by most of the people, using AS path length is a too rough measurement to calculate network proximity, we give another design in this section to make more accurate measurement about network conditions.

4.5.1 Path Latency Measurement

As shown in BGP, peers constantly exchange KEEPALIVE packets to probe each other to maintain knowledge of currently available neighbors. This is a waste of bandwidth and computation resource in most cases when peers are alive. We can utilize these packets for further measurements of the packet latency in the routing path. This is facilitated by the router's I/O architecture.

A typical router's architecture is shown in Figure 4-3.

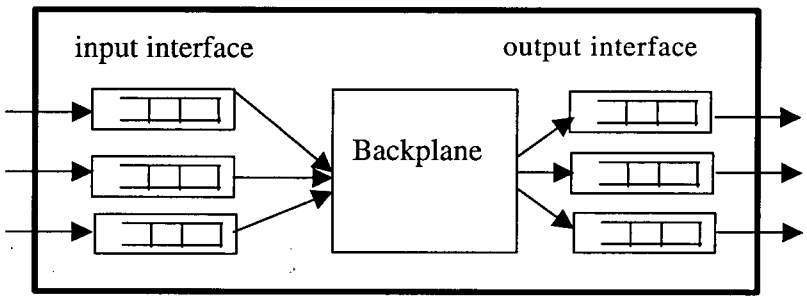


Figure 4.3: A typical router architecture

The KEEPALIVE packet is sent out through one of the output queues, just as any IP data packet being forwarded, it is not put in a queue with a high priority by most vendors (sometimes the update packet caused by routing table changes is sent through a high priority queue [33]). Also from [34], we find that when there is an overwhelming quantity of data packets to process, the KEEPALIVE message transfer can be greatly affected or even lost. So, by measuring this delay, we gain relatively accurate data regarding latency between BGP routers, which includes propagation, queuing, and transmission delay (the latter two also accommodate the factor of bandwidth). This is a fair measurement to all ASs with regard to inter-domain routing, as opposed to different intra-domain routing metrics.

The way to measure this latency is to use a triggered KEEPALIVE sending scheme, in which when one peer receives a KEEPALIVE packet with a sequence number in it, it acknowledges it immediately with a KEEPALIVE message; thus the peer initializing the sending process attains the round trip time for the packet.

4.5.2 Routing Decision Improvement without Changing IP Routing

When the latency between the *CBGs* is known, we can now include another route attribute, which we call "LATENCY", in the UPDATE packet. This attribute is also an optional transitive attribute and takes the attribute type code 21. The attribute value is the accumulated network latency along the way, and it is similar to the sum of the results measured by using the utility "traceroute".

In this approach, the IP routing still follows the original method by which BGP works. For content routing, however, we now have more accurate measurement for network proximity. When a *CBG* has multiple paths to the content, it compares the sum of the accumulated network latency along the way, and the server side metric of each path, then selects the best one. We prefer to use server response time as the server side metric here; if not, we need some way to do the corresponding conversion from other metrics. This is discussed in Chapter 6. If the update causes content route table changes, the new updates caused are sent to its peers and the accumulated network latency in the UPDATE packet increases by the network latency between those two peers.

In Figure 4-4, there are five ASs and we assume there is no Internal BGP (IBGP) communication inside each as we focus on interdomain communication here. The number on the link indicates the network latency between the peers through that link, measured using the method described in the previous section. CS2 and CS1 are two replicas of content A with ACN url-1 residing on AS I and II respectively. In AS V, the local preferences for the routes coming from AS III, IV and VI are 50, 100, 100 respectively. We also assume that the server side metric for CS2 and CS1 are 10 and 6 respectively; CS2 resides on a server with the IP address Addr2, which

is attached to subnet X; CS1 resides on a server with the IP address Addr1, which is attached to subnet Y. Under the conditions stated above, in AS V, we have the following IP routing table entries (not all the attributes of the route are listed) and content route table entry (we omit the Valid Time Period for the route and assume at most two routes are kept for each content):

Destination	AS_path (in AS_SEQUENCE)
X	IV, I (LOC_PREFERENCE & AS_PATH dominate the decision)
Y	VI, II (LOC_PREFERENCE & AS_PATH dominate the decision)

Table 4.1: IP routing table for the sample topology

Name	Routes
url-1	(Addr2, 17), (Addr1, 19)

Table 4.2: Content route table for the sample topology

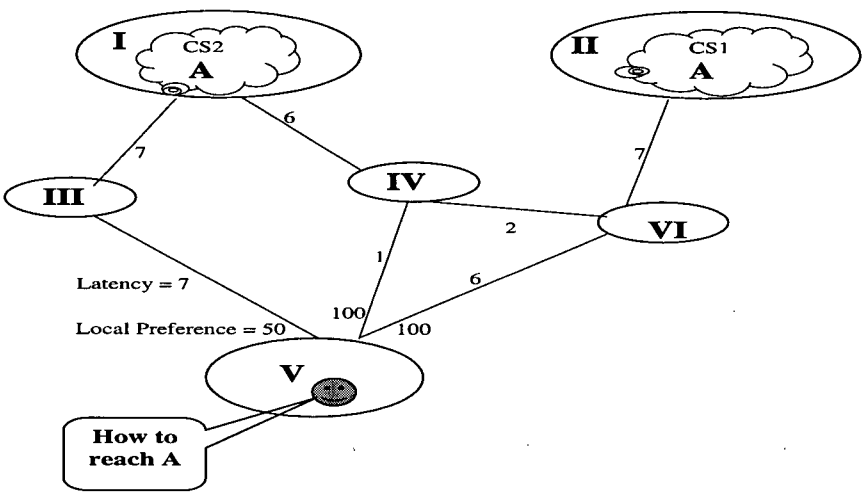


Figure 4.4: A sample network topology

Clearly, the accuracy of this approach matches the application layer ap-

proach. The application layer measurements follow the physical path to those content servers calculated by BGP, and in our approach, the network latency measurement of the paths is used to make close-to-best route selection. This method ameliorates the disadvantage of using hops as a measurement for network proximity. More importantly, it greatly reduces the application layer probing traffic to servers or networks, which could be repeatedly and constantly created by many applications.

Currently, we use the content server side metric as the trigger for content updates. It is possible that during the interval between content updates, some changes occur to the accumulated network latency along the path to the content server, but at present we put this scenario aside because we believe the change of network latency is largely caused by much heavier or lighter access to content and this should be reflected on the server side metric. We consider including both triggers in future work.

Even though BGP routers generally exchange only connectivity information, not performance information, and in the absence of explicit policy, the routers make decisions by minimizing the number of independent autonomous systems traversed along the way to the destination; this metric doesn't correlate with performance characteristics very well, but it doesn't affect our approach above. The rationale behind this is that the decision of the BGP router is fair to all users: as soon as the decision is made, the path is defined and every packet from any application program has to follow it. However, we can make further improvement for content routing by influencing the IP routing process, since we can attain measurements of network latency along the way. We discuss this in the next section.

4.5.3 Routing Decision Improvement by Influencing IP Routing

In each UPDATE packet, there is the "LATENCY" attribute which records the accumulated network latency of the path traversed, and we use this metric to find a better path (from the network routing point of view) to reach that server. For the IP routing, when a *CBG* receives an update, firstly, it still applies the policy routing, such as comparing the weight of routes specifically defined by some vendors, local preference of the path and so on, since we need to respect the subjective choice of the network operator. If all the previous comparisons result in a tie, it compares the length of the AS path. At this stage, instead of using the AS path length, it compares the accumulated network delay of multiple paths and selects the best one. For content routing, the sum of the network delay and server response time is used as the comprehensive metric to compare the routes to a content destination. If a better route results from either the IP or content routing, updates are sent to neighbors.

Again, referring to Figure 4-4, in AS V, we have the following IP routing table entries and content route table entry:

Destination	AS_path (in AS_SEQUENCE)
X	IV, I
Y	IV, VI, II

Table 4.3: IP routing table with influence of LATENCY

Name	Routes
url-1	(Addr1, 16), (Addr2, 17)

Table 4.4: Content route table with influence of LATENCY

The reason that the AS path to the network Y is (IV, VI, II) instead of (VI, II)

is because the accumulated network latency dominates, since the local preference to AS number IV and VI have the same value. For the content route table, considering both the network proximity and server side metrics, we get the result shown above.

Although the above method can produce a better route for accessing the content, we don't advocate using it, since the frequent changing of the IP routing behavior could cause route instability on the Internet.

4.6 More Design Issues

We discuss more design issues in this section including: how to control content update frequency in the Internet, how to make use of the periodic network latency measurements, how to deal with network proximity measurement inside an AS, and what is the behaviour of the convergence process for content routing.

4.6.1 Controlling the Content Update Frequency

The Internet is a huge distributed system. Considering its scale, making frequent route changes is not accepted for stability and scalability. Our intention is to interconnect content networks around the world and find a good replica to serve client requests, so it is not necessary to respond to all metric changes in real time, as many changes are transient. Thus we need to control the content update frequency.

As we know from the description in the previous section, in addition to selecting content routes based on a normal network level accessibility factor, extra effort must be made to take the server-side quality of service into consideration. This can be measured in terms of server response time to client requests, server CPU load, server health, maximum connections to the server, and so forth. To measure a server's processing capability at a certain time, we prefer to use server response time,

as this is the metric that reflects the service quality for a request, since our purpose for content networking is to improve content response time. As we can imagine, the server side metric is usually quite a transient parameter, and sometimes can change dramatically in a very short time. To smooth out the oscillation effects, this metric is usually recorded periodically, and reflects the average value in a moving window of time. (We refer to the size of this moving window as the *measurement interval*). Compared with the value in the previous measurement interval, when the value of the next interval has a significant change, then an update should be sent. The *Measurement interval* can be adjusted by each server.

Since the server-side metric is learned via the CGBP through the *CIG*, which in turn acquires this metric from the *CN* internal RR system, we have an alternative way to control the content update frequency. To keep the high accuracy and real time response inside a *CN*, the server response time can be updated at a relatively high frequency, but the *CIG* can do further processing to suppress the frequent updates to *CBG*, using monitoring and weighted calculations of historical changes. (Note: In our implementation, we use the technique described in the previous paragraph)

4.6.2 Usage of Network Latency Measurement

We calculate the latency between *CBGs* periodically, but we do not use all latency values at each moment, because the updates may not be sent as frequently as the latency measurement. Therefore, we need to discover a way to calculate the actual latency to use in the update packet. To calculate this value, we borrow the idea from TCP about how to compute the retransmission timer, since our purpose here has similarities with it. We have two variables, LT and M, representing the latency we want to calculate and the latest measurement of the latency, respectively. To

accommodate the effect of the variance between the new measurement and the historical value, we use a variable, D, to calculate the deviation by the following formula:

$$D = \delta D + (1\text{-}\delta) \quad | LT - M |$$

Where δ has the value of 7/8. The calculation of LT is:

$$LT = M + 4*D$$

Each time after an UPDATE packet is sent, LT and D are reset to start again.

4.6.3 Network Proximity Inside an AS

There is one important factor that affects content routing decisions: the delay in the original AS where the content update initiates. Due to the different size of the ASs, this factor can have a different weight in the influence to the whole route decision process.

To account for this factor, we take a rough measurement of latency inside an AS. As described in the previous sections, by measuring the latency between border routers through KEEPALIVE packets, each border router of the Internet Service Provider (ISP) stores the latency measured from this router to the other border routers in its network (the IBGP routers). By averaging the latencies stored, we arrive at the approximate latency in this AS. The content update packet then includes this number as the initial value for attribute LATENCY. This value can also be used as a reference to compare whether a content source inside an AS is better or worse than one outside an AS. This approach is scalable because the measurements

are performed locally (to an ISP's network) and the information stored at the border routers is only on the order of the number of border routers in the ISP's network. This idea is used in [21] as well.

4.6.4 Convergence Process Discussion

Since our name request resolution hinges on the *Name+*IP addressing mechanism that hardwires the IP address to the specified content in the routing process, concerns arise regarding the route convergence process. We claim that content routing in our approach does not have more serious problems in routing convergence and oscillation than that in the original BGP4 [36].

If a content server is down, content requests directed to that server before the BGP4 route converges drop, but this is no more serious than the convergence behavior of the original BGP4. The content withdrawal is quickly propagated through the content routing process, and *CBGs* remove that route from the content route table, if it is there, and the peering *CIGs* are informed. In addition, multiple paths kept by each *CBG* ameliorate this problem.

If the route to certain content is in the convergence process because of the existence of a better route when the *CBG* is serving a name resolution query for that content, the performance will not be affected much. The old destination is still reachable and can serve the content request, but with only a slight performance loss, as compared to the new one.

4.7 Deployment Considerations

Our content internetworking system allows an all-at-once deployment or incremental evolution, based on services needs.

To incorporate the individual *CN* into the content internetworking picture, the only requirement is to place *CIGs* at the edge of *CN*, which is a basic requirement to enable interconnection with other *CN*s. Running CGIP on *CIG* will enable one *CN* to locate other *CIG* peers in the same AS, hence interconnecting with them.

Enabling the *CN* interconnection on a global scale requires only the upgrade of BGP4 to support CBGP on existing Border Gateways. By using an UPDATE packet compatible with the original BGP protocol and defining the new attributes as transitive optional, we can make a step-by-step deployment for the new routing architecture. The justification is that paths with unrecognized transitive optional attributes should be accepted as defined in the BGP standard. If a path with an unrecognized transitive optional attribute is accepted and passed along to other BGP peers, then the unrecognized transitive optional attribute of that path must be passed along with the path to other BGP peers. Then, we can guarantee that attributes such as CONTENT can be transmitted across non-CBGP-enabled regions. An original Border Gateway that only has IP routing function ignores content attributes in routing advertisements and passes them along to BGP peers, so it can still work compatibly with the upgraded Content Border Gateway. An incremental deployment path greatly facilitates the extension of global content internetworking.

The deployment process can be based on user needs and an ISP's motivation to provide a better web experience to customers and superior service to co-located content providers. For replicated content, name resolution can be quickly solved with the result of returning a good server, eliminating the need for name requests to leave their network. For content without replicas, the name resolution fails over to normal DNS behavior. This initial deployment requires no change to end hosts.

With more and more content being replicated, the content route table may

become bigger in *CBG*s, which may run out of resources to store route table and answer queries. In that case, we can deploy an active server co-located with the *CBG*, to store content routes and processes content requests. ISPs that already peer at the IP routing level are motivated to peer at the content routing level to provide their customers faster access to nearby content servers and increase the benefit of placing content servers in their networks. As demand grows, the routers will be upgraded gradually to adapt to content internetworking requirements.

Chapter 5

Implementation and Experiment

5.1 Implementation Based on MRT

We introduce our prototype implementation in this section. The focus is on the implementation about CBGP. Those of other protocols can be found in [35].

5.1.1 Implementation Overview

To test the complete system framework, we implement all the components in Figure 3-1, including simulating the *CN* itself, as we don't have an existing *CN* system to experiment with. Thus, the implementation includes a server side metric updating protocol between content servers and *CIG*, CGIP, CGBP, and CBGP.

We have a monitoring agent installed on each server that records the server performance status each minute, and sends updates to the *CIG* through port 10789 whenever a significant change happens.

CIGs inside an AS communicate with each other, exchanging the content

status inside each *CN*. Each *CIG* has a name resolution thread running on port 53, and the CIG replaces the normal DNS server running on the machine. One of the *CIGs* is selected as the representative to communicate with the *CBG* through CGBP. That *CIG* is configured as an internal peer of the CBG. The focus in this chapter is on CBGP.

CBGP is implemented by extending the *Multi-threaded Routing Toolkit* (MRT) [14, 15, 16, 17] under Linux to support content internetworking.

5.1.2 Introduction of MRT

MRT is a routing toolkit developed by the University of Michigan/Merit Network. Besides tools such as the traffic generator and message format converter, the key part of this toolkit is a routing daemon called MRTd. It supports RIPng, BGP4+, multiple RIBs (route server), and RIP1/2. MRTd reads Cisco Systems-like router configuration files and supports a Cisco Systems router-like telnet interface.

The routing architecture design of MRT incorporates features such as parallel lightweight processes, multiple processor support, and shared memory. Although MRT has been designed with multi-threaded, multi-processor architectures in mind, the software will run in emulation mode on non-thread capable operating systems. The modular design of the software encourages the rapid addition and prototyping of experimental routing protocols and inter-domain policy algorithms.

MRT is written in C programming language, and contains about 120,000 lines of code, amongst which there are more than 30,000 lines specifically dealing with BGP (this excludes the code for data structures and operations shared by the whole MRT). Currently, we have only completed the implementation of the preliminary solution described in the previous chapter based on MRT due to limited time, and

we are still working on the refined solution.

Our focus is the extension of the routing daemon MRTd. In particular, implementation of BGP4 in the MRTd is modified and extended to support content internetworking. Content query/response handling functions on the *CBG* is implemented in a separate MRT thread.

5.1.3 Content Internetworking Implementation

- Data structures

In addition to the regular IP routing table which is represented by a radix tree in MRT, we need another data structure for the RIB of content routes. Currently, we use a hash table for experimention. A patricia tree is also a good data structure for this purpose since we assume that only a low percentage of websites on the Internet are replicated, but we have not implemented it. The hash key for the content route table is the name of the content, and each entry in the content route table is a data structure defined as the following:

```
typedef struct _croute_item_t{
char *cname; //content name
one_entry *entry; //the list of routes for this content
} croute_item_t;

typedef struct _one_entry{
char *prefix; //the server which holds the content
bgp_route_head_t *bhead; //the pointer to the bgp RIB node
unsigned int metricN; // metric of the network level
```

```
unsigned int metricS; // server side metric of the content
long expire_time; // the expire time for this entry
struct _one_entry *next; //next pointer in this link list
} one_entry;
```

Each route to the content is inserted into a linked list in order of the calculated preference value.

The *CBG* maintains one data structure for each of its peers, which contains all the information and data storing substructure needed when communicating with that peer. The main fields of the data structure are listed below:

```
typedef struct _cbgp_peer_t {
char *name; /* peer name */
prefix_t *peer_addr; /* peer address */
int peer_as; /* peer's AS number */
u_long peer_id; /* peer's router id */
int peer_port; /* port number in case it is not 179 */
nexthop_t *nexthop; /* immediate next hop */
int sockfd; /* socket connected to peer */
pthread_mutex_t mutex_lock;
LINKED_LIST *ll_announce; /* store announcement from the peer */
LINKED_LIST *ll_withdraw; /* store withdrawal from the peer */
cbgp_attr_t *attr; /* attributes of the path */
LINKED_LIST *ll_update_out; /* updates to be sent to peer */
radix_tree_t *routes_in[AFI_MAX][SAFI_MAX]; /*incoming routes*/
```

```
HASH_TABLE *content; /* content routes rib_in*/
} cbgp_peer_t;
```

- Content route updating process

Since the packet format is changed with the addition of new attributes, the first thing we need to do is to modify the encoding and decoding process for the UPDATE packet. When a new peer relationship is established, all the entries in the content route table are sent to the peer.

By processing all the route updates from individual peers, the *CBG* keeps on updating its view to the outside world. Each time a *CBG* finds a change in its view of the IP or content routes, the changes are sent to all the other peers, except the one that announces the new route.

The processing of the updates can be divided into the following three phases: (only the processing of a route announcement is described below, since content updates are not involved with the processing of the normal IP route withdrawal. We assume there is only one prefix in the NLRI field, since processing is the same for each).

Phase 1: In this phase, the announced route gets checked and put into the RIB-in for that peer. RIB-in stores route information that has been learned from inbound UPDATE messages. The entries in it represent routes that are available as an input to the decision process. A processing flow diagram is shown in Figure 5-1.

The Check Attributes step in the diagram aims at verifying the path attributes to guarantee it has the all the mandatory attributes, such as NEXT_HOP, AS_PATH.

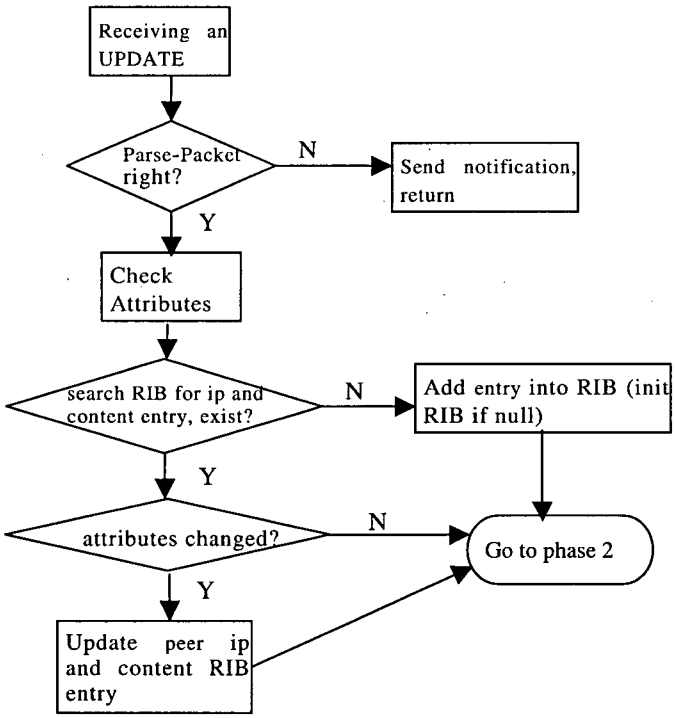


Figure 5.1: Phase 1 of content route updating

Phase 2: The view of the overall content routing map in the *CBG is* updated according to the route change resulting from phase 1. The best routes are chosen out of all those available for the content destination, and installed into the local RIB. The changes to the RIB are stored in the change list. The main processing flow is shown in Figure 5-2.

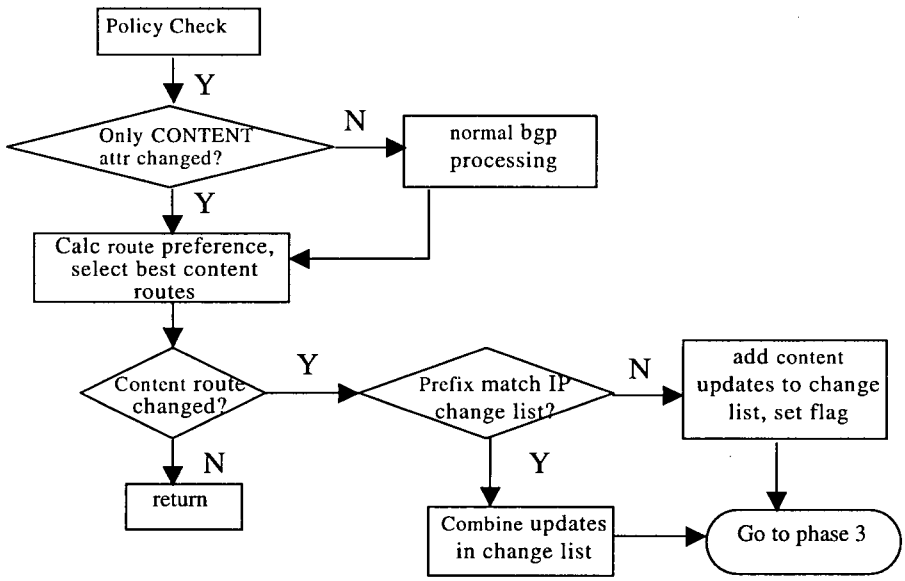


Figure 5.2: Phase 2 of content route updating

The Policy Check step filters the routes and path attributes manipulating to influence its own decision process as the local policy defines. It could filter certain networks coming from a peer while accepting others. In case of aggregation, the content server address only matches a less specific route (with a shorter prefix), it is necessary to recalculate the sub-host address in the CONTENT attribute to match that prefix.

In this phase, if a new route is added to the content route table, the expiration time is calculated by adding the Valid Time Period to the current system time.

Phase 3: For the changes in the list resulting from phase 2, each peer, except the one announcing the new route, gets scanned to check whether it has a contradiction with the predefined policy. The changes are disseminated to those peers if the policy allows it. Meanwhile, the route withdrawal and the route changes to the content existing in the local AS are also propagated to the *CIG* representative.

- Content resolution thread

CIGs send content requests to *CBG* whenever they can not find an answer in their own content route table, or the entry in their cache expires, either because the caching TTL expires, or because the Valid Time for that entry expires. There is an individual thread in *CBG* which deals with such requests.

The content request/response processing between *CIG* and *CBG* is internal in the whole system architecture, and does not relate to the end user's request directly, so the communication port does not have to be 53. We chose 553 as the listening port.

Each time a content request is received, the content route table is searched, and if the item is found, the routes stored in the linked list are retrieved one by one and then encapsulated into one singe data structure to pass back to the *CIG*. In the process of route retrieval, the expiry time for each route is checked and if it expires, the route is discarded. Otherwise, the new Valid Time Period is calculated by computing the difference of the current system time and the expected expire time.

- Tips in system implementation

MRTd is a routing daemon running on Linux. When it runs, no terminal is associated with the threads, so it is difficult to debug. To make it convenient

for debugging and testing, the daemon process is removed from the original MRTd program. Further, since the multi-threading system is very hard to debug due to constant thread switching by the process scheduler, in the system debugging stage, we set MRTd to run in emulation mode in a single process, just as if running on non-thread capable operating systems.

The other interesting point in the implementation is that the client side applications such as ping or Netscape on the Windows platform exhibit different behavior in accepting the name resolution answer from those on Linux. Clients on Windows accept answers sent from any port on the server system, while clients on Linux only accept answers sent through the standard service port with the number 53. This problem proves difficult, as we wrote the name resolution thread in Windows and tested it there, however a direct port did not work on the Linux platform.

5.2 Topology and Configuration of the Experiment

The network topology description, environment settings and tools used in the experiment are covered in this section.

5.2.1 General Description

We conducted experiments to test the viability of the system architecture and the performance of the system.

In simulating an Internet web environment, we set up *Apache* web servers to hold web pages, files etc.. As usual, the *CN* itself is considered a black box for content internetworking, so our simulation ignores the *CN* internal Request-Routing system. We use agents residing on the web server to collect real-time content availabilities and server side metric information, and these variations are

considered to be the origins of routing updates to be sent to the *CIG*.

We simulated a single AS using a Linux box running CBGP. Other machines Played the role of web servers and *CIG* by running the corresponding software. With limited number of machines, however, we could have only one machine acting as a multi-function box with any combination of CBG, *CIG* and web server. The *CIG* communicating with a certain *CBG is* configured as the same AS number as the CBG.

GNU web stress tool – *OpenLoad* [18], which provides near real-time performance measurements for web applications, is used to generate traffic for system performance evaluation and analysis. For example, "openload www.myContent1.com 50" simulates 50 clients requesting and accessing content www.myContent1.com simultaneously. The name requests are sent to our Request-Routing system for resolution; after getting the IP address of a content server, *Openload* visits the content servers. Accordingly, content servers' load and network usage are affected. The metric variations are reflected in the routing system. Thus, we can observe content routing behaviors and the load balancing performance of the content routing system. Openload is further modified to reflect the content route changes in the process of simulation.

5.2.2 Experiment Network Topology

We used a Bay450 T-24 Switch with VLAN capability to set up a simulation environment. The simulation environment contains 6 ASs with *CN*s scattered throughout, and 9 A*CN*s representing different content. We have clients sitting in different ASs access the content provided, and the overall performance data is recorded for comparison. The simulation topology is shown in Figure 5-3. Although the simulation

scale is small, the characteristic features of the Internet structure are simulated, and we plan to make a large scale simulation in future work. The local preference of each route coming into an AS is set to the same default value, and in this special environment, we set a very small weight (0.1) for both the AS path length and local preference, while we give a heavy weight to the server side metric.

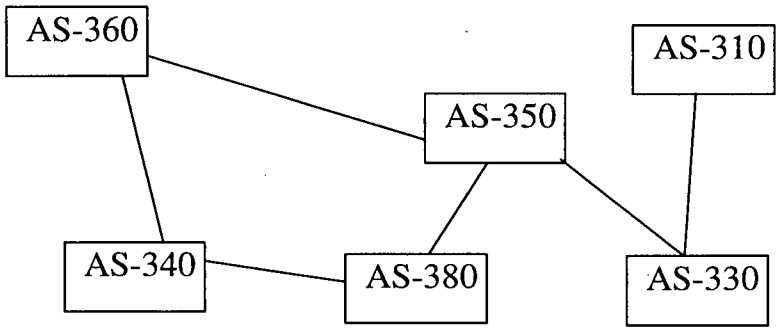


Figure 5.3: Network topology for the experiment

The above topology was configured with the computers in the lab. Computer names, the AS numbers they belong to, their IP addresses, the role they played in the network and the content they hold (if they take content server roles), are listed in Table 5-1. Some Linux boxes have several network cards installed to act as routers, and hence have several network addresses. The whole experiment is done on an internal network using the internal IP subnet 192.168.x.x. We also set up web servers delegating pseudo domain names, such as www.cisco.com, www.yahoo.com as shown in Table 5-1.

5.3 Evaluation Criteria

In the experiment, we mainly conduct a stress test for performance evaluation. That is, we simulate multiple clients' concurrent and intensive access to content.

Computer name	AS No. belonged to	IP address(es)	Role of the machine	Content held	*CN* No. in an AS
goodearth	310	192.168.1.7; 192.168.5.7	CBG, *CI*G, content server	Cisco, dior, lancom, yahoo	*CN*-1
venus	350	192.168.5.2; 192.168.3.2; 192.168.6.2; 192.168.8.2	CBG, *CI*G, content server	Cisco, dior, lancom, yahoo	*CN*-1
grimface	330	192.168.1.12; 192.168.3.12	CBG, content server	Netscape, elle	*CN*-1
slesse	330	192.168.3.15	*CI*G, content server	Netscape, elle	*CN*-2
hozameen	380	192.168.8.16	*CI*G, content server	Netscape, elle	*CN*-1
celestial	380	192.168.8.25; 192.168.7.25	CBG, *CI*G, content server	Google, cnn, ubc.ca	*CN*-2
robson	360	192.168.6.10; 192.168.4.10	CBG, *CI*G, content server	Google, cnn, ubc.ca	*CN*-1
moonlight	340	192.168.4.8; 192.168.7.8	CBG, *CI*G, content server	Cisco, dior, lancom, yahoo	*CN*-1

Table 5.1: The configuration of the testing environment

Client request response time, transactions completed per second, and total number of completed requests are measured by Openload and recorded for comparison and analysis. The load variations on the server side are also monitored.

We evaluate the system on the following aspects: first, we expect that if there ia a large quantity of client requests, content that has more replicas will deliver better overall performance than that which has fewer or no replicas, despite system management overhead. Second, with our load balancing strategy, content servers should demonstrate good overall load distribution performance. Third, compared to traditional DNS resolution schemes, our approach incurs one extra level of round trip time from the *CIG* to the *CBG* for non-replicated content request resolution. However, we think this overhead should be acceptable in contrast to the significant performance gains for resolving widely replicated content, which is most frequently requested and accessed. Fourth, our internetworking architecture should be compatible with the original border gateways, shown by good interoperability. The experiment results are shown in the next section.

5.4 Data Collection and Analysis

We did testing and data analysis on the following aspects:

1) Overhead on the content name resolution:

In this test, we measured the overhead of the content name resolution process (total for both request and response). The experiment is carried out on a content route table of 50,000 entries. These entries are randomly generated domain names with most of them in the .com domain and others in .org and .net.

The overhead we measured on a 667 MHz Pentium III system running Linux 2.4.13 is just 5.7 milliseconds for going through two hops of the content layer that

are *CIG* and *CBG*. Also, we believe most of this time is spent on packet processing, as any DNS server has to do.

The DNS delay measured by [32] is shown in Table 5-2. It assesses the delay by choosing top 100 URLs with largest number of Internet users and 100 random URLs. The 95th percentile delay listed in the table is the delay sufficient to serve 95% of the queries.

	Top 100 URL			100 Random URL		
	Median	95th percentile	Average	Median	95th percentile	Average
DNS delay (ms)	160	2159	498	298	6149	1706

Table 5.2: DNS QoS assessment in the internet

Compared with the data shown in Table 5-2, we conclude that the overhead in our approach is negligible.

2) Other performance testing for content accessing:

In the set of tests conducted, we compare situations where server loads change with the timeline, the response times when different numbers of surrogate servers exist, and overall request response time measured from different clients.

2.1) Server load comparison between two machines with similar configuration:

By having clients in different ASs access www.cnn.com continuously for 3 minutes, we observed the changing trend of the server load on *robson* and *celestial* as shown in Figure 5-4 in which server load is calculated in self-defined units. *Robson* and *celestial* are two machines with similar system configurations, a 200MHz CPU, and 64M and 96M memory respectively. Server load is computed synthetically from

the CPU load as well as memory and swap space usage. From the chart shown in the figure, we can see that these two servers have basically the same load during the service period, which showed good load distribution.

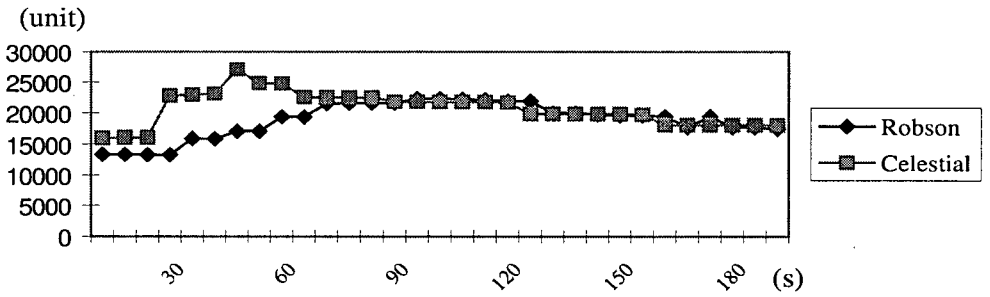


Figure 5.4: Load comparison on two similar configured machines

2.2) Faster average response in case of more surrogates:

We measured the request response times in cases when different numbers of surrogates exist in the system. For the content www.google.com, we first start the web server on *robson* only, and then simulate 20 clients access concurrently from both *venus* and *goodearth* using Openload. The average response time measured on *venus* is shown in Figure 5-5 with the line indicating "one surrogate." We then start the web service on *celestial*, and again measure the response time on *venus*. The data is shown in the chart with a line indicating "two surrogates". From the chart, we can see that apparently, the average request response time is faster in case where more surrogates exist.

2.3) Similar client response times with the same amount of surrogates:

The chart in Figure 5-6 shows the response times measured on *goodearth* and *venus* during the same period when they access the content of www.google.com. From the chart, we can see that the average response time from both clients is quite similar, which demonstrates the benefits of content internetworking.

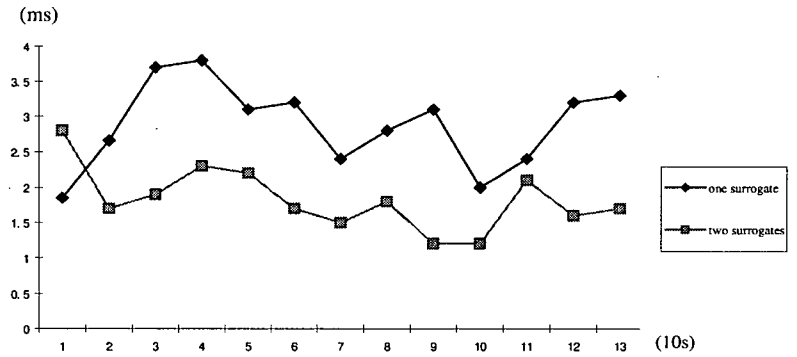


Figure 5.5: Response times for a single client

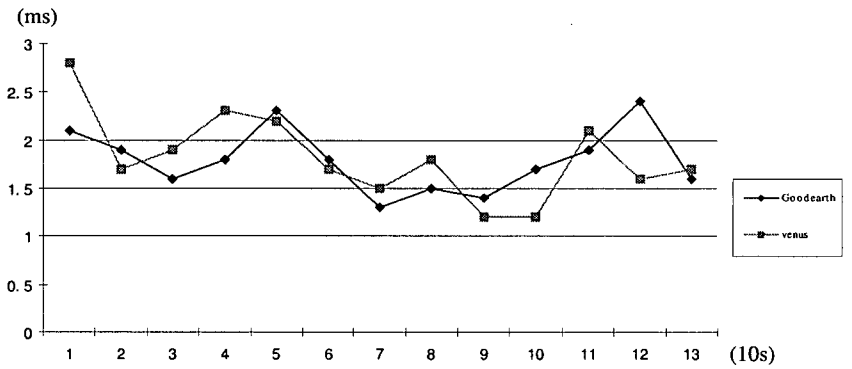


Figure 5.6: Response times for two concurrent clients

3) Interoperability between BGP and CBGP:

To show the feasibility of step-by-step deployment, we also perform interoperability testing of BGP and CBGP using the following simple topology, as shown in Figure 5-7:

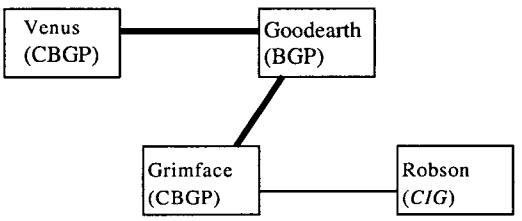


Figure 5.7: Interoperability testing topology

In the testing topology, the Linux box *Robson* plays the role of the *CIG* and the content server with the agent running on it.

After *venus, goodearth* and *grimface* have established connections, we start the *CIG* and the agent on the content server. We can see that the route table on *grimface* has the entries shown in Table 5-3 :

content	Route to server	metric	Valid Time
www.google.com	192.168.6.10	100	36000
www.cnn.com	192.168.6.10	100	36000
www.ubc.ca	192.168.6.10	100	36000

Table 5.3: Content route table in interoperability test

Immediately, the same content route table shown in Table 5-3 appears on *venus* too by traversing the non-CBGP area. This proves that CBGP is interoperable with BGP, and that routers upgraded to CBGP can provide customers with better content service.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

Although content networks are being deployed fairly actively on the Internet these days, content internetworking is still a new and important area. The proprietary nature of most content routing designs makes them undesirable for global use, and there is still no scalable and efficient way to interconnect content in different networks to handle increasing global demands for content access. To address the problem, we propose a hierarchical scalable architecture.

Our approach includes the following features: hierarchical internetworking architecture, which lends itself well to good scalability; hybrid routing platform integrating both IP and name based routing, which is compatible with existing internet architecture and make good use of existing network function; fully distributed routing mechanism, which eliminates the centralized node of the bottleneck; direct participation in the routing process of the server side metric, providing more complete metrics for routing decision; load balance considerations in content routing, which has apparent advantages for load distribution.

Our scheme is the first that we are aware of to use integrated IP and name-based routing mechanisms. When compared to most related solutions, our approach pushes content naming information out into the network, which greatly reduces the traffic volume made to the network infrastructure by application layer metric measurements.

Preliminary experiments show that our architecture is fully viable, and the overhead of adding an extra level of indirection for non-replicated content request is insignificant compared with the benefit gained.

Our approach can be easily deployed to provide immediate benefits to ISPs and their customers. It also helps to protect investment in infrastructure by upgrading software only on the border gateway, which should scale at least to the demands of content requests for popular content.

6.2 Future Work

We are considering the following work in the future to further improve and verify the viability of our approach.

1) Large scale simulation:

While we have conducted some experiments in a size-limited testing environment, we have not done any simulation on a large network yet. We plan to use OPNET or NS2 for this simulation, especially to simulate the refined solution proposed in the design.

2) Further consideration about scalability:

In the design of the framework, we can see that each *CBG* has to keep the whole content route table for all the replicated content around the world. This is not very efficient when the amount grows very large. We can observe that not all

content is popular all over the world. There is content that is only popular in a certain area or region, although there may be many replicas around that area, it is not necessary to propagate the content updates to all the *CBGs* in the Internet. The popularity of content can be measured through the usage at each network. Proper methods need to be found to deal with this problem.

3) Agent based content routing metric calculation:

There may exist different ways of calculating the server side metric in each *CN*. Even for defining server "load," different brands of hardware and different types of operating systems react differently under the same set of load criteria. Newer metrics have been developed recently that perform tests for response time by tracking how quickly packets are responded to, which is the preferable metric in our proposed solution. Not all the *CN*s have this metric available however, so we want to use mobile agents to coordinate and do conversion between different metric measurements, and come up with a mathematical correlation that's the same for all servers, so as to make the use of those metrics in a consistent and meaningful way.

Bibliography

[1] M. Day, B. Cain, G. Tomlinson, and P. Rzewski. A Model for Content internetworking(CDI). http://www.ietf.org/Internet-drafts/*draft-ietf-cdi-model-00.txt,* February 22, 2002

[2] M. Day, B. Cain, G. Tomlinson, and P. Rzewski. Content internetworking(CDI) Scenarios. http://www.ietf.org/Internet-drafts/*draft-ietf-cdi-scenarios-00.txt,* February 25, 2002

[3] Barbir, B. Cain, F. Douglis, M. Green, M. Hofmann, R. Nair, D. Potter, and O. Spatscheck. Known *CN* Request-Routing Mechanisms. http://www.ietf.org/Internet-drafts/*draft-ietf-cdi-known-request-routing-00.txt,* February 22, 2002

[4] M. Green, B. Cain, G. Tomlinson, S. Thomas, and P. Rzewski. Content internetworking Architectural Overview. http://www.ietf.org/Internet-drafts/*draft-ietf-cdi-architecture-00.txt,* Jun, 2002

[5] B. Cain, O. Spatscheck, M. May, and A. Barbir. Request-Routing Requirements for Content internetworking. http://www.ietf.org/Internet-drafts/*draft-ietf-cdi-request-routing-reqs-00.txt,* February 22, 2002.

[6] L. Amini, S. Thomas, and O. Spatscheck. Distribution Requirements for Content internetworking. http://www.ietf.org/Internet-drafts/*draft-ietf-cdi-distribution-reqs-00.txt,* February 22, 2002.

[7] D. Gilletti, and R.Nair. Content internetworking(CDI) Authentication, Authorization, and Accounting Requirements. http://www.ietf.org/Internet-drafts/*draft-ietf-cdi-aaa-reqs-00.txt,* February 22, 2002.

[8] Y. Rekhter, and T. Li. A Border Gateway Protocol 4 (BGP-4). *RFC 1771,* http://www.ietf.org/rfc/rfc1771.txt March 1995.

[9] P. Mockapetris. Domain Names - Concepts and Facilities. *RFC 1034,* http://www.ietf.org/rfc/rfc1034.txt November 1987.

[10] P. Mockapetris. Domain Names - Implementation and Specification. *RFC 1035,* http://www.ietf.org/rfc/rfc1035.txt November 1987.

[11] M. Gritter, and D.R. Cherition. A New Next-Generation Internet Architecture. http://gregorio.standford.edu/triad/index.html

[12] M. Gritter, and D.R. Cherition. An Architecture for Content Routing Support in the Internet. http://gregorio.standford.edu/triad/index.html

[13] C. Yang. Efficient support for content-based routing in Web server clusters. Proceeding of USENIX Symposium on Internet Technologies and Systems (USITS)'99.

[14] University of Michigan and Merit Network. Multithreaded Routing Toolkit (MRT) project. http://www.merit.net

[15] MRT Installation Guide, Version 2.0.0. Alpha, Nov. 5, 1999. http://www.merit.net

[16] MRT User/Configuration Guide, Version 2.0.0 Alpha. Nov. 5, 1999. http://www.merit.net

[17] MRT Programmer's Guide, Version 2.0.0 Alpha. Nov. 5. 1999. http://www.merit.net

[18] OpenLoad 0.1.2 for Linux, June 26, 2001. http://openload.sourceforge.net/

[19] L. Amini, H.Schulzrinne and A. Lazar. Observations from Router-level Internet Traces, DIMACS Workshop on Internet and WWW Measurement, Mapping and Modeling, Piscataway, Feb. 2002

[20] Fast Internet Content Delivery with FreeFlow, white paper, Akamai, 2000

[21] D. Katabi, and J. Wroclawski. A Framework for Scalable Global IP-Anycast (GIA) SIGCOMM'00, Stockholm, Sweden

[22] Cooper, I., Melve, I. and G. Tomlinson. "Internet Web Replication and Caching Taxonomy", RFC 3040, June 2000, http://www.rfc-editor.org/rfc/rfc3040.txt

[23] Bassam Halabi, Internet Routing Architectures, 1995, CISCO PRESS

[24] Using the Border Gateway Protocol for Interdomain Routing, *CISCO*, http://www.cisco.com/univercd/cc/td/doc/cisintwk/ics/icsbgp4.htm

[25] P.S.M. Sayal and P. Vingralek. Selection algorithms for replicated web servers. In The 1998 SIGMETRICS/Performance Workshop on Internet Server Performance, June 1998

[26] K.Obraczka and F. Silva. Looking at network latency for server proximity. TR 99-714, USC/Information Science Institute, 1999

[27] P. McManus. A passive system for server selection within mirrored resource environments using as path length heuristics. Available from http://pat.appliedtheory.com/bgp-prox, June 1999

[28] CISCO, http://www.cisco.com/univercd/cc/td/doc/pcat/dd.htm

[29] M. Grossglauser and B. Krishnamurthy. Looking for Science in the Art of Network Measurement, IWDC Workshop, Taormina, Italy, September 2001

[30] K.L. Johnson, J.F. Carr, M.S.Day and M.F.Kaashoek. The measured performance of content distribution networks. In proceedings of the 5^{th} International Web Caching and Content Delivery Workshop, May 2000

[31] L. Qiu, V. N. Padmanabhan, and G. M. Voelker. On the Placement of Web Server Replicas, infocom 2001

[32] Internet Quality of Service Assessment, May, 2002, http://www.netsizer.com/daily/quality_today.html

[33] JUNOS Internet Software Release 5.3, http://www.juniper.net/products/105009.html

[34] David M. Nicol, Challenges In Using Simulation to Explain Global Routing Instabilities, 2002 Conference on Grand Challenges in Simulation, *San Antonio, TX, January 2002*

[35] M. Gan, A hybrid Hierarchical Request-Routing Architecture for Content Internetworking, M.Sc. Thesis, in progress

[36] T. G. Griffin, and G. Wilfong. An Analysis of BGP Convergence Properties, SIGCOMM'00, Cambridge, USA