Content Delivery Optimization for the Future Internet Architecture

Zihao Liu

CMU-CS-16-132

December 2016

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Srinivasan Seshan, Chair
Peter Steenkiste

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science.

Keywords: Networks, Distributed Systems, Routing for Content Identifiers, Interdomain Cache Management

To my mother, Yuping Ke, father, Shaoyong Liu,
aunt Suling Ke, and uncle Longzhong Wu

Abstract

This thesis research presents content delivery optimization techinques in the Future Internet Architecture, specifically in the context of eXpressive Internet Architecture (XIA). First, we propose an intradomain CID routing protocol that achieves performance objectives of reducing request latency for ISP's customers by routing content requests off-path to nearby caches. This is beneficial for an ISP since it helps them attract more customer. The challenge for the CID routing design is to address the scalability of the protocol to a large number of CIDs in the domain. To address this challenge, we propose 1) scoping the advertisements and 2) advertising the delta of locally cached CIDs. We evaluate the scalability of the protocol by comparing it against traditional link-state and distance vector routing protocol adapted for CIDs.

The second part of the thesis discusses an interdomain cache management scheme. We discuss a cache sharing model that allows regional peering ISPs to collaboratively share their cache space to reduce cost of data exchange with their transit providers, and the tradeoffs between performance, availability, and bandwidth. Our results show that replicating the top few popular content locally in the domain achieves a good balance between these criterias for each coordinating ISP. Finally, we present a cache sharing algorithm that distributes the request load evenly among coordinating domains without incuring any explicit message exchange. We evaluate this approach against the oracle that would require coordination overhead but approximate the optimal solution. The result shows that the caching decision output by our approach differs little from the oracle in terms of the cache hit rate for each coordinating domain.

Acknowledgments

This thesis would not be possible without many people's support and encouragement. First and foremost, I would like to thank my advisors, Professor Srinivasan Seshan and Professor Peter Steenkiste for their guidance and lessons in networking research. Networking research does not come naturally to me, but despite their busy schedule they were very patient throughout the process. I thank them for the lessons from how to design systems and evaluate the design to how to give a research presentation. I will always cherish the intellectual discussion we had for various topics and ideas presented in this thesis. Finally, I thank them for making my times at Carnegie Mellon intellectually challenging but fruitful and memorable.

I would also like to thank Dan Barrett for his help and support regarding the XIA prototype. Dan is always available and patient for any questions I have regarding the prototype. I am grateful to have his guidance and support for implementations of ideas in this thesis.

I thank Professor Vyas Sekar for the discussion on designs of the video use case. I thank Nitin Gupta for discussion on the network joining in XIA. I thank Rui Meireles, Junchen Jiang, Matt Mukerjee and David Naylor for their advice in graduate school. I thank the rest of the XIA team for making my research experience during my MS studies here great.

I would also like to thank others outside of the XIA team for making the graduate school experience wonderful at Carnegie Mellon. I thank Dave Eckhardt for his academic advice and support during my MS studies at Carnegie Mellon. Dave is a great academic advisor. I thank him for introducing me to the XIA team at the beginning and his continuous support throughout my graduate studies at Carnegie Mellon. I would also like to thank Tracy Farbacher for her help on various administrative issues regarding this thesis and her help as an advisor for the MSCS program. Furthermore, I thank people I met in School of Computer Science at Carnegie Mellon for providing the intellectually challenging but memorable graduate school experience.

Finally, I want to thank my parents and my aunt and uncle for their encouragement and support throughout my life. I thank them for providing me with the best possible resources and environment for education.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The core foundation of the great technical success of the Internet can be largely attributed to the end to end design principle [30], which states that functions placed at the low level of the system may be redundant and of little value when compared with the cost of providing that function at that level. As a result, the end to end principle argues that functions in question can only be implemented with the help of applications at the end points. Indeed, Internet Protocol, the narrow waist of the Internet, implements minimal functionalities to provide internetwork routing as well as packets delivery between two end hosts identified by the source and the destination IP address. The success of the Internet builds upon the minimalistic as well as the end to end approach of the IP service model with many popular Internet applications such as Google that provide services for millions of people.

However, the end to end principle also implies that content delivery should be implemented at the end hosts between the customers and content providers. The network should assume minimal responsibilities for assisting in the content delivery because otherwise the services provided by networks would always be redundant and at an unjustifiable costs. The latter is especially true given the scale of the content available on the Internet. As a result, existing solutions such as Content Delivery Networks (CDN) build upon the predominnat Internet Protocol service model and optimize content delivery on top of the host to host based communication model. There are a couple of challenges in the existing CDN service model for content delivery:

1. **Limited flexibility**: content requests are bound to specifc host names or IP addresses rather than content itself.
2. **Content integrity**: Integrity of the content response is not guaranteed since host to host based communication secures communication pipe but the data.
3. **Limited awareness of network infrastructure**: CDN is an application layered solution to content delivery. The end result is that the delivery decisions can be sub-optimal without knowing the network conditions (such as congestion etc).

These challenges are especially magnified given the rise of Internet streaming traffics such as Video on Demand or Live Streaming services offered by the top content providers such as Netflix, Youtube with thousands of millions of customers around the globe.

To address these challenges, recently there have been proposals for clean slate network design in which contents are identified as a key network principal. Some of these proposals include

XIA [20], CCN [22] and etc. In these proposals, content in the network can be fetched using the Content Identifiers (CID), which is the crytographic hash of the content. The advantages of using CIDs are that 1) content request can be satisfied in a more flexible fashion and 2) using the cryptographic hash of the content means that the content response can be easily verified at the receiving end. In addition, on-path routers are generally equipped with caches so that popular content can be cached at the router locally. Future requests can be satisfied at these on-path routers. Caching on the routers means 1) lower latency for content requests, 2) reduced server load on the content origin, and 3) reduced upstream network congestion.

In-network caching provides not only performance but also economic benefits for the Internet Service Provider (ISP). A customer ISP can cache the content response from its providers in order to save the cost of data transfer from providers. A peering ISP can cache the content response from its peers so that its customer can enjoy lower content response latency. Caching can also help addressing the ”peering wars” since caching popular content from the other peers can reduces the link load and bandwidth consumption on the peering link. In this context, there are many research that aims at optimizing the content delivery for these Future Internet proposals with in-network caching capabilities. The key goal of this thesis is to look at two core components for CIDs: 1) CID routing and 2) Cache Management.

1.1 Contributions

This thesis aims at addressing intradomain routing as well as interdomain cache management for CIDs. The contributions of this thesis include

1. Design and evaluation of an intradomain CID routing protocol.
2. Design and investigation of tradeoffs for an interdomain cache management scheme.
3. Design and evaluation of an interdomain cache sharing algorithms that assign cache load evenly to ISP peers.

We use a combination of experiments and analytical modeling for evaluating the design. For the CID routing we want to show that it is scalable and more efficient for route computations compared to the traditional link state and distance vector routing protocol. For the interdomain cache management, we describe a scheme for a group of peering caches to collaboratively share their cache space with each other to increase each domain's cache hit rate and reduce its economic cost. We discuss the benefits in the context of tradeoffs between performance, availability and bandwidth consumptions. Finally, we discuss a cache management algorithm that balances these tradeoffs for each peering domain and its applications in the inter-domain settings.

1.2 Organization

The organizations of the thesis is as follows: chapter 2 and chapter 3 will primarily be discussion on the design and evaluations of intradomain CID routing as well as interdomain cache management. For CID routing, we will discuss how it achieves scalability in terms of the overhead to setting up the routes compared to traditional link state and distance vector. For interdomain cache

management, we propose a caching sharing scheme among a group of peering domains and show tradeoffs between performance, availability and bandwidth consumption. Finally, we propose a cache sharing algorithms with implicit coordination that achieves even division of cache load among coordinating peers without explicit exchange of coordination messages.

Finally, chapter 4 concludes the thesis and summarizes the contributions, limitations and future work.

Chapter 2

Intradomain routing on Content Identifiers

2.1 Introduction

2.1.1 Motivations

Intradomain routing on CIDs is motivated by the benefits of fetching content off-path. Fetching content off-path is primarily attractive for an ISP due to its performance benefits. The performance here means lower latency of content response to its customers, which could help attract more customers for the ISP. Another is the economic benefit. Rather than sending requests to content origin through transit providers, fetching content off-path from another router within the same domain means cost savings for ISP. Despite these benefits, the challenge for CID routing is scalability. Routing on CID flat identifiers without any scoping won't scale to a large network. In addition, advertising CIDs that turnover quickly means that CID routes on receiving routers would become stale with high probability. We will ellaborate on both the benefits and challenges of CID routing in intradomain settings in this chapter. The next chapter will have discussion on its application and modifications in the interdomain settings.

CID Routing Benefits

Two benefits for deploying CID routing within an ISP are: 1) lower response latency for ISP's customers, which increases the service quality for an ISP and hence can help attract more customers and 2) lower cost for the ISP since content requests can be satisfied within the domain without incurring the cost to fetch from its transit providers. Consider the following intradomain setting (see Figure 2.1):

In this case, suppose that routers in ISP1 all have on-path caching enabled. ISP1 could be a tier 3 network access provider such as Century Link [26] or XO Communications [13]. ISP2 could be a large tier 1 transit provider such as Level 3 [3] or Cogent [11]. Finally the content provider could be Netflix [27] or Hulu [21]. The client in ISP1 are sending requests to fetch his subscribed videos from the content provider. The first hop router of his requests is r1, which is connected to a nearby router r2. r1 does not have the content cached locally for client, but r2 do have it cached (r2's clients might request this popular videos first).

Without CID routing, client's requests would traverse all the way to the content provider

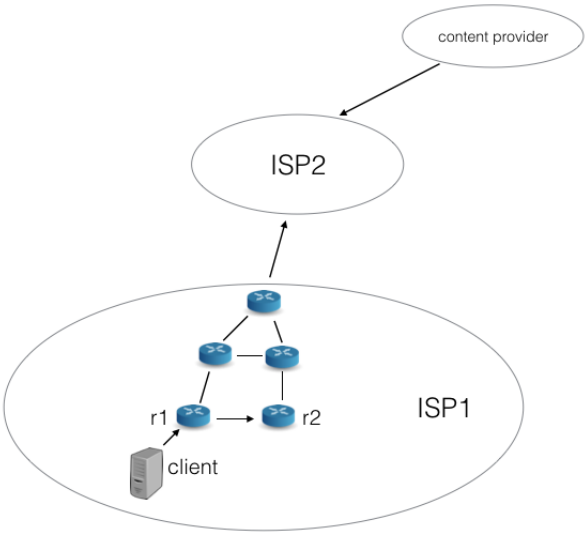

Figure 2.1: CID routing in intradomain scenario. Arrows between ASes mean customer pays provider.

through ISP2. This is bad for the clients because his requests would go through many hops in order to reach content provider, which could increase the delay for the response to requests. This is also bad for the ISP1 since now it needs to pay money to its provider ISP2 for both the in-bound as well as out-bound traffics from its own customers.

With CID routing, r1, r2 and the rest of routers in ISP1 can now advertise their locally cached content to each other. When clients send to r1 its content requests, these requests can now be forwarded off-path to r2, if r2 advertises the same content. The same goes for r2. For clients, CID routing provides lower latency access to their contents since these contents can be fetched off-path from nearby routers, rather than through content providers far away. For these ISPs that provide network access for customers, CID routing can increase the quality of their network services and hence attract more customers.

The second benefit of CID routing is that it provides economic benefits for the ISP. Without CID routing, ISP1 needs to forward the request to its transit provider ISP2. With CID routing, content requests can be satisfied within ISP1 as much as possible from nearby routers in the same domain.

CID Routing Challenges

Despite the benefits, CID routing faces the challenge of scalability. There are two main points that affect scalability for CID routing. First is the scalability of the routing protocol to adapt to a large network topology. Traditional routing protocols such as link state and distance vector rely on the hierarchical nature of IP addresses to address this challenge. On the other hand, CID in XIA is a flat identifier. It is a well-known philosophy from routing design that routing on flat identifiers won't scale to a large network. Second is the scalability of the routing protocol to adapt to the scale of the content on the Internet. The number of CIDs is much greater than the number of end hosts on the Internet. The majority of the traffics on the Internet now is dominated

by content, especially given the rise of Internet Video [10]. Hence, the same routing techniques used by the traditional routing protocols such as link state and distance vector won't be scalable to the size and number of contents in a large network.

2.2 Related Work

2.2.1 Distance vector and Link state

Two traditional classes of intradomain routing protocols are distance vector [33] and link state [31]. Distance-vector routing protocol is designed based on "routing by rumor", in which each router periodically broadcasts a vector of distance to all nodes in the network. On the other hand, link state routing protocol is designed based on "telling the worlds about my neighbors". Each router periodically broadcasts its neighbors to the rest of the network. After hearing the advertisements, every router constructs a map of connectivity and computes the shortest path to each destination based on its local connectivity map. These traditional routing protocols provide intradomain reachability to end hosts on the Internet. Scalability is achieved by taking advantage of the hierarchical structure of IP addresses. Nonetheless, adapting to routing on flat identifiers such as CID remains a challenge since the address is no longer hierarchical. In addition, the number of CIDs can far exceed the number of end hosts handled by these traditional protocols.

2.2.2 Routing on Location Independent Identifiers

There are several research efforts that provide routing on location independent identifiers for data and services. In TRIAD [19], the concept of route-by-name is introduced to address the scalability issues of DNS lookup to content servers. In TRIAD, content server advertises the content names (URLs) it has locally to content routers. Upon receiving this advertisement, content routers would map these URLs to the next-hops to reach these content. To address scalability, TRIAD's content routing relies on the aggregation in the URL names and advertises the name suffix reachability just like BGP advertises address prefix for ASes. Nonetheless, routers do not have their own cache space in this model as in XIA. In addition, URL names are hierarchically structured whereas CIDs are not.

In DONA [23], a name-based anycast primitive is proposed to replace name-to-location translation scheme by traditional DNS service to achieve better persistence, availability and authenticity. Only names, which are flat identifiers, are used to identify data and services through a resolution handler in each domain. A nearby content in another domain can be located using route-by-name approach similar to TRIAD to find the shortest path to the content. However, the goal of DONA is not to argue for an in-network layer intradomain routing for CIDs, but rather to argue for the benefits of name-only based resolution primitives. Nonetheless, the idea of routing to the nearest close-by content can still be valuable for the design of CID routing protocol.

Similar to DONA, ROFL [8] argues for an identity and location split in the architectural design in which routing to end hosts is done on flat identifiers only. ROFL uses DHT like ring where each identifier is assigned some positions on the ring. Routing is performed much like the DHT lookup. Although ROFL addresses routing on flat identifiers, these identifiers are

primarily used for identifying end hosts, not content, which is of much larger scale. In addition, scalability is hard to achieve by only routing on flat identifiers to provide global reachability to end hosts. The goal of intradomain CID routing is not to provide global reachability, as discussed in Section 2.3.1. Finally, the problem with using DHT is the long path stretch, which can impact the latency of content requests.

2.2.3 Routing in Information Centric Networks (ICN)

More recently, there have been many proposals in the clean slate network designs that have the content as the first principal. Content routing in ICN has become an active area of research. In [29], a hash routing scheme is proposed to allow efficient routing to content. In their model, each edge router implements a hash function that map a content to a cache in the domain. When requests arrive, edge router can simply compute the hash value to locate which cache in the domain to forward the requests to. Scalability is achieved since there is no signaling overhead in the network to setup the routing table. The drawback in this scheme is the long path stretch since content placement among caches is determined based on the value computed from the hash function, which may not be the most optimal.

In SCAN [24], content routing is only scoped to nearby routers in order to achieve scalability. The CID routing table is set up using the distance vector protocol. To further limit the periodic signaling overhead, router in SCAN uses bloom filter (BF) to condense the CID advertisements sent to its neighbors. When receiving this BF, router compresses this information into the BF of its local interface where the advertisement is received. During forwarding of a CID request, if k hashed values of request content's CID are matched to BF of a particular interface, then the router can conclude that content can be located through the interface with high probability. Scoping the CID advertisement can be very valuable to address the scalability challenge. However, the use of BF in SCAN also presents the problem of false positive, which means that CID forwarding path might be invalid. This can increase the path stretch and hence the latency for clients' requests.

2.3 Design

Previous approach of off-path content routing addresses scalability challenge at the expense of longer path stretch and higher latency for content requests. On the other hand, addressing the scalability of CID routing shouldn't be at the expense of performance for content requests. Hence, CID routing primarily aims at optimizing for performance in a scalable fashion. Hence, we will discuss the CID routing design that addresses scalability and how it maintains the performance benefits of off-path content fetching in the following section.

2.3.1 Design Goals

Scalability and Performance

Design for performance means that the key goal of CID routing is to optimize for latency of content requests. Hence, techniques that optimize for scalability should not mitigate the perfor-

Figure 2.2: Benefits of scoping the advertisements when routers are 1) far away and 2) close to each other

mance benefits of CID routing. Design for scalability means that: 1) the ability of the routing protocol to adapt to a larger and more complex network topology and 2) the ability of the routing protocol to adapt to an increasing amount of content cached locally on each router.

1. Scoping the Advertisements

To achieve performance and scalability goals, the first idea is scoping the advertisements. Traditional routing protocol provides global reachability to end hosts on the Internet. On the other hand, the goal of routing for CID is not to provide global reachability but performance improvements for content requests. The reachability of the content can always be gauranteed as long as the content origin server has the content. However, global reachability does not make sense for CID routing since the performance penalty would not justify the overhead paid in the routing protocol to set up routes for content on caches very far away. For instance, we can consider the following two cases (Figure 2.2):

In the following discussion, we assume that routers that are far away in distance have longer hops and higher latency between them. We use this assumption to simplify the discussion. In the first case, two routers, r1 in Pittsburgh and r2 in San Francisco, are very far from each other. If the advertisement is not scoped, performance for content requests would suffer since requests through r1 in Pittsburgh to content origin in Wisconsin can go off-path to r2 in San Francisco, which has much longer distance and much higher latency. In addition, setting up the CID routes between r1 and r2 that are far away is not scalable since it requires significant signaling overhead to the network as well as the CPU processing and storage consumption among many routers betweeen r1 and r2.

In the second case, two routers r1 and r2 are all in the same city in Pittsburgh. Hence, performance for content requests would improve through off-path since requests through r1 that go off-path to r2 have lower distance and hence lower latency than requests to content origin far

away in Wisconsin. In addition, the overhead to set up the CID routes for routers between r1 and r2 is low since r1 and r2 are just few hops away from each other.

2. Advertising the Delta of locally cached CIDs

The first design point addresses the ability of the routing protocol to handle large and complex network topology since each router can have a specific TTL to advertise its cached CIDs independent of the complexity of the network topology. Nonetheless, it doesn't fully address the case when routers have large amount of locally cached CIDs. If they are advertised periodically, the total size of advertisements sent to the network by all caches in the domain can be quite large. In addition, routers receiving these messages will spend significant CPU processing power and storage capacity to handle them.

The key to address scalability to a large number of CIDs cached in a domain is to reduce the redundant routing information among consecutive advertisements. This redundancy is minimized when each advertisement only contains unique CID routing information. Hence, each router only needs to advertise the delta of the locally cached content between consecutive advertisements. However, the challenge is to handle CID routes setup and teardown. To achieve these goals, when CIDs on the router are evicted, router can flood the CID route eviction message to others within its TTL. When CIDs are cached on the router, router can flood CID route addition message to others within its TTL. Finally, advertisements need to be transmitted over the reliable communication channel to ensure that messages is correctly delivered and in the same order that they are sent. The last point is of particular importance since the delivery of route eviction message for CIDs can never precede the delivery of their route addition message. Otherwise, there will be invalid CID routes in the network.

Can current RIB and FIB handle the scale of CID routes? The short answer is no for now. Intuitively, scoping the advertisements can control the number and hence size of CID routes. Nonetheless, CIDs do not have prefix aggregation as IP addresses. In addition, CID and HID identifiers are implemented as 40 bytes md5 hash [20]. Hence, route entries for CID and HID could potentially take large amount of space. To understand this question, we consider the total size required to store CID routes in FIB and RIB for a single router R, given a typical configuration for a cached network. The following are the configurations (Table 2.1):

average number of 1-hop neighbors	5
TTL	2
chunk size for each CID	2MB
CID hash size	40 bytes
HID hash size	40 bytes
port number	4 bytes
cache size	100 GB

Table 2.1: Cache network configurations for understanding the scale of CID routes to RIB and FIB

We choose these parameters to understand the scale of CID routes when these CIDs represent

video workloads, which represent significant traffics on the Internet. The average number of 1-hop neighbors for a router within domain is set to 5 since the median of average degrees of intradomain routers is 5 [16]. TTL is set to 2 due to the scoping of advertisements. Chunk size is set to 2MB to represent a single high quality video segment of 1 second for DASH videos[14]. CID and HID hash size as well as port number are chosen from XIA [20]. Finally, cache size is set from [32] since authors argue that the benefits of in-network caching for video workloads with caches smaller than 100GB are marginal.

In this context, each router could generate $\frac{100\text{GB}}{2MB} = 50000$ CIDs. Then the total number of entries for R's CID routing table would be $50000 \times 5 \times 4 = 1000000$ entries, which have a total size of $1000000 \times (40 + 40 + 4)$bytes $= 84$MB. These CID routes can fit into the RIB since RIB is generally stored in RAM. On the other hand, it poses a challenge for FIB since most FIBs today are implemented with TCAM. TCAM is a low capacity but high speed hardware specialized for network packet forwarding. For instance, the NL9000 family TCAMs [2] from NetLogic Microsystems contains contain up to 1 million 40-bit entries, which has only 5 mega bytes.

The core of the limitation is that most current FIBs are designed for IP forwarding, not forwarding for flat identifiers such as CID. Nonetheless, we are still optimistic that FIBs for flat identifiers can be made efficient with more capacity in the future. For instance, there are existing solutions that could scale the FIB table for ethernet MAC address (6 bytes) to up to 1 billion entries with good packet processing rate [36].

2.3.2 Design Details

The design of CID routing protocol is motivated by efficiency and scalability to a large number of CIDs in a domain. Traditional link state and distance vector routing protocols have shown to work quite well to provide intradomain reachability to end hosts. However, there are two issues adopting them for CID routing: 1) they are soft state style routing protocol where advertisements are sent periodically and 2) they are designed for providing reachability to end hosts. The design goal of the CID routing is to provide performance benefits for content requests rather than reachability to all CIDs in a domain. Hence, low advertisement overheads, efficient route computation, and low storage cost of the routing protocol are important goals.

To achieve these goals, the high level design decisions for CID routing are: 1) each router reliably floods the delta of its locally cached content to nearby routers within TTL. This achieves the scalability benefits of scoping the advertisements as well as advertising the delta of locally cached CIDs. Reliability of the flooding can be achieved through sending advertisements using TCP or other retransmission schemes. 2) router receiving CID advertisements keeps for each destination CID a list of all possible routes. This is motivated by the performance benefits. For instance, when a route is removed due to receiving the delta of advertisements, routers can still recover the best route from remaining CID routes to the same CID. 3) the route with the shortest distance to a CID is used to forward the CID request packet since it provides lower latency for requests from off-path. 4) when R receives advertisements for new CIDs from another router, it will insert these new CID routes into its CID routing table. If any of these routes in the message is new to R or has shorter distance than existing CID routes in R's CID routing table, it will be updated to R's forwarding table. 3) When R receives advertisement for evicted CIDs

from another router, it will remove the corresponding routes from its CID routing table. If no remaining routes to the same CIDs in the eviction message are available on R, it will remove them from R's CID forwarding table. Otherwise, R will update to its CID forwarding table the shortest CID routes from those remaining routes in its CID routing table.

Below we list out comparisons of design features between CID routing, link state and distance vector (Table 2.2):

	CID routing	Link state	Distance vector
Goal	performance	reachability	reachability
Scope	neighborhood	domain	domain
Advertisement frequency	topology changes	periodic	periodic
Advertisement content	delta of local content	all content	all content
Advertisement filtering	no	no	yes
Advertisement channel	reliable and ordered	unreliable and not necessarily ordered	unreliable and not necessarily ordered
Route visibility for a router	all routes to CIDs on caches in the neighborhood	all routes to CIDs on caches in the domain	best route to CIDs on caches in the domain (due to filtering by distance)
Routing information format on router	hash table of destination CID to a list of all routes to that CID from routers in the neighborhood	connectivity map among all routers in the domain	vector of distance to all routers in the domain
Route computation	shortest route among list of all routes to the same CID with cost metric from the message	Dijsktra shortest path to all routers with received advertisements in the domain	Bellman Ford to all routers with received advertisements in the domain
Route computation frequency	topology changes	periodic	periodic
Network topology changes (router/link join/failure)	neighbors of affected routers inform other routers	by default periodic advertisements	by default periodic advertisements
Issues	handling network topology changes is more complex	route computation is expensive	convergence time is slow, count to infinity problem

Table 2.2: Comparison of design features between CID routing, distance vector and link state

Hence, to achieve performance objectives, CID routing is designed to limiting the advertisement and processing overheads to the network and improves the efficiency of route computation. However, the drawback of adopting these optimizations is that CID routing needs to have an explicit mechanism to handle network topology changes such as router or link failures, since advertisements are sent only once for content cached locally. We will elaborate on how to handle the topology change for CID routing in later section.

CID Advertisement Message Format

The advertisement message would need to contain a sequence number as well as HID of the originator of the advertisement message. The combination of these ensures that duplicate messages are filtered out to reduce the overheads to the network. Scoping the advertisements means that each router advertises its cached content with a TTL to its neighbors. Advertising the delta means that CID advertisements need to contain fields that denote: 1) the evicted CIDs and 2) the newly cached CIDs at router that originates the advertisements. Furthermore, to identify the cost of CID routes in the advertisement, message needs to contain distance or the number of hops. Finally, originator's HID is also used for other purposes including 1) handling CID topology changes and 2) handling network HID topology changes. For instance, when CIDs from a cache on router R_i are evicted, router R_j, upon receiving the eviction message from R_i, needs to make sure that it purges the CID routing information for exactly R_i not someone else.

Hence, the advertisement message has the following information:

1. seq: sequence number of the advertisement
2. ttl: number of hops to broadcast the advertisement
3. distance: number hops to the orignator of the advertisements
4. originatorHID: the HID of router that originates this advertisement
5. newCIDs: new CIDs from originatorHID
6. delCIDs: evicted CIDs from originatorHID

CID Routing Table

Due to the size and scale of CIDs in the network, CID routing table should support fast and efficient lookup, insertion and deletion of CID routes. Hash table is a good data structure to support these operations since it takes on average $O(1)$ insertion, deletion and lookup. In addition, a given CID can originate from multiple routers within the neighborhood. Providing visibility for all routes to the same CID in the neighborhood means that each CID route entry can be uniquely identified by $CID_i + HID_x$ pair, where CID_i is the destination CID and HID_x is the HID of router that originates the advertisement for CID, or HID of router that caches the CID_i locally. CID routing table can then be represented as the following:

```
typedef struct {
  string nextHop;// nexthop HID
  int32_t port;  // interface (outgoing port)
  uint32_t cost; // distance to reach this CID
} CIDRouteEntry;

// CID -> originatorHID -> CIDRouteEntry
map<string, map<string, CIDRouteEntry>> CIDRoutingTable;
```

This data structure means that for any given CID, multiple HIDs of routers within the neighborhood might cache the same CID locally. Hence, looking up a CID to the hash table will get

back a list of all possible routes from different routers that also cache the CID within neighborhood. Looking up any of these routes identified by CID_i and HID_x needs two hash table lookups, which takes O(1) time in total. The same goes for inserting a new route and deleting an existing route. Finding the best route for CID_i would take $O(k)$ time, where k is the number of routes within neighborhood to CID_i. This is because the algorithm needs to iterate through each options of a total of k routes and find the best one with the minimum distance. In practice, due to advertisement scoping, k is generally small and can be treated as a constant number. Hence, finding the best route to CID_i would only take $O(1)$ time.

Routing Protocols

1. Sending Advertisements

Sending the CID advertisements is quite simple since router just needs to compute the delta between two versions of its locally cached CIDs: the one that it used to send the advertisements previously and the one that it has cached currently. The advertisement procedure below can run periodically on each router to update CID routes on nearby routers (see Algorithm 1).

Algorithm 1 Procedure for advertising CIDs at $router_i$

procedure ADVERTISECIDS
initialize advertisement message, msg, with $router_i$'s TTL and HID
$currCachedCIDSet \leftarrow$ snapshot of CIDs for current local content of $router_i$
$prevCachedCIDSet \leftarrow prevCIDSnapshot_{router_i}$
$msg.newCIDs \leftarrow currCachedCIDSet \setminus prevCachedCIDSet$
$msg.delCIDs \leftarrow prevCachedCIDSet \setminus currCachedCIDSet$
$prevCIDSnapshot_{router_i} \leftarrow currCachedCIDSet$
send advertisements to $router_i$'s neighbors via reliable communication channel

2. Receiving Advertisements

When a router receives an advertisement, it will first check if the message is duplicate using the sequence number and orignator HID in the message. In addition, receiving a CID advertisement means potential changes to CID routing table and forwarding table since the message might contain information for new or better CID routes than the existing ones in router's CID routing table. It might also contain information for evicted CIDs on the originator's cache. Hence, receiving router needs to remove those routes from its local CID routing table and in certain cases forwarding table as well. Finally, router will update the TTL as well as the distance metric in the message before sending it to its neighbors. We will present the procedure for receiving the CID advertisements here (see Algorithm 2) and procedure for handling changes in CID routing table and forwarding table due to CID advertisements in the next section (see Algorithm 3 and Algorithm 4).

Algorithm 2 Procedure for handling the received advertisement, msg, from $router_i$'s one-hop neighbor $router_k$

```
procedure HANDLERECEIVEDCIDADVERTISEMENTS(msg, router_k)
    DROPMESSAGEIFDUPLICATE(msg)
    DELETEEVICTEDCIDROUTES(msg)          // procedure for route computation
    HANDLENEWCIDROUTES(msg, router_k)    // procedure for route computation
    if msg.ttl − 1 > 0 then
        msg.ttl ← msg.ttl − 1
        msg.distance ← msg.distance + 1
        for router_j that is router_i's first hop neighbors do
            if router_j ≠ router_k and router_j ≠ msg.originatorHID then
                replay msg to router_j via reliable communication channel
```

3. Route Computations

The main job of route computation procedures is to ensure that shortest routes to CIDs are always set in the CID forwarding table for routers that receive the advertisements. The key design challenge for route computation is efficiency due to the potential scale of CIDs in a domain. The efficiency is partially achieved by designing the CID routing table with a hash table that has on average $O(1)$ insert, remove and lookup time. Route computation procedures take advantage of this data structure and only need to be run whenever a CID advertisement is received, rather than periodically. There are two cases for handling the route computations. The first case is handling the new CIDs in the advertisement. When new CIDs are cached on the originator of the advertisement message msg, receiving routers of this advertisement need to make sure that its local shortest paths to all CIDs of $msg.newCID$ in its forwarding table are updated if $msg.originatorHID$ indeed provides a shorter path to reach those CIDs than any existing CID routes in its routing table. The shortest CID routes are then set in the CID forwarding table. The procedure for handling new CIDs in the advertisement message is presented in Algorithm 3. The running time for the algorithm is at most $O(c_{new}k)$, where c_{new} is the number of new CIDs in the message and k is the average number of existing routes for a CID in the CID routing table. This is true since the algorithm iterates through each $newCID_x$ in the $msg.newCIDs$ and there are c_{new} of them. For each $newCID_x$, there are at most k iterations when the router needs to interate through existing routes to $newCID_x$ to update the shortest path to the CID forwarding table.

The second case is handling the evicted CIDs in the advertisement. Receiving routers need to make sure that they can recover the best route among remaining routes in its CID routing table for any CID evicted from the CID advertisement message. If there is no such route, CIDs evicted need to be removed from the CID forwarding table. Otherwise, the best routes among the remaining CID routes are set to the CID forwarding table. We present the algorithm in Algorithm 4. The running time of this algorithm is $O(c_{evict}(k-1))$, where c_{evict} is the number of CIDs in $msg.delCIDs$ and k is the average number of existing CID routes in the CID routing table. The algorithm needs to iterate over each CID_{evict} in $msg.delCIDs$ and there are c_{evict} of them. For each CID_{evict}, the algorithm need at most $k-1$ iterations to determine the best route among the remaining $k-1$ routes. We have $k-1$ here since the algorithm first would remove

the route in CID routing table for CID_{evict} from $msg.originatorHID$ and then iterate.

4. Handling HID topology changes

The design for scalability means that each router in the CID routing protocol advertises the changes in the locally cached content. In addition, advertisements in CID routing do not contain full information about the network topology, only the distance and who originates the advertisement message. Hence, there is less visibility for each router to know the complete network topology around its nearby environments. Nonetheless, for a newly joined router, its connected end hosts would preferrably want to enjoy the performance benefits of CID routing immediately. On the other hand, for a router that left the network, its CID routes might still be on nearby routers. This is bad for performance since if content requests follow through these dangling CID routes, their performance would suffer. Therefore, when such changes do occur, there needs to be an explicit mechanisms in the routing protocol to inform new routers of the existing CID routes, as well as informing the involving routers of changes in CID routes when a router that caches those CIDs leaves the network.

There are two approaches to this. The first approach is to go back to the soft state style approach of advertisments used by distance vector and link state, in which HID topology changes can be handled by default since routes can be invalidated or validated by periodic refresh messages. However, as we will show in the evaluation section, soft state style advertisements introduce significant signaling overhead to the rest of the network, impacting the scalability of the CID routing. In addition, CID routing is primarily designed for the situation where the network topology is quite stable. Frequent changes of network topology (such as vehicular network) mean that the performance benefits of fetching content off-path would be low, since routers that cache the content might not be available or available for just a short amount of time.

Another approach is to let neighbors of routers that just leave or join the network inform the rest of the network of changes in CID routes due to HID topology changes. The advantage of doing this is that it preserves the scalability benefits of CID routing in most of the cases when HID topology is quite stable. This introduces minimum overhead to the rest of the network since the set of routers whose CID routes are affected by HID topology changes are in the local neighborhood. Hence, the signaling traffics would be low. This approach can achieve correctness because CID routing tables of affected neighbors contain enough routing information from advertisements received prior to the changes of the HID network topology. When a node joins, they can reconstruct those advertisements from their CID routing table and send them to the newly joined node. When a node leaves, they can inform the rest of the routers within TTL of the changes in CID topology by sending CID route eviction advertisement messages on behalf of the node that left the network.

2.4 Evaluation

The goal of the evaluation is to show that CID routing is more scalable and efficient compared to the traditional distance vector and link state routing protocol. We used a combination of experiments and analytical models to show the results.

2.4.1 Scalability and Efficiency

We first show that CID routing introduces less overhead to the network compared to the soft state style advertisements that broadcast the locally cached content periodically (scheme used by link state). Then, we show that the size of advertisements of CID routing is low compared to that of link state and distance vector. We show this with experiments on a network with 28 nodes deployed on GENI. Finally, we will show the comparison of efficiency of route computation as well as the storage consumptions between three routing protocols

Comparison of the number of advertisements

CID routing uses "hard state" advertisement in which only the delta of the cached content is sent reliably to the network, whereas link state and distance vector use "soft state" advertisement to refresh routes periodically. We show here that CID routing is able to send much less number of advertisements to set up the CID routes as the number of CIDs cached on each router in the network increases. We begin by stating some simplified assumptions to the model we have: 1) all routers have the same TTL, 2) all routers have the same number of neighbors, 3) all routers have the same number of content cached locally, 4) churn rate for newly arrived chunks are uniform, and 5) advertisements are sent through reliable channel. Our goal is to understand the trend of differences of signaling overhead between two different types of routing protocol: 1) those that advertises periodically (soft state) and 2) those that only advertises the delta (hard state).

We begin by defining some parameters to the cache network (see Table 2.3):

Variable	Definitions
ttl	number of hops to broadcast advertisements for a router
n	number of 1-hop neighbors
C	capacity of cache in terms of maximum number of CIDs
s	size of chunk represented by each CID hash
h	size of a CID hash
r_a	number of CID advertisements per second sent by each router (for soft state advertisement)
r_{ca}	churn rate for newly arrived chunks on each router
r_{ce}	churn rate for evicted local chunks on each router. Note that $r_{ce} = 0$ if $c \leq C$, else $r_{ce} = r_{ca}$
$c(i)$	number of locally cached CIDs for each router at ith soft state advertisement. $c(i) = c(i-1) + \frac{r_{ca}}{r_a}$ if $c \leq C$ else $c(i) = C$

Table 2.3: Variable definitions for evaluating the signaling overhead between soft state and hard state protocol

Equations

We count the overheads between two styles of advertisements in rounds. Each round has length $\frac{1}{r_a}$ seconds. Then, we compare the overheads of soft state and hard state advertisements upto multiple rounds of advertisements. Finally, we assume that the duplicate messages are not filtered out by the receiving router, or that the network topology has no loops, in order to simplify the equation.

The total number of CIDs propagated to the network due to advertisements for c number of locally cached CIDs on router R, denoted $sendCost$ is

$$sendCost(c) = nc(1 + \sum_{i=1}^{ttl-1}(n-1)^i)$$

This can be derived from the following argument: the total number of CIDs propagated at the first hop of R is nc , the total number of CIDs propagated at the second hop of R is $n(n-1)c$, and so on until the advertisement reaches the ttl number of hops. Note that each step other than the first hop we have factor $(n-1)$ since advertisements do not need to go back to its sender. When we sum these value all together, we get the equation above.

The total number of CID advertisements received by a router for each round of advertisements in the network, denoted $recvCost$ is the same as $sendCost$ since each router has the same number of CID cached locally and the same number of 1-hop neighbors.

$$recvCost(c) = sendCost(c)$$

Hence, for soft state routing (distance vector and link state), the cost of control traffics generated by nth advertisement for each router is thus

$$c' = c(n)$$

$$cost_{soft}(c') = h \times (sendCost(c') + recvCost(c')) = 2h \times sendCost(c')$$

For hard state routing (CID routing), the cost of control traffics is caused by changes in the local cache. Hence its cost with each round of advertisements is

$$c' = \frac{(r_{ca} + r_{ce})}{r_a}$$

$$cost_{hard}(c') = h \times (sendCost(c') + recvCost(c')) = 2h \times sendCost(c')$$

In both cases, the data traffic gained by a router for each round of advertisements is

$$dataTraffic = s \times \frac{r_{ca}}{r_a}$$

Results

We plot out the analytical cost of control traffics for a single router in a large network using the following configurations. For instance, $n = 5$ since the median average router degree within an AS on the Internet is 5 [16]. We set $ttl = 3$ since the CID advertisements need to be scoped to within a small neighborhood, and the maximum number of hops on the Internet is 20 hops. Cache size is 1GB since we want to know the lower bound on the advertisement costs and 1GB is considered a smaller cache size for video workloads [32]. $s = 1$MB to represent a high quality DASH video segments for one second. $h = 40$ bytes since it is the size of CID in XIA. Finally, r_a and r_{ca} are chosen to understand how chunk arrivals affect the cost of advertisements.

n	5
ttl	3
C	1000 CIDs
h	40 bytes
s	1MB
r_a	1 advertisement per second
r_{ca}	1 new chunk arrives every 2 second

Table 2.4: Cache configurations in a large network for understanding the signaling overhead between soft state and hard state routing protocols

The results is the following (Figure 2.3):

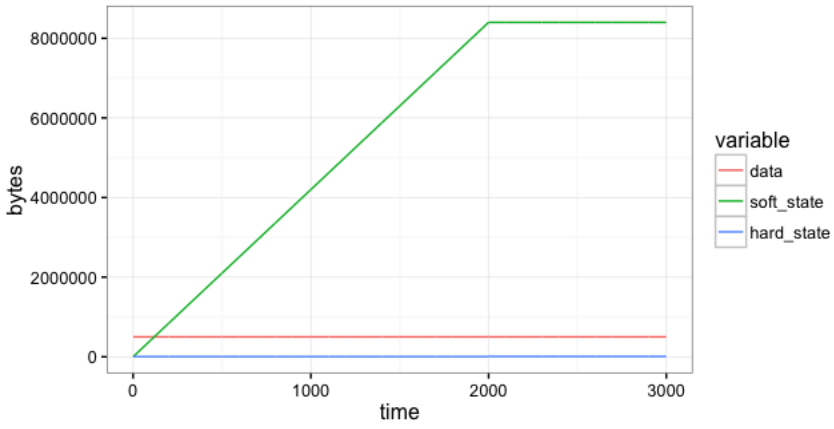

Figure 2.3: Comparison of signaling overhead between soft state and hard state style protocol with respect to data traffic

Clearly, the overhead of hard state advertisements for CID routing is much lower compared to soft state advertisement for link state and distance vector routing protocol. The initial growth of the signaling overhead by soft state style protocol is due to the fact that the cache size is not full. Once the cache size is full (at time 2000), soft state style protocol will advertise exactly the maximum number of CIDs of a cache. On the other hand, advertising only the changes in locally cached CIDs can avoid this redundancy in the advertiement since what is advertised before do

not need to be advertised again. Hence, its overhead in control traffic to the network is much lower. In fact, it is even lower than the data traffic since the size of a CID hash in general is much smaller than the size of chunks it represents.

Comparison of the size of advertisements

We implemented the CID routing protocol in XIA and compared it against routing for CIDs implemented using the standard link state and distance vector with scoping. For standard link state, each router would maintain a local topological database from advertisements received within the TTL of each advertisement. For each node in the topology, router would also maintain the locally cached CIDs at that node, which is included in the standard LSA advertisement message. For the standard distance vector, each router would send its CID routing table to others within its advertising TTL. When a router receives the advertisements for the CID routing table, it will filter out those with longer distance or those with about to expired TTL, and then broadcast it to the rest of the neighbors. There are roughly 1300 lines of code for CID routing, 500 lines of code for distance vector, and 1000 lines of code for link state. We use the following topology deployed on GENI to emulate an intradomain network (see Figure 2.4)

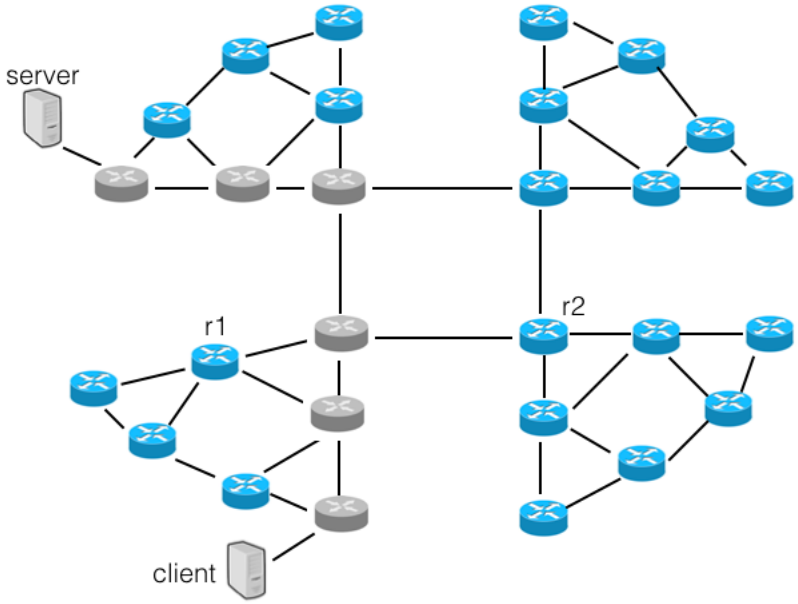

Figure 2.4: Network topology deployed on GENI used for comparing the size of advertisement messages among CID routing, link state and distance vector

The gray router means the shortest path (or on-path route) between server and client. We compare the size of advertisements received at each router between three routing protocols under different TTL. The server publishes a video file of 36MB with 36 CIDs, from which the client machine fetches. We pick r1 and r2 that would always receive advertisement messages regardless of the TTL (1 to 3). Routers in the same experiment have the same TTL. We plot the CDF for size of the received advertisements over the course of transfering of this file. Note that the size of the received message for link state and distance vector is always smaller than 1200 bytes since

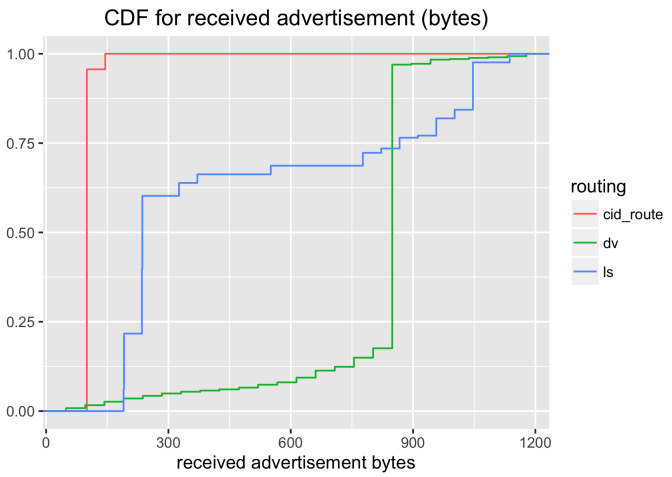

Figure 2.5: CDF for the size of received messages at r1 when TTL = 1

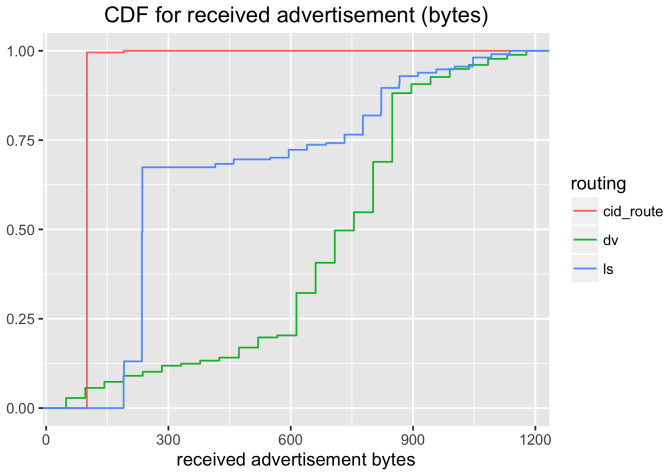

Figure 2.6: CDF for the size of received messages at r1 when TTL = 2

the implementation of XIA's UDP used for this experiments can only send up to 1200 bytes of messages. Hence a single advertisement might be broken down into multiple small advertisement messages. Finally, all three routing protocols are advertising their routes at the same rate.

As shown in figure 2.5, 2.6, 2.7, CID routing has the minimum size of advertisements among all three routing protocols. We observe similar result in figure 2.8, 2.9 and 2.10. This is because CID routing will only advertise the delta of locally cached content to the network, which is much smaller compared to the entire routing table (distance vector) and local content (link state) periodically.

Comparison of route computation efficiency and storage consumptions

We compare the asymptotic running time and space complexity between three routing protocols. We begin by variable definitions for a cache network:

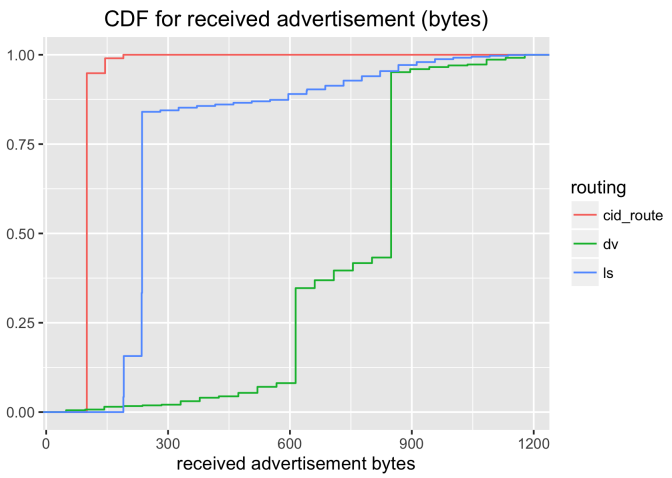

Figure 2.7: CDF for the size of received messages at r1 when TTL = 3

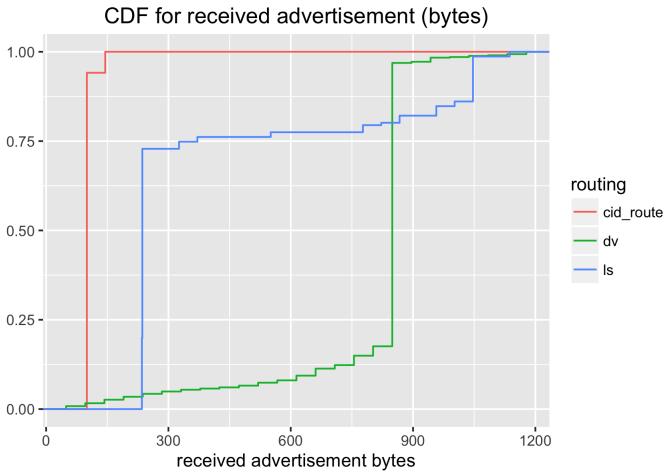

Figure 2.8: CDF for the size of received messages at r2 when TTL = 1

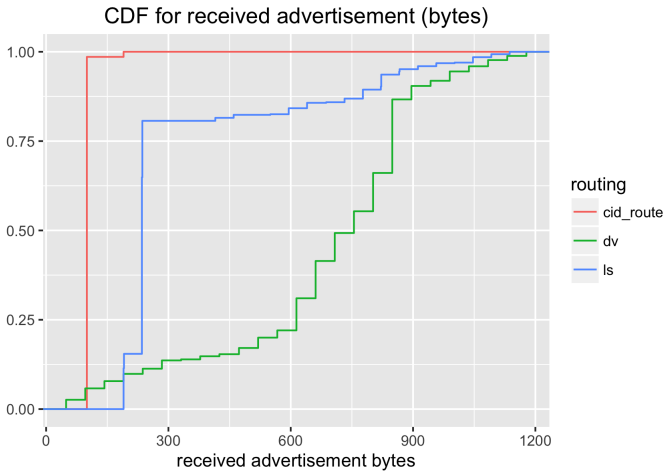

Figure 2.9: CDF for the size of received messages at r2 when TTL = 2

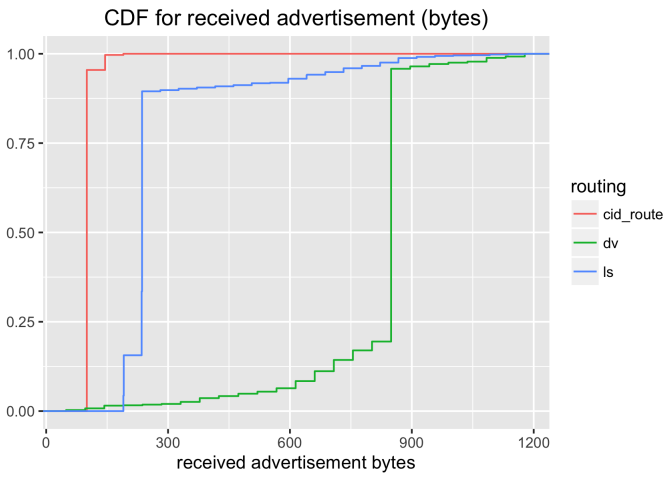

Figure 2.10: CDF for the size of received messages at r2 when TTL = 3

Variable	Definitions
k	number of routes to each destination CID on R
m	number of routers that advertise to R ($m \geq k$)
c	number of content cached on routers in the network

For finding the best route for a single CID on router R, here are the time complexity for each routing protocol (Table 2.5)

	time complexity	frequency
CID routing	$O(k)$	topology changes
link state	$O(mc) + O(m\log(m))$	periodic
distance vector	$O(m^4c)$	periodic

Table 2.5: Cost comparison of route computation between CID routing, link state and distance vector

CID routing is $O(k)$ due to the hash table design. A hash table lookup is needed to find all routes to CID, which have size k. Link state is $O(mc) + O(m\log(m))$ since Dijsktra takes $O(m\log(m))$ to find the shortest path within the local neighborhood of m routers. After Dijsktra, finding shortest path to each CID means iterating through each node's cached content in the topology database, which takes $O(mc)$. Finally, distance vector implements bellman ford algorithm, which takes $O(m^3) \times \frac{mc}{k} = O(\frac{m^4c}{k}) = O(m^4c)$ since k is small in practice. This is derived from that bellman ford to each router takes $O(m^3)$ and the length of a distance vector is $\frac{mc}{k}$ since there are a total of mc CIDs that R would receive advertisements from, and for each unique CID, there are k duplicates. So the total number of unique CIDs R would receive is $\frac{mc}{k}$. The relationship of time complexity among three routing protocol is $O(k)$ is asymptotically less than $O(mc + m\log(m))$, which is asymptotically less than $O(m^4c)$. Hence, CID routing is the

most efficient for finding the best route to a CID.

For space complexity, we have the following (Table 2.6)

	space complexity
CID routing	$O(mc)$
link state	$O(mc)$
distance vector	$O(mc)$

Table 2.6: Comparison of space complexity between CID routing, link state and distance vector

CID routing is $O(mc)$ since its routing table identifies each CID route with $CID+HID$ pair, where CID is the destination CID and HID is one of the nearby originators that cache the CID. Since there are a total of m routers that advertise to R and each router stores on average c number of content, the total size used by CID routing is $O(mc)$. The same goes for link state since it keeps track of locally cached content for each router from a total of m routers in its topology database. Hence the space is also $O(mc)$. Distance vector is $O(\frac{mc}{k})$ since it filters the routes that are not the shortest for R. There are a total of $O(mc)$ possible CID routes for router R, and for each CID distance vector only chooses the best one among k options. Since in practice k is small and a constant, $O(\frac{mc}{k}) = O(mc)$. Hence, the total number of routes distance vector will keep track of for router R is just $O(mc)$. Therefore, in terms of space complexity, all three routing protocols are the same.

2.5 Summary

In summary, this chapter discusses the design and evaluation of CID routing, a routing protocol for content identifiers for XIA future Internet architecture. The core objective of CID routing is to improve performance, namely lower content request latency, for ISP's customers. In addition, CID routing needs to achieve scalability in terms of signaling overhead as the network becomes more complex and as number of content cached on router increases. The design decisions to address these two challenges are: 1) scoping the advertisements and 2) advertising only the delta of locally cached CIDs. To show that CID routing's scalability benefits, we compare it against the tradtional link state and distance vector and show that CID routing provides higher efficiency for route computation and lower overheads to the network.

Nonetheless, the benefits of CID routing is not just performance within a single domain. The design can be adapted to interdomain settings where a business policy module can be added to the CID routing. Each domain can advertise the most popular content cached locally to other domains subject to the bussiness policy of interdomain routing. A scenario of particular research interest is that of peering relationships where a group of peering caches collectively share the cache to increase each peer's cache hit rate locally. CID routing in that scenario can provide information to each peer as to what content other peers have cached locally and set up CID routes to content on the other peer. Nonetheless, a cache management scheme is needed to enable this design since peers need to decide the collaboration scheme of who should keep what content, which will be the focus of next chapter.

Algorithm 3 Procedure for handling new CIDs in the received advertisement, msg, from neighbor $router_k$ at $router_i$

procedure HANDLENEWCIDROUTES(msg, $router_k$)
  for $newCID_x$ in $msg.newCIDs$ **do**
    if $newCID_x$ not in $CIDRouteTable_{router_i}$ **then** // hash table lookup
      // $router_i$ has never seen advertisement for $newCID_x$

      // initialize $CIDRouteEntry$
      $CIDRouteEntry.cost \leftarrow msg.distance$
      $CIDRouteEntry.port \leftarrow router_k.port$ // port to neighbor $router_k$
      $CIDRouteEntry.nextHop \leftarrow router_k.HID$

      // hash table insert
      $CIDRouteTable_{router_i}[newCID_x][msg.originatorHID] \leftarrow CIDRouteEntry$

      $setCIDForwardTable_{router_i}(newCID_x, router_k.port, router_k.HID)$
    else
      // $router_i$ has seen advertisement for $newCID_x$ before

      // update the CID routing table if $newCID_x$ is never received from
      // $msg.originatorHID$ or this message contains a shorter route to
      // $newCID_x$ on $msg.originatorHID$
      if $msg.originatorHID$ not in $CIDRouteTable_{router_i}[newCID_x]$
        or $msg.distance$
        $< CIDRouteTable_{router_i}[newCID_x][msg.originatorHID].cost$ **then**
        $CIDRouteEntry.cost \leftarrow msg.distance$
        $CIDRouteEntry.port \leftarrow router_k.port$
        $CIDRouteEntry.nextHop \leftarrow router_k.HID$

        // hash table insert
        $CIDRouteTable_{router_i}[newCID_x][msg.originatorHID]$
          $\leftarrow CIDRouteEntry$

      // check if this message has minimum cost route to $newCID_x$
      $minCost \leftarrow +\infty$
      for $eachRouteToCID$ in $CIDRouteTable_{router_i}[newCID_x]$ **do**
        if $eachRouteToCID.cost < minCost$ **then**
          $minCost \leftarrow eachRouteToCID.cost$

      // if $newCID_x$ in this message is indeed the minimum cost route for $router_i$
      // set it to CID forwarding table
      if $msg.distance = minCost$ **then**
        $setCIDForwardTable_{router_i}(newCID_x, router_k.port, router_k.HID)$

Algorithm 4 Procedure for handling evicted CIDs in the received advertisement, msg at $router_i$

```
procedure DELETEEVICTEDCIDROUTES(msg)
    for delCID_x in msg.delCIDs do
        // hash table lookup for if condition below
        if delCID_x in CIDRouteTable_router_i
                and msg.originatorHID in CIDRouteTable_router_i[delCID_x] then
            // router_i has seen advertisement from msg.originatorHID for delCID_x

            // hash table remove route entry to delCID_x at msg.originatorHID
            erase msg.originatorHID from CIDRouteTable_router_i[delCID_x]

            if size of CIDRouteTable_router_i[delCID_x] is zero then
                // no more CID routes to delCID_x, remove the delCID_x from
                // CID forwarding table

                removeCIDForwardTable_router_i(delCID_x)
            else
                // there are CID routes remaining to reach delCID_x, find out the shortest path
                // to delCID_x among the remaining CID routes.

                minCost ← +∞
                initialize minEntry
                for eachRouteToCID in CIDRouteTable_router_i[delCID_x] do
                    if eachRouteToCID.cost < minCost then
                        minCost ← eachRouteToCID.cost
                        minEntry ← eachRouteToCID

                // if there are remaining routes to delCID_x,
                // set to CID forwarding table the shortest one
                if minCost is not +∞ then
                    setCIDForwardTable_router_i(delCID_x,
                                minEntry.port, minEntry.nextHop)
```

Chapter 3

Interdomain Cache Management

The goal of the interdomain cache management is to allow collective sharing of cache space among a group of peering ISPs in the same region. There are both economic as well as performance benefits for each ISP in the peering group to collaboratively share their cache space. The economic benefit means that instead of fetching content through an ISP's provider, which incurs both in-bound and out-bound traffic cost for the ISP, ISP can fetch the content off-path from caches deployed by one of its peers. Peering relationships are generally "settlement-free". This means that cache sharing would provide them with great economic benefits since peer does not need to pay, or the cost would be low, for transit traffics to other peers.

The second benefit is performance in terms of requests' latency for ISP's customers. Most of these peering relationships are established at the Local Internet Exchange point (IXP) [28] generally located in the same city or state among peers. The advantage of establishing peering relationships via IXP is low latency and high bandwidth since local links are much faster and offer better quality of service. The key reason is that data does not need to traverse the fragile middle-mile links in the wide area Internet that are subject to congestions. Hence, peers that share cache with each other can enjoy this latency benefit at a much lower cost since IXP charges at lower prices, or even free, for data exchange. On the other hand, peers can also establish private peering points by building their own infrastructure if they prefer even better quality of data exchange on peering links. It's more costly, but the performance benefits are higher. Nonetheless, to further illustrate performance as well as economic benefits, consider the following scenarios (see Figure 3.1):

Here the tier 3 regional ISPs (ISP1, ISP2 and ISP3) are the customers of their provider ISP4. They are colocated in the same city or state and are the network access providers for the local businesses, enterprises and home owners. Examples of these providers are XO Communications [13], CenturyLink [26], Fairpoint Communications [12] and etc. ISP4 is a tier 1 or tier 2 transit provider. Examples of it includes Verizon [34], Level 3 [3], Cogent [11] and etc. Content provider could be Netflix [27], Hulu [21] and etc. In this scenario, suppose that all ISPs have built caches on their routers in its domain, and that ISP1, ISP2 and ISP3 are serving their customers' content requests for the content from the same popular content provider [1]. Finally, assume that they maintain peering relationships through the local Internet Exchange Point (IXP),

[1] We use this assumption to simplify the discussion.

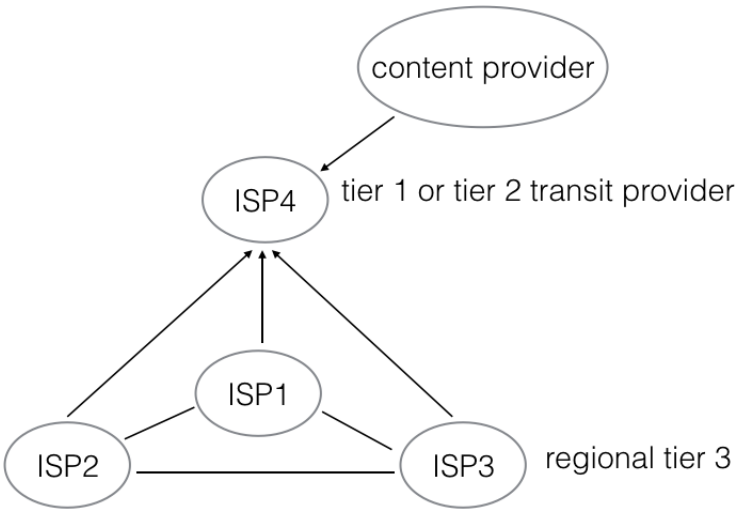

Figure 3.1: An interdomain scenario for illustrating the benefits for interdomain cache management. Arrow means customer pays provider. Straight line means peering relationships

which provides low cost and low latency exchange of traffics between two peers.

Without cache sharing, ISP1, ISP2 and ISP3 serve their customers' requests and utilize their cache space independently. Specifically, if the CID in the request is not found in caches in its own domain, each ISP will use fallback to send it to the content origin. If ISP4 has it cached in its own domain, then the CID response can be sent back immediately. Otherwise, customers in each of these ISPs need to wait a round trip time of delays between these ISPs and content providers in order to get the content response. Moreover, every request sent to the provider ISP4 incurs a cost to ISP1, ISP2 and ISP3.

With the interdomain cache management, ISP1, ISP2 and ISP3 can now collectively share their cache space with each other and eliminate the redundancy in their own respective caches. Upon hearing the advertisements from other peers for the same CIDs cached in its own domain, each peer can make decision to evict the locally cached content or let the peers evict the same CIDs. The saved cache space for each ISP can be used to cache other popular content that would not have been able to fit into the cache had collaborative cache sharing not existed. When customers send to any of these ISPs content requests, these requests can now be forwarded off-path to other peers if the corresponding content is cached collaboratively at other peers. Since these peers are located in the same region connected through IXP, requests can be satisfied with minimum latency and low cost. For these regional ISPs that provide network access for customers, interdomain cache management can increase the quality of their network services and hence attract more customers. It will also save them the cost of fetching content from their providers in the interdomain settings.

3.1 Related Work

There are several existing works that discuss the benefits of cache coordination. In [35], an intradomain cache coordination scheme is proposed to reduce the redundancy in caches in intradomain settings. The problem they address is that CCN en-route default caching scheme creates highly synchronized cache redundancy along the chunk return path. This is especially problematic since cache has limited CPU resources and storage capacity. The solution they propose is that caches in a neighborhood can collaborate to purge the duplicate items through a periodic summary of what is cached on each nearby router. By utilizing this explicit coordination scheme, they were able to increase the hit rate within a domain and reduce the inter-ISP cost since more content can be cached in the same domain.

There are a couple of similarities and differences between [35] and our interdomain cache management scheme. The similarity is that both try to reduce the cost for a domain. In [35], costs are reduced by judiciously removing redundancy in caching so that more caching resources can be allocated to other chunks in the domain. This reduces the inter ISP traffic footprints. Our interdomain cache management also tries to reduce the cost, except that it does so by allowing a group of peering domain to collaboratively share their cache space so that more popular content can be cached among peers to reduce the cost of fetching content through providers.

In [6], authors propose the idea of optimizing user-centric performance (e.g. latency) rather than network centric performance (e.g. cache hit rate) for cache coordination schemes. The argument is that a caching policy that provides best network centric performance might provide very poor user-centric performance. To address this, authors propose the idea of caches that would preferentially retain popular content forwarded over the congested links. The results show that desipite lower cache hit rate, their scheme actually achieves lower average download delay.

The idea of optimizing for bandwidth consumption for cache coordination can be useful. Nonetheless, the peering links in our interdomain cache management scheme have sufficient bandwidth [28]. In addition, the primary goal of our interdomain cache management is to reduce the cost of cache sharing for each peer in the collaboration group.

In [25], the problem authors addresses is also the insufficient hardware resources such as cache capacity for in-network caches. Their goal is to develop a careful object placement scheme with the objectives to reduce the average hop count and inter-ISP bandwidth consumption. The object placement scheme is intradomain. Compared to other coordination schemes as well as default en-route caching, they achieve higher saving rate of hop count and inter-ISP traffics. The problems and objectives in this paper is very similar to [35].

In conclusion, most previous literatures focus on intradomain cache management to address the problem of insufficient caching resources and inefficency in their utilization by existing ICN schemes. They also argue for the benefits of having intelligent intradomain cache coordination schemes that would achieve performance objectives such as maximizing cache hit rate and minimizing users' request latency within a domain. Few addresses the problem of cache coordination in interdomain settings where ISP's cost savings are the primary goal. Nonetheless, the idea of removing redundancy among coordinating caches is still attractive in interdomain settings. It not only provides performance benefits (requests can be routed through high bandwidth and low latency IXP links to nearby peer), but also economic benefits (more content can be cached with larger cache space).

3.2 Design

The key goal of interdomain cache coordination is to design a cache sharing scheme among coordinating peers that aims at minimizing the redundancy in their caches. Hence, the saved cache space can be utilized to cache other popular content that otherwise would not had been cached. More content cached means more cost savings for the ISPs. However, simply caching everything uniquely among peering domains would provide poor availability when the cache resources in peering domains are down and being repaired. In addition, caching content on other peering domains means bandwidth consumption to fetch content from other peers. Ideally, an ISP would want to cache everything locally so it does not need to pay the cost of bandwidth to IXP (even though the cost is much lower than cost to provider) and have negligible latency for its customers (content cached locally can be available immediately to its customers). Therefore, there is a tradeoff between performance (cache hit rate by participating in coordination), availability (cache failures of other coordinating domains), and bandwidth consumption (fraction of requests to other peering domain).

After the discussion on tradeoffs, we will propose an interdomain cache sharing algorithm that each ISP can run in its domain independently. The goal is to allow even division of cache sharing load among peers that incurs minimum coordination overhead through implicit coordination scheme. We will then discuss the application of this algorithm in interdomain settings and its evaluations.

3.2.1 Interdomain Caching Metrics and Model

Before we describe the cache sharing model among interdomain peers, we need to define the metrics for ISPs. For the following discussions, assume that peering ISPs share the same content and serve it for their customers (e.g. popular movies on Netflix). This is mainly to simplify the discussion, since if peering ISPs have nothing in common, it is questionable whether it is beneficial to do any cache sharing at all. In addition, we assume that caching decisions for the coordination group are not persistent. Hence, when a cache fails in a domain, its peers' local cache would cache some of the more popular content on the failed domain after they stabilizes.

Performance

The performance objective is to reduce as much cost as possible for an ISP. It is defined as the total cache hit rate for each ISP in coordination for cache sharing. Higher cache hit rate means lower cost for the ISP since more requests from ISP's customers can be satified through its peers rather than its providers. Hence, the strategy to maximize performance would be to maximize the hit rate through coordination. Note that this hit rate is maximized when every ISP caches content uniquely and provides local routes for other peers.

Availability

Availability metric is defined as the cache hit rate through coordination for each peering ISP under failures. Since no money is involved, the quality of caching service offered by peers in

general would be lower compared to the services offered by ISP's provider. Hence, periods of unavailability for caches on peers are more frequent. When other coordinating peers fail, an ISP can no longer access the cached content collaboratively shared on those peers and would have additional cost to fetch them from its content providers for content cached on those failed peers. This is also not ideal for client's content request latency before domain's cache stabilizes. Therefore, higher availability means that the cache hit rate through coordination would be higher when a fixed number of failures from coordinating peers occur. This implies that the ISP would cache popular shared contents even if those contents are already cached on its peers.

Bandwidth

Each request satisfied through peering link introduces a small cost of bandwidth for ISP. Hence, the bandwidth cost metric is defined as the fraction of total requests to other coordinating peers, excluding those that can be satisfied locally at the ISP. The higher the fraction of total requests satisfied through coordination, the higher the cost of bandwidth for the ISP. In our model, we assume that each request for cached content on coordinating peers would always result in a cache hit. For example, 50% of total requests through ISP's coordination to other peers also means 50% cache hit rate through coordination for the same ISP. Therefore, bandwidth consumption is proportional to the total cache hit rate to other coordinating peers. Each peer ideally would want to cache everything locally to minimize bandwidth consumption since its customers' requests do not need to traverse through peering links.

Fairness

Fairness is defined as whether cache hit rate is evenly distributed among coordinating peers. A coordination group of 4 peering domain means each domain would ideally serve 25% of the total requests for the group. The cache sharing decision should achieve fairness among the peers since if a single peer provides significant percentage of hit rate for other peers, it would effectively become a content provider and hence would want to charge for its services.

Summary

In summary, there is clearly a tradeoff between performance and availability, and between performance and bandwidth. The performance objective is to replicate as few content as possible on each domain so that more content can be cached among peers to increase hit rate through cache coordination. On the other hand, for bandwidth and availability the objective is to replicate as much content locally as possible in a domain since more requests can be satisfied locally in case of cache failures on peering domains and high bandwidth cost. Moreover, it is also important that each peering domain in coordination shares the cache space fairly so that no one is serving significantly more traffics for others. We first study these tradeoffs, and then will come back to the fairness issues in section 3.4. Consider the following caching model:

In the model, we are considering N peers cordinating with each other, forming a complete graph (see Figure 3.2 for an example when $N = 5$). Each domain has a single cache for cache sharing. The key parameter in balancing the tradeoff mentioned above is the number of top

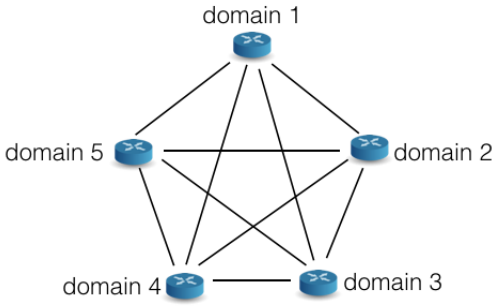

Figure 3.2: Interdomain cache sharing Example. Each domain has a cache that will share with its peers. They form a complete graph configuration since peer do not carry traffics between two other peers.

popular content replicated on each router. For instance, when the number top popular content replicated is 0, each router (r1 to r5) all caches content uniquely, maximizing the performance. When the number of replicated content is the size of cache (assuem each peer has the same cache size), availability is maximized and bandwidth consumption is minimized since requests do not need to go off-path to other peers. Hence, by varying this parameter, we can get an estimate of how to choose a design point in this tradeoff.

3.3 Tradeoffs

We begin with variable definitions:

Variable	Definitions
x	number of top popular replicated in each peering domain
R	number of coordinating peers
S	size of cache for each peer
L	total number of content objects shared by coordinating peers. Note that not all objects can fit into the total size of the cache over all peers.
$zipf(i)$	fraction of requests and hit rate for ith content objects . Generated from L number of content objects shared by coordinating peers. This is also the hit rate since we assume that every request to non-failed caches would result in a cache hit
F	number of failed caches among peering domains

3.3.1 Performance versus Availability tradeoff

The formula to calculate the hit rate for a single domain by participating in the coordination under different number of cache failures of peering domains is

$$\sum_{i=1}^{x} zipf(i) + \frac{R-F}{R} \sum_{i=x+1}^{x+RS-Rx} zipf(i)$$

The first sum calculates the cache hit rate of the replicated content on ISP's own cache. The second sum first calculate the sum of the rest of the hit rate by cache sharing, excluding those hit rates through replication. To ensure fairness, we are assuming that the rest of the hit rate are distributed evenly among coordinating peers. Hence, there is a factor of $\frac{R-F}{R}$ for the second sum when failure occurs.

3.3.2 Performance versus Bandwidth tradeoff

The formula to calculate the fraction of content requests (or cache hit rate) to all other coordinating peers is

$$\frac{R-1}{R} \sum_{i=x+1}^{x+RS-Rx} zipf(i)$$

Using the same method as before, here we are assuming that request ratios are distributed evenly among coordinating peers. Hence there is a factor of $\frac{R-1}{R}$ at the beginning of the sum.

3.3.3 Frequency of failures

It is also important to have a notion of how frequent the cache failure is for coordinating domains. The more frequent the failure, the better it is to replicate content locally for an ISP. Hence we begin by some definitions.

Variable	Definitions
$MTBF$	Mean time between cache failure for coordinating domain
$MTTR$	Mean time to repair cache failures for coordinating domain

Probability that caches on all coordinating domains fail

$$\frac{MTBF}{MTBF+MTTR}(\frac{MTTR}{MTBF+MTTR})^{R-1}$$

Probability that cache on any coordinating domain fails

$$\frac{MTBF}{MTBF+MTTR}(1-(\frac{MTBF}{MTBF+MTTR})^{R-1})$$

Probability that there are exactly k caches among coordinating domains that fails at the same time

This can be modeled with the binomial distribution, we have

$$\frac{MTBF}{MTTR+MTBF}\binom{R-1}{k}(\frac{MTTR}{MTTR+MTBF})^{k}(\frac{MTBF}{MTTR+MTBF})^{R-1-k}$$

3.3.4 Results

We first evaluate the performance versus availability tradeoff. The x-axis is the number of top popular content replicated on each domain. The y-axis is the cache hit rate through coordination for each domain. Finally, we plot out how this hit rate changes as the number of cache failures on coordinating domains increases (see Figure 3.3). The parameters we used are: 5 peering domains, 1000 CIDs cache size on each peer, and a total of 10000 CIDs. We use these numbers since most tier 3 peering domains are not serving content for a large number of users such as those served by popular ISP (Verizon, Comcast etc). In addition, the number of tier-3 ISP providers that peer with each other in the same region is generally small.

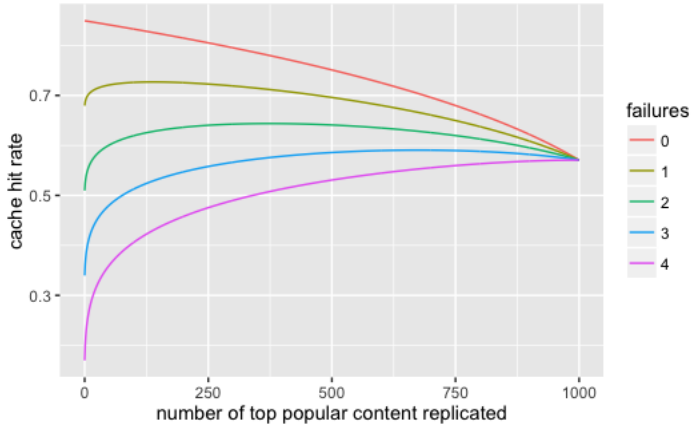

Figure 3.3: Performance versus availability tradeoff for peering ISPs to share the cache space

As the number of cache failures increases, the hit rate for each coordinating peer decreases since each failure means a decrease in the shared cache size, which in turn means there are less number of unique content cached among peers. When everyone replicates fully without coordination (x axis is 1000), the hit rate for all failures is the same since a peer's hit rate in this case is independent of other domains' cache failures. The plot for failure > 0 shows an increase in hit rate when x is small. This is because there are a few very popular content on the failed caches for a peer. When this cache fails, other domains can no longer access the shared popular content from that domain. If each peer chooses to replicate several top popular content, the amount of hit rate gained when other peers fail is greater than the amount of hit rate lost by not caching more unpopular content through coordination. This can be seen from the case when caches on every other coordinating domains fail. In this case, the remaining peer would always be better off replicating since failed caches have most popular content.

This implies that the degree of replication for popular content is different based on how often each peer sees cache failures from other peering domains. When other peers fail frequently, a better strategy would be to increase the number of top popular content replicated. The extreme case is to cache content independently of other peers if they fail too often. When other peers fail infrequently, peer can reduce the number of local replication for popular content. To understand this, we need to know how often is the cache failure for peers. Using simple probability method as described in Section 3.3.3, we have the following results.

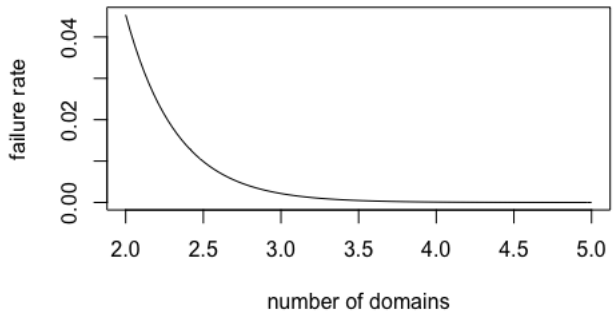

Figure 3.4: Probability that caches on all coordinating domains fail.

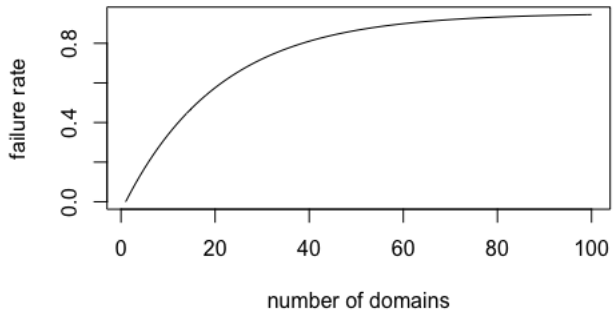

Figure 3.5: Probability that cache on any coordinating domain fails.

The parameters we used are: MTTR for cache in peer is 20 hours, and MTBF is 400 hours. We pick these numbers to provide an upperbound on the estimate of the failure rate (In fact, Cisco's 800 series Integrated service router has MTBF above 200000 hours [1]). We assume that failure on each cache happens independently of each other. We calculate the proportion of time that caches on all coordinating domains fail as we increase the number of coordinating domains (Figure 3.4). Then we plot out the porportion of time that cache on any coordinating domain fails as we increases the number of domains (Figure 3.5). Two graphs above suggest that the

case where caches on every coordinating domain fail happen quite infrequently. On the other hand, the probability that caches on any coordinating domain fail increases as we increase the number of coordinating domains. Nonetheless, this probability is quite small when this number is small (when $R < 5$).

We also plot out the probability that there are exactly k simultaneous cache failures among coordinating domains as the number of coordinating domains increases using the binomial model (Figure 3.6).

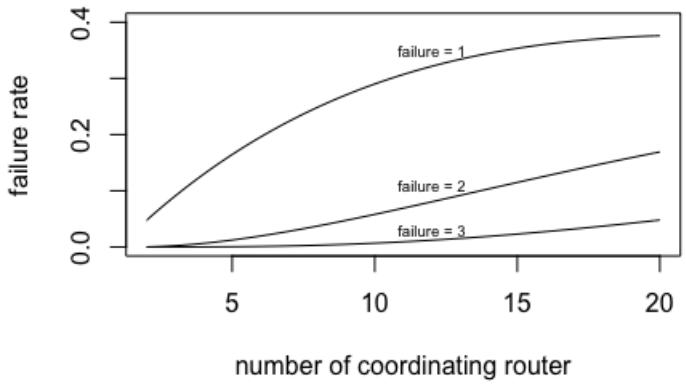

Figure 3.6: Probability for exactly k cache failures among all coordinating domains.

The above graph shows the fraction of time there are exactly 1, 2 and 3 simultaneous failures as the number of coordinating domains increases. When the number of coordinating domains is small (< 10), failure of one cache happens the most likely. Therefore, replicating a few top popular content (as shown in Figure 3.3) can achieve good balance between performance and availability.

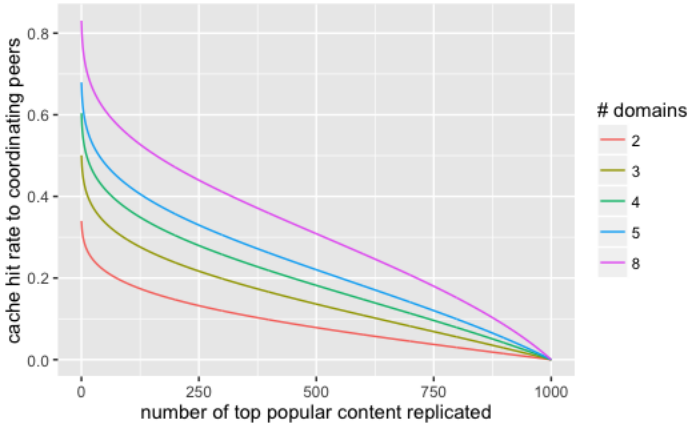

Figure 3.7: Performance versus bandwidth tradeoff for peering ISPs to share the cache space

Replication of popular content can also reduce the bandwidth cost for a domain. We plot out the fraction of the content requests (or cache hit rate) to the other coordinating domains as we increases the number of replication of top popular content (see Figure 3.7). Here the cache size is also 1000 CIDs and there are a total of 10000 CIDs shared among peers. The values of these parameters are the same for the performance versus availabiliy model. Compared to figure 3.3, this also shows that replicating the top most popular content can also significantly reduce the bandwidth consumption, while minimally affects the hit rate among the coordinating domains.

3.3.5 Discussion

In summary, what we have shown using the analytical model is that replicating the top few popular content can be useful strategy to achieve good performance, availability and bandwidth consumption for a peering ISP. From the probabilistic method, we know that when the number of peers in coordination is small (< 10), the failure of 1 peer happens the most frequently (Figure 3.4, 3.5, 3.6). Replicating the top few popular content does not significantly impact the hit rate under 1 failure (Figure 3.3). In addition, it also significantly reduces the bandwidth consumption for each peer (Figure 3.7).

3.4 Cache Sharing Algorithm

The cache sharing algorithm aims at achieving one goal: to collaboratively remove the redundant content cached among peers and decide who should cache the content. Ideally, it shouldn't introduce coordination cost (e.g. bandwidth consumption) due to the scale of CIDs in each domain. In addition, it should achieve fairness in the caching decision to ensure that no single peering ISP serves significantly more requests for other peers. There are a couple of approach to this. The first approach is to use explicit coordination scheme where decision for who should cache the content can be made by peers exchanging coordination messages. However, a couple of drawback to this approach is that 1) the signaling overhead can be high due to the scale of the content cached in each domain, and 2) there is a delay in the consensus since coordination message needs to first be received by each peer and then a consensus protocol needs to be run to determine who should cache what. To address this challenge, we uses implicit coordination scheme, which does not require any exchange of coordination messages between peers. In addition, since no coordination message exchange is needed, routers in each domain do not need to run consensus protocol that can introduce delay in the cache sharing decision. On the other hand, the drawback of using implicit coordination scheme is that peers need to place trust to each other since the caching decision is done without explicit agreement through consensus. A free riding problem can occur if a peer chooses to not cache any content for others. Nonetheless, it can be prevented since a peering ISP can monitor the traffic imbalance. If the imbalance is significant, an ISP can withdraw some of its CID routes to other peering domains, or opt out of the peering relationships in the worst case.

Implicit coordination scheme is beneficial since it does not incur any coordination cost for bandwidth on peering links. It can be achieved through the use of a deterministic functions shared by all peers that takes as input a list of domain names and CID of content shared by do-

main. Then it would deterministically output which domain should cache the content identified by the CID. The key design goal is to ensure that the cache sharing decision should be random among a set of domains for each CID so that no one domain serves significantly more requests for the group than others. To justify why random distribution strategy works well to distribute requests load evenly among peers, we have the following theorem

Theorem 3.4.1. *A random distribution strategy where each content is cached on only one single domain chosen randomly among n peering domains can achieve $\frac{1}{n}$ fraction of total requests loads for each domain in expectation.*

Proof. For each domain, D_i, we can first express the expectation of the fraction of requests it receives from random distribution strategy. Define Y_{D_i} to be the total fraction of requests received by D_i. Define X_{k,D_i} define the indicator that equals 1 if content k from the shared set of content K within the coordination group is cached in D_i. $zipf(k)$ is the fraction of requests for content k. Hence we have

$$Y_{D_i} = \sum_{k \in K} zipf(k) X_{k,D_i}$$

It's expectation is just

$$\begin{aligned} E[Y_{D_i}] &= E[\sum_{k \in K} zipf(k) X_{k,D_i}] \\ &= \sum_{k \in K} zipf(k) E[X_{k,D_i}] = \sum_{k \in K} zipf(k) Pr[\text{k is cached on } D_i] \\ &= \frac{1}{n} \sum_{k \in K} zipf(k) \end{aligned}$$

The above argument follows from the linearality of expectation and the fact that $E[X_{k,D_i}] = Pr[\text{k is cached on } D_i] = \frac{1}{n}$ since a random domain is chosen among n domains that share the set of content K. □

Hence, random cache sharing strategy works well to distribute load evenly among domains. In addition, since replicating the top few popular content is beneficial from the tradeoff analysis in previous section, each peering cache in the coordination group can maintain a locally defined threshold K, which defines the number of top popular content cached locally not involving in the coordination protocol. Then, for the rest of the popular content cached locally, each peer can run the cache sharing algorithm presented below (see Algorithm 5).

Line 7 to line 9 in the above algorithms ensures that domains that cache the same CIDs can arrive at a deterministic decision as to who should cache which CIDs. This deterministic property is enabled by sorting the list of names of domains that have the CID cached locally. Hence each domain would eventually get the same list in the same order. The randomness in the decision is ensured by computing the hash value of the CID_k, which output integer that approximates randomness since the CID hash itself approximates a random hash. It uses this integer to compute which domain to cache the CID. Finally, line 10 checks if the domain to cache the CID is not itself, it will evict the CID.

Algorithm 5 Cache Management Algorithm for $domain_i$

```
1:  procedure FINDCIDSTOEVICT
2:      initialize result to be an empty array
3:      for each CID_k cached locally at domain_i do
4:          if CID_k has popularity below threshold K at domain_i then
5:              if CID_k is also cached locally at other peering domains then
6:                  allDomainsWithCID ← peers that also cache CID_k ∪ domain_i
7:                  allDomainsWithCID.sort()
8:                  domainToCacheIdx ← hash(CID_k) % allDomainsWithCID.size()
9:                  domainToCache ← allDomainsWithCID[domainToCacheIdx]
10:                 if domain_i != domainToCache then
11:                     append CID_k to result
        return result
```

This algorithm can be run periodically on each coordinating peer. Within each period, we want that the decision output by the algorithm to be persistent in that period to ensure cache sharing decisions are correctly preserved. On the other hand, when domains have cache failures, this algorithm can be rerun among the remaining peers in the coordination group.

3.4.1 Cache Management Algorithm in the Interdomain settings

The cache management algorithm primarily works among the caches that are peer with each other in the interdomain settings. Nonetheless, provider of an ISP can benefit from the above algorithm by coordinating with its customers to avoid caching the same content so that it can more effectively utilize its own cache space. It saves for provider ISP the cost of provisioning additional caching resources and also provides better services for its customers since the additional cache space gained by running the coordination algorithm can be used to cache other popular content for its customers. For instance, in an interdomain network, a customer AS would always advertise to its providers, peers and its own customers. Provider of an AS, after hearing this advertisement, can run the same cache management algorithm above. On the other hand, it is questionable whether customers would want to coordinate with its providers since they need to pay providers for every traffic that goes through the provider. Consider the case where customer is deciding whether to evict the content since it is cached at its providers. If it does evict the chunk, then every access to the content now need to go through the provider, which incurs a data transfer cost for the customer. From the economic incentives standpoint, this seems not so realistic.

3.4.2 Evaluation

We evaluate the cache sharing algorithm for its ability to distribute content requests evenly among a set of peers with analytical model. To do so, we compare the cache hit rate among shared content on each peer between decisions output by our algorithm and those output by an oracle. To find out the oracle, we know that this problem is essentially a partition problem [4] where the

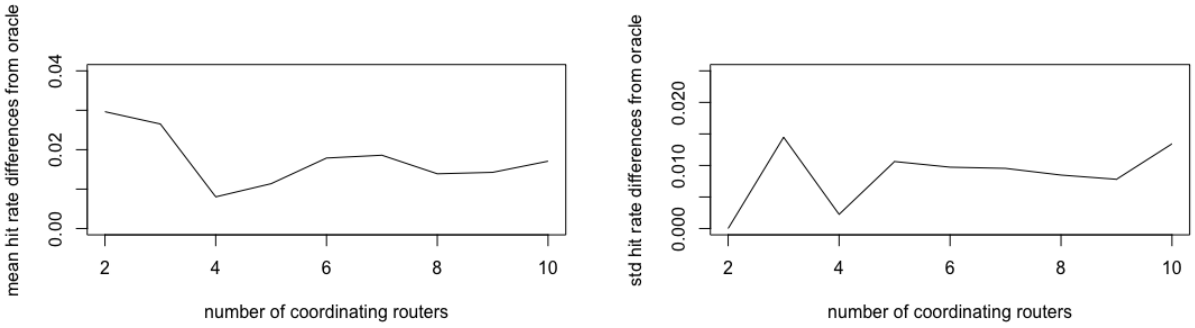

Figure 3.8: Plots of mean and standard deviation of cache hit rate differences between our approach and oracle as number of coordinating router increases (number of shared content is 10000)

goal is to decide if a set can be partitioned into multiple subsets such that each subset produce equal sum (in our case, equal fraction of requests). This problem is known to be NP-hard, but we can use a greedy procedure that achieves good approximation to optimum solution, which iterates through numbers in descending order and assign each number to whichever subset that has the smaller sum [17]. We implement this greedy approach and compare it against our solution. Note that in reality this oracle approach requires explicit coordiantion overhead among peers, which would be costly for network bandwidth on peering links given the scale of CIDs among peering domains.

In our analytical model, we assume that customers for all ISPs are accessing the same set of content following the same popularity distribution. We do so primarily to understand the effectiveness of our algorithm and the practicality of cache sharing in interdomain settings since if all coordinating ISPs' customers have requests for different content, it is questionable whether there is any benefit to do cache sharing at all. Then for both oracle and our approach, we will assign to each coordinating domain a CID as well as fraction of requests for that CID generated using its popularity distribution. The CID is then assigned to a domain with the least total fraction of requests so far. In practice, implementing this strategy requires domains to explicitly coordinate CID placement decisions. On the other hand, for our cache sharing algorithm, domains use the shared hash function to compute random placement of CIDs within coordination group. In the end, each domain has a list of CIDs assigned and total fraction of requests from these CIDs.

To see the differences for the cache placement decisions, we plot out the mean and standard deviation of differences in cache hit rate among shared contents on each peer between our approach and the oracle. Note that the cache hit rate here is the same as the fraction of requests since each request would result in a cache hit in our settings. To see how these data change, we plot out the results as the number of coordinating domain and as the number of shared content increases. The results are shown in Figure 3.8 and Figure 3.9.

As shown from the graph, the mean cache hit rate differences between our cache sharing algorithm and oracle in different configurations of cache network is quite small in general. The

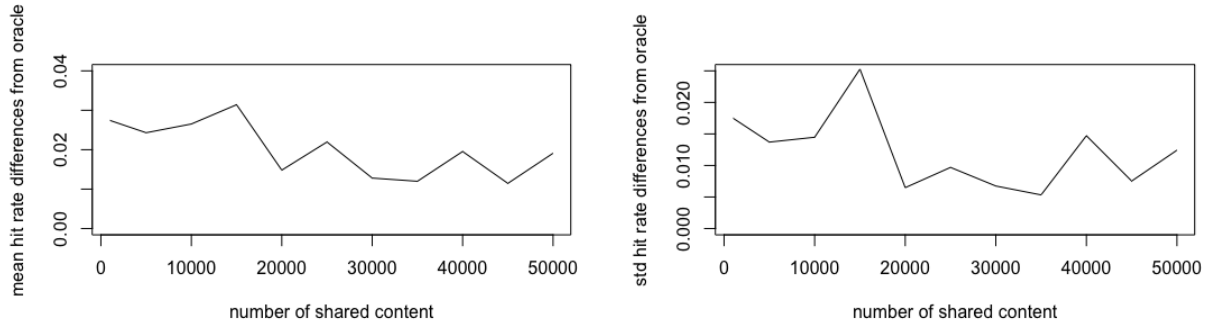

Figure 3.9: Plots of mean and standard deviation of cache hit rate differences between our approach and oracle as number of content increases (number of coordinating routers is 3)

same goes for standard deviation. The small differences are due to the fact that CID hashes are not perfectly random. Hence some domains might occationally be assigned with more popular contents. Also, there is no direct relationship between this difference and the cache configurations since our approach essentially assigns a random domain for each unique CID, which is independent from the number of content or coordinating domains. The above results show that our approach can approximate the oracle in assigning the cached content evenly to coordinating peers, with the benefits of not incuring any coordination message exchange.

3.5 Summary

In summary, this section discusses an interdomain cache management scheme in terms of the tradeoff between performance, availability and bandwidth. We found out that by replicating a few top popular content across the domain, a good degree of availability and reduction of bandwidth consumption can be achieved, without sacrificing significantly the performance. Finally, we propose, based on the discussion on these tradeoffs, a cache management algorithm that uses implicit coordination scheme to achieve consensus among coordinating domains without signaling overhead. We discuss its application in the interdomain settings and evaluate it against the oracle cache sharing algorithms. The results show that our algorithm can closely approximate the oracle to evenly share the cache load among coordinating peers.

Chapter 4

Conclusion and Future Work

In conclusion, this thesis presents content delivery optimization techniques for the Future Internet Architecture. There are several components considered: 1) design and evaluation of an intradomain CID routing protocol that achieves low latency performance goal for content requests and is scalable and more efficient compared to distance vector and link state, 2) design and investigation on interdomain cache management scheme that discusses various tradeoff of cache sharing among peers in interdomain settings, and 3) design and evaluation of an interdomain cache sharing algorithm that balance cache load evenly among peers without coordination overhead.

4.1 Limitations

The limitations of this thesis is in evaluation of CID routing. It should be simulated with larger and more realistic network topology and workload to fully understand various properties proposed.

4.2 Future Work

The future work includes completing the detailed evaluation on CID routing and interdomain cache management schemes using realistic workload and larger network topology. It would also be interesting to investigate the interactions between caching policies and the effects of cache churn on the CID routing. It would be interesting to investigate what are the best routing strategies for different caching policy to handle high cache churn.

In addition, it would be useful to see the full system of interdomain cache management built to better understand various properties proposed and discussed (for instance various tradeoffs discussed in chapter 3)

Bibliography

[1] Cisco 800 series integrated services router. URL http://www.cisco.com/c/en/us/td/docs/routers/access/800/hardware/installation/guide/800HIG/appendix.html. 3.3.4

[2] Netlogic nl9000 family. URL http://www.netlogicmicro.com/Products/Layer4/index.asp. 2.3.1

[3] Level 3. Level 3. URL http://www.level3.com. 2.1.1, 3

[4] Partition Algorithm. Partition algorithm. URL https://en.wikipedia.org/wiki/Partition_problem. 3.4.2

[5] Ashok Anand, Chitra Muthukrishnan, Aditya Akella, and Ramachandran Ramjee. Redundancy in network traffic: Findings and implications. In *Proceedings of the Eleventh International Joint Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '09, pages 37–48, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-511-6. doi: 10.1145/1555349.1555355. URL http://doi.acm.org/10.1145/1555349.1555355.

[6] Mikhail Badov, Anand Seetharam, Jim Kurose, Victor Firoiu, and Soumendra Nanda. Congestion-aware caching and search in information-centric networks. In *Proceedings of the 1st ACM Conference on Information-Centric Networking*, ACM-ICN '14, pages 37–46, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-3206-4. doi: 10.1145/2660129.2660145. URL http://doi.acm.org/10.1145/2660129.2660145. 3.1

[7] L. Breslau, Pei Cao, Li Fan, G. Phillips, and S. Shenker. Web caching and zipf-like distributions: evidence and implications. In *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 126–134 vol.1, Mar 1999. doi: 10.1109/INFCOM.1999.749260.

[8] Matthew Caesar, Tyson Condie, Jayanthkumar Kannan, Karthik Lakshminarayanan, Ion Stoica, and Scott Shenker. Rofl: Routing on flat labels. In *Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '06, pages 363–374, New York, NY, USA, 2006. ACM. ISBN 1-59593-308-5. doi: 10.1145/1159913.1159955. URL http://doi.acm.org/10.1145/1159913.1159955. 2.2.2

[9] Hong-Tai Chou and David J. DeWitt. An evaluation of buffer management strategies for relational database systems. In *Proceedings of the 11th International Conference on Very Large Data Bases - Volume 11*, VLDB '85, pages 127–141. VLDB Endowment, 1985. URL

http://dl.acm.org/citation.cfm?id=1286760.1286772.

[10] Cisco. White paper: Cisco vni forecast and methodology, 2015-2020. URL http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html. 2.1.1

[11] Cogent Communications. Cogent communications, . URL http://cogentco.com. 2.1.1, 3

[12] Fair Point Communications. Fair point communications, . URL http://fairpoint-hsi.com. 3

[13] XO Communications. Xo communications, . URL https://www.xo.com. 2.1.1, 3

[14] DASH. Dynamic adaptive streaming over http. URL https://en.wikipedia.org/wiki/Dynamic_Adaptive_Streaming_over_HTTP. 2.3.1

[15] Nick Feamster, Jennifer Rexford, and Ellen Zegura. The road to sdn: An intellectual history of programmable networks. *SIGCOMM Comput. Commun. Rev.*, 44(2):87–98, April 2014. ISSN 0146-4833. doi: 10.1145/2602204.2602219. URL http://doi.acm.org/10.1145/2602204.2602219.

[16] Center for Applied Internet Data Analysis. Internet topology at router- and as-levels, and the dual router+as internet topology generator. URL https://www.caida.org/research/topology/generator/. 2.3.1, 2.4.1

[17] R. L. Graham. Bounds on multiprocessing timing anomalies. *SIAM JOURNAL ON APPLIED MATHEMATICS*, 17(2):416–429, 1969. 3.4.2

[18] Albert Greenberg, Gisli Hjalmtysson, David A. Maltz, Andy Myers, Jennifer Rexford, Geoffrey Xie, Hong Yan, Jibin Zhan, and Hui Zhang. A clean slate 4d approach to network control and management. *SIGCOMM Comput. Commun. Rev.*, 35(5):41–54, October 2005. ISSN 0146-4833. doi: 10.1145/1096536.1096541. URL http://doi.acm.org/10.1145/1096536.1096541.

[19] Mark Gritter and David R. Cheriton. An architecture for content routing support in the internet. In *Proceedings of the 3rd Conference on USENIX Symposium on Internet Technologies and Systems - Volume 3*, USITS'01, pages 4–4, Berkeley, CA, USA, 2001. USENIX Association. URL http://dl.acm.org/citation.cfm?id=1251440.1251444. 2.2.2

[20] Dongsu Han, Ashok Anand, Fahad Dogar, Boyan Li, Hyeontaek Lim, Michel Machado, Arvind Mukundan, Wenfei Wu, Aditya Akella, David G. Andersen, John W. Byers, Srinivasan Seshan, and Peter Steenkiste. Xia: Efficient support for evolvable internetworking. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, NSDI'12, pages 23–23, Berkeley, CA, USA, 2012. USENIX Association. URL http://dl.acm.org/citation.cfm?id=2228298.2228330. 1, 2.3.1, 2.3.1

[21] Hulu. Hulu. URL https://www.hulu.com. 2.1.1, 3

[22] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs, and Rebecca L. Braynard. Networking named content. In *Proceedings of the*

5th International Conference on Emerging Networking Experiments and Technologies, CoNEXT '09, pages 1–12, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-636-6. doi: 10.1145/1658939.1658941. URL http://doi.acm.org/10.1145/1658939.1658941. 1

[23] Teemu Koponen, Mohit Chawla, Byung-Gon Chun, Andrey Ermolinskiy, Kye Hyun Kim, Scott Shenker, and Ion Stoica. A data-oriented (and beyond) network architecture. In *Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '07, pages 181–192, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-713-1. doi: 10.1145/1282380.1282402. URL http://doi.acm.org/10.1145/1282380.1282402. 2.2.2

[24] M. Lee, K. Cho, K. Park, T. Kwon, and Y. Choi. Scan: Scalable content routing for content-aware networking. In *2011 IEEE International Conference on Communications (ICC)*, pages 1–5, June 2011. doi: 10.1109/icc.2011.5963381. 2.2.3

[25] Jun Li, Hao Wu, Bin Liu, Jianyuan Lu, Yi Wang, Xin Wang, YanYong Zhang, and Lijun Dong. Popularity-driven coordinated caching in named data networking. In *Proceedings of the Eighth ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, ANCS '12, pages 15–26, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1685-9. doi: 10.1145/2396556.2396561. URL http://doi.acm.org/10.1145/2396556.2396561. 3.1

[26] Century Link. Century link. URL http://www.centurylink.com. 2.1.1, 3

[27] Netflix. Netflix. URL https://www.netflix.com. 2.1.1, 3

[28] Local Internet Exchange Point. Local internet exchange point. URL https://en.wikipedia.org/wiki/Internet_exchange_point. 3, 3.1

[29] Lorenzo Saino, Ioannis Psaras, and George Pavlou. Hash-routing schemes for information centric networking. In *Proceedings of the 3rd ACM SIGCOMM Workshop on Information-centric Networking*, ICN '13, pages 27–32, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2179-2. doi: 10.1145/2491224.2491232. URL http://doi.acm.org/10.1145/2491224.2491232. 2.2.3

[30] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. *ACM Trans. Comput. Syst.*, 2(4):277–288, November 1984. ISSN 0734-2071. doi: 10.1145/357401.357402. URL http://doi.acm.org/10.1145/357401.357402. 1

[31] Link State. Link state. URL https://en.wikipedia.org/wiki/Link-state_routing_protocol. 2.2.1

[32] Yi Sun, Seyed Kaveh Fayaz, Yang Guo, Vyas Sekar, Yun Jin, Mohamed Ali Kaafar, and Steve Uhlig. Trace-driven analysis of icn caching algorithms on video-on-demand workloads. In *Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies*, CoNEXT '14, pages 363–376, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-3279-8. doi: 10.1145/2674005.2675003. URL http://doi.acm.org/10.1145/2674005.2675003. 2.3.1, 2.4.1

[33] Distance Vector. Distance vector. URL https://en.wikipedia.org/wiki/

Distance-vector_routing_protocol. 2.2.1

[34] Verizon. Verizon. URL http://www.verizon.com. 3

[35] Jason Min Wang, Jun Zhang, and Brahim Bensaou. Intra-as cooperative caching for content-centric networks. In *Proceedings of the 3rd ACM SIGCOMM Workshop on Information-centric Networking*, ICN '13, pages 61–66, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2179-2. doi: 10.1145/2491224.2491234. URL http://doi.acm.org/10.1145/2491224.2491234. 3.1

[36] Dong Zhou, Bin Fan, Hyeontaek Lim, Michael Kaminsky, and David G. Andersen. Scalable, high performance ethernet forwarding with cuckooswitch. In *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies*, CoNEXT '13, pages 97–108, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2101-3. doi: 10.1145/2535372.2535379. URL http://doi.acm.org/10.1145/2535372.2535379. 2.3.1