5-2013

Economic Incentives in Content-Centric Networking: Implications for Protocol Design and Public Policy

Patrick Kwadwo Agyapong

Economic Incentives in Content-Centric Networking: Implications for Protocol Design and Public Policy

Submitted in partial fulfillment of the requirements for

the degree of

Doctor of Philosophy

in

Engineering and Public Policy

Patrick Kwadwo Agyapong

B.S., Electrical Engineering and Computer Science, Jacobs University Bremen
M.S., Electrical Engineering, Jacobs University Bremen
M.S., Engineering and Public Policy, Carnegie Mellon University

Carnegie Mellon University
Pittsburgh, PA

May 2013

For my grandmother (Ante Maggie) and my parents

Abstract

Content-centric networking (CCN) has emerged as a dominant paradigm for future Internet architecture design due to its efficient support for content dissemination, which currently dominates Internet use. This dissertation shows how economic and social welfare analysis can be used to inform the design of a CCN architecture that provides network stakeholders with incentives to deploy and use.

Firstly, the dissertation investigates the economic incentives of different stakeholders to deploy content-centric network Internet architectures and shows that network operators will fail to deploy sufficient storage infrastructure to support CCN without payment flows from publishers. However, the level of payment required differs for different network players, which gives them different competitive advantages in providing storage infrastructure and content delivery services.

Secondly, it evaluates the social welfare implications of different storage deployment scenarios in a CCN-based architecture and identifies two deployments that maximize social welfare. In the first, edge networks provide the storage infrastructure through a transaction broker. In the second, edge networks pay third-parties an amount, equivalent to the realized benefits from a storage node, to deploy storage infrastructure in the network. All other deployment scenarios lead to a deadweight loss.

Thirdly, the dissertation identifies content delivery functionalities that break in a CCN-based architecture and shows how these functionalities can be successfully replicated and enhanced by a careful design of the structure of routable content, content naming and the meta-information added to content. The proposed design supports several content delivery applications and can be easily extended to other networking principals.

Finally, the dissertation identifies and discusses threats in the CCN content delivery model and proposes some mechanisms to address these threats. In addition, the dissertation identifies some policy implications of the CCN content delivery model and proposes some policy interventions that may lead to desirable deployment outcomes.

Acknowledgments

This thesis would not have been possible without the financial support of the Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) through the CMU-Portugal Program under award SFRH/BD/33507/2008 and the National Science Foundation (NSF) under award number CNS-1040801. I am very grateful to these two institutions for supporting my work.

In addition, I sincerely thank my advisor, Professor Marvin Sirbu, for being always available, for encouraging me through tough times and for challenging me to go beyond my comfort zone. I would also like to thank my co-advisor, Professor Rui Prior, for the time he dedicated to my research and for being available whenever I needed his advice and guidance. The insightful discussions between the three of us greatly contributed to the direction of the research presented in this thesis. Furthermore, I am grateful to the members of my thesis committee, namely Professor Jon Peha, Professor Pedro Ferreira, Professor Luis M. Correia and Professor José Rui Figueira for their support.

I also owe a debt of gratitude to all the people who made my regular travels between Pittsburgh and Lisbon bearable. First, Ana Mateus and Vicki Finney partially insulated me from the required administrative bureaucracies on both sides of the Atlantic, which enabled me to dedicate more time to my research. Second, all the wonderful friends I made in Lisbon and Pittsburgh helped me to feel at home, regardless of where I was located. Thank you Tony, Francis, Apurba, Stefan, Kelly, Rebecca, Brandon, Sarah, Maria João, Alex and all the people I met at Carnegie Mellon and Instituto Superio Técnico.

Finally, I would like to thank my family for their love, support and encouragement. I especially would like to thank my dear Ivana, for her understanding, patience, encouragement and unflinching support throughout this period.

Contents

1 Introduction **1**

1.1 Background . . . 4
1.1.1 Addressing schemes . . . 5
1.1.2 Middleboxes . . . 8
1.1.3 Overlay networks . . . 10
1.2 Future Internet architectures . . . 15
1.3 Content-centric networking . . . 19
1.4 Contributions . . . 24

2 Economic Incentives in Content-Centric Networking **27**

2.1 Introduction . . . 27
2.2 Model and assumptions . . . 29
2.3 Economic incentives to provide content delivery . . . 36
2.3.1 Publisher . . . 37
2.3.2 Content distribution network (CDN) . . . 39
2.3.3 Transit network . . . 41
2.3.4 Eyeball network . . . 44
2.4 Economic viability of cache service provisioning . . . 48
2.4.1 Requests originate mainly from a single geographic area served by a single eyeball network . . . 48
2.4.2 Requests originate uniformly from multiple geographic areas served by different eyeball networks . . . 52
2.5 Discussion . . . 56
2.6 Lessons for CCN architecture and protocol design . . . 60
2.6.1 Accounting, reporting and payment mechanisms . . . 60
2.6.2 Cache hit ratios . . . 61
2.7 Conclusion . . . 63
A Appendix . . . 65
A.1 Notation . . . 65
A.2 Publisher's incentives to pay for caching services . . . 67
A.3 Eyeball network's incentives to offer caching services . . . 69
A.4 Transit network's incentives to offer caching services . . . 73
A.5 Content distribution network's incentives to offer caching services . . . 75

3 Social Welfare Implications of Different Cache Deployment Scenarios 77
3.1 Introduction . . . 77
3.2 Social welfare implications . . . 79
3.2.1 Computing surplus, social welfare and deadweight loss . . . 79
3.2.2 Eyeball network transparent caching . . . 83
3.2.3 Eyeball network commercial caching . . . 85
3.2.4 CDN pays eyeball network to co-locate caches . . . 88
3.2.5 Eyeball network pays CDN to deploy caching . . . 90
3.3 Discussion . . . 91
3.4 Conclusion . . . 93

4 Chunk Design to Support a CCN Content Delivery Ecosystem 95
4.1 Introduction . . . 95
4.2 The TCP/IP content delivery model . . . 98
4.2.1 Content preparation . . . 98
4.2.2 Content discovery . . . 102
4.2.3 Content request . . . 102
4.2.4 Content transfer/delivery . . . 103
4.2.5 Meta-information gathering . . . 104
4.3 Content delivery functionalities that must be replicated in a content-centric network . . . 107
4.3.1 Delegation of content delivery to a paid third-party . . . 109
4.3.2 Third-party cache control . . . 109
4.3.3 Identifying content discovery agent . . . 110
4.3.4 Accounting and billing for content delivery . . . 111
4.3.5 Independent verification of content delivery . . . 111
4.4 Content-centric network model . . . 113
4.4.1 Overview of XIA . . . 113
4.4.2 Content delivery ecosystem and infrastructure . . . 116
4.4.3 Structure of routable content . . . 121
4.4.4 Chunk meta-information . . . 124
4.4.5 Chunk naming . . . 131
4.5 Proposed XIA content delivery model . . . 131
4.5.1 Content preparation . . . 132
4.5.2 Content discovery, content request and content transfer . . . 134
4.5.3 Meta-information gathering . . . 139
4.6 Accounting and billing for content delivery . . . 141
4.6.1 Example . . . 144
4.7 Lessons . . . 147
4.8 Related work . . . 148
4.8.1 Structure of routable content . . . 148
4.8.2 Design of content-centric routers . . . 149
4.9 Conclusion . . . 149

5 Applications of the Proposed CCN Content Delivery Model 151
5.1 Introduction . . . 151
5.2 Web-based content delivery . . . 152
5.2.1 Publisher does not pay for third-party content delivery . . . 153
5.2.2 Publisher pays a third-party for content delivery . . . 156
5.2.3 Keeping track of content delivery from multiple distribution partners 159
5.3 Email . . . 162
5.4 Near real-time streaming . . . 166
5.5 Extensions to other principals . . . 171
5.5.1 Service delivery ecosystem . . . 171
5.5.2 Service delivery model . . . 173
5.6 Conclusion . . . 179

6 Dealing with Different Threats in the CCN Content Delivery Model 181
6.1 Threats to publishers . . . 182
6.1.1 Denial-of-service attacks . . . 182
6.1.2 Billing fraud . . . 183
6.1.3 Deep linking . . . 184
6.1.4 Click fraud . . . 185
6.2 Threats to end-users . . . 186
6.2.1 Censorship . . . 186
6.2.2 Privacy . . . 187
6.3 Dealing with specific threats in our model . . . 190
6.3.1 Billing fraud . . . 190
6.3.2 Deep linking . . . 193
6.3.3 Censorship . . . 194
6.3.4 Privacy . . . 194
6.3.5 Other threats . . . 196
6.4 Conclusion . . . 197

7 Implications and Future Research Directions 199
7.1 Content-centric networking . . . 200
7.1.1 Support for payment mechanisms . . . 200
7.1.2 Support for transaction brokers and other intermediaries . . . 200
7.1.3 Minimum chunk sizes . . . 201
7.1.4 Content router design . . . 202
7.2 Future Internet architecture design . . . 202
7.3 Public policy . . . 204
7.4 Future research . . . 205

References 209

List of Figures

2.1 General network model 32
2.2 Transit network model 42
2.3 Viability of cache service provisioning when end-users are concentrated in a single eyeball network 50
2.4 Viability of cache provisioning when end-users are located in multiple eyeball networks 54

3.1 Computing welfare parameters 81
3.2 Social welfare when eyeball network deploys transparent caching 84
3.3 Social welfare when eyeball network deploys commercial caches 86
3.4 Constraints on the profit-maximizing price charged for content delivery by an eyeball network with market power. 87
3.5 Social welfare when CDN pays an eyeball network to co-locate caches . . . 89
3.6 Social welfare when eyeball network pays a CDN to deploy caches 91

4.1 DAG addressing - Example 1 114
4.2 DAG addressing - Example 2 116
4.3 A simplified illustration of the CCN content delivery ecosystem and infrastructure 118
4.4 Relationship between chunks, master chunks and objects. 124
4.5 Illustration of the structure of a chunk 125
4.6 Cache-level meta-information 126
4.7 Fixed field encoding scheme for cache-level meta-information 128
4.8 Application-level meta-information 130
4.9 Content discovery, content request and content delivery in our CCN model. 135
4.10 Illustration of a chunk request in our CCN content delivery model 136
4.11 Optional referrer field in a chunk request packet 138
4.12 Information flows to support accounting, billing and reporting in our CCN content delivery model 144
4.13 Example of the local version of the CDB's database stored at a cache module 145
4.14 An example of a cache database kept by a cache module 146
4.15 An example of the content delivery log kept by the cache module 147

5.1 Serving content through transparent caches 153
5.2 Serving content through a content delivery broker 157

5.3 Serving advertisements through multiple distribution partners 160
5.4 Supporting email applications with the CCN content delivery model 163
5.5 Structure of an email message in the CCN content delivery model. 164
5.6 Treating an email as a chunk for content delivery 165
5.7 Supporting near real-time streaming with the CCN content delivery model 169
5.8 Extension of the CCN model to a service delivery ecosystem. 173
5.9 Supporting service delivery with different DAG addressing types 175
5.10 Service delivery process for a service provider who does not use a service delivery broker 177
5.11 Service delivery process for a service provider who uses a service delivery broker. 178

List of Tables

1.1 Examples of clean slate Internet architecture proposals and the design paradigms they emphasize. . . . 19
1.2 Main differences between content-centric and host-centric networking. . . 21

2.1 Description of network players considered in the model. . . . 32
2.2 Benefits and costs of deploying caches for different network players 36
2.3 Caching decision space of different network players 46
2.4 Factors that affect the incentives of network players to deploy caches . . . 47
2.5 Overview of notation used - Part I: Latin alphabets. . . . 65
2.6 Overview of notation used - Part II: Greek alphabets. . . . 66

4.1 Network players in the TCP/IP content delivery model for the web. . . . 99
4.2 Meta-information needs of different players and how they are satisfied in the TCP/IP content delivery model 105
4.3 Key facilitators in the TCP/IP content delivery model and the functionalities they enable 107
4.4 Description of cache-level meta-information types. . . . 127
4.5 Minimum chunk sizes with a fixed-field encoding scheme 128
4.6 Meta-information needs of different players and how they are satisfied in the proposed XIA content delivery model 140
4.7 Key facilitators in the proposed XIA content delivery model and the functionalities they enable 142

6.1 Examples of threats in different content delivery models. . . . 189

Chapter 1

Introduction

A huge effort is currently underway in the network research community to design an Internet architecture that is more efficient, secure and evolvable for our future communication needs. While some research proposals advocate evolutionary changes to current Internet protocols to address content delivery, security, mobility and performance challenges, others propose more ambitious clean slate designs that do away with key design paradigms of the current Internet.

Notable among such clean slate proposals are those that advocate new networking principals, such as content and services, to replace or co-exist with hosts, which currently serve as the sole networking principal. In particular, the idea of using content as a networking principal, popularly referred to as content-centric or information-centric networking, has gained widespread traction because of its potential to more efficiently handle information dissemination, which constitutes the bulk of current and anticipated Internet usage.

In order to enjoy widespread deployment and usage, any new network architecture must not only address technical challenges, but must also deal with socio-economic issues that directly affect the incentives of different stakeholders to deploy and use the architecture. In particular, network architects must identify and explicitly make provisions to deal with incentives and potentials for conflict among stakeholders in a future content-centric network architecture.

This dissertation illustrates a new paradigm of network architecture design, which is motivated by economic and stakeholder considerations. Specifically, this dissertation shows how we can employ an in-depth understanding of the economic incentives of various stakeholders to inform content-centric network architecture design. The design, which results from such an approach has inbuilt mechanisms to facilitate business models of stakeholders such as publishers and network operators.

At the same time, the resulting design provides several means to deal gracefully with conflicts between the interests of various stakeholders. For instance, users can trade-off privacy with performance when privacy conflicts arise, without significantly affecting other network participants. This approach to architecture design contrasts sharply with traditional architecture design (e.g., the current Internet), which is motivated firstly by technical challenges and often does not consider the incentives of different stakeholders and thus, lacks mechanisms to deal gracefully with conflicts that arise between different stakeholder interests.

Overall, this thesis sets out to achieve four goals.

- Investigate the economic incentives of network stakeholders to deploy content-centric networking-based Internet architectures and identify design choices that align the architecture with these incentives.
- Evaluate the conditions under which the deployment of such architectures enhance social welfare and identify design choices that can enable such an outcome to be realized.
- Analyze the implications of these architectures on policy issues, such as competition, and identify design choices and regulatory interventions that can mitigate potential issues.
- Use the knowledge gained from the previous understanding to inform the design of a content-centric network architecture that is incentive-compatible and possesses inbuilt mechanisms to deal gracefully with conflicts that may arise between different stakeholder interests.

Hopefully, the insights obtained from this dissertation will help network architects to make design choices that achieve fundamental objectives such as efficiency, security and evolvability while putting in place the right incentives to encourage deployment and the appropriate mechanisms to deal effectively with potential conflicts that may arise between various stakeholders in a future content-centric network-based Internet architecture.

1.1 Background

Today, the Internet plays a major role in many aspects of the daily lives of individuals, institutions, organizations and governments. Individuals use the Internet to communicate, shop, gather information and for entertainment. Organizations use the Internet to reach out to customers and clients, streamline operations with business partners and reduce costs. Governments use the Internet to keep citizens informed, improve communications between various branches and reduce costs. Increasingly, the Internet forms a critical component of infrastructures such as communications, energy, health care and transportation.

Despite its commercial success, it has become clear that some key design decisions that underpin the Internet architecture are not well-suited to the way we use the Internet today and must be addressed going forward [1–3]. For instance, content retrieval and other forms of information dissemination currently dominate Internet usage [4, 5]. However, the Internet was primarily designed to support pairwise host-to-host connections and is thus, poorly suited for information dissemination among multiple hosts[1] [1].

Moreover, the current Internet architecture uses hosts as the sole networking principal. As a consequence, content (and other services) are tightly coupled with the location of

[1]Besides inefficient support for information dissemination, the Internet also employs very weak notions of identity, which introduces several security vulnerabilities [6]. Specifically, the Internet uses easily spoofed Internet Protocol (IP) addresses to identify hosts. As a result, it is hard to prove provenance, which is exploited by malware creators and phishing scams. Further, the Internet lacks effective mechanisms for senders to control the path of packets and for receivers to control who they receive packets from [7, 8]. Denial of service (DoS) attacks exploit these weaknesses to limit the availability of certain network resources [8, 9]. Moreover, the Internet lacks effective mechanisms to provide privacy, accountability or mobility support [1, 10]. The challenges faced by the Internet are numerous, but we will limit our attention to addressing inefficient content delivery in this dissertation.

the host. This implies that end-users must first locate a particular host, which is usually the publisher of the content, in order to obtain the content. But in most cases, users are indifferent to the host that delivers the content [3]. For example, instead of establishing another connection with the original publisher, an end-user may very well be satisfied with retrieving a copy of a free video from a neighbor who recently fetched a copy from the original publisher. This reduces network traffic and improves latency.

Attempts to provide more efficient content delivery support for the Internet fall under three main categories. The first expands the addressing capabilities of the Internet beyond the simple host-to-host, or unicast communications, which is the default communication mode supported by the original Internet architecture. The second introduces middleboxes, such as caches, within the network to store and serve frequently requested content. Finally, different kinds of overlay networks, such as content delivery networks and peer-to-peer networks, built on top of the Internet, use various technologies to direct end-users to nearby copies of content in order to improve reliability, minimize network traffic and improve end-user latency. We briefly discuss these attempts next.

1.1.1 Addressing schemes

Each host is assigned an Internet Protocol (IP) address as a prerequisite for communications on the Internet. In unicast communications, which is the default communications mode, packets are sent to an address that is associated with a single host. Thus, a publisher, who receives requests for the same video from three different end-users sends the video

separately to the IP address associated with each end-user. This is true even when all the users are located in the same destination network. Clearly, this scheme of addressing is inefficient, especially when multiple users request the same content at nearly the same time.

IP multicasting provides one means to overcome this inefficiency. Specifically, IP multicasting enables a source to send a packet only once to an IP address, which is temporarily or permanently associated with a group of interested receivers [11, 12]. The Internet Protocol (both IPv4 and IPv6) has a reserved address block for multicasting, from which a unique address can be selected for such communications[2]. Interested receivers send a control message to this multicast address in order to join the multicast group. In this way, the source only requires knowledge of the multicast address and not of individual receivers. Network elements, such as routers and switches, possess knowledge about the receivers that are associated with a given multicast address and use this knowledge to ensure that the packet gets replicated, when necessary, to reach all interested recipients of the packet.

Unlike broadcasting, which sends a packet to all potential recipients, IP multicasting ensures that a packet is sent only to interested receivers, which reduces network traffic. Nevertheless, IP multicasting is only suitable for real-time applications, such as radio streaming, videoconferencing and Internet television (IPTV). In applications, such as video on demand (VoD), where users are interested in the same content at different times, mul-

[2]Managing a multicast communication session is not a trivial task. Several steps need to take place in order to transparently support IP multicast. These include multicast address management, group management, multicast tree construction and management, resource reservation and multicast session teardown. For a more detailed discussion of the various steps and their challenges, please refer to [11, 12] and the references therein.

ticasting fails to provide efficient content delivery. Moreover, the requirements for routers to keep state for multicast groups poses global scalability challenges, which have limited its deployment to mostly enterprises, very popular real-time applications and managed internets (e.g., AT&T's U-verse video) [12].

Another addressing scheme that provides efficient content delivery support is IP anycast [13, 14]. Just like in IP multicast, a source sends a packet to an IP address that is associated with multiple recipients. The source is usually unaware of the fact that the address is associated with multiple recipients. Unlike IP multicast however, the packet is delivered to only a single receiver. This receiver is selected by a router based on proximity to the requester, where the metric used for proximity depends on the particular routing algorithm employed [13, 15]. Because of this property, IP anycast is best suited for achieving high availability and load balancing for stateless services delivered from multiple distributed servers [15]. Thus, IP anycast finds applications in transparent server selection and service location [16]. For instance, it is employed by commercial operators of the domain name service (DNS) to improve reliability and reduce latency of DNS queries [15].

Anycasting assumes that all potential recipients that share an anycast address are capable of performing the same functions. Consequently, the benefits of anycasting are mostly enjoyed by large entities with the capabilities to both deploy multiple infrastructure across the Internet and replicate services on the infrastructure. A small publisher who runs a single server obtains no benefits from anycasting. Moreover, the use of anycasting on a global scale increases the size of global routing tables, possibly affecting route lookups

and routing performance for the network as a whole [16]. Thus, it is unlikely that anycast will be widely deployed on a global scale without sufficient advances in route aggregation techniques for anycast routing [16].

Together, IP multicast and IP anycast enhance content delivery support. Nevertheless, their usefulness is limited because of the narrow range of use cases that they cater to. Next, we discuss middleboxes, which improve content delivery support in some typical use cases that are neither supported by IP multicast nor IP anycast.

1.1.2 Middleboxes

The original Internet architecture was designed according to the end-to-end principle [1, 17]. This architectural principle stipulates that the network functions as a mere conduit for packet transport, leaving all intelligence concentrated at the edge of the network. In practice, adherence to this principle at the packet level requires network elements to forward all packets along to their intended destinations and process only packets that are addressed to the network element [18]. At the application level, the end-to-end principle requires data to be transferred directly from the source to the destination, without any alteration by the network [19]. Thus, the network will forward a request for a video to the video publisher for processing, without gaining any knowledge about the request. Further, the network will not modify any data that flows between the publisher and the end-user.

With a little intelligence in the network, it is possible to avoid unnecessary traffic and reduce the latency to deliver frequently requested content. Middleboxes are devices deployed to add some intelligence to the network. They may intercept packets to learn more about the communication intent or to provide other services like security for the host that will process the packet. By virtue of the role that they perform, middleboxes necessarily break the end-to-end principle [18, 20]. Several middleboxes exist, which perform different functions such as network address management, traffic filtering for security purposes and efficient content delivery support (see e.g., [18, 21] for a more thorough discussion about middleboxes). In the context of content delivery, the most popular middlebox is the web cache [22, 23].

A cache is usually deployed at the network edge to store frequently requested objects. Once an object is cached, the network intercepts and transparently serves subsequent requests for the object from the cache, as long as the object in the cache is still fresh [24]. In this context, freshness implies that the copy of the object stored in the cache remains identical to the copy served directly by the publisher. Serving requests from a cache eliminates the need to contact the publisher for subsequent object requests, which reduces network traffic and improves latency [22, 25, 26]. For instance, Frontier Communications, a service provider that mostly caters to small and medium-sized towns and cities in the U.S., estimates that the caches deployed in its network reduce backhaul traffic by at least 25% [27].

Due to their usefulness, caches are widely deployed by many network operators and currently contribute greatly to improving content delivery efficiency. In addition, web browsers also maintain caches to store frequently accessed objects by the client. However, studies show that the coupling of content with location currently limits the full potential of caching [28]. This is because the same object may not be served from a cache simply because it is hosted by a different publisher. Thus, by decoupling content from location, it may be possible to realize greater content delivery efficiencies with caches.

1.1.3 Overlay networks

Clark *et al.* define an overlay network as "*a set of servers deployed accross the Internet that a) provide infrastructure to one or more applications, b) take responsibility for the forwarding and handling of application data in ways that are different from or in competition with what is part of the basic Internet, and c) can be operated in an organized and coherent way by third parties (which may include a collection of end-users)*" [19]. In other words, overlay networks provide a means to add new functionality to the basic Internet to meet the needs of specific applications or user groups, without requiring a change in the underlying Internet architecture.

Overlays are seen as a means to evolve the Internet architecture [19]. This is because overlay networks emerge to satisfy a need which is not currently addressed by the basic architecture. If successfully deployed and widely adopted, an overlay may eventually become part of the basic Internet architecture. Several types of overlays exist to meet various needs.

For example, routing overlays, such as the Resilient Overlay Network (RON), enhance basic network-controlled routing by providing end-hosts with the capability to specify paths that accomplish desirable goals like reduced routing delays, rapid failover and security [29]. In addition, virtual private networks (VPNs) and onion routing (e.g., Tor) enhance end-user security and privacy [30–33]. Besides the previous examples, overlays that enhance content delivery, such as peer-to-peer (p2p) networks and content delivery networks (CDNs) also exist. We briefly discuss p2p networks and CDNs next.

Peer-to-peer networks

Until the advent of peer-to-peer networks, most applications (e.g., web, FTP, Telnet, email) employed the client-server architecture paradigm for content distribution. In the client-server model, one host, called the server, services requests from all other hosts, called the clients. In this model, the server acts as an always-on resource that hosts the content. All clients, regardless of their location, establish a connection with the server in order to obtain the content. This means that clients that are closer to the server enjoy better latency than clients located farther from the server. It also means that the resources available at the server become a bottleneck when several clients simultaneously request content, which can degrade performance for all clients. This is because the resources consumed by the server and the time it takes to transfer content to clients increase linearly in the number of clients [34, Chapter 2].

One way for a publisher to deal with the drawback of the client-server model is to invest in more resources for the server infrastructure. For instance the publisher could invest in a massive server farm or data center and purchase more bandwidth to address the worst-case client access scenario. However, this approach is very costly and inefficient, because the resources will remain idle most of the time given variable demand. In addition, it still fails to address the different latencies realized by clients in different locations.

Peer-to-peer application architectures provide a low-cost alternative to distribute content to many clients without degrading the quality of service (QoS) experienced by the clients. In a p2p model, each host acts as both a server and a client. This is possible because, in a p2p model, a client can establish a direct connection with any host that has a copy of the content of interest. Thus, a host (say Host A) acts as a server when fulfilling content requests from other hosts, who act as clients during that communication session. The roles become reversed when Host A requests content. In this case, Host A becomes the client, whereas other hosts that help to fulfill the content request become servers.

Unlike the client-server model where all servers are always-on and are provided by a publisher or a service provider for a fee, most servers in a p2p model are voluntarily provided and may exhibit intermittent connectivity. Nevertheless, p2p networks possess excellent scalability properties for content dissemination between multiple clients at very low cost. To illustrate, Kumar and Ross show that whereas the minimum content distribution time increases linearly, without bound, with the number of clients in a client-server model, the minimum content distribution time in a p2p model is bounded, and does not increase

beyond the bound as the number of clients increases [35]. This property is very useful for content dissemination. As a result, p2p architectures have been adopted for numerous file distribution applications. For instance, it is used to distribute new versions of several Linux distributions and open-source software. In addition, it is used by some multimedia publishers to distribute their content [36].

Several p2p protocols exist, of which BitTorrent is the most popular[3] [5]. Estimates suggest that p2p traffic accounts for nearly 15% of aggregate Internet traffic during peak times in North America, second only to real-time entertainment and web browsing, with BitTorrent alone accounting for more than 90% of that traffic [5]. This situation is not much different in other parts of the world. The immense popularity of p2p networks suggests that the content delivery functionalities that they provide are currently very useful for a large proportion of Internet users and thus, must be considered for incorporation within the basic architecture itself.

Content delivery networks

Content delivery networks (CDNs) provide an alternate means to scale the client-server model for large-scale content distribution. The main idea behind CDNs is very simple. Rather than serve all content from a single always-on server, called the origin server, a CDN distributes surrogate always-on servers across multiple locations in the Internet and employ several techniques to direct client requests to the nearest surrogate server [39, 40].

[3]See e.g., [37, 38] for very comprehensive surveys about different peer-to-peer networking protocols.

The metrics used to determine the suitable server for a client request take factors such as physical proximity, dynamic link characteristics (e.g., round trip time, packet loss), server load and the likelihood of locating the requested content on the server into account [39, 41]. Thus, a CDN allows a publisher to distribute content to clients in multiple locations simultaneously, without degrading the quality of service experienced by different users.

In general, the reliability and the time required to distribute files improve as the number of surrogate servers and the number of locations increase [39, 42]. Hence, CDN operators negotiate to place surrogate servers close to clients, in as many network locations as possible. Unlike p2p networks where most servers have intermittent connectivity and are voluntarily provided by end-users, surrogate servers are always-on and are provided by the CDN operator. This allows the CDN operator to carefully optimize content replication strategies on the servers in order to meet different performance goals [39, 43]. Nevertheless, installing surrogate servers in many network locations is capital and labor-intensive. As a result, CDN services are mostly provided for a fee by dedicated third-party operators and large network providers. Very few publishers operate their own CDNs. The ones that do are typically very large publishers with a huge user base. Google is a typical example.

CDNs have evolved, since their initial deployment in the mid-nineties, to meet diverse needs such as the delivery of static and dynamic objects (e.g., documents, images, video, audio, software), application replication and streaming services [44, 45]. Currently, there are over forty CDN operators in the market (excluding regional operators) [44, 46]. Some

of these operators provide a wide range of content delivery services, whereas others focus on specific niches, such as video streaming. Akamai Technologies, with over 100,000 surrogate servers distributed in over 1000 networks in more than 70 countries is currently the dominant player in the CDN market, with a market share by traffic volume close to 50% [39, 47, 48]. Other major players include Limelight Networks, EdgeCast Networks, Level 3 Communications and Amazon CloudFront.

Purchasing content delivery services from a CDN operator allows a publisher to provide a consistent quality of service, regardless of the number of clients that request content, without investing in costly infrastructure, which may remain idle most of the time. As a result, many publishers pay CDN operators to realize resilience to flash crowds and low-latency content delivery [19, 39]. By some estimates, CDNs now account for almost half of all traffic currently delivered on the Internet [5, 48, 49]. This fraction is even higher in some smaller networks and is expected to increase in the future [48]. The widespread adoption and utilization of CDN services strongly suggest that CDN surrogate servers currently perform an essential role, which should be incorporated in the basic Internet architecture.

1.2 Future Internet architectures

The current limitations of the Internet have generated a flurry of activity in the networking research community to design an Internet architecture that is more efficient, robust and evolvable for our future needs. This effort, fully supported by governments[4] and industry

[4]The National Science Foundation in the U.S. has devoted up to $30 million to support the Future Internet Architecture Project (FIA) [50]. In addition, the EU and industrial partners have pumped almost

has two main motivations. First, it is evident that overlays, such as CDNs and p2p networks provide useful functionalities that should be integrated in the basic Internet architecture. Second, it has become clear that the patchwork of currently deployed ad hoc fixes fail to adequately address the challenges faced by the Internet.

Research on future Internet architectures fall under two broad categories namely evolutionary approaches and clean slate approaches [7]. Evolutionary approaches design protocols and technologies to improve the Internet, taking the current architecture as a constraint. Thus, evolutionary solutions still assume a host-centric networking paradigm and IP addresses for identification, among other things. Evolutionary research to improve content delivery focuses on designing better caching techniques, overlay networks and protocols to help hosts retrieve content more efficiently [38, 43, 52–59]. Evolutionary ideas have also been proposed and, in some cases, deployed to address other challenges such as security. These include Internet Protocol Security (IPSec) [60, 61], Transport Layer Security (TLS) [62], Secure Border Gateway Protocol (S-BGP) [63] and Domain Name System Security Extensions (DNSSEC) [64–66].

Clean slate approaches, on the other hand, design solutions to address the root cause of identified problems. As such, clean slate architectures and protocols are not constrained by the current Internet architecture. Clean slate designs usually advocate a break from three main design paradigms of the Internet. First, these approaches propose a departure from the host-centric networking (HCN) design of the current Internet. They suggest the use of

€45 million into the Scalable and Adaptive Internet Solutions (SAIL) and 4WARD projects which are geared towards future Internet architecture design [51].

other principals, such as content or service, directly for networking. This paradigm shift is intended to make the Internet more flexible and more efficient in dealing with content and other future principals that may dominate Internet use.

Given the prominent role content plays in the current Internet, it is not surprising that it is frequently suggested as a new networking principal. The idea of networking based on content names, rather than host names is usually referred to as content-centric networking (CCN) or information-centric networking (ICN), and is supported by a significant number of clean slate architectures [3, 67–76]. We will use CCN rather than ICN in the rest of this dissertation. Similarly, service-centric network (SCN) architectures use service as a networking principal and form the basis of architecture proposals such as SCAFFOLD [77]. Other architectures, such as XIA, embrace a flexible approach that supports CCN, HCN, SCN and other future principals that may arise [74].

Besides new networking principals, clean slate approaches also emphasize a break from the current network-controlled inter-domain routing. They propose to equip communicating entities with the capability to specify end-to-end communication paths that meet their needs. We refer to the routing paradigm that allows senders and receivers of traffic to specify the end-to-end communication path as edge-directed routing. This design choice helps networks and end-hosts to choose more secure paths and to control how and from whom they receive traffic. Examples of architecture proposals that emphasize more routing control include NIRA [78] and SCION [8].

Finally, clean slate approaches advocate stronger notions of identity. For instance, the concept of intrinsic security, where security properties are decoupled from path and location attributes has gained a lot of attention [79, 80]. Such intrinsic security properties can be achieved by using cryptographic primitives to name content, hosts and services. For instance, intrinsic security can be achieved by using a hash of the content or the hash of a public key representing the host or service as the name. This shift is intended to make the Internet fundamentally secure. In addition to these three main paradigm shifts, clean slate proposals also emphasize mobility support, which is severely lacking in the current Internet [1, 10]. For example, the MobilityFirst project focuses on providing efficient mobility support in a future Internet [81]. Table 1.1 summarizes the main paradigm shifts advocated by a selection of clean slate Internet architecture proposals.

Content-centric networking forms a fundamental component of many of the architecture proposals in Table 1.1. In addition, it affects all stakeholders in the current Internet ecosystem and has the potential to fundamentally alter business relationships and the roles of various network players. Moreover, CCN directly affects policy issues such as network neutrality and competition, among other things. Due to its prominent role in future Internet research and its disruptive potential, this dissertation will focus on content-centric networking. The next section provides a background and a review of the relevant literature on content-centric networking.

Table 1.1: Examples of clean slate Internet architecture proposals and the design paradigms they emphasize.

Architecture	New networking principals	Routing choice	Intrinsic security
Delegation Oriented Architecture (DOA) [18]		✓	✓
Data Oriented Network Architecture (DONA) [70]	✓		✓
Framework for Internet Innovation (FII) [10]	✓	✓	
Incorporating Consent in the Internet's Next Generation (ICING) [82] (part of NEBULA Project)		✓	✓
Layered Naming Architecture (LNA) [69]	✓	✓	✓
Named Data Networking (NDN) [3]	✓		✓
New Inter-Domain Routing Architecture (NIRA) [78]		✓	
Network of Information (NetInf) [71, 72]	✓		✓
Service-centric architecture for flexible object localization and distribution (SCAFFOLD) [77]	✓		✓
Scalability, Control and Isolation on Next Generation Networks (SCION) [8]		✓	✓
Translating Relaying Internet Architecture Integrating Active Directories (TRIAD) [67, 68]	✓		
eXpressive Internet Architecture (XIA) [83]	✓	✓	✓

1.3 Content-centric networking

Content-centric networking offers a new and a more efficient way to distribute and retrieve content. The main idea behind CCN is to elevate content to a networking principal. To achieve this, CCN dissociates content from its location attributes and enables content to exist and be retrievable by its own attributes, typically by name.

Unlike in a host-centric network where a client looks for a specific host in order to find a specific content, CCN enables the client to find a specific content on any host or network element that has stored a copy. In other words, CCN equips the network with the intelligence to locate the nearest copy of content to satisfy a client request. One could think of CCN as a means to integrate caches and the useful overlay functionalities provided by p2p networks and CDNs at the network layer, thus, making these useful functions an integral part of the basic Internet architecture.

CCN differs from HCN in three main areas namely naming, content discovery and security. The main differences are summarized in Table 1.2. CCN proposals mostly differ in content naming, content routing and how they achieve intrinsic security properties. Naming approaches are either based on hierarchical, human-friendly names (e.g., [3, 67, 68]) or flat self-certifying, human-unfriendly names (e.g., [69, 70, 72, 83]). Self-certifying names could take the form of the hash of the content [76]. The main advantage of flat self-certifying names is that they can be used to achieve intrinsic security properties. However, they pose a challenge for content discovery and routing. They also require a service to map from human-friendly names to flat names, a process which could introduce new security vulnerabilities [79].

Hierarchical names overcome issues with content discovery and routing, but encounter challenges in achieving intrinsic content security. Some architectures use a secure binding between the hierarchical name and other content properties in order to achieve intrinsic security [79]. Nevertheless, this approach requires trust in the keys used to sign this

Table 1.2: Main differences between content-centric and host-centric networking.

Attribute	Content-centric networking (CCN)	Host-centric networking (HCN)
Naming	Content is named directly, independent of its location.	Content name is dependent on its location. The same content retrieved from different locations will have different names.
Content discovery and transfer	Users request content directly by name and obtain a copy from any host or network element that has a copy. CCN requires content to be stored in many locations in the network in order to achieve good latency performance.	Users first locate the host for the content before requesting the content from the host or its authorized delegates. Authorized delegates could be content delivery networks or networks that cached content marked cacheable by the publisher. Responsible hosts for a content must provide enough resources to handle surges in demand.
Security (data integrity)	Security properties are derived directly from the content, which require schemes to make content intrinsically secure. For instance, content can be named after its hash.	Security depends on the location of the content and the path from the user to the location. In order to ensure data integrity, the user must ensure that communication has been established to the right host and the path to the host has not been compromised by any unauthorized intermediary [79].

binding, which could introduce new security vulnerabilities [84, 85]. Ghodsi *et al.* [80] perform an analysis of various desired characteristics of a future Internet and show that flat, self-certifying names are best suited to meet the requirements of security, scalability and flexibility in a future Internet.

There are two main approaches employed by CCN architectures to handle content requests. Some CCN architectures employ a name resolution service to first find a list of hosts that possess the desired content before establishing a connection between the requester and the host. Examples of such architectures include PSIRP [86], TRIAD [67],

LNA [69], NetInf [72] and CURLING [87]. Employing a name resolution service makes it easier to reuse most of the functionalities provided by a host-centric network in a content-centric network. This is because the name resolution services acts as a bridge between the two networking principals. However, designing a name resolution service for a potentially huge content space could prove challenging [76]. Recent research efforts suggests progress on this front. For instance, D'Ambrosio *et al.* propose a name resolution service, MDHT, which can resolve 10^{15} flat-named objects [88].

On the contrary, other architectures directly forward content requests without first resolving the content name to a specific location. In other words, these architectures perform name-based routing of content requests [76]. Name-based routing has a huge potential to improve latency [3]. However, it makes it harder to reuse functionalities from a host-centric network for content delivery. Further, routing on a large, flat namespace could prove challenging. Nevertheless, Caesar *et al.* propose an approach to route data on flat labels, called ROFL, and show that routing on a potentially large flat namespace could be feasible [89].

A few studies have used mathematical models and simulations to show the content delivery efficiencies that could be realized with a CCN architecture [90–92]. Others have sought to demonstrate the technical feasibility of CCN. For example, Jacobson *et al.* implement a software prototype of NDN and demonstrate that it achieves similar data transfer efficiency as current Internet protocols but offers better content distribution efficiency [3]. They also implement and demonstrate a working prototype of voice-over CCN, which shows

that CCN can support real-time applications [93].

Similarly, Zhu *et al.* build an audio conference tool over NDN and demonstrate that it achieves comparable performance to host-centric audio conferencing tools [94]. Further, Anand *et al.* build a prototype for XIA and demonstrate XIA's ability to support content delivery under different conditions at current line speeds [74]. In addition, Perino and Varvello [95] systematically evaluate the ability of current router technology to support NDN and conclude that current technology can only support NDN deployment within autonomous domains. However, they argue that further development of router technology, specifically in software and memory, could enable global NDN support in the future.

Lastly, some research efforts have analyzed the socio-economic and policy implications of CCN architectures. Aarianfar *et al.* argue that CCN poses significant privacy threats when compared to HCN because the uniqueness of content in CCN makes it possible for an adversary (e.g., government) to know users that accessed a piece of content [96]. They propose a technique that makes use of the widespread storage infrastructure envisaged in CCNs to help users protect their privacy and they show that the technique, which leverages computational asymmetry, can achieve some degree of privacy for users. DiBenedetto *et al.* also note that decoupling content identity from location opens up the possibility for finer-grained routing policies, but the economic incentive to reduce costs could limit the policies that are implemented in practice, which could have a detrimental effect on performance [97].

1.4 Contributions

It is clear from the above that most of the initial research activity on CCN focuses on addressing technical challenges. There has been very little effort to understand the incentives of network players to partake in a CCN-based Internet. Furthermore, the effects of CCN on policy issues such as network neutrality and competition have been largely ignored.

This dissertation makes the following contributions

- First, we show that network operators will fail to deploy sufficient storage infrastructure to support CCN without payment flows from publishers. This implies that the design of CCN architectures and protocols must incorporate mechanisms to facilitate payment flows. We also show that the level of payment required to make the deployment of storage infrastructure beneficial differs for different network players, which gives them different competitive advantages in providing storage infrastructure and content delivery services in a CCN ecosystem.
- Second, we identify two CCN deployment scenarios that maximize social welfare. In the first, edge networks provide the storage infrastructure through a transaction broker. In the second, edge networks pay third-parties an amount, equivalent to all the realized benefits from a storage node, to deploy storage infrastructure in the network. All other deployment scenarios lead to a deadweight loss. Furthermore, we show that the most likely of the two is the one where edge networks deploy caching infrastructure through a content delivery broker.
- Third, we identify key content delivery functionalities that break in a CCN archi-

tecture and propose a CCN ecosystem and an incentive-compatible CCN content delivery model capable of replicating and enhancing these desirable functionalities. We show that the proposed model is versatile enough to support several content delivery applications and general enough to be extended to other networking principals.

- Finally, we identify some threats introduced by the CCN content delivery model and propose some mechanisms to address these threats. We also identify policy interventions that could lead to desirable CCN deployment outcomes.

The rest of the dissertation is organized as follows. In Chapter 2, we perform an analysis of the economic incentives of different network players to deploy content-centric networking and evaluate the conditions under which cache service provisioning becomes viable for different network players in a content-centric network. In Chapter 3, we evaluate the social welfare implications of different cache deployment scenarios in a content-centric network and identify deployment scenarios that maximize social welfare. In Chapter 4, we identify the content delivery functionalities that break in a CCN and propose a CCN ecosystem and content structure, in the context of the eXpressive Internet Architecture (XIA), which replicate these functionalities and provide other desirable incentive-compatible mechanisms to encourage real-world deployment and adoption of XIA and similar CCN-based architectures. We discuss several applications and extensions of our proposed CCN content delivery model in Chapter 5. In Chapter 6, we identify and address various threats introduced by the proposed CCN content delivery model. Finally, we discuss the implications of our findings and directions for future research in Chapter 7.

Chapter 2

Economic Incentives in Content-Centric Networking

2.1 Introduction

Many proposals for a future Internet architecture envision widely distributed storage infrastructure within the network that facilitates delivery of content [3, 67, 69, 70, 72, 73, 76, 83]. Such proposals, generally referred to as content-centric networking (CCN), aim to address several challenges of the current Internet, such as the lack of efficient in-network content delivery support, lack of data persistence and security [3, 75, 79, 83]. Even though the benefits of CCN rest on the assumption of widespread storage infrastructure, such as content stores on routers and dedicated caches, previous studies fail to evaluate the economic in-

centives of various stakeholders to deploy the required storage infrastructure. In addition, policy issues, such as the effect of CCN on competition have been largely ignored.

Recently, Trossen *et al.* highlighted the importance of socio-economic issues when evaluating future Internet architectures [73]. In addition, Clark *et al.* point out the need to identify and explicitly make provisions to deal with incentives and potentials for conflict in any network design [2]. To the best of our knowledge, Rajahalme *et al.* provide the only previous analysis of the economic incentives to deploy CCN Internet architectures [98]. They qualitatively show that top-level transit providers lack incentives to deploy CCN architectures because it robs them of transit revenues. In a related work, Chun *et al.* use game theory to show that monetary payments are necessary to achieve the optimum deployment of caches within the network [99]. However, they fail to identify the specific incentives for different types of network players because they consider a non-hierarchical network scenario.

In this chapter, we provide an assessment of the economic incentives of different types of network players to partake in a content-centric network and the factors that affect these incentives. Specifically, we build a simple engineering-economic model to study the incentives of different network players to deploy storage infrastructure to support CCN and the economic viability of cache service provisioning by different network players. We find that without some explicit monetary compensation from publishers, networks will fail to deploy sufficient storage infrastructure to support CCN.

However, the level of compensation required to make cache deployment economically viable differs for different network players. For instance, large eyeball networks enjoy substantial benefits from caches and will widely deploy caching infrastructure when there are low barriers to entry and publishers incur negligible transaction costs. Further, we show that CCN architectures provide numerous opportunities for large eyeball networks to leverage their terminating access monopoly to extract more profits from other network players.

The rest of this chapter is organized as follows. In Section 2.2, we describe our model and assumptions. We follow this with an analysis of the economic incentives of different network players to deploy caching infrastructure in Section 2.3. In Section 2.4, we study the economic viability of paid content delivery by different network players and evaluate their competitiveness in providing content delivery services. We relate our findings to the current state of the content delivery industry and discuss the policy implications and the limitations of our findings in Section 2.5. In Section 2.6, we outline how the design of CCN protocols can benefit from our findings. We conclude the chapter in Section 2.7.

2.2 Model and assumptions

At its core, CCN relies on widely distributed storage within the network to facilitate delivery of content. Some authors suggest that this storage could take the form of extra memory integrated on routers, usually referred to as content stores [3]. Others suggest

the deployment of dedicated storage infrastructure to handle large volumes of data [75, 90–92, 100]. Hence, any evaluation of the incentives to deploy CCN-based architectures involves an analysis of the incentives of network stakeholders to deploy the necessary storage infrastructure to support content delivery.

In the rest of this chapter, we will refer to both content stores and dedicated storage as caches, with the understanding that content stores will likely keep frequently requested content and other less-frequently requested content from publishers who pay a premium for content delivery. Unlike the current TCP/IP model where caches are deployed as application layer overlays, CCN assumes caches are integrated at the network layer. Thus, compared to the status quo, the relationships that exist between networks will likely change in a CCN ecosystem and alter incentives and disincentives for different network players.

We build a simple engineering-economic model, which provides a framework to analyze the incentives to deploy caches in network architectures that incorporate caching. We abstract most technical cache implementation details and account for them through the various costs incurred to provide the caching service[1]. For instance, an architecture that utilizes numerous small caches incurs different storage, processing and bandwidth costs compared to one that utilizes a few large caches. Thus, our model can be used to evaluate the incentives to deploy caches in the current Internet ecosystem, where caches exist at the application layer. At the same time, it can be used to evaluate the incentives to deploy caches in a future CCN-based architecture, where caches exist at the network layer. It can

[1]Henceforth, we use caching services and content delivery services interchangeably.

also be used to understand incentives of cloud based providers, who provide both storage and computation. This is possible because we can use bandwidth and interconnection costs, as well as, processing and storage costs to reflect the technical differences between the different approaches.

In our model, publishers pay separately for network access and content delivery. The publisher has the option to buy network access and content delivery services from the same or different network providers. We identify four types of network players namely publishers, eyeball networks, transit networks and content distribution networks (CDNs). Table 2.1 provides a description of each type of network player. Additionally, one could consider large and sophisticated publishers, such as Google, who operate an extensive backbone and caching infrastructure, as a fifth type of network player. However, we have chosen to treat such entities as publishers or CDNs, depending on whether they deploy network and caching infrastructure for their private use or for commercial purposes respectively.

We consider the network model shown in Figure 2.1. Publishers connect to a transit network to reach clients located in N eyeball networks. For our purposes, eyeball networks serve a relatively small geographic area. Thus, large eyeball networks, such as AT&T and Comcast Corporation are treated in our model as a collection of small eyeball networks under a common management. The transit network connects directly to some eyeball networks, but uses transit and peering arrangements with other transit networks to reach the remaining eyeball networks [49, 101].

Table 2.1: Description of network players considered in the model.

Network player	Description	Examples
Publisher	Produces content for end-users	Google, CNN, eBay, Netflix
Eyeball network	Predominantly provides network access to end-users	Comcast Corporation, British Telecom
Transit network	Primarily provides transport services to publishers and eyeball networks to reach the Internet	Level 3 Communications, Tata Communications
Content distribution network (CDN)	Primarily caches and delivers content on behalf of publishers for a fee	Akamai Technologies, Limelight Networks

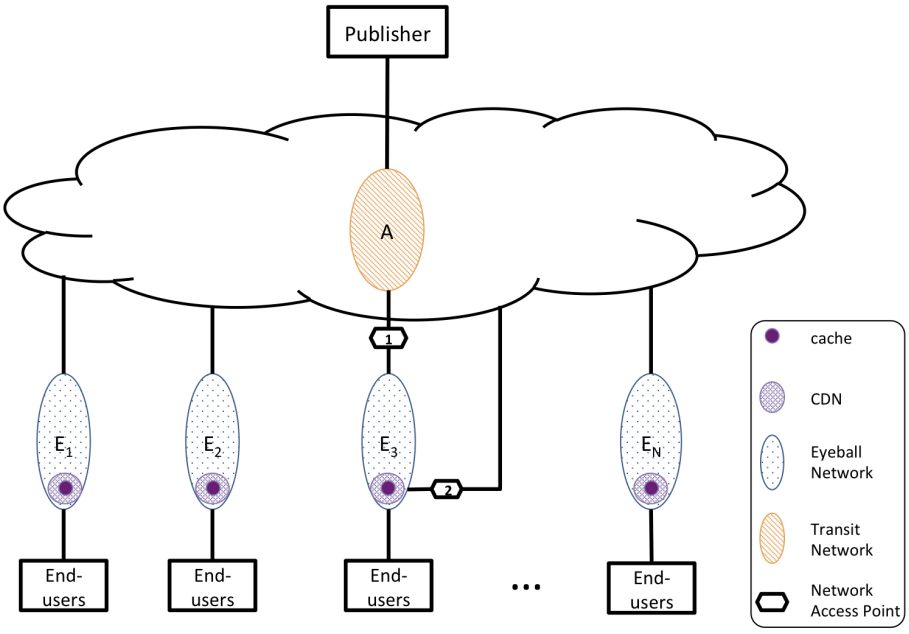


Figure 2.1: Illustration of the network model. Publishers purchase content delivery services from content distribution networks (CDNs), transit networks or eyeball networks. In general, CDNs locate their servers and caches directly within the eyeball network (e.g., Akamai Technologies) or outside the eyeball network (e.g., Limelight Networks). We consider only the former in our model, as it more accurately reflects the scenario envisioned in content-centric networking.

Transit networks either employ hot potato or cold potato routing. In cold potato routing, a network carries traffic across its own network and hands it off to another network at the closest interconnection point to the final destination. In our model, cold potato routing means that the transit network exchanges traffic with the eyeball network at the second network access point (NAP2). On the contrary, when networks engage in hot potato routing, then they exchange traffic with the next network at the closest interconnection point regardless of the final destination of the traffic. Thus, when the transit network engages in hot potato routing, it will hand off traffic from the publisher destined for the eyeball network at NAP1. The decision to implement either hot or cold potato routing depends on both technical and economic tradeoffs [102–104]. In general, we assume that networks employ hot potato routing by default, but will use cold potato routing if there exists some monetary incentive for doing so, in line with current practices [105]. However, we do not account for costs incurred in addressing any technical challenges that result from using either hot or cold potato routing.

In our model, we assume that CDNs and transit networks can pay eyeball networks to locate caches directly in the eyeball network or employ high capacity links to connect external caches to multiple locations in the eyeball network in a way that makes them logically indistinguishable from caches that will be placed by the eyeball network. In the current Internet ecosystem, CDNs pay some eyeball networks in order to locate caches [106].

However, employing high capacity links to connect external caches to multiple locations in the eyeball network does not reflect the current interconnection model where eyeball networks pay for transit. In this case, eyeball networks prefer fewer interconnection points in order to achieve economies of scale in equipment, maintenance and transit costs. Nonetheless, multiple interconnection points become desirable for eyeball networks in an environment where they receive payments for terminated traffic. As a consequence of the above assumptions, performance issues related to cache location do not form the basis of differentiation between different cache providers in our model. Rather, networks differentiate themselves through the price they charge publishers for content delivery, and each other for interconnection.

Additionally, we assume that publishers use a single cache provider to serve the same content to the same end-user. This reflects the current situation where publishers prefer to use a single cache provider to reduce transaction costs and benefit from economies of scale in pricing. For instance, annual surveys to about a thousand content providers, conducted by Rayburn since 2008, show that sixty to seventy percent of content providers use a single content delivery network to serve their content [107, 108]. The remaining thirty percent utilize multiple CDNs primarily as a means to provide the scale needed to reach end-users located in different geographic areas [108]. Nevertheless, we assume that each caching provider implements internal mechanisms to achieve redundancy and load balancing. In addition, we assume the existence of a mechanism (offline or online), which enables caches to determine whether to store and deliver an object on behalf of the object's publisher.

We limit our analysis to the delivery of a single cacheable object, whose total request rate far exceeds the update rate[2] of the object, since caching only makes sense in this case [109–111]. This could be a single file, a collection of files usually requested together or the contents of an entire webpage. This simplifies our analysis but does not affect the generality of our results since, in practice, caching decisions are made on individual files or objects [55, 112–114]. Studies show that about forty percent of web objects are currently uncacheable due to various reasons such as dynamic content generation, cookies and special HTTP headers [110, 115, 116]. However, in a future Internet where video and other relatively static multimedia content form more than ninety percent of consumer Internet traffic, it is reasonable to assume that a high fraction of objects will be cacheable [4, 117]. Moreover, technologies like Edge Side Includes (ESI) enable edge caches to assemble dynamic content, which could potentially make a larger fraction of content cacheable [118].

Our model assumes that caches are deployed in existing network locations (e.g., edge routers, NAPs, Internet exchange points (IXP) and private peering points). Therefore, we only consider (long run) incremental costs associated with bandwidth, network access, storage, processing, accounting and billing needed to provide caching at these locations. For instance, we model the delivery of a static or dynamic object by the storage and processing costs required. The network incurs mostly storage costs to deliver static objects from cache, whereas dynamic objects incur both storage and processing costs. We believe that the model described in this section contains the basic building blocks necessary to make

[2]For content with self-certifying identifiers, the concept of update rate is non-existent since in principle, a change of the content necessarily implies a different content name and hence a different object. Hence, there is no need to consider update rate in the context of self-certifying identifiers.

generalizable observations about the incentives of different network players in a complex heterogeneous system such as the Internet.

2.3 Incentives for network players to deploy caches

In general, a publisher purchases content delivery services if the benefits of doing so exceed the costs. Similarly, other types of network players deploy caches when the benefits exceed the costs. We summarize the benefits and costs for the different network players in Table 2.2.

Table 2.2: Benefits and costs of deploying caches for different network players. Publishers purchase content delivery services from other networks and all other networks deploy their own caches.

Network Player	Benefits	Costs
Publisher	Increased revenues from improved end-user latency Resilience to flash crowds and DDoS attacks Savings in network access charges Savings in processing costs	Payment for content delivery Transaction costs to set up business relationships Potential loss of content access information Potential loss of control over content access
Transit network	Bandwidth/transit savings Revenue from publishers for content delivery	Lost network access/transit revenue Processing costs Storage costs Billing and accounting costs Traffic termination and interconnection charges
Eyeball network	Bandwidth/transit savings Revenue from end-users due to improved latency Revenue from publishers for content delivery Traffic termination and interconnection charges	Processing costs Storage costs Billing and accounting costs
CDN	Revenue from publishers for content delivery	Processing costs Storage costs Billing and accounting costs Traffic termination and interconnection charges

In the next four sections, we use Table 2.2 and the model depicted in Figure 2.1 to develop quantitative models for the publisher's willingness to pay for the delivery of an object from cache and the minimum price for content delivery that makes it economically viable for other network players to cache content, given various costs, inter-domain routing policies and interconnection agreements.

2.3.1 Publisher

The publisher can choose to deliver an object from its servers or pay a content delivery service provider to deliver the object on its behalf. When the publisher performs content-delivery in-house, the publisher usually deploys just enough infrastructure to cope with the xth percentile (e.g., 95th) demand for the object and is thus, not immune to flash crowds when the demand exceeds the xth percentile. At that investment level, the publisher pays a marginal price of transit of W dollars per byte transferred. The marginal computational processing cost per request at this investment level is given by P dollars per request.

The publisher's marginal costs for transit and computational processing are lower when it delegates object delivery to a content delivery service provider. This is because the content delivery service provider delivers most objects on behalf of the publisher and the publisher only sends object updates and responses to cache misses. Furthermore, the publisher obtains benefits in the form of reduced latency and resilience to flash crowds. The resulting latency reduction and resilience to flash crowds obtained from delivering objects from N cache locations translates to monetizable benefits of $\beta(N)$ dollars per request delivered from a cache location.

At the same time, the publisher incurs costs when it offloads content delivery to a third-party. First, the publisher pays a dollar amount, γ_{P}, for every byte delivered from cache. Further, the publisher incurs transaction costs, which consist of fixed and variable components. The fixed component includes costs associated with finding and negotiating contracts and service level agreements (SLAs) with the cache providers, modifying content to facilitate content delivery and potential loss of control over content access and meta-information related to content access. We represent the fixed component, in dollars per second, by $\tau_{F}(M)$, where M is the number of cache providers contracted by the publisher. To simplify things, we assume that $\tau_{F}()$ increases linearly with the number of cache providers and thus, $\tau_{F}(M) = M\tau_{F}(1) = M\tau_{F}$.

The variable transaction cost component consists of costs to convert object delivery reports to a useful format, as well as to keep track of processes and resolve conflicts and performance bottlenecks. We represent the variable component, in dollars per request, by $\tau_{V}(M)$. We assume that $\frac{d\tau_{V}(M)}{dM} > 0$. In general, the fixed component decreases as the number of objects served from cache increases. On the other hand, the variable transaction costs increase as the number of cache providers increase.

Given the setting above, the publisher will opt to purchase content delivery services only when the amount paid for content delivery satisfies[3]:

$$\gamma_{P} \leq \underbrace{W + \frac{\beta(N)}{\eta F} + \frac{P}{F}}_{\text{marginal benefits}} - \left(\underbrace{\frac{M\tau_{F}}{\eta DF} + \frac{\tau_{V}(M)}{F}}_{\text{marginal costs}} \right) , \tag{2.1}$$

[3]See the Appendix at the end of this chapter for more details.

where F represents the size of the object in bytes, η represents the cache hit ratio realized by the contracted caching provider, and $D = \sum_{i=1}^{N} D_i$ represents the demand for the object in requests per second and D_i is the request rate from eyeball network i . For very popular objects, where the cache hit ratio is close to one, the publisher's willingness to pay for content delivery services is given by $\gamma_P \leq W + \frac{\beta(N)}{F} + \frac{P}{F} - \frac{\tau_V(M)}{F}$. We see from this that with all things equal, contracting with multiple cache providers reduces the publisher's willingness to pay for content delivery services.

2.3.2 Content distribution network (CDN)

CDNs deploy caches solely for the purpose of obtaining content delivery revenue from publishers. They negotiate with eyeball networks to place caches within their networks. They usually pay large eyeball networks to co-locate and terminate traffic [106]. At other times, they are able to co-locate for free or get paid in the form of free electricity, rack space and bandwidth. This usually happens with small and medium-sized eyeball networks, who obtain significant bandwidth savings and performance benefits from hosting the CDN. Larger eyeball networks have more negotiating leverage for extracting payments from CDNs. We represent the co-location and termination fee paid by the CDN to eyeball network i by X_{CE}^{i} dollars per byte. This payment is positive when the CDN pays the eyeball network and negative when the eyeball network pays the CDN.

Let us represent the per file processing cost by P_C dollars per request, the storage cost by S_C dollars per byte second, the cost to account and bill for the delivery of a request by

K_C dollars per request and the cache hit ratio by η. In general, a CDN can put in place a caching infrastructure that is able to handle flash crowds and achieve good performance at the same or lower marginal cost than the publisher incurs to provide a service without such benefits. This is because, in practice, the CDN can invest in a smaller infrastructure, compared to that required from individual publishers to achieve resilience to flash crowds, and take advantage of statistical multiplexing of traffic from various publishers to deal with flash crowds for all publishers. This would imply that $P_C \leq P$.

The CDN offers content delivery services only if it can charge a price, γ_c, in dollars per byte, which satisfies:

$$\gamma_c \geq \frac{\sum_{i=1}^{N} P_C^i D_i}{\eta_c DF} + \frac{\sum_{i=1}^{N} S_C^i}{\eta_c D} + \frac{\sum_{i=1}^{N} K_C^i D_i}{DF} + \frac{\sum_{i=1}^{N} X_{CE}^i D_i}{D} \ , \tag{2.2}$$

where P_C^i, S_C^i, K_C^i and X_{CE}^i represent the respective marginal costs incurred by the CDN in eyeball network i. If we assume that the CDN deploys caches in all eyeball networks and the requests for the publisher's content originate uniformly from all networks, then the above condition can be simplified as:

$$\gamma_c \geq \frac{P_C}{\eta_c F} + \frac{NS_C}{\eta_c D} + \frac{K_C}{F} + X_{CE} \ , \tag{2.3}$$

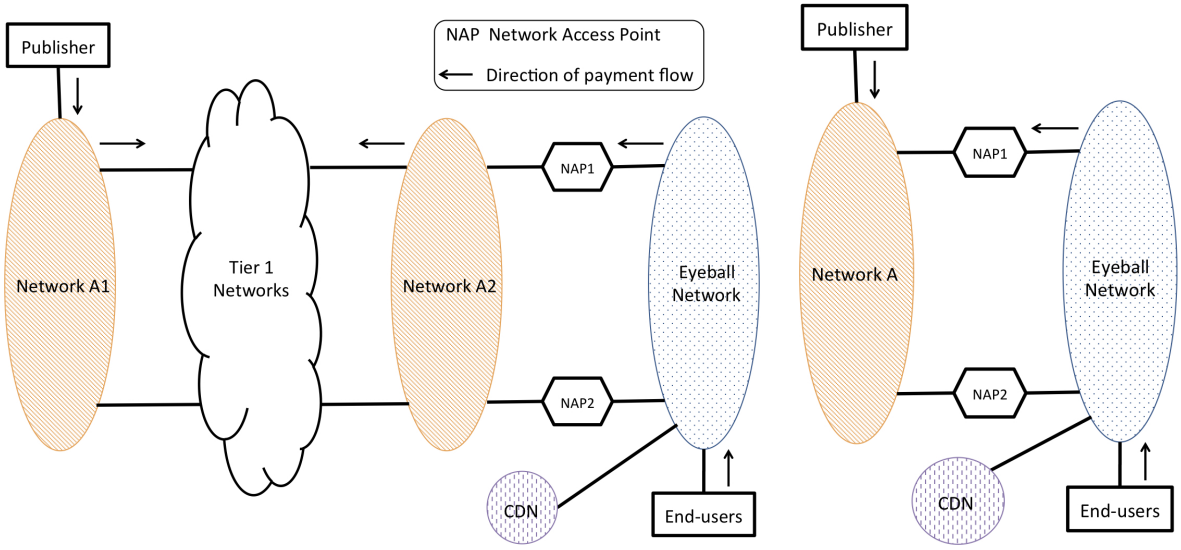
where P_C, S_C, K_C and X_{CE} represent the respective marginal costs across all cache locations.

In our model above, we assume that the CDN has configured its caching infrastructure to deliver objects directly from the cache located in the eyeball network from which the request originates. Wolman *et al.* show that in most cases, the benefits of caching can be fully realized by utilizing only the cache located in the network without any cooperation with external caches [119]. In practice, cache misses may occur that will require the CDN to fetch the requested object from other locations, which introduces additional bandwidth and/or transit costs that is not accounted for in our model. However, we assume that these costs occur rarely or are incorporated in the co-location and termination fees that the CDN pays the eyeball network.

2.3.3 Transit network

In general, there could be multiple transit networks between the eyeball network and the publisher, and the incentives for each type of transit network will be different depending on the payment flows it receives with and without caching. We illustrate this in Figure 2.2. In the setting depicted in Figure 2.2(a) for instance, Network A1 loses network access revenues when it performs caching but avoids bandwidth costs to transfer packets from the publisher. On the other hand, Network A2 saves on upstream transit payments when it performs caching. Due to its proximity to the eyeball network, it could even charge publishers for lower latency content delivery.

For our analysis, it is more interesting to consider transit networks that connect directly to eyeball networks. This is because publishers are likely to buy access directly from a tier-1



(a) Multiple transit networks connect the publisher and the eyeball network

(b) A single transit network connects the publisher and the eyeball network

Figure 2.2: Illustration of two network models. In the model in Figure 2.2(a), the publisher buys network access from a tier-2 network, which uses transit and peering relationships with other networks to reach the eyeball network. In the model in Figure 2.2(b), the publisher buys network access from a tier-1 transit network that connects directly to eyeball networks. In both models, the publisher has the choice to buy content delivery services from only one of three types of network players namely transit networks (Networks A, A1 and A2), eyeball networks and content distribution networks. We employ the simplified model in Figure 2.2(b) in the rest of the chapter.

network that in turn serves large eyeball networks directly. Thus, we employ the simplified model in Figure 2.2(b) for our analysis. Further, we limit the use of transit network to a transit provider that has a direct connection to an eyeball network. Our setting eliminates the possibility of more complex network hierarchies but does not affect our conclusions about the incentives of individual eyeball networks, content distribution networks and transit networks to deploy caching infrastructure.

In order to achieve comparable latency to a cache deployed by an eyeball network, the transit network performs cold-potato routing when it caches content for publishers. If we represent the marginal bandwidth cost to transfer a byte from the publisher to NAP1 by

B_{AP} dollars per byte and the marginal bandwidth cost to transfer a byte from NAP1 to NAP2 by B_{AE} dollars per byte, then we see that offering content delivery services implies an additional bandwidth cost of $B_{AE} - B_{AP}$ dollars per byte anytime there is a cache miss.

If the transit network places caches in L locations, where $L \leq N$, to serve a total demand of $\tilde{D}$, where $\tilde{D} \leq D$, then the transit network will find caching beneficial only if it obtains a payment, γ_{A}, in dollars per byte, which satisfies:

$$\gamma_{A} \geq W + \frac{P_A}{\eta_{A} F} + \frac{L S_A}{\eta_{A} \tilde{D}} + \frac{K_A}{F} + \frac{1 - \eta_{A}}{\eta_A} B_{AE} - B_{AP} \ . \tag{2.4}$$

The above equation assumes that the nature of the transit payment relationship between the eyeball network and the transit network remains the same when the transit network offers content delivery services. This assumption is reasonable because the same traffic flows to the eyeball network, regardless of whether it is exchanged at NAP1 or NAP2. Nevertheless, as was demonstrated recently by the dispute between Level 3 Communications and Comcast Corporation, the nature of the transit payment relationship may change in practice when large eyeball networks with market power are involved [106]. In such a scenario, we require an additional term in (2.4) to account for the change in the nature of the payment relationship. In order to keep things simple, we ignore this for now.

Furthermore, (2.4) assumes that the transit network obtains negligible indirect monetizeable benefits from end-users by introducing caching. This assumption is reasonable since a transit network deals with publishers and eyeball networks and not directly with

end-users.

The transit network can trade off between bandwidth and storage costs by changing the number of cache locations. With a smaller number of cache locations (i.e., reducing L), the transit network reduces its overall storage costs, but increases the bandwidth costs to reach different eyeball networks. On the other hand, providing caches in multiple locations reduces bandwidth costs at the expense of increased storage costs. The choice that makes sense depends on the transit network's bandwidth costs, the prevailing market price for storage and the networking architecture. For instance, some CCN architectures favor co-location of storage infrastructure with routers (e.g., NDN [3]), whereas others could perform equally well, regardless of the storage location (e.g., PSIRP [86]).

2.3.4 Eyeball network

An eyeball network obtains bandwidth and transit savings from locating a cache within its network. The end-users also obtain better performance in terms of latency, which can be monetized by the eyeball network. With a monopoly in the end-user access market, the eyeball network can monetize the additional improvement in end-user latency from deploying caches by raising the end-user network access charge [120]. We denote the monetizeable benefit that the eyeball network obtains from end-users, as a result of improved latency from deployed caches, by σ dollars per byte.

The eyeball network will find caching desirable if it receives a payment from a publisher, $\gamma_{_E}$ dollars per byte, which satisfies:

$$\gamma_{_E} \geq \frac{P_E}{\eta_{_E} F} + \frac{S_E}{\eta_{_E} D_i} + \frac{K_E}{F} - (B_E + X_{EA} + \sigma) \ , \tag{2.5}$$

where B_E represents the bandwidth costs from NAP1 to the cache location and X_{EA} represents the positive transit payments from the eyeball network to its transit provider.

In a monopoly, the end-user network access charge when the eyeball network does not undertake caching, W_{eu} dollars per byte, and charge when it does, W'_{eu} dollars per byte, are related to σ by (see [120])

$$W'_{eu} - W_{eu} \leq \sigma \ . \tag{2.6}$$

Thus, in a competitive end-user network access market, the eyeball network finds caching desirable when

$$\gamma_{_E} \geq \frac{P_E}{\eta_{_E} F} + \frac{S_E}{\eta_{_E} D_i} + \frac{K_E}{F} - (B_E + X_{EA}) \ . \tag{2.7}$$

Table 2.3 summarizes the decision space of a publisher to purchase caching services and of other network players to offer caching services. An overview of the notation used in the model, together with a detailed derivation of the caching decision space of each network player is provided in the appendix at the end of this chapter.

Table 2.3: Summary of the constraints on the price, in dollars per byte, that networks must charge in order to provide economically viable caching services to a publisher. When all end-users are located in a single eyeball network, then the eyeball network is in a position to offer the lowest prices for caching services, since it gains a lot from bandwidth savings and does not lose any revenue from network access.

Network Player	Price constraint for viable caching services (dollars per byte)
Publisher	$\gamma_{P} \leq W + \frac{\beta(N)}{\eta F} + \frac{P}{F} - \left(\frac{M\tau_{F}}{\eta D F} + \frac{\tau_{V}(M)}{F} \right)$
Transit network	$\gamma_{A} \geq W + \frac{P_A}{\eta_{A} F} + \frac{L S_A}{\eta_{A} \tilde{D}} + \frac{K_A}{F} + \frac{1-\eta_{A}}{\eta_{A}} B_{AE} - B_{AP}$
CDN	$\gamma_{C} \geq \frac{P_C}{\eta_{C} F} + \frac{N S_C}{\eta_{C} D} + \frac{K_C}{F} + X_{CE}$
Eyeball network	$\gamma_{E} \geq \frac{P_E}{\eta_{E} F} + \frac{S_E}{\eta_{E} D_i} + \frac{K_E}{F} - (B_E + X_{EA} + \sigma)$

We provide a summary of the impact of some parameters on the economic viability of cache deployment for all network players in a content-centric network in Table 2.4. Some current trends provide encouraging signs for the viability of a market for caching in content-centric networks while some other evidence paints a gloomy picture for content delivery services. For example, publishers increasingly offer high resolution multimedia, which typically have bigger file sizes. This tendency towards bigger file sizes has a positive impact on the viability of content delivery services. In addition, the cost of storage has experienced a rapid decline in the last few years [121]. Such a trend improves the economic viability of content delivery services.

However, recent evidence suggests that bandwidth prices are decreasing over time [122]. For a fixed per file processing and billing cost, decreasing bandwidth prices reduce the incentives to provide caching services. Hence the overall desirability of providing content delivery services in the future depends on the relative rates at which processing and bandwidth costs decline.

Table 2.4: The effect of different parameters on the viability of providing in-network content delivery services. The impact of each parameter is estimated under *ceteris paribus*. Up arrows in the first column indicate that the parameter is increasing. In the second column, up arrows indicate that the viability of providing content delivery services improves, whereas a down arrow indicates that the viability deteriorates. These observations are consistent with intuition.

Parameter Variation	Market Viability of Caching Services
External bandwidth cost $(B) \uparrow$	$\uparrow$
Demand for file $(D) \uparrow$	$\uparrow$
File size $(F) \uparrow$	$\uparrow$
Accounting and bill processing cost $(K) \uparrow$	$\downarrow$
Number of cache locations $(N) \uparrow$	$\uparrow$
File processing cost $(P) \uparrow$	$\downarrow$
Storage cost $(S) \uparrow$	$\downarrow$
Cache hit-ratio $(\eta) \uparrow$	$\uparrow$

In the next section, we evaluate the viability of cache service provisioning for each type of network player, given the publisher's willingness to pay, distribution of end-users and the object popularity.

2.4 Economic viability of cache service provisioning

In the previous section, we developed generic expressions for the publisher's willingness to pay for content delivery services and the price for content delivery that makes caching desirable for each type of network provider. Next, we determine the economic viability of cache service provisioning for each type of network player, under two end-user distribution scenarios. In the first, we consider a situation where the majority of requests for the publisher's content originate mainly from one geographic area served by a single eyeball network. In the second, we study the case where requests originate uniformly from multiple geographic areas served by different eyeball networks.

2.4.1 Requests originate mainly from a single geographic area served by a single eyeball network

When requests originate mainly from one geographic area, the publisher only needs to contract with a single eyeball network, a single transit network or a single CDN for paid content delivery. In this case, the maximum willingness of the publisher to pay for content delivery services is the same for each type of provider and is given by $\gamma_P^* = W + \frac{\beta(1)}{F} + \frac{P}{F} - \frac{\tau_V(1)}{F}$. For comparison purposes, we consider the case where all networks incur the same marginal costs for storage, processing and accounting and billing.

Even though economies of scale may exist in storage, processing and accounting and billing costs, the current content delivery ecosystem suggests that such scale advantages are, at best, minimal. Furthermore, we can reasonably assume that different networks can

employ state of the art technology to realize similar cache hit ratios. Hence, the marginal costs for the transit network, CDN and eyeball network are represented respectively by $\gamma_A^* = W + \frac{P}{\eta F} + \frac{K}{F} + \frac{1-\eta_A}{\eta_A} B_{AE} - B_{AP}$, $\gamma_C^* = \frac{P}{\eta F} + \frac{K}{F} + X_{CE}$ and $\gamma_E^* = \frac{P}{\eta F} + \frac{K}{F} - (B_E + X_{EA} + \sigma)$.

For a large eyeball network, it is reasonable to expect the CDN to pay for co-location and traffic termination [106]. Further, recent trends suggest that transit profits are very close to zero [122]. Thus, the relative positions of the marginal costs curves for the CDN and the transit network depend on the difference between the co-location and termination charges paid by the CDN and the metropolitan bandwidth cost incurred by the transit network. The recent competition between Level 3 Communications, a large transit network that provides content delivery services, and Akamai Technologies, the dominant CDN, suggests that the two curves are either very close, or that the marginal cost curve for the transit network is just slightly below that for the CDN [123–125]. Hence, we only consider a CDN in our subsequent analysis in the rest of this chapter, with the understanding that the curve for the transit network could be at a higher, similar or lower position depending on its metropolitan bandwidth costs.

In Figure 2.3, we plot the minimum price required to make cache provisioning viable for an eyeball network and a CDN (transit network), as a function of the object request rate. The marginal cost, γ_C^*, for the CDN comprises per-file processing, co-location, termination, accounting and billing costs. The net marginal cost for an eyeball network, γ_E^*, is lower because it realizes bandwidth savings and increased end-user revenue from deploying caches. Strictly speaking, γ_E^* decreases with increasing demand for an object because of increased

monetizable benefits from improved latency. However, we assume that the monetizable benefits are exhausted at a relatively low level of demand, and are constant as demand increases further, thereby resulting in a horizontal γ^*_E. The shaded areas between different curves represent the region where content delivery from caches becomes viable.

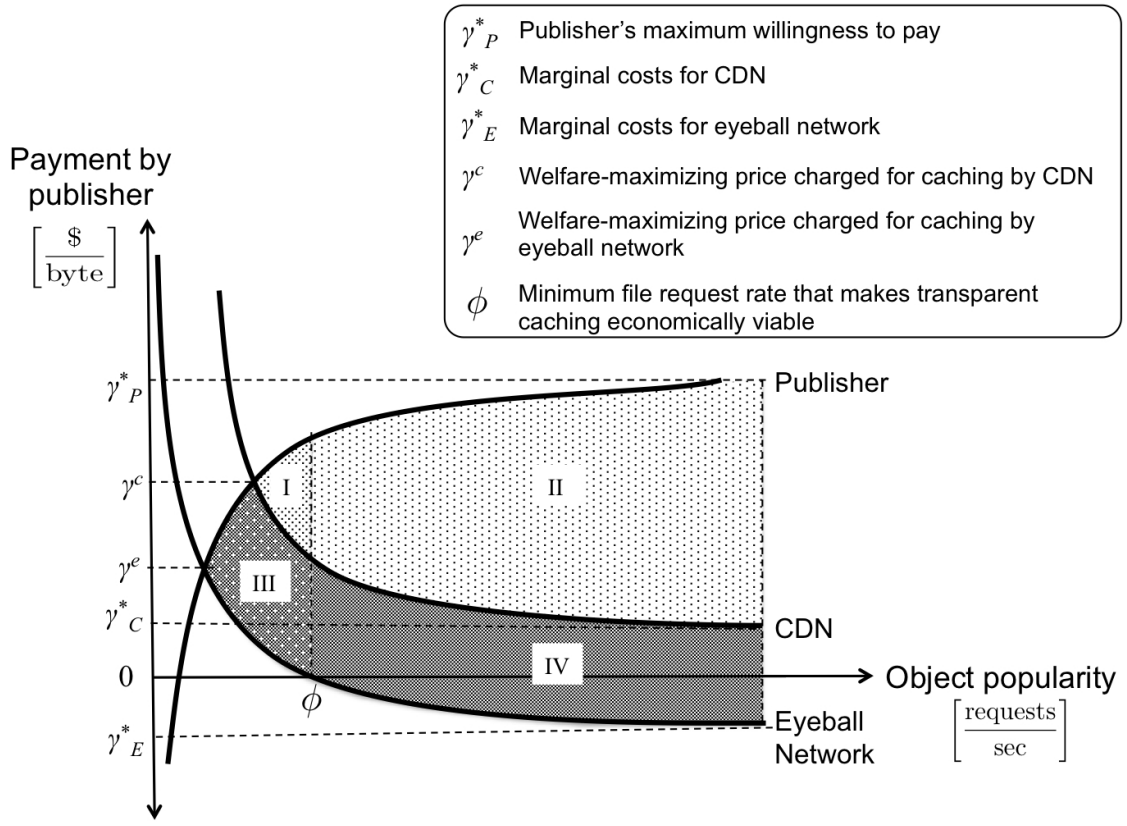


Figure 2.3: Illustration of the viability of cache provisioning for different network providers. The shaded areas between different curves represent the region where content delivery from caches becomes viable. In the absence of monetary transfers between the publisher and the network, the eyeball network only caches objects in regions II and IV, whereas the CDN caches no objects at all. This suggests that payment transfers between publishers and network players will be necessary to incentivize networks to deploy sufficient caching infrastructure in a CCN architecture. Given a payment transfer mechanism, the graph shows that an eyeball network has a larger potential content delivery market (regions $I - IV$) than the CDN (regions I and II). The figure assumes a cache hit ratio that is very close to one.

We see from Figure 2.3 that in the absence of monetary transfers between the publisher and other network players for paid content delivery, the eyeball network will only cache objects whose popularity exceeds ϕ, the minimum demand for an object that makes trans-

parent caching economically viable[4] . Thus, the eyeball network will only cache objects in regions II and IV. All objects in regions I and III will not be cached by the eyeball network. On the contrary, the CDN will not provide any caching without monetary payments from the publisher. This suggests that networks will fail to deploy the economically optimum amount of caching infrastructure in content-centric network architectures, which lack provisions for paid caching.

Figure 2.3 also shows that when mechanisms exist for payment transfers between the publisher and the network for content delivery, then the eyeball network has a larger potential market for content delivery than the CDN. That is, for any given payment from a publisher, the eyeball network is in a position to provide economically viable content delivery services for objects with a lower request rate than what the CDN can offer. Whereas the eyeball network can provide viable content delivery services for all payment and object request rate combinations in regions $I - IV$, the CDN can only provide viable content delivery services for combinations in regions I and II. Therefore, a publisher that delivers content mostly within a single geographic area served by a single eyeball network may find it beneficial to purchase content delivery services from the eyeball network rather than a CDN, when either player charges welfare-maximizing prices.

The eyeball network can directly influence the viability of cache provisioning for the CDN through the co-location and termination fees it charges. To illustrate, an eyeball

[4]In transparent caching, the network caches content purely for cost savings and performance improvement on its network and does not demand any form of payment from the publisher. It is done without any knowledge (or permission) from the publisher of the content. See [126] for the distinction between transparent caching and other forms of caching.

network can move the marginal cost curve of the CDN up or down by raising or lowering the co-location and termination fees respectively. An eyeball network could even lower the marginal cost curve of the CDN to overlap with its own marginal cost curve simply by paying the CDN an amount equivalent to the bandwidth savings and increased end-user revenue it realizes from the CDN's caches. On the other hand, eyeball networks can also make caching undesirable for CDNs by raising the co-location and termination fees.

When an eyeball network does not possess market power, then the cost curve for the transit network serves as the upper bound on the extent to which it can increase co-location and termination fees [127]. However, when market power exists, then the eyeball network can raise co-location and termination fees till the point where the additional deadweight loss created by the raise offsets the additional gain in surplus. We discuss this in detail in the next chapter, when we examine the social welfare implications of different cache deployment scenarios.

2.4.2 Requests originate uniformly from multiple geographic areas served by different eyeball networks

We consider a scenario where requests for a publisher's content originate uniformly from M geographic areas, each served by a different eyeball network. Thus, $D_i \approx \frac{D}{M}$ requests per second originate from each eyeball network. This simplifies our analysis without loss of generality. We further assume that M is reasonably large.

CDNs perform two key functions. First, they serve as transaction brokers between different publishers and different eyeball networks. Second, they deliver content. The first role becomes very important when requests for a publisher's content originate from multiple eyeball networks. By contracting with a CDN, a publisher obtains access to multiple eyeball networks through a single point of contact. Thus, publishers only need to worry about paying for content delivery when they contract with a CDN. Consequently, publisher's have a higher willingness to pay for content delivery from a CDN. In the absence of a transaction broker, each publisher takes on the brokerage function of contracting with each of the eyeball networks, which increases transaction costs for the publisher. This has the effect of reducing the publisher's willingness to pay for content delivery from individual eyeball networks.

Stated differently, the publisher's maximum willingness to pay for content delivery from a CDN is given by $\gamma_P^* = W + \frac{\beta(M)}{F} + \frac{P}{F} - \frac{\tau_V(1)}{F}$, whereas that for an individual eyeball network is given by $\gamma_P' = W + \frac{\beta(M)}{F} + \frac{P}{F} - \frac{\tau_V(M)}{F}$. In effect, the maximum willingness to pay for content delivery from an eyeball network decreases as the number of eyeball networks from which requests originate increases. However, if there exists an entity that performs the transaction brokerage function on behalf of all eyeball networks (e.g., a federation or a third-party broker like a CDN that aggregates services from multiple eyeball networks), then the maximum willingness to pay for content delivery through this single point of contact is also given by γ_P^* and increases as the number of eyeball networks increases in a similar way to the maximum willingness to pay for caching from a CDN.

We illustrate the situation when the publisher contracts individually with eyeball networks and when it contracts with a single entity representing all eyeball networks in Figure 2.4. The shaded areas between different curves represent the region where content delivery from caches becomes viable.

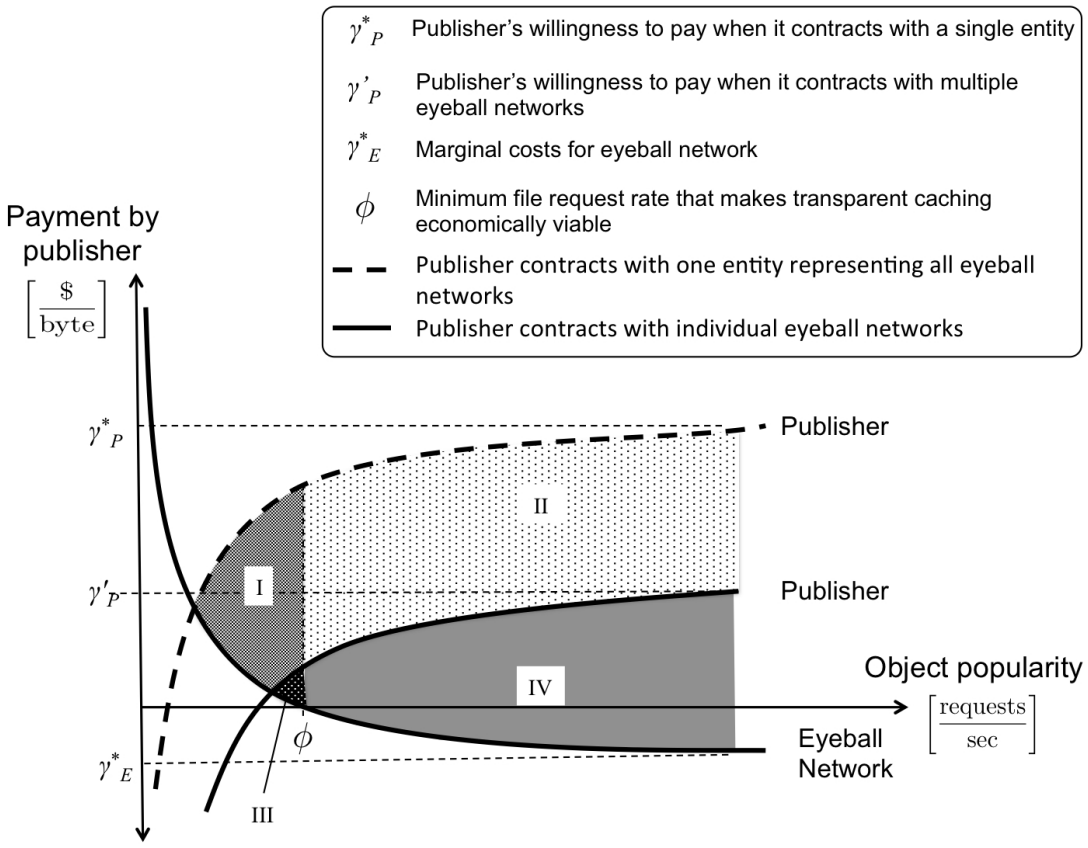


Figure 2.4: Illustration of the viability of cache provisioning when end-users are located in multiple eyeball networks. The publisher's willingness to pay for content delivery is lower when it contracts with multiple eyeball networks (solid curve) when compared to the case where it contracts with a single entity representing all eyeball networks (broken curve). In either case, an eyeball network has incentives to cache objects whose popularity exceed ϕ (regions II and IV). However, the size of the potential market for paid content delivery reduces from regions $I - IV$ to only regions $III - IV$ when the publisher contracts individually with eyeball networks. When the number of eyeball networks gets very large, regions III and IV may be non-existent. We do not show the curve for the CDN because it remains the same as shown in Figure 2.3.

We see from Figure 2.4 that, the eyeball network finds it economically viable to cache all objects whose popularity exceeds ϕ, regardless of whether the publisher contracts individ-

ually with eyeball networks or with a single entity representing all eyeball networks. This means that the eyeball network only caches objects in regions II and IV in the absence of monetary transfers from the publisher for content delivery. However, the size of the potential market for paid content delivery reduces from regions $I - IV$ to only regions III and IV when the publisher contracts individually with eyeball networks. The situation for the CDN remains unchanged from our discussion in Section 2.4.1. Again, the above figure shows that in the absence of monetary transfers from the publisher, networks will fail to deploy sufficient caching infrastructure. Additionally, the figure shows that paid content delivery becomes more viable when the publisher contracts with very few entities.

Our analysis in this section shows that eyeball networks enjoy favorable conditions to provide paid content delivery services. This is especially true when requests for a publisher's content are concentrated in a single network. Even when users are distributed in multiple networks, an eyeball network content delivery federation provides members with favorable conditions to undertake paid content delivery. Alternatively, eyeball networks could host CDNs and still enjoy the bandwidth savings and latency improvements that comes from serving requests from a cache, without making any investments.

The choice that makes sense for an eyeball network depends on two factors namely the barriers to entry and the extent to which it can leverage caching infrastructure to obtain revenue streams. The barriers to entry include capital costs to deploy caching infrastructure and access to caching technology. These barriers could be high if CDNs have a monopoly on caching technology, in the form of patents. In such a scenario, eyeball networks will

likely host CDNs and use their terminating access monopoly to extract rents from CDNs in the form of co-location and termination fees.

The implications of the preceding analysis are clear. When a publisher's willingness to pay for content delivery services does not depend on the type of network that provides the service, then eyeball networks enjoy a significant pricing advantage over CDNs when both face similar costs. Despite the pricing advantage enjoyed by eyeball networks, publishers may take factors other than price into account when making a decision to purchase content delivery services. For example, publishers may not want to pay for content delivery from a network that provides competing content, even if such a network offers the lowest prices. Under such a scenario, our conclusion about the advantage enjoyed by the eyeball network fails to hold.

Even when publishers take the type of caching provider into account when making purchasing decisions or face significant transaction costs to establish relationships with eyeball networks, it may still make sense for eyeball networks to deploy limited transparent caching within their networks.

2.5 Discussion

In Section 2.4, we showed that when transaction costs and barriers to entry are low, large eyeball networks have significant incentives to deploy caching infrastructure in order to appropriate the benefits of caching. Even when transaction costs for publishers are fairly high or when publishers remain reluctant to purchase content delivery services from

eyeball networks with competing content, eyeball networks still have incentives to deploy transparent caching for publishers with popular content. A logical question to ask is whether these results reflect the current situation and the extent to which they can be applied to a future CCN ecosystem.

The first observation we make is that, until recently, content delivery was dominated by third-party CDNs. This is mostly due to the low transaction costs that publishers incur when they do business with CDNs and the added value they obtain through aggregate content delivery reports across multiple eyeball networks. In addition, CDNs, such as Akamai Technologies, erected significant barriers to entry through patents on numerous caching and content delivery technologies[5]. However, we see that large eyeball networks have recently taken steps to eliminate the technological barriers through partnerships and acquisitions. As our model shows, large eyeball networks have significant incentives to deploy caching infrastructure when barriers to entry are low. This is a trend we currently see in the Internet and we expect it to continue in a future CCN architecture.

A single transaction broker lowers the transaction costs publishers incur to purchase content delivery services. Thus, one could imagine a content delivery federation of eyeball networks, which serves as a single point of contact for members of the federation. The Open Carrier Exchange (OCX) represents one such effort [128]. Alternatively, third-party CDNs could evolve into a new role, where they license their technology to multiple eyeball networks and act as a transaction broker for all eyeball networks that utilize their tech-

[5]See `http://bit.ly/eKZbRT` for an interesting discussion of some key technologies that were patented by Akamai, which gave it a huge initial advantage in the CDN marketspace.

nology. We are beginning to see such trends in the market, with managed and licensed CDN offerings from major players, such as Limelight Networks, Edgecast Networks and Akamai Technologies [129–132]. The design of CCN architectures could seek to facilitate such measures aimed at minimizing transaction costs across multiple eyeball networks.

In our model, we have assumed that other cache providers interconnect their caches deeply within the eyeball network. However, this does not reflect current reality. For instance, even with the presence of a CDN or a transit network that provides caching, eyeball networks may still find it desirable to deploy transparent caching to save the metropolitan bandwidth required to transport files across distant locations within their networks. This is especially true in small towns and rural areas, where the caches placed by a CDN may be very far away from some clients. For the same reasons, cellular operators may find it beneficial to locate transparent caches at cell towers in order to save on back haul. Thus, it is entirely possible that multiple caches may be deployed by different network players to serve the same content to the same end-users in an CCN ecosystem. We leave the analysis of the competitive effects of such a scenario for future work.

Even though CCN could improve on security and content delivery, it also introduces a few issues that deserve consideration. For starters, caching and content delivery infrastructure provides networks with a means to circumvent transport-focused net neutrality regulations. For example, the recent net neutrality rules enacted by the Federal Communications Commission in the U.S. focused solely on transport infrastructure [133]. Hence, networks can, in principle, implement and monetize differential quality of service by lever-

aging their caching infrastructure. In particular, eyeball networks can favor particular content publishers through caching algorithms or the prices they charge for caching content. Furthermore, large eyeball networks can also influence the competitiveness of content delivery providers and publishers by adjusting the prices they charge for co-location and traffic termination.

We end this section by pointing out that our analysis does not account for the fact that the actual bit delivery business forms a decreasing fraction of the revenues of CDNs. Many third-party CDNs are increasingly moving into the business of providing value-added services such as application delivery, dynamic site acceleration, real-time analytics, security, consulting and providing tailor-made solutions to enterprises for content and advertisement management [134–137]. For instance, value-added services now form more than half of Akamai's annual revenues [136]. Given this shift, one could easily imagine a scenario where CDNs bundle actual bit delivery with other value-added services in order to remain competitive. Thus, the level of CDN cache deployment in a CCN ecosystem may be higher than our model suggests.

In addition, we have assumed similar cost structures for all types of network players. In particular, we have assumed separate cost components for processing and storage and we have not accounted for the fact that storage can be bundled as part of modern routers, which changes the cost structure of transit networks and eyeball networks. In such circumstances, the competitiveness and incentives of these networks to provide caching services may be better than our model suggests.

Finally, we have not explicitly modeled end-user behavior as part of our analysis. For instance, users can react in several ways when they have a bad quality of experience of a particular website. In the short term, they could switch to other websites for similar information. In the medium to long term, they could even switch network providers. Taking such dynamic user-behavior into account can influence the incentives of publishers and network providers in ways that are not captured by our model, which could lead to slightly different conclusions.

2.6 Lessons for CCN architecture and protocol design

In this section, we discuss the implications of our findings for the design of CCN architectures and protocols.

2.6.1 Accounting, reporting and payment mechanisms

Our preceding analysis implies that without some form of payment flow, networks will fail to deploy sufficient caching infrastructure. Thus, CCN architectures must incorporate features that make it easy to implement payment mechanisms to support caching-based business models. In order to be useful, these features should exhibit low transaction costs and provide a means for transparent verification.

For instance, CCN protocols should make it easy for publishers to indicate their willingness to partake in the caching market and the conditions under which they will participate. In addition, the protocols must enable networks to easily determine whether to cache an

object based on easily identifiable and trustworthy information about the publisher of the object or its authorized intermediaries. These features will ideally provide networks with the capability to aggregate information about content delivered on behalf of publishers across multiple networks. Furthermore, the architecture must minimize of the number of entities with whom a publisher must establish business relationships for content delivery.

Moreover, payment mechanisms must provide disincentives for fraud. For example, network operators could have incentives to over-report the volume of content delivered in order to extract more revenue from publishers. In the absence of effective mechanisms to verify content delivery, publishers may not have incentives to participate in the caching market for fear of potential fraud. Furthermore, CCN architectures must provide reporting mechanisms that enable publishers to obtain information about who accesses their content, when they access it, and how they access it. This kind of information is necessary for publishers to build models for targeted advertisement and to protect against click fraud.

2.6.2 Cache hit ratios

Among the factors that directly affect the economic viability of caching, the cache hit ratio is perhaps the one that the design of CCN can directly influence. Since networks have more incentives to deploy caches when the cache hit-ratio increases, CCN design must emphasize cache performance in addition to primary concerns, such as security and efficient transport.

Previous research efforts have demonstrated that the manner in which big files are split up or chunked for storage affects the cache hit ratio [138]. For example, cache replacement

policies that are optimized to maximize the cache hit ratio tend to favor smaller chunk sizes, whereas those optimized for byte hit ratio tend to favor bigger chunk sizes [139]. Thus, it is reasonable to expect that without any explicit standards and guidelines on chunking, different publishers will split up big files in different ways to optimize particular parameters, or in response to different cache charging strategies.

However, when the same file is split up differently by different publishers, a chunk naming scheme based on the hash of the content (e.g., [69, 70, 72, 73, 83]) results in different names and properties for individual components of the file from each publisher, which limits overall cache performance for networks serving the content. In addition, the manner of chunking also has an impact on routing. For instance, splitting a file into many smaller chunks increases the number of unique names and affects routing scalability. Thus, in order to optimize the cache hit-ratio and improve routing, it may be necessary for CCN designers to propose (and potentially standardize) universal chunking and naming guidelines.

Furthermore, mechanisms to enforce access controls should also take the impact on the cache hit ratio into account. For example, uniquely encrypting content for each user as part of digital rights management (DRM) greatly reduces the cache hit ratio, which makes caching unattractive. Thus, the trade-offs between cache hit ratios and access controls must be carefully studied in order to design CCN protocols that incentivize cache deployment. This problem is not unlike the problems encountered in secure multicast (see e.g., [140, 141]), since hierarchical caching can be viewed as a form of asynchronous multicast.

2.7 Conclusion

In this chapter, we evaluated the economic incentives of different types of network players to partake in a content-centric network. We showed that the level of caching supplied with transparent caching is inefficient. With a payment flow from publishers for content delivery, the level of caching supplied increases because cache providers find it economically viable to cache less popular objects. Further, we showed that contracting with a single entity increases a publisher's willingness to pay for content delivery services, which suggests that CCN architectures must support paid content delivery through intermediaries that act on behalf of several network players.

Moreover, we pointed out that, while publishers may have legitimate rights to limit access to content using various means, it is important that the means employed do not negatively hamper cache hit ratios. For instance, per-user encryption based techniques for digital rights management (DRM) successfully achieve access control at the expense of cache hit ratio. Thus, content owners and CCN designers need to think of creative ways to balance the needs of access control and the performance of caches in order to improve the incentives for network players to deploy caches in an CCN ecosystem.

Finally, we identified caching infrastructure as one avenue for networks to circumvent transport-focused net-neutrality regulations. In particular, networks can use caches as a strategic asset to open up new revenue streams from differential QoS business models. For instance, networks can favor a publisher's content through the algorithms employed at the caches. Hence, policy measures taken to address net-neutrality must also take caching

infrastructure into account. Moreover, eyeball networks can implement differential pricing for interconnection and co-location, depending on the CDN and the CDN's customers. Thus, regulators will need to address interconnection issues in a CCN-based architecture.

A Appendix

A.1 Notation

Table 2.5: Overview of notation used - Part I: Latin alphabets.

Symbol	Definition	Unit
B_{ij}	cost of bandwidth to transfer a file from network i to network j. B_i represents the cost of bandwidth to transfer a file from one access point at the edge of network i and the point where network i exchanges traffic with other networks.	$\frac{\$}{\text{byte}}$
D_i	demand for a given file originating or terminating on network i and $D = \sum_i D_i$.	$\frac{\text{requests}}{\text{sec}}$
F	file or object size.	$\frac{\text{bytes}}{\text{request}}$
K_i	costs associated with accounting and bill processing for network i.	$\frac{\$}{\text{request}}$
L	number of locations where a transit network has placed caches.	
M	number of cache providers contracted by a publisher.	
N_i	number of caches deployed by network i.	
$P_i(F)$	cost of processing capacity needed by network i to handle a single request for a file with a size of F bytes. In general, the processing cost depends on both the request rate and the file size. We assume that for a given file size, the processing cost increases linearly with demand, i.e., $P_i(D, F) = DP_i(F)$. We use $P_i = P_i(F)$ in all analyses.	$\frac{\$}{\text{request}}$
S_i	cost of storage for network i.	$\frac{\$}{\text{byte}\times\text{sec}}$
W_i	network access price charged by network i to a publisher.	$\frac{\$}{\text{byte}}$
X_{ij}	payment received by network j from network i when traffic is terminated in an eyeball network. $X_{ij} = -X_{ji}$	$\frac{\$}{\text{byte}}$

Table 2.6: Overview of notation used - Part II: Greek alphabets.

Symbol	Definition	Unit
$\alpha(N)$	additional increase in end-user demand for an object, which results from the latency and reliability benefits from delivering the object from N cache locations.	$\frac{\text{requests}}{\text{sec}}$
$\beta(N)$	net indirect benefits to the publisher for serving requests from N cache locations.	$\frac{\$}{\text{request}}$
γ_i	price charged (or paid) by network (publisher) i for caching services.	$\frac{\$}{\text{byte}}$
η_i	cache hit ratio in network i. The corresponding miss ratio is denoted by μ_i	
$\sigma(N)$	the revenue that the network obtains from additional end-user demand for an object that results from the latency and reliability benefits from delivering the object from N cache locations.	$\frac{\$}{\text{byte}}$
$\tau_{_F}(M)$	fixed component of transaction costs when a publisher contracts with M cache providers. $\tau_F(M) \approx M\tau_{_F}$.	$\frac{\$}{\text{sec}}$
$\tau_{_V}(M)$	variable component of transaction costs when a publisher contracts with M cache providers. We assume that $\frac{d\tau_{_V}(M)}{dM} > 0$.	$\frac{\$}{\text{request}}$
ϕ	the minimum demand for an object that makes transparent caching economically viable	$\frac{\text{requests}}{\text{sec}}$

A.2 Publisher's incentives to pay for caching services

Publisher does not purchase caching services

Costs[6]

$$C_{pub}^{nocache} = \underbrace{[D \times F \times W \times T]}_{\text{network access cost}} + \underbrace{[P \times D \times T]}_{\text{processing costs}}$$

Revenues

$$R_{pub}^{nocache} = \underbrace{R}_{\text{revenue from serving requests without caches}}$$

Profits

$$\Pi_{pub}^{nocache} = R - DFWT - PDT$$

Publisher purchases caching services

Costs

$$C_{pub}^{cache} = \underbrace{[\mu \times D \times F \times W \times T]}_{\text{network access cost to service cache misses}} + \underbrace{[\mu \times D \times P \times T]}_{\text{processing costs}}$$

$$+ \underbrace{[\gamma_P \times \eta \times D \times F \times T]}_{\text{content delivery costs}} + \underbrace{[M \times \tau_F \times T + \eta \times D \times \tau_V(M) \times T]}_{\text{transaction costs}}$$

[6] We do not consider the publisher's storage costs because we assume it is the same regardless of whether the publisher purchases caching services or not.

Revenues

$$R_{pub}^{cache} = \underbrace{R}_{\substack{\text{revenue from serving} \\ \text{requests without caches}}} + \underbrace{\beta(N) \times D \times T}_{\substack{\text{additional benefits from serving} \\ \text{requests from N cache locations}}}$$

Profits

$$\Pi_{pub}^{cache} = R + \beta(N)DT - \mu DFWT - \mu DPT - \gamma_{P}\eta DFT - M\tau_{F}T - \tau_{V}(M)\eta DT$$

Publisher's willingness to pay for caching services

The publisher pays for caching services only when the profits it obtains with caching exceeds the profits it obtains without caching services (i.e., $\Pi_{pub}^{cache} \geq \Pi_{pub}^{nocache}$). From the equations above, we see that the publisher is willing to purchase caching services when it pays a price that satisfies the following:

$$\gamma_{P} \leq \underbrace{W + \frac{\beta(N)}{\eta F} + \frac{P}{F}}_{\text{marginal benefits}} - \left(\underbrace{\frac{M\tau_{F}}{\eta DF} + \frac{\tau_{V}(M)}{F}}_{\text{marginal costs}} \right) .$$

For very popular documents, where D is very large and $\eta \approx 1$, the above condition simplifies to:

$$\gamma_{P} \leq W + \frac{\beta(N)}{F} + \frac{P}{F} - \frac{\tau_{V}(M)}{F}$$

A.3 Eyeball network's incentives to offer caching services

Let us assume that an eyeball network (Network E) has a monopoly in the client access market. Further, let us a assume that the publisher buys access from a transit network (Network A) who peers with the eyeball network and only performs hot-potato routing.

Eyeball network does not offer caching services

Lets assume that the demand for an object, i, is given by:

$$D_i = a + \alpha_i(N) - bF_iW_{eu} \ ,$$

where a and b are positive constants; and $\alpha_i(N)$ represents the additional end-user demand for object i, which results from the latency and reliability benefits from delivering the object from N cache locations. We assume that $\alpha_i(N) > 0$ and $\alpha_i(0) = 0$. F_i represents the size of object i and W_{eu} denotes the network access charge levied on the end-user. Because we consider a single object, we drop the subscripts in the following analysis.

When the eyeball network has a monopoly in the client access market, it will set W_{eu} in order to maximize its profits. In this case, the costs, revenues and corresponding profits are given below.

Costs

$$C_{NetE}^{nocache} = \underbrace{[D \times F \times B_E \times T]}_{\text{transport cost}} + \underbrace{[D \times F \times X_{EA} \times T]}_{\text{transit}} + \text{Other Costs}$$

Revenues

$$R_{NetB}^{nocache} = \underbrace{[W_{eu} \times D \times F \times T]}_{\text{network access}}$$

Profits

In order to obtain monopoly profits, the eyeball network sets the client access charges as:

$$W_{eu} = \frac{1}{2}\left(\frac{a}{bF} + B_E + X_{EA}\right) .$$

The corresponding monopoly profit is given by:

$$\Pi_{NetE}^{nocache} = \frac{1}{2}\left(\frac{a}{bF} - (B_E + X_{EA})\right) DFT - \text{Other Costs} .$$

Eyeball network offers caching services

Costs[7]

$$\begin{aligned} C_{NetE}^{cache} &= \underbrace{[\mu_{_E} \times D \times F \times B_E \times T]}_{\text{transport for cache misses}} + \underbrace{[\mu_{_E} \times D \times F \times X_{EA} \times T]}_{\text{transit for cache misses}} \\ &+ \underbrace{[S_E \times F \times T]}_{\text{storage}} + \underbrace{[P_E \times D \times T]}_{\text{file processing}} + \underbrace{[\eta_{_E} \times D \times K_E \times T]}_{\text{accounting and bill processing}} \\ &+ \text{Other Costs} \end{aligned}$$

[7]We neglect the bandwidth cost to transport the file from the cache location to the client because it is incurred with and without a cache.

Revenues

$$R_{NetE}^{cache} = \underbrace{[W_{eu} \times D \times F \times T]}_{\text{network access}} + \underbrace{[\gamma_{E} \times \eta_{E} \times D \times F \times T]}_{\text{content delivery}}$$

Profits

The eyeball network obtains monopoly profits by setting the end-user access prices as:

$$W'_{eu} = \frac{1}{2}\left(\frac{a+\alpha(N)}{bF} + \frac{\eta_{E}K_{E}}{F} + \frac{P_{E}}{F} + \mu_{E}\left(B_{E}+X_{EA}\right) - \eta_{E}\gamma_{E}\right) .$$

The monopoly profits that the eyeball network earns is then given by:

$$\begin{aligned}\Pi_{NetB}^{cache} &= \frac{1}{2}\left(\frac{a+\alpha(N)}{bF} + \eta_{E}\gamma_{E} - \left(\frac{\eta_{E}K_{E}}{F} + \frac{P_{E}}{F} + \mu_{E}(B_{E}+X_{EA})\right)\right)DFT \\ &\quad -S_{E}FT - \text{Other Costs} ,\end{aligned}$$

Eyeball network's willingness to sell caching services

The eyeball network will only offer caching services when the profits it obtains from doing so exceeds the profits it obtains without providing caching services (i.e., $\Pi_{NetE}^{cache} \geq \Pi_{NetE}^{nocache}$). Therefore, it will provide caching services if it can charge a price given by

$$\gamma_{E} \geq \underbrace{\frac{P_{E}}{\eta_{E}F} + \frac{2S_{B}}{\eta_{E}D} + \frac{K_{E}}{F}}_{\text{incremental costs from caching}} - \underbrace{\left(B_{E}+X_{EA}+\sigma(N)\right)}_{\text{incremental savings from caching}} ,$$

where $\sigma(N) = \frac{\alpha(N)}{\eta_E bF}$. For a very popular object, this can be simplified as

$$\gamma_E \geq \underbrace{\frac{P_E}{F} + \frac{K_E}{F}}_{\text{incremental costs from caching}} - \underbrace{(B_E + X_{EA} + \sigma(N))}_{\text{incremental savings from caching}} .$$

On the other hand, if we assume that the end-user access market is fairly competitive, then the eyeball network must recover the costs of providing caching from the publisher. Under this scenario (and following a similar reasoning as in as above), the eyeball network will set a price for caching that satisfies:

$$\gamma'_E \geq \frac{P_E}{\eta_E F} + \frac{S_E}{\eta_E D} + \frac{K_E}{F} - (B_E + X_{EA}) \ .$$

As the cache hit ratio approaches unity for a very popular object, the above equation simplifies to:

$$\gamma'_E \geq \frac{P_E}{F} + \frac{K_E}{F} - (B_E + X_{EA}) \ .$$

Thus, the price charged for caching in the case of monopoly and in a competitive market are related by

$$\gamma'_E - \gamma_E \geq \sigma(N) \ .$$

In other words, the eyeball network can charge lower prices for caching services when it has a monopoly in the access market because it can recover some of the cost of providing caching services by charging end-users for the improved services from caching.

A.4 Transit network's incentives to offer caching services

We assume that the publisher purchases transit services from the transit provider. We consider the simple case where all end-users are located in a single eyeball network. Extension to multiple eyeball networks is straightforward.

Transit network does not offer caching services

In the absence of caching, the transit network performs hot-potato routing (i.e., exchanges traffic with the eyeball network at NAP1 in Figure2.1).

Costs

$$C_{NetA}^{nocache} = \underbrace{[D \times F \times B_A \times T]}_{\text{transport cost}}$$

Revenues

$$R_{NetA}^{nocache} = \underbrace{[D \times F \times W_A \times T]}_{\text{network access}} + \underbrace{[D \times F \times X_{EA} \times T]}_{\text{transit}}$$

Profits

$$\Pi_{NetA}^{nocache} = W_A DFT + X_{EA} DFT - B_A DFT$$

Transit network offers caching services

Costs

$$C_{NetA}^{cache} = \underbrace{[\mu_{_A} \times D \times F \times (B_A + B_{AE}) \times T]}_{\text{transport for cache misses}} + \underbrace{[S_A \times F \times T]}_{\text{storage}}$$

$$+ \underbrace{[P_A \times D \times T]}_{\text{file processing}} + \underbrace{[\eta_A \times D \times K_A \times T]}_{\text{accounting and bill processing}}$$

Revenues

$$R_{NetA}^{nocache} = \underbrace{[\mu_A \times D \times F \times W_A \times T]}_{\text{network access for cache misses}} + \underbrace{[D \times F \times X_{EA} \times T]}_{\text{transit}} + \underbrace{[\gamma_A \times \eta_A \times D \times F \times T]}_{\text{content delivery}}$$

Profits

$$\Pi_{NetA}^{cache} = \left(\mu W_A + X_{EA} + \gamma_A \eta_A - \mu_A \left(B_A + B_{AE}\right) - \eta_A \frac{K_A}{F} - \frac{P}{F} - \frac{S_A}{D}\right) DFT$$

Transit network's willingness to sell caching services

The transit network offers caching services only when $\Pi_{NetA}^{cache} \geq \Pi_{NetA}^{nocache}$. This implies that caching becomes desirable when the price paid for caching satisfies

$$\gamma_A \geq W + \frac{P_A}{\eta_A F} + \frac{S_A}{\eta_A D} + \frac{K_A}{F} + \frac{1-\eta_A}{\eta_A} B_{AE} - B_A \ .$$

For very popular documents, where the cache hit ratio is close to one, this condition simplifies to

$$\gamma_A \geq W + \frac{P_A}{F} + \frac{K_A}{F} - B_A \ .$$

A.5 Content distribution network's incentives to offer caching services

Unlike eyeball and transit networks, a content distribution network (CDNs) obtains revenues in our model only when an object is delivered from a cache. In practice, CDNs can obtain revenue streams from other value added services (e.g., through bundling of services) but we do not consider that since our focus is on content delivery. In this case, it is straightforward to derive (for a single eyeball network) that the CDN finds caching desirable when it can obtain a payment flow which satisfies

$$\gamma_C \geq \frac{P_C}{\eta_C F} + \frac{S_C}{\eta_C D} + \frac{K_C}{F} + X_{CE}.$$

Chapter 3

Social Welfare Implications of Different Cache Deployment Scenarios

3.1 Introduction

Content delivery from distributed storage nodes within the network provides both an evolutionary (e.g., CDNs) and a revolutionary (e.g., CCN) means to deal with the proliferation of content retrieval on the Internet. Despite the important role played by distributed storage nodes in a CCN, it is not yet obvious who should deploy them. Further, the social welfare implications of different deployment scenarios are not well understood.

In this chapter, we use the model described in Chapter 2 to study the welfare implications of different cache deployment scenarios. We show that the level of caching supplied with transparent caching is inefficient and results in a deadweight loss. With a payment flow from publishers for content delivery, the level of caching supplied increases because caching service providers can extract some of the publisher surplus for themselves. Moreover, we show two deployment scenarios that maximize social welfare when a payment flow exists between the publisher and the caching service provider.

In the first, eyeball networks provide the caching infrastructure through a federation or a third-party entity that serves as a transaction broker between publishers and various networks. In the second, eyeball networks pay content delivery networks an amount, equivalent to the realized bandwidth and transit savings from caching, to deploy caches. Even though both deployment scenarios maximize welfare, we show that the latter deployment scenario is unlikely to be realized in practice because eyeball networks do not share in the resulting value created. This explains the emergence of caching federations and content distribution networks (CDNs) becoming brokers for eyeball network caching services.

The rest of this chapter is organized as follows. In Section 3.2, we evaluate the social welfare implications of various cache deployment scenarios and how the value created from caching gets distributed among different network players. We follow this with a discussion of the implications of our findings and the limitations of our model in Section 3.3. We conclude the chapter in Section 3.4.

3.2 Social welfare implications

The manner in which the value created from any cache deployment scenario gets distributed among the different network players affects the incentives for the network players to realize that deployment scenario. In this section, we study how the value created from caching gets distributed among the different network players under four cache deployment scenarios.

In the first scenario, we consider the case where an eyeball network deploys only transparent caching. In the second, we consider the case where an eyeball network undertakes paid content delivery through a transaction broker (e.g., as part of an eyeball federation or through a third-party broker). In the third, the CDN performs all the caching within the network, but pays eyeball networks to co-locate and terminate traffic. Finally, we consider the case where eyeball networks pay CDNs an amount, equivalent to the realized bandwidth savings and additional monetary benefits, to perform caching within the network.

3.2.1 Computing surplus, social welfare and deadweight loss

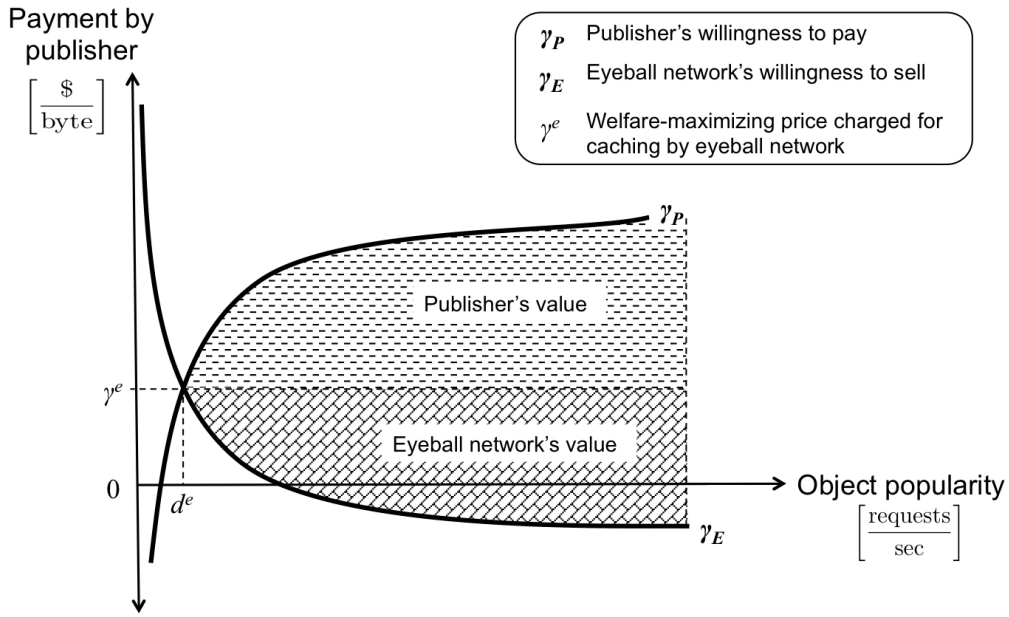
Our notions of social welfare and value are based on the surplus that accrues to various network participants. In our model, the value created for end-users does not depend on the cache provider and remains the same under different deployment scenarios. This is because end-users pay the same network access fee and enjoy the same level of performance, regardless of the entity that performs content delivery. Thus, we only consider the surplus for publishers, CDNs and eyeball networks, which depend on the cache deployment

scenario. We use Figure 3.1 to illustrate how we compute the publisher surplus, eyeball network surplus, CDN surplus, overall social welfare and deadweight loss when all requests originate from a single eyeball network (the scenario studied in Section 2.4.1). We only consider a single publisher for ease of exposition, but an extension to multiple publishers is straightforward.

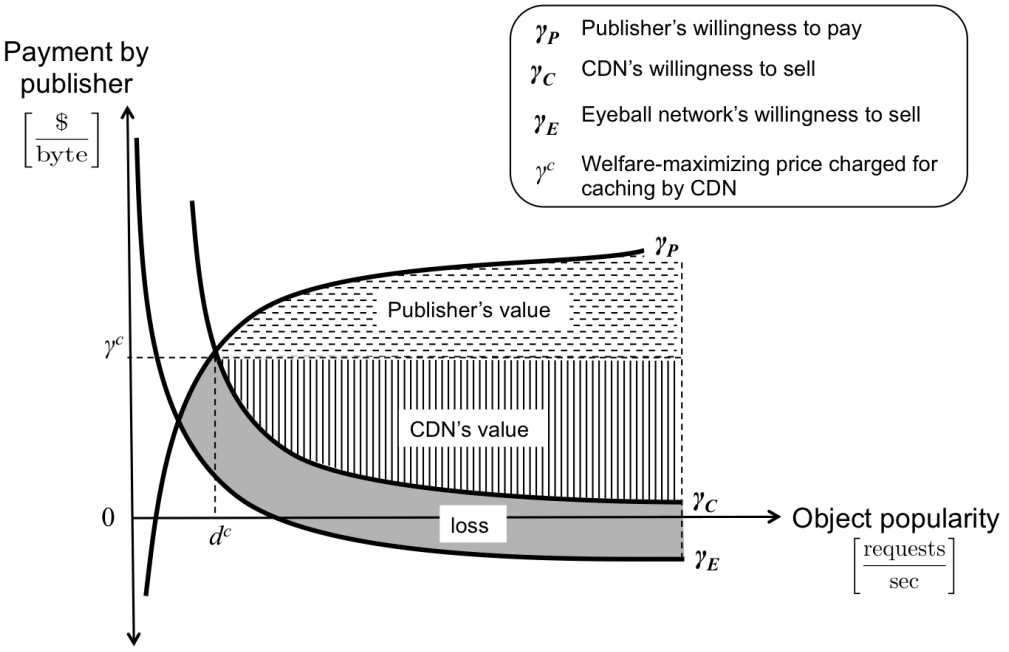
In our analysis in Chapter 2, we considered only a single object. However, in computing the surplus, we need to take all objects into account. Let us assume that there exists a continuously differentiable function $g(D)$, which gives the number of objects that have popularity D. Several studies show that object popularity follows a Zipf-like distribution [142–147]. Thus, we assume that $g(D)$ possesses similar characteristics, with very few objects exhibiting very high popularity and the majority of objects exhibiting low popularity. Given this function and an average object size, F bytes, for all objects, and assuming that object size and object popularity are uncorrelated (see e.g., [142]), we compute the surplus that accrues to different players and the overall social welfare as follows.

When the eyeball network performs caching and charges a welfare-maximizing price, γ^{e} dollars per byte, then the overall social welfare, in dollars per second, can be computed from Figure 3.1(a) as:

$$\begin{aligned}\text{Social welfare in Figure 3.1(a)} &= \underbrace{\int_{d^e}^{\infty} \left(\gamma_{P}(\varphi) - \gamma^{e}\right) F g(\varphi) d\varphi}_{\text{Publisher surplus}} + \underbrace{\int_{d^e}^{\infty} \left(\gamma^{e} - \gamma_{E}(\varphi)\right) F g(\varphi) d\varphi}_{\text{Eyeball network surplus}}, \\ &= \int_{d^e}^{\infty} \left(\gamma_{P}(\varphi) - \gamma_{E}(\varphi)\right) F g(\varphi) d\varphi \,. \qquad (3.1)\end{aligned}$$



(a) Value created for different network players when an eyeball network performs caching



(b) Value created for different network players when a CDN performs caching and co-locates for free

Figure 3.1: Illustration of how the surplus, overall social welfare and deadweight loss are computed. When the eyeball network performs caching, the social welfare is computed from the areas that correspond to the publisher value and the eyeball network value in Figure 3.1(a). When the CDN performs caching and co-locates for free, the network incurs a deadweight loss, which is computed from the loss region in Figure 3.1(b).

Similarly, we compute the social welfare when the CDN performs caching and charges welfare-maximizing prices as:

$$\text{Social welfare in Figure 3.1(b)} = \underbrace{\int_{d^c}^{\infty} \left(\gamma_P(\varphi) - \gamma^c\right) Fg(\varphi)d\varphi}_{\text{Publisher surplus}} + \underbrace{\int_{d^c}^{\infty} \left(\gamma^c - \gamma_C(\varphi)\right) Fg(\varphi)d\varphi}_{\text{CDN surplus}},$$

$$= \int_{d^c}^{\infty} \left(\gamma_P(\varphi) - \gamma_C(\varphi)\right) Fg(\varphi)d\varphi \ . \tag{3.2}$$

In this case, the deadweight loss, which is a function of the loss region in Figure 3.1(b) is given by

$$\text{Deadweight loss} = \underbrace{\int_{d^e}^{\infty} \left(\gamma_P(\varphi) - \gamma_E(\varphi)\right) Fg(\varphi)d\varphi}_{\substack{\text{Welfare from} \\ \text{eyeball network caching}}} - \underbrace{\int_{d^c}^{\infty} \left(\gamma_P(\varphi) - \gamma_C(\varphi)\right) Fg(\varphi)d\varphi}_{\text{Welfare from CDN caching}} \tag{3.3}$$

The integrals above are bounded because of the Zipf-like nature of $g(D)$. In the rest of this section, we will use figures similar to Figure 3.1 to discuss the surplus that accrues to different players, the overall social welfare and the deadweight loss from different cache deployment scenarios. In these discussions, we will use value and surplus interchangeably because the surplus is a monotonically increasing function of the value region. But strictly speaking, the surplus is computed from the value region using the equations described above.

3.2.2 Eyeball network transparent caching

In transparent caching, the eyeball network caches content purely for cost savings and performance improvement on its network and does not demand any form of payment from the publisher. It is done without any knowledge (or permission) from the publisher of the content [1]. Transparent caches could take the form of router buffer memory or content stores envisaged in content-centric networking (see [3]). They could also take the form of a dedicated cache hierarchy attached to routers. Network vendors, such as Cisco and Juniper, are pushing transparent caching solutions for network operators and they have gained a lot of attention lately among eyeball networks [126, 148].

In Figure 3.2, we illustrate the distribution of the resulting value when an eyeball network deploys transparent caching. We see that the publisher and the eyeball network share in the value created from caching. The actual positions of the willingness to pay and the marginal cost curve determines the share of the value that accrues to each entity. However, this deployment scenario fails to realize the maximum value from caching, since the eyeball network refrains from caching objects whose request rate fall below ϕ. This leads to a welfare loss, as shown in Figure 3.2. Additionally, the publisher loses vital information about who accesses the content, when they accessed it and how they accessed it.

It has been suggested that publishers label their content as uncacheable in order to avoid transparent caching. Yet, several networks are known to ignore such instructions

[1] See [126, 148, 149] for the distinction between transparent caching and other forms of caching.

[149]. This indicates that publishers may be willing to give up some of the surplus from transparent caching in order to obtain such information. Hence, it is reasonable to expect some sort of payment flow between publishers and eyeball networks to incentivize eyeball networks to cache objects whose popularity is less than ϕ, and also to provide content access and delivery information to publishers.

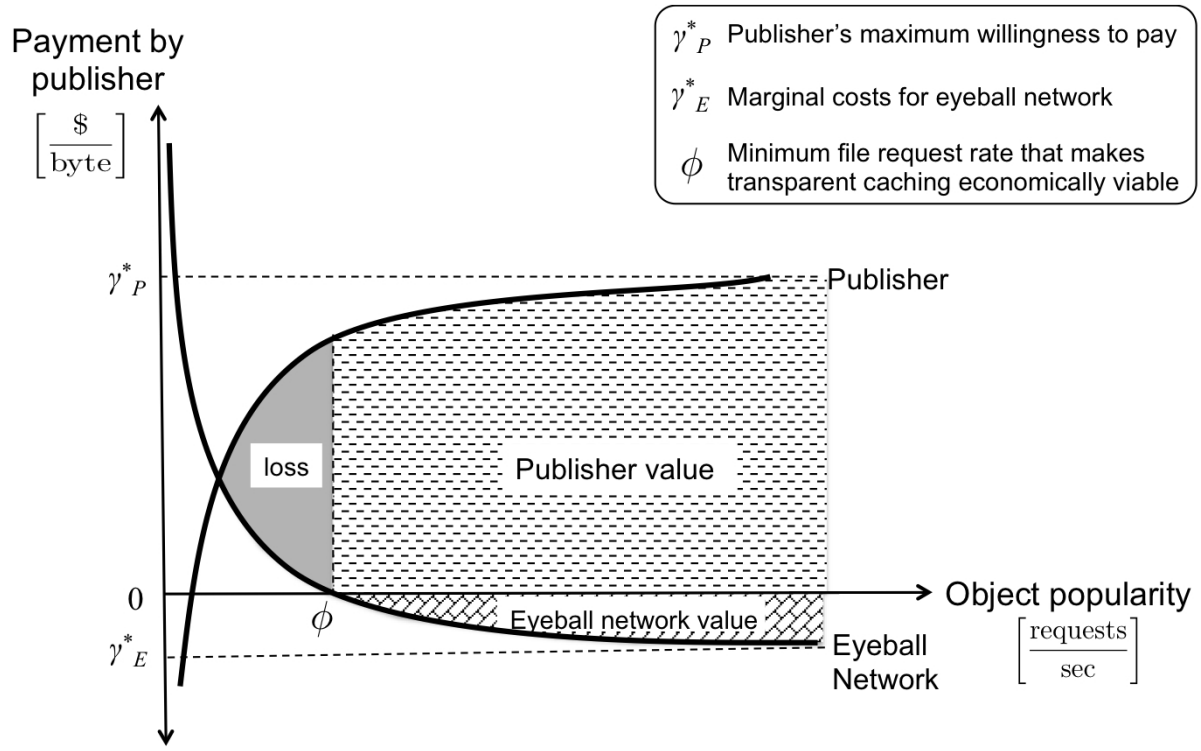


Figure 3.2: Transparent caching fails to capture all the possible value from caching because only objects whose request rate exceed ϕ gets cached. The publisher and the eyeball network share in the value created from transparent caching. However the publisher loses information about who accessed the content, how they accessed it and when they accessed it. The publisher may be willing to give up some of its surplus in order to obtain this information. Hence, it is reasonable to expect some sort of payment flow between publishers and eyeball networks to incentivize eyeball networks to cache objects whose popularity is less than ϕ, and also to provide content access and delivery information to publishers.

3.2.3 Eyeball network commercial caching

We consider the case where an eyeball network deploys its own caching infrastructure and offers paid caching services through a transaction broker. We assume that there is sufficient competition in the transaction broker market. Thus, we do not consider rent-seeking behavior from transaction brokers. Several authors argue that a federated content delivery model reduces transaction costs for publishers and could make eyeball networks competitive with CDNs [128]. In fact, we are already beginning to see efforts by networks towards the formation of such a federation, such as the Open Carrier Exchange (OCX) [128]. With several traditional CDN vendors offering licensed and managed solutions for CDN technology (e.g., Edgecast, Limelight Deploy and Akamai Aura Network Solutions), this seems like a feasible deployment scenario for eyeball networks.

In Figure 3.3, we show how the value from such a cache deployment is shared among the network players. Again, we see that the publisher and the eyeball network share in the value created. Unlike the transparent caching deployment scenario, however, the network as a whole realizes the maximum value from caching when the eyeball network charges a welfare-maximizing price for caching services. This is because the eyeball network caches all objects for which a payment flow from the publisher makes content delivery from the cache economically viable. Thus, the eyeball network caches more objects than it does when it performs transparent caching, which creates additional value in the form of bandwidth and transit savings and monetizable improvements in end-user latency. Moreover, the publisher obtains content access information by virtue of paying for content delivery.

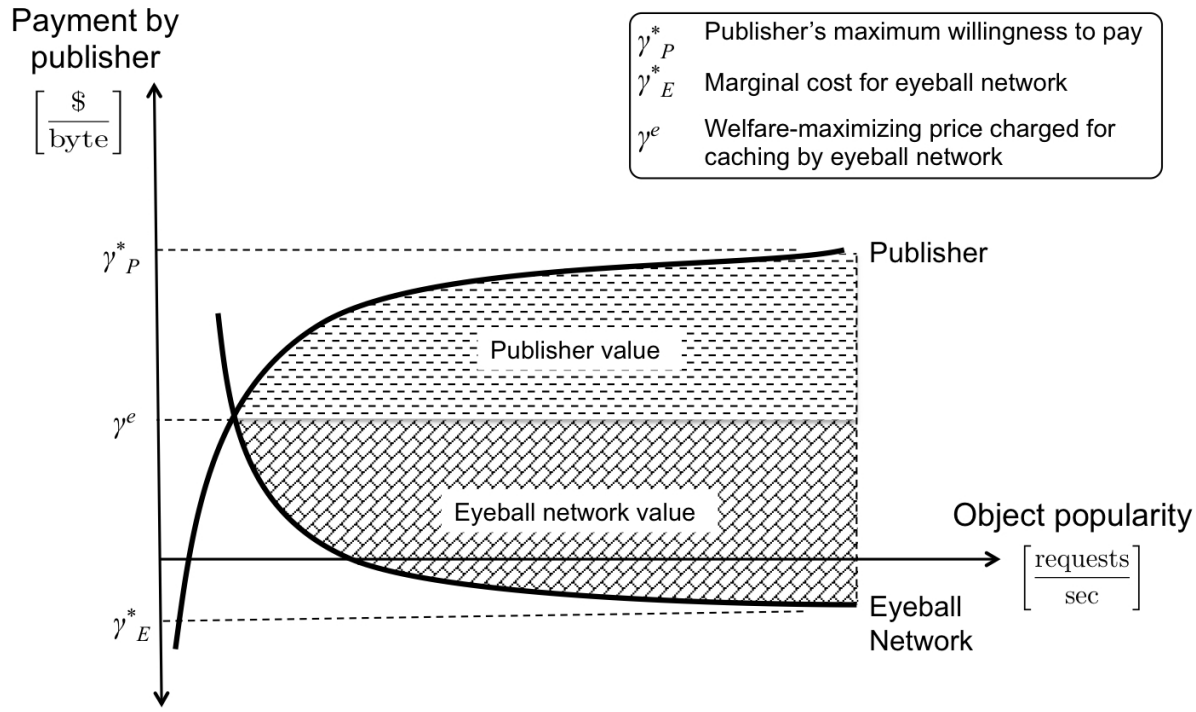
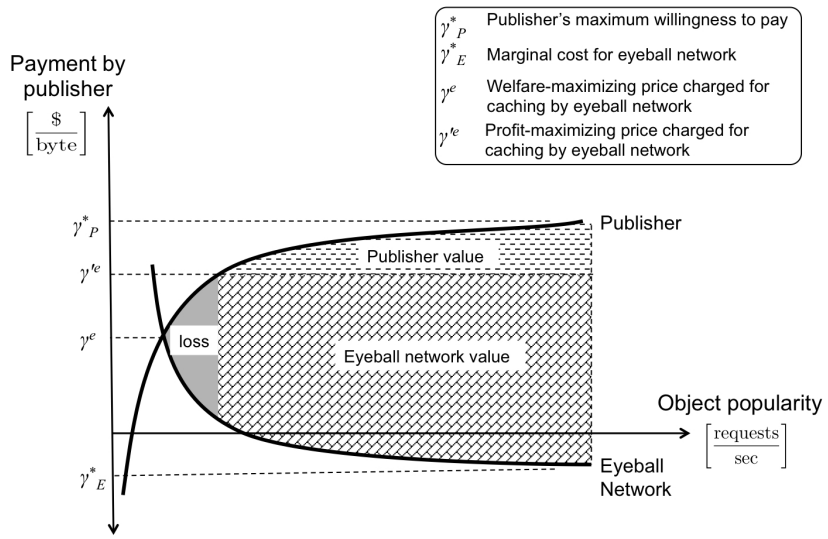


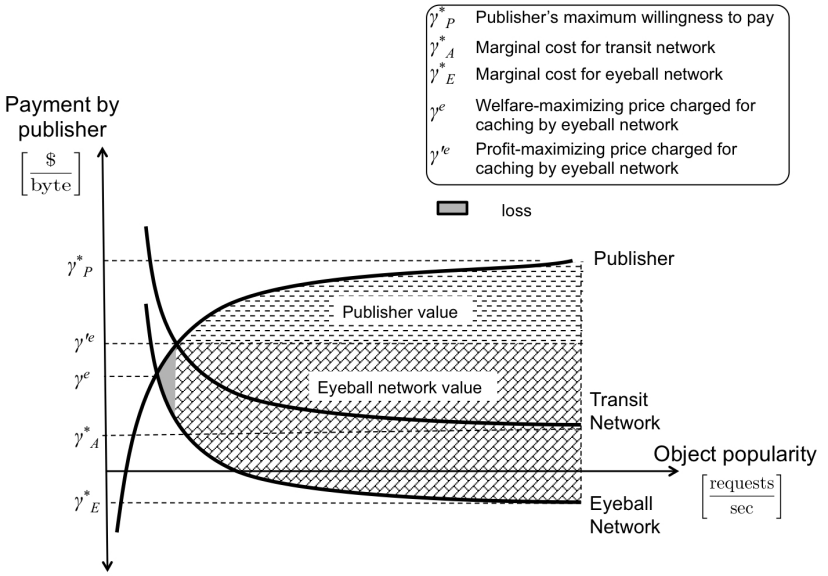
Figure 3.3: When an eyeball network deploys caches and charges welfare-maximizing prices for content delivery, then the network realize the maximum social welfare. This is because the eyeball network caches all objects for which a payment flow from the publisher makes content delivery from the cache economically viable. In this case, the created value is shared between the publisher and the eyeball network.

In practice, a profit-maximizing eyeball network may not have incentives to charge welfare-maximizing prices. As shown in Figure 3.4, charging prices other than the welfare-maximizing price, γ^e, leads to a deadweight loss. The profit-maximizing price, γ'^e, which the eyeball network can charge in equilibrium is bounded by two constraints. First, the eyeball network raises prices until the additional surplus it obtains is exactly offset by its share of the additional deadweight loss created from the price increase. The second bound on the price occurs only when the eyeball network lacks market power in setting termination prices in transit arrangements. In this case, the profit-maximizing price is also bounded by the price that the transit network with the lowest costs will charge for caching. The price charged by the profit-maximizing eyeball network in practice, γ'^e, will be the

lower of the two prices in Figure 3.4.



(a) Constraint 1: The eyeball network sets the profit-maximizing price at the point where the additional surplus it obtains is exactly offset by its share of the additional deadweight loss created from a price increase



(b) Constraint 2: In the absence of market power, the eyeball network sets the profit-maximizing price at the same price point as the lowest cost transit network, if that price is lower than that in Figure 3.4(a)

Figure 3.4: In practice, the profit-maximizing price charged by the eyeball network, γ'^e, will be different from the welfare-maximizing price, γ^e. The profit-maximizing price is bounded by two constraints. First, the eyeball network raises prices until the additional surplus it obtains is exactly offset by its share of the additional deadweight loss created from the price increase. Second, in the absence of market power in setting termination charges, the maximum price that the eyeball network can charge for caching is bounded by the price that will be charged for caching by the transit network with the lowest costs.

When the eyeball network possesses significant market power, then only the first constraint limits the profit-maximizing price. Therefore, it is important that regulators take steps to ensure that eyeball networks do not obtain too much power in setting termination charges for transit networks. One way to do this may be to provide guidelines on termination and co-location charges, if the market fails to yield a favorable socio-economic outcome. By doing so, the potential deadweight loss that arises when the eyeball network sets profit-maximizing prices for caching can be reduced. Another way to achieve the welfare-maximizing outcome is to facilitate competition in the end-user access market. A competitive end-user access market forces eyeball networks to charge welfare-maximizing prices.

3.2.4 CDN pays eyeball network to co-locate caches

Currently, this deployment scenario is arguably the most common in the Internet. We illustrate the value that accrues to different players in Figure 3.5. We see that the CDN, the publisher and the eyeball network all share in the value created. The value obtained by the CDN comes from charging publishers for content delivery. On the contrary, the value that the eyeball network obtains comprises bandwidth and transit savings, as well as, revenues from co-location and termination fees. In effect, the eyeball network captures some of the CDN's surplus by levying a fee to co-locate and terminate traffic, in addition to the bandwidth and transit savings it realizes from the cache.

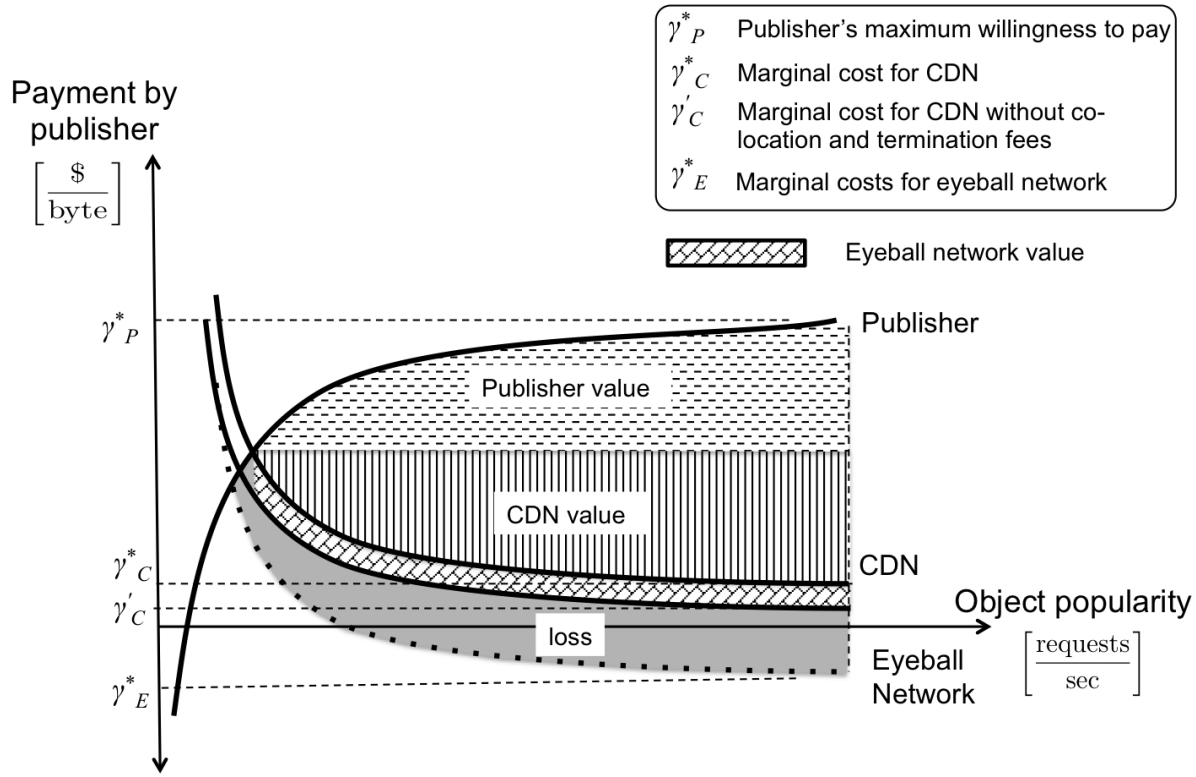


Figure 3.5: The network incurs a deadweight loss when the CDN pays an eyeball network to co-locate and terminate traffic. The deadweight loss comprises two parts. The first part results from the CDN paying to co-locate and terminate traffic within the eyeball network, whereas the second part arises because the eyeball network fails to pay the CDN to co-locate and terminate traffic.

Although currently popular, we see from Figure 3.5 that this deployment scenario leads to an inefficient result because of the deadweight loss. The deadweight loss comprises two parts. The first part results from the CDN paying to co-locate and terminate traffic within the eyeball network, whereas the second part arises because the eyeball network fails to pay the CDN to co-locate and terminate traffic. In practice, the eyeball networks will extract as much value as possible from the CDNs by raising the co-location and termination charges till the point where the additional deadweight loss created offsets the additional gain in surplus or till the point where using transit networks become more competitive than CDNs. This suggests that eyeball networks can improve overall social welfare, albeit at the expense

of their share of the resulting value, by allowing CDNs to co-locate and terminate traffic for free in their networks.

3.2.5 Eyeball network pays CDN to deploy caching

Finally, we consider the situation where an eyeball network pays a CDN to deploy caching within the network. This deployment scenario reflects only a subset of current reality. For instance, this scenario occurs when a small eyeball networks host a CDN in their network and provide rack space, cooling, power and other necessities without charge. We extend this deployment scenario to an extreme case where an eyeball network pays a CDN an amount, equivalent to the realized bandwidth and transit savings and other monetary benefits, to deploy caching within the network. This has the effect of pushing the CDN cost curve down to the point where it overlaps with the cost curve of the eyeball network. Admittedly, this deployment scenario does not fully reflect current reality. However, we consider it to show an alternate way to maximize social welfare.

We show the distribution of value from this deployment scenario in Figure 3.6. Even though this deployment scenario also maximizes social welfare, we see immediately from the figure that it will not be realized in practice because only the CDN and the publisher share in the resulting value created.

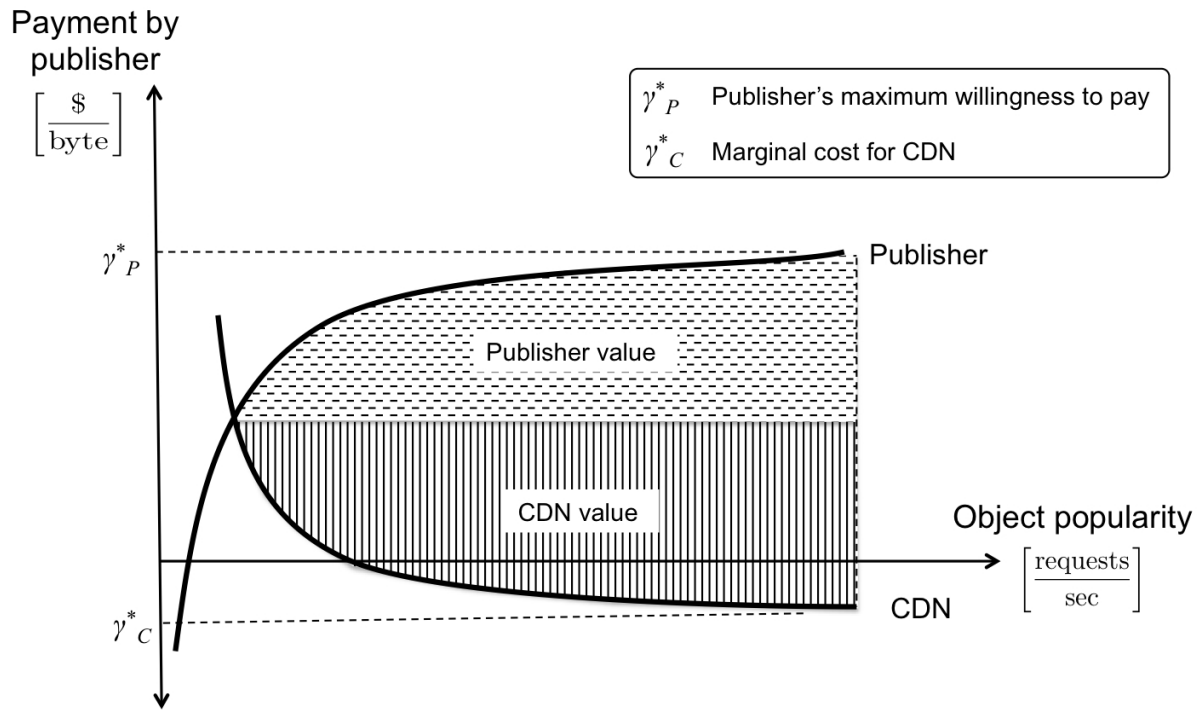


Figure 3.6: Social welfare is maximized when the eyeball network pays the CDN an amount, equivalent to the realized bandwidth and transit savings and the additional monetary benefits, to deploy caches. Even though this situation maximizes social welfare, eyeball networks do not have incentives to realize this outcome because only the publisher and the CDN share in the value created.

3.3 Discussion

Our analysis shows that when networks charge welfare-maximizing prices for caching, then it is welfare-maximizing for the eyeball network to undertake content delivery from caches when requests for a publisher's content mainly originate from one geographic area. On the other hand, a CDN should undertake content delivery when requests for a publisher's content originate from multiple eyeball networks and eyeball networks offer caching services individually. Better still, the eyeball network should pay the CDN to do this, in order to realize the maximum social welfare. However, we have seen that such a deployment scenario will not be realized in practice because the eyeball network does not share in the value

created. Another deployment scenario that maximizes social welfare, and the one that is more likely to be realized in practice is for eyeball networks to offer caching services as part of a federation or through a third-party broker.

Therefore, it seems likely that large eyeball networks will prefer to offer paid content delivery services for publishers through a transaction broker, in order to appropriate as much of the value from caching as possible. Medium and small eyeball networks may prefer to deploy transparent caching solutions or host CDNs in their networks for free. In this way, they still share in the value created from caching without incurring any significant costs. The current trend in the market, where established CDNs offer licensed and managed technologies for content delivery could lower any technological barriers to entry. The only stumbling block, perhaps, could be the form that the transaction broker for eyeball networks would take in a future CCN-based ecosystem.

It also remains to be seen how an eyeball network caching federation will function. Several challenges need to be overcome. For instance, standards to facilitate content delivery, accounting and reporting must be developed. In addition, transparent mechanisms to allocate costs and share revenues must also be put in place. Furthermore, a federation consisting of many large eyeball networks may face legal challenges on the potential anti-competitive effects of such a federation. Perhaps, traditional CDNs could serve as a third-party broker for their deployed licensed and managed caching solutions for eyeball networks. This is an area that could benefit from more research.

As we pointed out earlier, publishers do not like the idea of transparent caching because they lose vital information about who accesses their content, when they accessed it or how they accessed it. This kind of information is useful for advertisement purposes, among other things. Thus, publishers mark their content as uncacheable to prevent transparent caching [110, 115, 116]. Nonetheless, it is known that some networks routinely ignore such instructions, in order to improve the performance of transparent caches [149]. It is very likely that publishers and networks will clash over this issue, as transparent caching becomes more widespread in a CCN-based architecture.

For instance, many proposals for a future Internet architecture, such as named-data networking (NDN) [3], envision on-board storage on routers (called content stores) where content gets transparently cached. Nevertheless, these architectures do not provide mechanisms to deal with the potential tussle highlighted above. Hence, network architects and policy makers may need to design elegant solutions to resolve the tussle between the network's need for bandwidth and transit savings and the publisher's need to obtain content delivery information.

3.4 Conclusion

In this chapter, we evaluated the social welfare implications of different cache deployment scenarios to support content delivery. We showed that the level of caching supplied with transparent caching is inefficient and leads to a deadweight loss. By instituting side payments for content delivery, the network gets incentivized to deploy sufficient level of

caching. We also showed that when payment flows exist between the publisher and the caching service provider and all requests for a publisher's content originate from a single eyeball network, then it is welfare-maximizing for the eyeball network to provide caching services.

We further showed that the network realizes the maximum social welfare in two scenarios. In the first, eyeball networks deploy caching infrastructure through a federation or a single broker. In the second, eyeball networks pay CDNs to co-locate caches and terminate traffic in their networks. Moreover, we showed that the system would not reach the latter outcome because the value created is shared between the publisher and the CDN and not the eyeball network. We also highlighted the potential tussle between the network's need for bandwidth and transit savings and the publisher's need to obtain content delivery information. This tussle arises when the network undertakes transparent caching. Network architects may need to design solutions to facilitate the resolution of such tussles when they arise.

Looking ahead, we expect large eyeball networks to get into the content delivery business, as part of a federation or through a transaction broker, in order to appropriate most of the value created from caching. Medium and small eyeball networks would find transparent caching and hosting CDNs beneficial, even though it results in a deadweight loss. Thus, our model predicts a future where transparent caching, CDN caching and eyeball network caching all co-exist to meet different needs.

Chapter 4

Chunk Design to Support a CCN Content Delivery Ecosystem

4.1 Introduction

In order to be considered a viable networking model for the future, content-centric networking must not only provide new capabilities but also provide superior or at least equivalent means of performing currently useful functions.

As an example, CCN achieves desirable new robustness properties by decoupling content from its location [3, 74, 79]. At the same time, decoupling content from its location breaks some useful functionalities. For instance, publishers currently rely on the coupling between content and its location to obtain information about who accesses their content, when

they access it and how they access it. End-users also rely on this coupling to obtain some degree of trust in the content they retrieve. In addition, networks that deliver content on behalf of publishers for a fee, such as content delivery networks (CDNs), use technologies and accounting methods that rely on the coupling between content and location. Thus, by breaking the coupling between content and its location, CCN removes these useful functionalities, which must be replicated by some other means.

Several studies demonstrate that the implicit assumption of small storage onboard all routers for the purpose of transparent caching in CCN architectures will fail to provide significant gains in content routing efficiency and throughput [90–92, 100]. This is because objects may not stay long enough in the cache to make them useful for serving other requests. Larger storage units are needed to provide improvements in content routing efficiency [90].

We showed in Chapter 2 that networks will fail to deploy sufficient storage infrastructure to support CCN-based architectures unless there is some form of payment flow between publishers and other network participants. This need for payment flows makes it all the more important to replicate the desirable meta-information gathering, authentication, accounting and verification functionalities, which become broken when CCN decouples content from its location. The absence of these functionalities could seriously dampen the incentives of network stakeholders to deploy CCN-based architectures [150].

In this chapter, we address the issue of accounting and billing for content delivery in a content-centric network. Specifically, we identify key TCP/IP content delivery facilitators

that break in a CCN and propose a model that provides all desirable content delivery functionalities. We propose a structure for organizing content into basic routable units in a content-centric network. Using this structure in the context of the eXpressive Internet Architecture (XIA) [74], we define the minimum meta-information that every piece of routable content must carry in order to make it self-sufficient for the purposes of aiding network caching decisions and for accounting and billing for content delivery. We also specify the minimum information necessary in a content request in order to facilitate meta-information gathering for publishers and proper accounting and billing for content delivery by networks.

The rest of the chapter is organized as follows. In Section 4.2, we provide an overview of the TCP/IP content-delivery model, paying particular attention to the dependence of some desirable functionalities for various network players on the coupling between content and its location. In Section 4.3, we identify the basic functionalities that must be replicated in a content-centric network when key facilitators break. We describe our content-centric network model and propose a structure for organizing content into basic routable units and the minimum information necessary in every piece of routable content in Section 4.4. We follow this with a description of our CCN content delivery model in Section 4.5. In Section 4.6, we discuss the mechanisms provided by our model to account and bill for content delivery. We identify applicable lessons for other CCN models in Section 4.7. In Section 4.8, we provide an overview of some related work and we conclude in Section 4.9.

4.2 The TCP/IP content delivery model

Content transfer is the mainstay of several Internet applications, such as email and the world wide web. However, we will limit our attention to the web in the rest of this chapter. This is because the evolution of the web more accurately reflects the shift in Internet usage from setting up conversations between two hosts to information dissemination between multiple hosts, which CCN seeks to seamlessly facilitate. In Chapter 5, we generalize our findings to other content delivery applications besides the web.

We introduce two additional types of players in the current TCP/IP web content delivery model, namely search engine providers and end-users, in addition to the four players already identified in Chapter 2. Table 4.1 describes the roles and provides examples of each type of player. In practice, a single player could take on multiple roles in the content delivery chain. For example, Google serves as a publisher, a search engine provider and a content delivery network.

The content delivery process in the web is characterized by five distinct stages namely content preparation, content discovery, content request, content transfer and meta-information gathering. We describe each of these in turn in the next sections.

4.2.1 Content preparation

Content preparation involves all the activities performed before making content available to end-users. For instance, if a publisher wants to make a video clip available to multiple users

Table 4.1: Network players in the TCP/IP content delivery model for the web.

Network player	Role	Examples
Publisher	Produces content for end-users. Publishers could be simple one-man operations that cater to a few enthusiasts in a small locality. They could also be large corporations that serve content to millions of users worldwide. The content could be provided for free or for a fee.	Google, CNN, eBay, Netflix
Search engine provider	Provides a service that facilitates content discovery. Typically, an end-user enters a search query that describes the content of interest and the search engine either returns the content or provides a list of addresses that could possess that content.	Google, Yahoo!, Baidu, Microsoft
Eyeball network	Predominantly provides network access to end-users for a fee. Eyeball networks range in size from small networks serving a few hundred users to large networks serving millions of end-users.	Comcast Corporation, British Telecom, Deutsche Telekom, Telekom Italia, AT&T
Transit network	Primarily provides transport services to publishers and eyeball networks to reach the Internet for a fee. Top-tier transit networks use only peering relationships to reach the entire Internet, whereas lower-tier transit networks use a combination of peering and transit relationships to reach the whole Internet (see [49, 151] for more details about transit and peering relationships).	AT&T, Level 3, Tata Communications, NTT Communications, Deutsche Telekom, Centurylink
Content delivery network (CDN)	Caches and delivers content on behalf of publishers for a fee. They serve as transaction brokers between the publisher and various eyeball networks, eliminating the need for the publisher to negotiate with multiple eyeball networks for low-latency content delivery. The operations of CDNs could be local in scope, serving only a small geographic area. At the other end, CDNs could be global, spanning multiple countries and network locations.	Akamai Technologies, Limelight Networks, Level 3, Edgecast Networks
End-user	Runs an application on an end host that requests content from a publisher for consumption. The application could be run manually or automatically.	Human user, application

over the web, then the publisher first creates a web page that hosts the video. Alternatively, the publisher could host the video on an existing web page. Additionally, the publisher chooses the encoding scheme to use for the video. If the publisher is sophisticated enough, he may choose to support the needs of diverse end-users by encoding the video in multiple formats and using technologies such as Apple HTTP[1] Live Streaming (HLS) or Dynamic Adaptive Streaming over HTTP (DASH) to dynamically select the most appropriate format for each end-user [152, 153]. Choosing any of these technologies may impose additional requirements. For example, HLS requires each encoded video stream to be broken down into small ten-second segments, which are all stored and requested separately [152].

Arguably, specifying the uniform resource identifier (URI) for the content constitutes the most important activity during content preparation. The web uses a URI scheme, commonly called the uniform resource locater (URL), of the form *http://domain/path*, which identifies content by a coupling between the location of the content and the name given to the content by its publisher [154, 155]. The domain specifies the authoritative host that knows how to find the path to the content. The URL specified by the publisher depends on two factors. First, it depends on whether the publisher chooses to serve requests for the content from its own servers or pay a third-party CDN to deliver the content. For example, if the publisher serves the content from its servers, then the URL takes the form *http://publisher_domain/path_to_content*.

[1]The hypertext transfer protocol (HTTP) is the *de facto* application protocol for content request and retrieval on the web. More details about HTTP could be found in [24].

On the other hand, if the publisher pays a CDN to deliver the content, then the publisher pushes the content to the servers of the CDN [39, 41]. In this case, the URL depends on the technology used to direct requests to the appropriate CDN server. For example, the publisher could use the same URL as above and employ HTTP redirection to reroute content requests to the CDN [39, 40, 156]. With this scheme the publisher obtains access information for all content delivered by the CDN, at the price of additional latency.

Alternatively, the publisher could specify the CDN as the domain responsible for the content. In this case, the URL takes the form *http://cdn_domain/path_to_content*, where the path may contain the publisher's domain to aid the CDN in identifying the entity to bill for the content. The CDN uses DNS resolution techniques to locate the nearest server to respond to requests for the content [39, 41]. Although this approach reduces network latency, it deprives the publisher of content access information. Hence, the publisher must either rely on the CDN for access information or employ other techniques to obtain this information. For instance, the publisher could choose to serve the main web page from its servers and thus, use access to the main web page as a proxy for access to the content. The publisher could also employ the PING attribute in HTML5 for this purpose [157] .

Another item in content preparation involves the decision to make the content available to all users or restrict access to a specific group of users, perhaps to users who pay some fee. In the latter, the publisher must decide on the content protection scheme to use. For instance the publisher could limit access based on a user's IP address. However, this approach is not robust and is easy to circumvent [6]. In order to achieve better protection,

the publisher may encrypt the content with a single key that is common for all legitimate users. Alternatively, the publisher could encrypt the content on a per-user basis. The disadvantage of encrypting content on a per-user basis is that it makes transparent caching inefficient and useless [120]. The content preparation stage is completed when the publisher makes the content available on the Internet.

4.2.2 Content discovery

Once publishers make content available on the web, end-users can access the content with the right URL. We refer to the process of obtaining the correct URL for a desired content as content discovery. In general, end-users could obtain the URL for a piece of content in one of three ways. They could either recall the URL from memory or make an educated guess. The structured nature of URLs makes this a feasible approach. Alternatively, an end-user could follow a link provided by an entity with knowledge of the URL. This could be a friend, a website or a document, among other things. Better still, end-users could use search engines to discover URLs for content of interest. Studies show that most human end-users prefer to use search engines as the means to obtain URLs because they perceive them as convenient and safe [158].

4.2.3 Content request

A web client located on the end-user's machine uses HTTP to request content [24]. The HTTP request message contains the path to the content, the domain name (host) of the

publisher or its authorized delegate and other parameters. The request message also contains a referrer field, which specifies how the URL for the content was discovered. For instance if the end-user clicked on a link on some website, then that website's URL is entered in the referrer field. The referrer information is important for publishers in several ways, which we discuss in more detail in Section 4.3.3. The client passes on the HTTP request to the TCP layer in the protocol stack.

Before a content request can be sent to the network, the domain name part of the URL must be resolved to the network address of the host that serves the domain. This resolution is performed by the domain name service (DNS) [159]. When the publisher provides a URL of the form *http://publisher_domain/path_to_content*, then the DNS always resolves the domain to a unique server (or cluster of servers located at the same place), regardless of the location of the end-user. However, when the URL is served by a CDN and is of the form *http://cdn_domain/path_to_content*, then the domain name is resolved to the nearest server to the end-user [41]. The nearest server is determined by making use of inputs such as the network address of the end-user, which can be used to estimate the end-user's approximate geographic location [41, 156].

4.2.4 Content transfer/delivery

In general, an HTTP request is forwarded to the publisher's server or to the nearest server of its affiliated CDN. When a valid path for content is specified, then the server responds with the content in an HTTP response message. Because of the way in which the URL is

constructed, a CDN can only serve requests for content of paying publishers. Requests for content of non-paying publishers will contain paths that do not exist on the CDN.

Networks along the path from the publisher to end-users may find it desirable to cache frequently requested content in order to save on transit costs. This is referred to as transparent caching. When a request for a previously cached content reaches a router with access to the cache, then the router directs the request to the proxy server for the cache, which sends the content to the end-user. This process is transparent to both the publisher and the end-user's client.

Publishers and their CDNs can control the extent to which intermediate networks cache content by using implicit and explicit cache-control directives in the HTTP response message[2] [24]. For instance, the publisher can implicitly prevent networks from caching content by specifying short validity periods for the content. Furthermore, the publisher can explicitly instruct networks to avoid caching their content by specifying "no-cache" as a parameter for cache-control. A publisher may do this to prevent networks from serving stale content or to minimize the loss of content access information.

4.2.5 Meta-information gathering

Meta-information gathering forms a vital part of the content delivery process. In Table 4.2, we summarize the meta-information needs of different players in the content delivery chain

[2]It is known that cache control directives are sometimes ignored by transparent caches [149].

and identify the means provided by the TCP/IP content delivery model to obtain the required meta-information.

Table 4.2: Summary of the meta-information required by different network players in the TCP/IP content delivery model for the web and the means by which they gather the required meta-information.

Network player	Meta-information required	Means to obtain required meta-information
End-user	Trustworthiness of content retrieved	URL, search engines, server certificates in HTTPS [62]
Publisher	Content access information (who, when and how)	HTTP server logs, CDNs
	How do end-users discover content?	HTTP referrer field, HTML5 PING attribute
	Independent content access information to verify bills received from a CDN	HTTP redirection, single-pixel/index-page retrieval, HTTP cache-control directives, HTML5 PING attribute
CDN	Content delivery information (how much and to whom)	HTTP server logs
	Entity to bill for content delivery	URL for content delivered
Eyeball network	Content access information (who accesses what content, when and how) for law enforcement purposes or revenue generating activities, such as targeted advertising	Proxy server logs, deep packet inspection (DPI)

Some of the means in Table 4.2 are not foolproof. For instance, it is easy to blank out or insert false information in the HTTP referrer field [24]. There also exist tools and add-ons for most modern browsers that allow end-users to manage the referrer information sent to

websites[3]. End-users may want to alter the referrer information for privacy reasons. Similarly, the URL does not always provide reliable information about the trustworthiness of content. Nowadays, malware creators and phishing websites possess very creative means to make fake URLs appear legitimate, and in the process fool many users to trust the content [160]. Thus, users require additional tools and technologies, such as the HTTPS Everywhere browser add-on for Firefox and Chrome, which relies on secure HTTP connections, in order to obtain any reasonable level of trust in the content they retrieve [62, 161].

As detailed in the preceding paragraphs, network players rely on several important functionalities to realize different goals. In Table 4.3, we identify the key facilitators in the TCP/IP content delivery model and provide examples of the functionalities that they enable. Search engines, the DNS, HTTP and URLs all play important roles in facilitating content delivery on the web. Hence, any networking model that replaces the current TCP/IP content delivery model must either find ways to integrate these facilitators or find alternative means to realize the same functionalities that they enable. In the next section, we identify the key functionalities that must be replicated in a content-centric network.

[3]Opera has a built-in referrer control feature, whereas extensions like RefControl for Firefox enable a user to blank out the referrer field or insert fake entries.

Table 4.3: Illustration of some key facilitators in the TCP/IP content delivery model and examples of the functionalities that they enable.

Facilitator	Functionality enabled
Search engine	Content discovery
Uniform resource locater (URL)	Content discovery Content authentication Delegation of content delivery to a third-party Identifying publisher to bill for content delivery
Domain name service (DNS)	Network address resolution Delegation of content delivery to a third-party
Hypertext transfer protocol (HTTP)	Delegation of content delivery to a third-party Third-party cache control Identifying content discovery agent Gathering content access information Accounting and billing for content delivery Independent verification of content delivery

4.3 Content delivery functionalities that must be replicated in a content-centric network

Even though all CCN architectures aim to provide an intrinsic and more efficient support for content delivery, they differ in the details of how they achieve these goals.

Regardless of the approach taken, CCN-based architectures must provide all the useful functionalities that facilitate content delivery in the current TCP/IP model. Generally speaking, the functionalities that must be replicated in a particular CCN architecture depend on the extent to which the architecture reuses some of the key facilitators in

the TCP/IP content delivery model. For instance, CCN architectures that employ URI schemes which are similar to URLs can use the URI to accomplish the same functionalities in Table 4.3. Examples of such architectures include TRIAD [67] and NDN [3].

Reusing protocols like HTTP minimizes the number of content delivery functionalities that must be replicated in the CCN architecture. The degree to which CCN architectures can reuse protocols like HTTP depends on how the architecture treats content requests. CCN architectures that employ a name resolution service to first find a list of hosts that possess the desired content can use HTTP to establish a connection between the end-user and a host that possesses the content and thus reuse all the functionalities facilitated by HTTP. Examples of such architectures include PSIRP [86], TRIAD [67], LNA [69], NetInf [72] and CURLING [87].

On the other hand, CCN architectures like NDN, DONA [70] and XIA [74], which perform name-based routing of content requests, eliminate all the functionality provided by HTTP. These CCN architectures must find alternative ways to replicate HTTP functionalities. When CCN architectures combine name-based routing with URI schemes that completely decouple content identifiers from location and/or publisher then the number of functionalities that must be replicated increases. In the next paragraphs, we briefly summarize the functionalities that must be replicated in a CCN architecture that employs name-based routing together with a URI scheme that completely decouples content from its location and/or publisher.

4.3.1 Delegation of content delivery to a paid third-party

Some publishers prefer to delegate content delivery to paid third-parties in order to reduce latency and achieve resilience to flash crowds. For such publishers, it is important for the network architecture to provide a means to perform delegation efficiently. Currently, a publisher can delegate content delivery to paid third-parties during the content preparation stage by specifying a URL that includes the domain of the third-party as the authoritative host. They can also perform delegation of content delivery during the content request stage by employing HTTP redirection to direct requests to the paid third-party [41, 156].

A URI scheme that completely decouples the object's name from its location and/or publisher removes the delegation capability. In addition, name-based routing of content requests eliminates HTTP and with it, the second means of delegating content delivery to paid third-parties. As a result, a new means of delegating content delivery to paid third-parties must be designed for CCN architectures that perform name-based routing of content requests on URIs that decouple content identifiers from location.

4.3.2 Third-party cache control

In many cases, publishers may want to control the extent to which their content gets cached in the network. For instance, some content may change frequently and the publisher may want to prevent caching in order to limit the likelihood of end-users receiving stale content. At other times, publishers may restrict caching in order to obtain information about who

accesses their content, when they access it and how they access it. The publisher loses this information when networks are able to cache and deliver content unconditionally. Currently, publishers use HTTP cache control directives to control the behavior of caches [24]. However, alternative means to enable publishers to control the behavior of caches must be designed for CCN architectures that perform name-based routing of content requests.

4.3.3 Identifying content discovery agent

There are many reasons why a publisher may want to know how an end-user obtained the URI for content. For example, a publisher may want to provide a piece of content to an end-user only if the end-user discovered the content through the publisher's website. In this way, the publisher can mitigate problems such as deep linking [162, 163]. Moreover, the publisher can modify the content served to the end-user based on how the end-user discovers the URI.

To illustrate further, an advertising agent needs to know how many times different websites serve advertisements in order to compensate them appropriately. Without the means to identify how end-users obtained the URI of the advertisement, the advertiser cannot effectively meter click-throughs and appropriately compensate its distribution partners. Currently, the HTTP referrer field enables publishers to obtain information about how end-users discovered the URI. With name-based routing of content requests, new means must be designed to accomplish the same functionality.

4.3.4 Accounting and billing for content delivery

A content delivery network must be able to identify the correct publisher to bill for content delivery. In the TCP/IP content delivery model, the tight coupling between content name and its location provides the CDN with a means to identify the publisher to bill for content delivery. Even in cases where two different publishers serve identical content from the same CDN, the CDN can still account and properly bill for delivery because the structure of URLs ensures that the content served for different publishers have different identifiers. However, name-based routing on a URI scheme that is completely decoupled from the location and/or the publisher introduces new challenges in accounting and billing for content delivery, which must be addressed.

4.3.5 Independent verification of content delivery

Content delivery networks charge publishers based on the volume of content they deliver on their behalf. In the current TCP/IP content delivery model, publishers can employ several means to independently obtain an upper bound on the volume of content delivered by the CDN. For instance, some publishers choose to serve the text of the web page and only delegate delivery of large objects like videos and images to the CDN. A variation of this approach involves embedding a single pixel image on a web page, which is delivered exclusively by the publisher. In this way, the publisher can use delivery information for the web page or the single pixel image to estimate an upper bound for the number of times that the images and videos on that page were delivered.

Another approach to independently verify content delivery employs HTTP redirection techniques [156]. With this approach, the publisher always receives requests for content before redirecting them to the CDN. Thus, the publisher obtains an accurate estimate of the volume of data delivered by the CDN. Without any independent means of verifying access to content, the publisher would have to trust the CDN to report delivery information truthfully. In such a situation, a CDN may have incentives to over-report the volume of content delivered in order to obtain more revenue. The presence of simple and effective means for publishers to independently verify content delivery reduces the incentives for networks to over-report the volume of content delivered.

Name-based routing eliminates HTTP redirection techniques for independent verification of content delivery. Also, in the absence of a means to identify the content discovery agent (see Section 4.3.3), approaches that depend on single pixel retrieval or index web page retrieval directly from the publisher may prove ineffective as a means to independently verify content delivery. For example, a video on a publisher's web page could be posted on multiple websites. In the absence of a means to limit delivery and billing to the video that originates from the publisher's web page, the copies from other websites will all be delivered and billed to the original publisher. Thus, it is important for CCN architectures that employ name-based routing to implement simple and efficient mechanisms for publishers to independently verify content delivery and billing information.

Although the idea of name-based routing has been widely embraced, very little attention has been paid to replicating the functionalities that are lost by eliminating a protocol like HTTP. We believe that this results from the implicit assumption that the network will provide sufficient storage infrastructure without any payment flows for content delivery, which we have shown to be false (see Chapter 2 and [120, 150, 164]). In the next section, we describe our content-centric network model, which forms the basis of the mechanisms we later design to replicate the functionalities lost by eliminating HTTP and URLs in CCN architectures that perform name-based routing of content requests.

4.4 Content-centric network model

We assume a content-centric network that performs name-based routing of requests. Specifically, we consider the eXpressive Internet Architecture (XIA) [74]. In the next section, we provide a brief overview of the main features in XIA. For a more detailed description, please refer to [74, 83].

4.4.1 Overview of XIA

By design, XIA provides intrinsic support for different networking principals such as hosts, administrative domains, content and services. XIA uses self-certifying identifiers for all routable principals and enjoys their numerous benefits, highlighted in the literature (see e.g., [76, 80, 100]).

In XIA, communicating entities express the intent of their communication to the network and the network determines the best way to satisfy that intent. The network will use name-based routing to fulfill content requests by retrieving the content from any network location that has a copy. At the same time, XIA provides the possibility for the user to tell the network how to satisfy that request in the event that the original intent is not supported. The ability to express the intent and to provide fallback options for network entities that lack support for the original intent are all made possible through the use of directed acyclic graph (DAG) addresses in XIA [74].

We provide two examples to illustrate the DAG addressing style in XIA for content retrieval[4]. Let us consider an end-user who is interested in a piece of content, say a video clip. Further, let us assume that the end-user believes that he is very likely to get the the video clip from publisher B and somewhat likely to obtain it from publisher A. This scenario is represented by the directed acyclic graph in Figure 4.1.

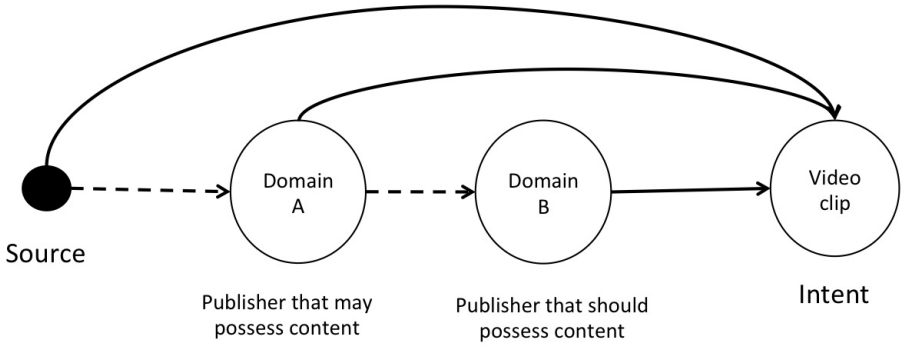


Figure 4.1: Illustration of the DAG addressing style in XIA. The user is primarily interested in obtaining a video clip from any location that has a copy. Nevertheless, the user knows that in the absence of nearby copies, the video may be obtained from publisher A or publisher B. A directed acyclic graph, where each node represents an entity and each edge represents a path can be used to capture the intent of the end-user. Solid arrows represent paths to the intent and the dash arrows represent paths to a fallback.

[4]The interested reader should refer to [74, 83] for a more detailed treatment of DAG addressing in XIA.

Basically, each node in the graph represents an entity and each edge represents a path component or segment. In XIA, nodes must be XIA principals, which at the moment include hosts, administrative domains, services and content. The end-user expresses priority for a path through the order in which the edge is listed in the address. Thus, a router processing the DAG address illustrated in Figure 4.1 knows that the user is really interested in the video clip and can respond directly to the request if it possesses the video. However, if the router does not have the video (or does not know how to interpret the request for the video), it simply forwards it along the first edge to publisher A, who may be able to serve the content. The process is repeated until the last edge is reached. When the content is still not found, an error message is returned to the source.

In our second example, we consider the case of a publisher who wants to serve routable content. Ideally, the publisher would like the content to be served by caches within the network. However the publisher provides fallback options for routers that do not have content routing capability to know how to handle requests for the content. Let us assume that the publisher runs a service (e.g., web service) on a host that definitely knows how to find the routable content. The publisher can capture this intent with the DAG illustrated in Figure 4.2. End-users obtain this DAG as an identifier for the video clip and use it to locate the content.

Like other content-centric network architectures, XIA assumes the presence of widespread caching within the network. However, caching in XIA is not limited to transparent caching. XIA provides for the possibility that caches may alter their caching policies based on pay-

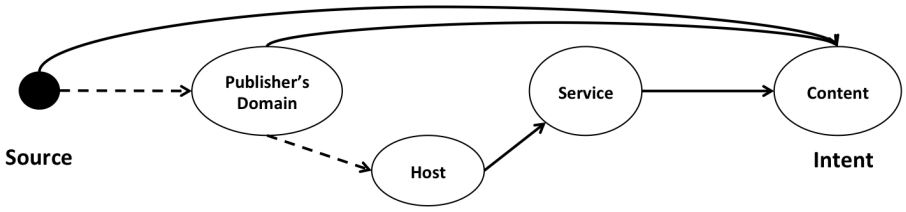


Figure 4.2: Illustration of the DAG addressing style in XIA. The publisher of the video clip provides users with a DAG address that captures the following intent. First, users can obtain the video clip from any content router in the network that has a copy. In the event that a router does not have a copy or does not know how to route on content, then the router forwards the request along the path to the publisher's domain. Once the request reaches the publisher's domain, there is a host that knows a service that knows how to find the video clip. Solid arrows represent paths to the intent and the dash arrows represent paths to a fallback.

ment flows from content owners. This stance is driven by the economic reality that network operators will not provide sufficient caching infrastructure to support the CCN content delivery model unless there is some payment flow from publishers for content delivery [99, 120, 164].

In the rest of this section, we describe the content delivery ecosystem we envisage and the infrastructure needed to support paid content delivery in CCN-based architectures. In addition, we describe our proposed model to support paid content delivery in XIA.

4.4.2 Content delivery ecosystem and infrastructure

One of the most appealing aspects of CDNs is that they provide a single point of contact for publishers to reach a large number of eyeball networks [19, 165]. In other words, in addition to their content delivery role, CDNs also serve as transaction brokers between multiple publishers and multiple eyeball networks and thus, minimize transaction costs for publishers and eyeball networks. If each Internet service provider (ISP) provided caching

services, publishers would have to contract with multiple eyeball networks in order to achieve low-latency content delivery, which introduces significant transaction costs for the publisher [150, 164]. Likewise, an entity that acts as a broker between publishers and the numerous cache owners envisaged in CCN is required in order to minimize transaction costs between publishers and cache operators.

We refer to the network player that performs the transaction brokerage role in our CCN model as a content delivery broker (CDB). CDBs could operate on a non-profit or for-profit basis. Nevertheless, competition is required in order to eliminate rent-seeking behavior from a CDB. Thus, we assume the presence of multiple CDBs who compete for brokerage services in a CCN ecosystem.

We envisage two types of CDBs. The first type provides only the infrastructure required to facilitate information and money flow between publishers and the networks that own and operate caches. These CDBs do not own or operate their caches. An example is a third-party, such as a trade organization or a federation, which provides the brokerage function on behalf of its members. The second type of CDB owns and operates caches, in addition to providing the brokerage required to facilitate information and money flow between publishers and other networks that own and operate caches. Current CDNs could evolve into this type of CDB in a content-centric network. We illustrate the content delivery ecosystem and infrastructure in Figure 4.3.

Publishers either rely on transparent caching or pay for content delivery through one or more CDBs. Likewise, networks operate caches transparently or establish paid content

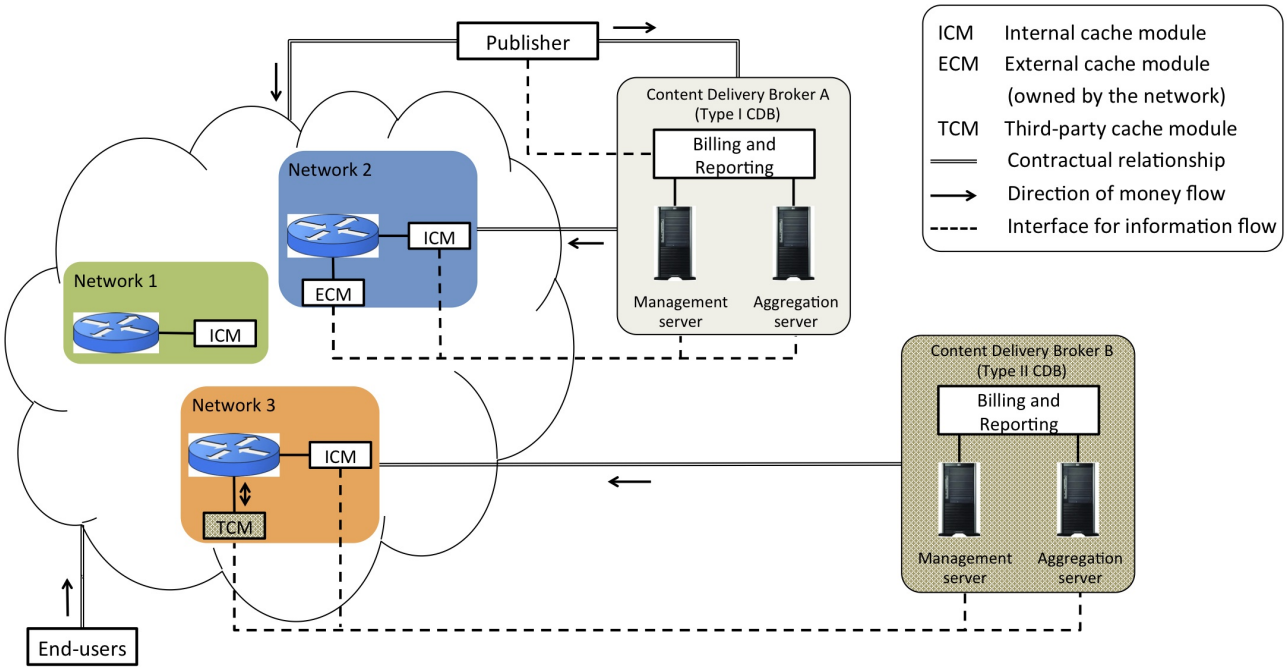


Figure 4.3: A simplified illustration of the CCN content delivery ecosystem and infrastructure. Caching and content delivery are performed by one or more cache modules, which interface with the router either internally or externally. There are two categories of external cache modules - those owned directly by the network and those owned by other entities, which we refer to as third-party cache modules. Content delivery brokers (CDBs) serve as transaction brokers between publishers and cache owners, in order to minimize transaction costs for paid content delivery. They provide the infrastructure to facilitate information and money flow between publishers and network operators. Some CDBs, such as CDB B, may also operate their own cache modules. Networks contract with one or more CDBs to offer paid caching services. Publishers also contract with one or more CDBs to purchase paid caching services. Money flows from publishers to CDBs and from CDBs to networks. In addition, money may flow between third-party cache module owners and the network in either direction, depending on the negotiating power of the parties involved.

delivery relationships with one or more CDBs. Caching is performed by cache modules attached to routers. Routers can instruct the cache module to serve content requests directly if they possess the requested content. In addition to forwarding content responses to the next hop, routers also pass along a copy of every content response to the cache module for potential caching.

A cache module could be integrated on the router (e.g., extended router buffer memory or content stores envisaged in NDN [3, 166]). Alternatively, the cache module could be an

external module that connects to the router through a standardized interface. A router could have multiple external cache modules, which could be owned by multiple entities. In the rest of the chapter, we refer to external cache modules owned by a third-party as third-party cache modules to distinguish them from external cache modules owned by the network.

One could think of third-party cache modules as the network-layer equivalent of surrogate servers placed by CDNs in different networks in the TCP/IP content delivery model [41, 42]. Thus, in line with current practices, we expect third-party cache modules owners to pay networks to place cache modules in their network or get paid by the network in the form of free co-location, depending on the negotiating power of the third-party cache module owner and the host network [106, 167].

In Figure 4.3, the publisher purchases paid content delivery services through CDB A[5]. The publisher negotiates the required level of service and payment, and a contract is established before content delivery occurs. The management server at the CDB maintains a database of valid paying publishers and their contracted services. It also keeps a database of valid network partners. At regular intervals, the management server sends database updates of valid paying publishers to all the network partners, such as Network 2. We assume that secure protocols exist for transferring databases between the management server and the cache modules. Depending on the architecture of the partner networks, this information could be sent to a centralized point before eventual dissemination to the cache

[5]The publisher also buys network access through other network providers, in addition to purchasing content delivery through a CDB. There could be instances where the network access provider also serves as a CDB.

modules or it could be sent directly to the cache modules[6]. It is also possible for the cache module to poll the management server for this information.

There are two ways to get the publisher's content in the cache modules. In the first, the publisher pushes content to the CDB, which then replicates the content across its partner networks to meet service level agreements (SLAs) and performance targets [43, 169]. This centralized approach is similar to the model currently used by CDNs and requires the CDB to procure transport services to reach all partner networks. It may be possible to reuse the same connections to the management servers for content transfer from the CDB.

The second way to get content in the cache module is more decentralized and reflects one of the key themes in the CCN literature. As the publisher's content moves through the network, all cache modules make independent caching decisions. For networks, such as Network 1, which do not partake in paid content delivery, the decision to keep the content in the cache is driven purely by the popularity of the content. For other networks, such as Network 2 and Network 3, which engage in paid content delivery, the decision to keep an object in the cache depends on factors such as the object's popularity, the publisher, the CDB contracted by the publisher and the type of service contracted by the publisher. For instance, Network 3 will only transparently cache very popular content from the publisher, since the publisher does not have a contractual relationship with CDB B. On the other hand, Network 2 may cache less popular content from the publisher simply because it gets paid by CDB A for delivering content for the publisher.

[6]See [168] for examples of possible ways to architect information flow between management servers and the cache modules.

All cache modules keep a log of content delivered. This log contains all the relevant information necessary to aid in accounting and billing for content delivery. It also contains meta-information that may be useful for the publisher. The cache modules periodically send logs to an aggregation server at the CDB. We assume that mechanisms and secure protocols exist for transferring the logs. The aggregation server processes all logs to determine bills for publishers and content delivery revenues for cache owners. The CDB may generate and charge for different flavors of content delivery reports with varying degrees of meta-information for publishers, depending on the nature of the market. As an example, CDNs currently report only the volume of data delivered and usually charge publishers for additional meta-information, such as sources of requests [41].

To summarize, networks perform transparent and/or paid caching using cache modules attached to routers. Further, CDBs serve as transaction brokers between publishers and network operators, facilitating information and money flow in a way that minimizes transaction costs for both publishers and networks. In the next section we describe the structure of routable content we propose for XIA in particular and name-based routing CCN architectures in general.

4.4.3 Structure of routable content

The granularity of naming content has significant implications for caching in a content-centric network. It also impacts communication and packet overhead. To illustrate, architectures such as NetInf [72] and DONA[70] treat entire objects as routable content units (

in this context, an object refers to a complete file, e.g., movie, image, audio or document). This design decision reduces the size of the namespace to manage, since a single name is associated with each object.

However, it also reduces the efficiency of caching and may introduce significant communication overhead. These problems occur when large objects are segmented for transport in the network. Segmentation of large objects becomes an issue when it is used in conjunction with connectionless protocols, such as user datagram protocol (UDP), because any lost segments result in a new request for the entire object and its eventual retransmission. Over time, constant retransmissions of large objects could lead to congestion in the network.

Moreover, it is entirely possible for different segments to take different paths in the network, which deprives caches of the possibility to accumulate all segments within a specified time frame and reconstruct the object for caching. Thus, large popular objects, which could benefit significantly from caching, may end up not being cached. Further, caching at the granularity of objects reduces the number of objects that can be stored in the cache, which reduces the cache hit ratio [139, 170].

In order to achieve a reasonable balance between communication and packet overhead and caching efficiency, we adopt a flexible approach in the granularity of naming content. We refer to the basic routable content unit as a *chunk*. Each chunk contains some meta-information and the payload. We leave the discussion of the required meta-information for Section 4.4.4.

A chunk is identified by a chunk identifier (CID), which is obtained by hashing the contents of the chunk. Thus, chunks have self-certifying identifiers and enjoy all the benefits of self-certifying names outlined in the literature (see e.g., [76, 80, 100]). Chunks have a minimum size, which is determined by the size of the required meta-information. However, chunks can take on any size beyond the minimum. It is the responsibility of the application to transform an object into chunks in a manner that best meets the needs of the application and any requirements imposed by content delivery brokers and their network partners.

A small object could be represented by a single chunk. In this case, the object is identified by the CID. On the other hand, large objects may be represented by multiple chunks, which must all be obtained to reconstruct the object. When this happens, a special *master chunk*, whose payload contains a list of all the chunks that constitute the object and how they must be put together to reconstruct the object, must be created.

In effect, a master chunk performs a similar role to a media presentation description (MPD) file in DASH [153], the index file in Apple HLS [152] or the manifest in Microsoft Smooth Streaming [171]. The master chunk is identified by the master chunk identifier (MCID), which is obtained by hashing the contents of the master chunk. An object that is composed of several chunks is named after the master chunk identifier. We illustrate the relationship between chunks, master chunks and objects in Figure 4.4.

In practice, CIDs and MCIDs are indistinguishable. We only distinguish between them for ease of explanation. In addition, our model does not limit the degree to which master chunks can be nested. Thus, very large objects can be represented by several mini-master

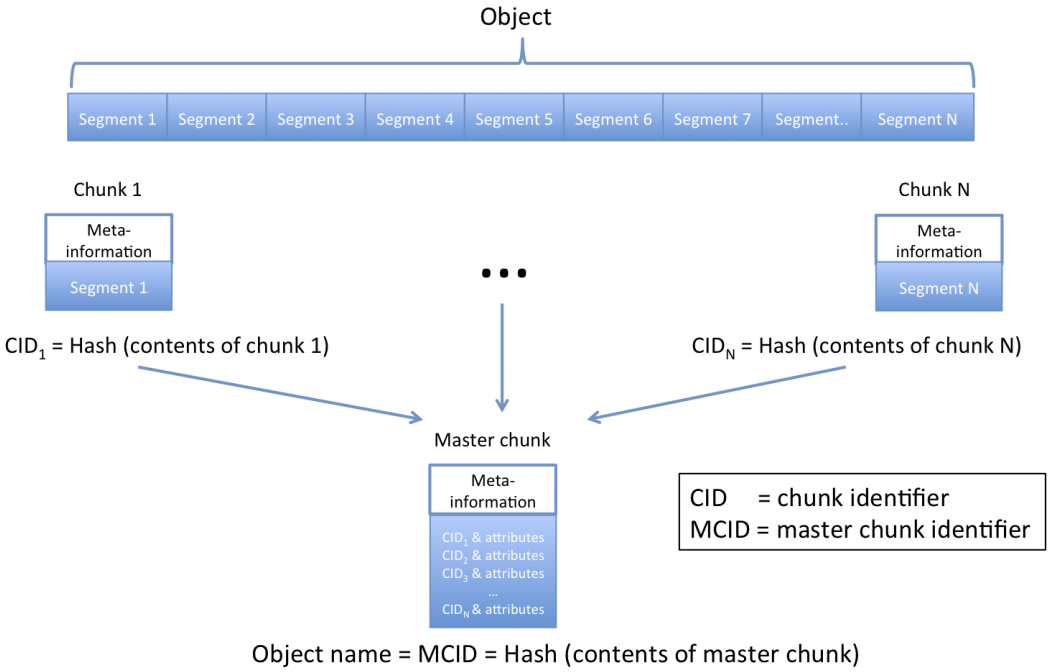


Figure 4.4: Relationship between chunks, master chunks and objects. A chunk describes the basic routable content unit. An object may be composed of several chunks. A master chunk contains a list of all the chunks that compose an object. When an object is composed of a single chunk, then the object is named after the hash of the contents of the chunk. On the other hand, when an object is composed of multiple chunks, then the object is named after the hash of the master chunk.

chunks and a master chunk. Given the above background, we next turn our attention to the meta-information contained in each chunk and how chunks are named.

4.4.4 Chunk meta-information

One of the main goals of CCN is to decouple content properties from location so that content can be served from any network location. In order to achieve this goal, end-users, cache modules and other network entities must be able to deduce all desired properties, such as content integrity and provenance, from the content itself. In addition, cache modules must be able to make caching decisions based on trustworthy information carried with

the content. Consequently, the meta-information included in the chunk and the scheme employed to make the chunk intrinsically secure play a vital role in the success of the CCN content delivery model.

In our model, all chunks contain two layers of meta-information (Figure 4.5). The first layer is used by cache modules to make caching decisions whereas the second layer is used by applications to know how to process the payload. We briefly describe the two layers of meta-information in the next section.

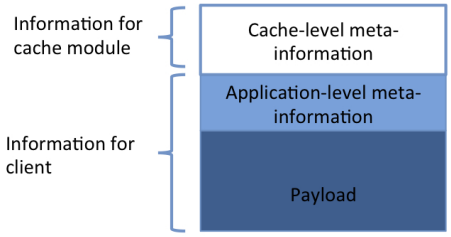


Figure 4.5: Structure of a chunk. All chunks contain two layers of meta-information. The first layer is used by cache modules to make caching decisions whereas the second layer is used by applications to know how to process the payload.

Cache-level meta-information

Cache-level meta-information includes all the information necessary for a cache module to make a caching decision. Figure 4.6 illustrates the set of meta-information required by the cache module under different scenarios. We also provide a description of each element of meta-information in Table 4.4.

There are many ways to encode the cache-level meta-information. Several factors affect the choice of encoding scheme. First, cache modules must be able to process the chunk

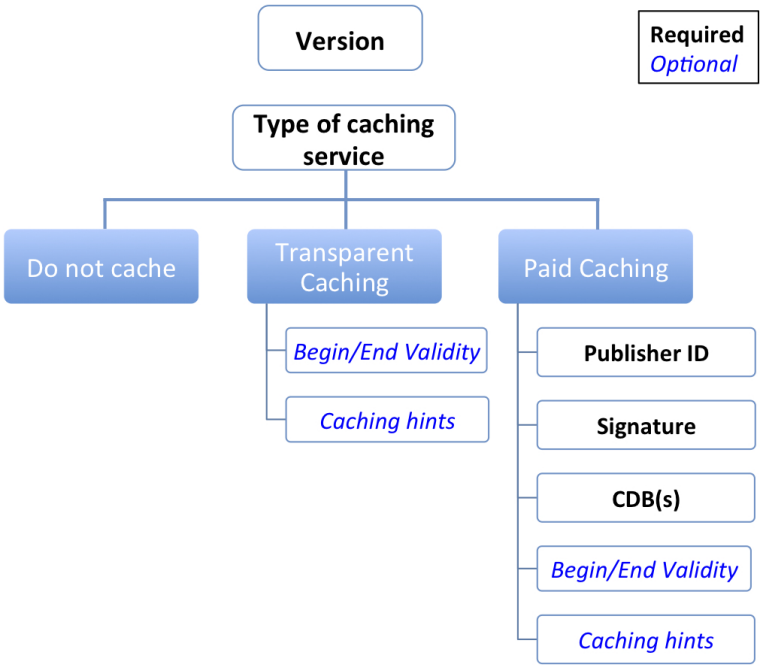


Figure 4.6: Cache-level meta-information include all information necessary for the cache module to make a caching decision. The type of caching service specified determines whether or not Publisher ID and Signature are present in the cache-level meta-information.

header at line speed at the network edge, where caching provides substantial benefits [90, 92, 172]. In addition, the scheme must minimize chunk overhead and enable easy debugging and troubleshooting. Finally, the encoding scheme must be flexible enough to support future evolution. For our purpose, we consider a fixed field encoding scheme.

The scheme, illustrated in Figure 4.7, encodes information in a binary format, using a fixed location for different pieces of information. The advantages of fixed field encoding include fast processing and low packet overhead. Dedicated hardware can be built to directly process a fixed field encoding scheme, allowing cache modules to operate at line speeds [95, 166, 173, 174]. However, this encoding scheme is not particularly user-friendly, and may not be flexible enough to support future evolution.

Table 4.4: Description of cache-level meta-information types.

Meta-information	Description
Version	Required in all chunks. Indicates the protocol version number for the cache-level meta-information. The version number determines how values for other meta-information are interpreted.
Type of caching service	Indicates the type of caching service required by the publisher of the chunk. By default, cache modules cache chunks transparently if no type of service is specified. Still, a publisher can explicitly specify transparent caching as a type of service. A publisher who does not want caching for the chunk must specify "do not cache" as a caching service. Publishers that pay for content delivery must also specify the type of paid caching service that they require for the chunk. If a publisher specifies a paid caching service, then the publisher must also specify a publisher ID, a signature and a list of content delivery broker(s).
Begin validity	Optional. Indicates the period (date and time) from which cache modules can serve the chunk from cache.
End validity	Optional. Indicates the period (date and time) after which cache modules cannot serve the chunk from cache.
Caching hints	Optional. Provide hints to the cache module to aid in caching and cache replacement decisions. This specifies, for example, whether the chunk is a master chunk. It also specifies chunk dependency information (e.g., whether chunk is part of a larger object or a standalone object).
Publisher ID	Required when paid caching service is specified. The publisher ID is a fingerprint of the entity that contracts with the content delivery broker for paid delivery of the chunk. It could be the creator of the object (i.e., the original publisher) or a legitimate distributor of the content.
Signature	Required when paid caching service is specified. The signature is obtained by signing a hash of all contents of the chunk (meta-information and payload) with the private key of the entity that contracts with the content delivery broker for paid content delivery.
CDB	Required when a publisher specifies a paid caching service. The CDB is a fingerprint of the content delivery broker's public key. Multiple CDBs can be specified, with each assigned different priorities.

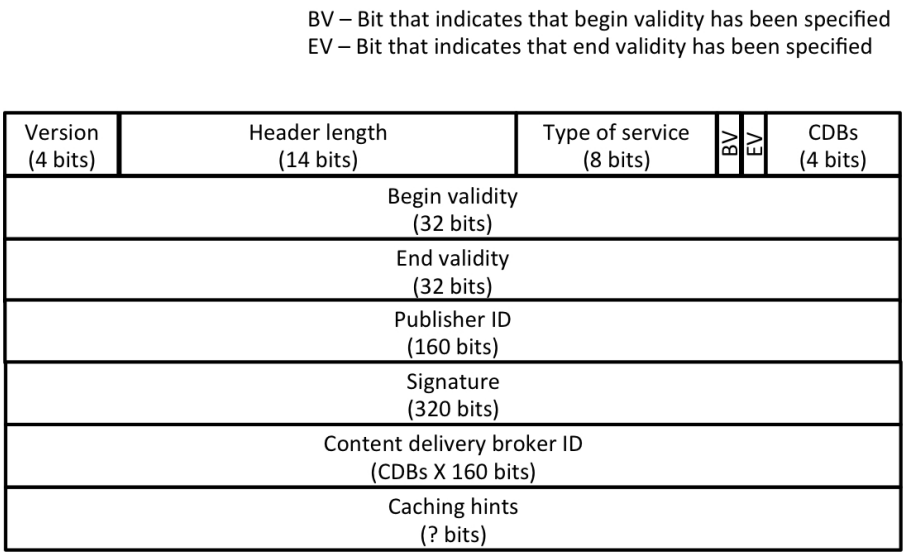


Figure 4.7: Encoding cache-level meta-information using a fixed field scheme. The header length is the number of bits contained in the header. The type of service indicates the type of caching service desired by the publisher. Possible values include: do not cache, transparent caching and other values that depend on the specific content delivery broker contracted by the publisher. The publisher ID and the content delivery broker ID are both hashes of public keys. The begin and end validity specifies the date and time period during which the content can be served from cache. The caching hints provide information for cache modules to facilitate caching and eviction decisions. Caching hints could include information about the dependency of the chunk (e.g., standalone object, part of larger object, etc), the type of chunk (e.g., chunk or master chunk) among other things.

The fixed field encoding scheme shown in Figure 4.7 limits the minimum size of chunk headers. For example, publishers who do not pay for caching services or who do not want their content to be cached must still include a header of at least 4 bytes for all chunks. We summarize the minimum chunk header required for a few typical scenarios in Table 4.5.

Table 4.5: Minimum chunk headers corresponding to different publisher scenarios when a fixed field encoding scheme is employed.

Scenario	Minimum chunk header [bytes]
No caching	4
Transparent caching	4
One content delivery broker	84
Two content delivery brokers	104
Three content delivery brokers	124

The values in Table 4.5 suggest that in order to achieve a chunk overhead of at most one percent, publishers who do not pay for caching must have chunk sizes of at least 400 bytes, whereas publishers who pay for caching from a single content delivery broker must have chunk sizes of at least 8 kilobytes. We choose not to propose chunk size standards since evolving caching technology and content types may suggest that optimal content size should evolve over the life the architecture. We believe that CDBs and cache module operators can drive chunk sizes to optimal levels for the current state of the technology by, for example, pricing policies.

Application-level meta-information

Application-level meta-information is required by the client to know how to process the payload. This information is independent from the cache-level information required by caches. The different kinds of application-level meta-information are illustrated in Figure 4.8. Like many application layer protocols, this meta-information is encoded with a parameter:value scheme.

All chunks must contain the content type in the application-level meta-information (Figure 4.8). The content type specifies the Internet media type of the payload [175, 176]. The Internet media type is a standardized means of identifying all file formats on the Internet and it is employed in protocols such as SMTP [177], HTTP [24], RTP [178] and SIP [179]. It consists of two or more parts namely a type, subtype and optional parameters. The type field could take on values such as application, audio, image, message, model, multipart,

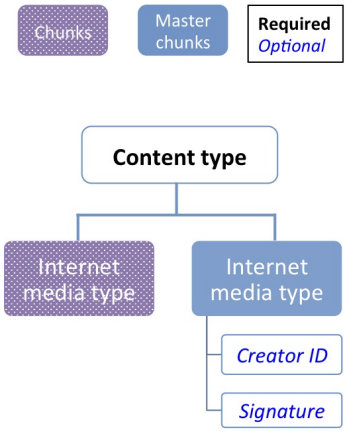


Figure 4.8: Application-level meta-information includes all information necessary for the client to process the payload. The content type specifies the Internet media type (see [175, 176]) to which the payload belongs. Optionally, the application-level meta-information may also specify authentication information of the original publisher of the content for the client to prove provenance. This may be used by the client to obtain some trust in the contents of a master chunk.

text and video [175]. For each type, there are standardized subtypes and options for each subtype, which help to further identify different formats and encoding schemes. There is also the possibility to specify vendor-specific and unstandardized or private subtypes [176]. Given the wide array of possibilities, we believe that the Internet media type provides enough flexibility to cover all possible content types in the payload of a chunk.

Optionally, the application-level meta-information may contain a creator ID and a signature over the contents of the chunk, independent of the information specified in the cache-level meta-information. The creator ID specifies the identifier (e.g., 160-bit hash of the public key) of the original content creator. The signature is generated with the private key of the content creator. Signatures in master chunks attest to the authenticity of the identifiers of the chunks that compose the object, which therefore do not need their own signatures. We assume that clients will make their own decisions on how to treat master chunks that lack authentication information.

4.4.5 Chunk naming

We make a design decision to use a self-certifying identifier to securely bind all meta-information to the chunk. This is necessary to prevent any modifications in the meta-data specified by a publisher. To securely bind all meta-information to a chunk, we name a chunk after the hash of all the contents of the chunk. This includes the actual payload together with the cache-level and application-level meta-information.

An alternative approach would be naming the chunks after the hash of the payload or after the hash of the payload and the application-level meta-information. With this approach, the same chunk would have the same identifier regardless of the type of caching service, which might be desirable for caching efficiency. The drawback would be that cache-level meta-information could be easily changed by other entities besides the original publisher, without detection. This could be used by an attacker, for instance, to change the caching service from paid to transparent or do not cache, in order to reduce the quality of service for a competitor's content. To prevent such attacks and ensure that the publisher's wishes regarding a chunk are always respected, we choose to name the chunk after the hash of the payload and all meta-information. Our approach is also more general, in that no parts of the chunk are treated as special for naming purposes.

4.5 Proposed XIA content delivery model

Our CCN content delivery model consists of the same stages introduced in Section 4.2, namely content preparation, content discovery, content request, content transfer and meta-

information gathering. We describe each stage in the next sections.

4.5.1 Content preparation

The publisher's application determines the granularity of chunking and the meta-information to include in each chunk during content preparation. The application may decide to make the object available as a single chunk or break the object into multiple chunks, which must be reconstructed by the end-user's client. Objects that fit in a single chunk are identified by the CID, whereas objects that comprise multiple chunks are identified by the MCID.

The type of cache-level meta-information included in the content preparation stage depends on the particular needs of the publisher. For instance, a publisher who desires transparent caching only needs to specify the version number. This information is sufficient because cache modules make transparent caching decisions based solely on the popularity of the object. Thus, in the absence of any explicit instruction to avoid caching, cache modules will eventually cache popular objects. On the contrary, publishers must specify the CDB(s), the type of service contracted and authentication information, in addition to the publisher ID and the version number, when they pay for content delivery. This is because cache modules need this information in order to make non-transparent caching decisions.

The publisher makes the content available on the web by specifying a URI in an HTML document. XIA provides a lot of flexibility for the publisher to decide how to best serve the content, either through traditional host-based means, through a service or as routable

content. We employ a URI scheme for content that allows the network to directly serve the content through name-based routing or to use a host-based fallback to serve the current version of the desired content or through a service.

We employ a URI scheme for content that takes the form *xid:DAG_address_ for_content;content_URL*. DAGs in URIs are encoded using the "Base 64 Encoding with URL and Filename Safe Alphabet" defined in RFC 4648 to avoid the use of + and / [180]. The publisher has the option to specify only the DAG address in order to provide early binding, specify only the URL[7] in order to provide late binding or specify both the DAG address and the URL. Thus, the URI for content could take one of three possible forms namely *xid:DAG_address_ for_content*, *xid:;content_URL* or *xid:DAG_address_ for_content;content_URL*. All three options are useful for different purposes.

A publisher specifies the first form of the URI in order to take full advantage of name-based routing and caching in the network. However, for content that changes fairly rapidly, it may be better for the publisher to use the second form of the URI in order to ensure that end-users always get the current version of the content through late binding. For instance, in response to a request sent to the URL, the publisher could directly send the desired content or send the current DAG address for the content which can then be retrieved using name-based routing. The second approach is useful for publishers who employ cookies to keep track of or to authenticate end-users. Without the ability to directly contact the web

[7]We assume that URLs specified in this way have been encoded/escaped safely (see e.g. [155]).

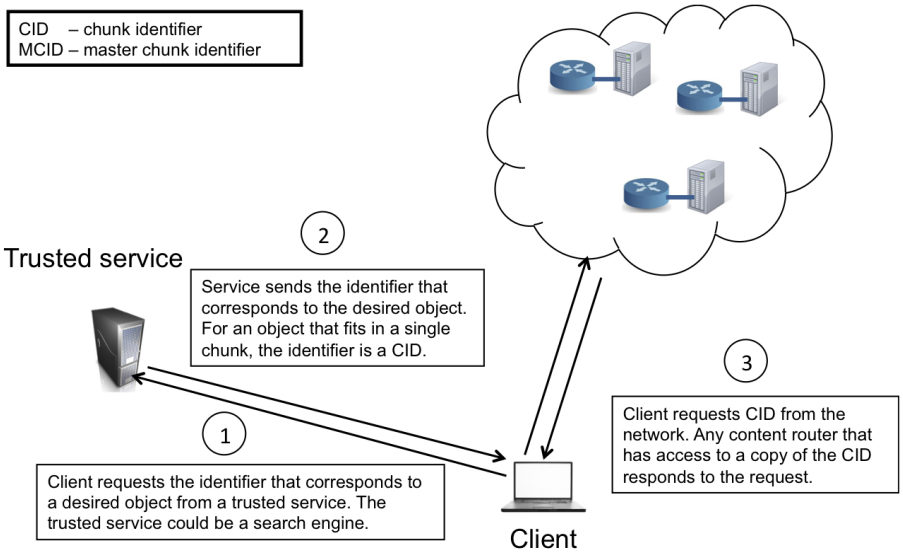
server for the publisher, cache modules would need to keep state in order to offer cookie functionality, which is undesirable for scalability and other reasons.

Finally, the third form of the URI could be used for content that may change over a longer time period. Providing both a DAG and a URL allows the publisher to take advantage of in-network caching for the lifetime of the current version of the content while providing the means for clients to locate a newer version of the content in the future. It also gives clients the ability to satisfy their desired intent. A client interested in the current version of the content employs the URL (late binding), whereas a client interested in a particular version of the content specifies the DAG address for that version of the content (early binding).

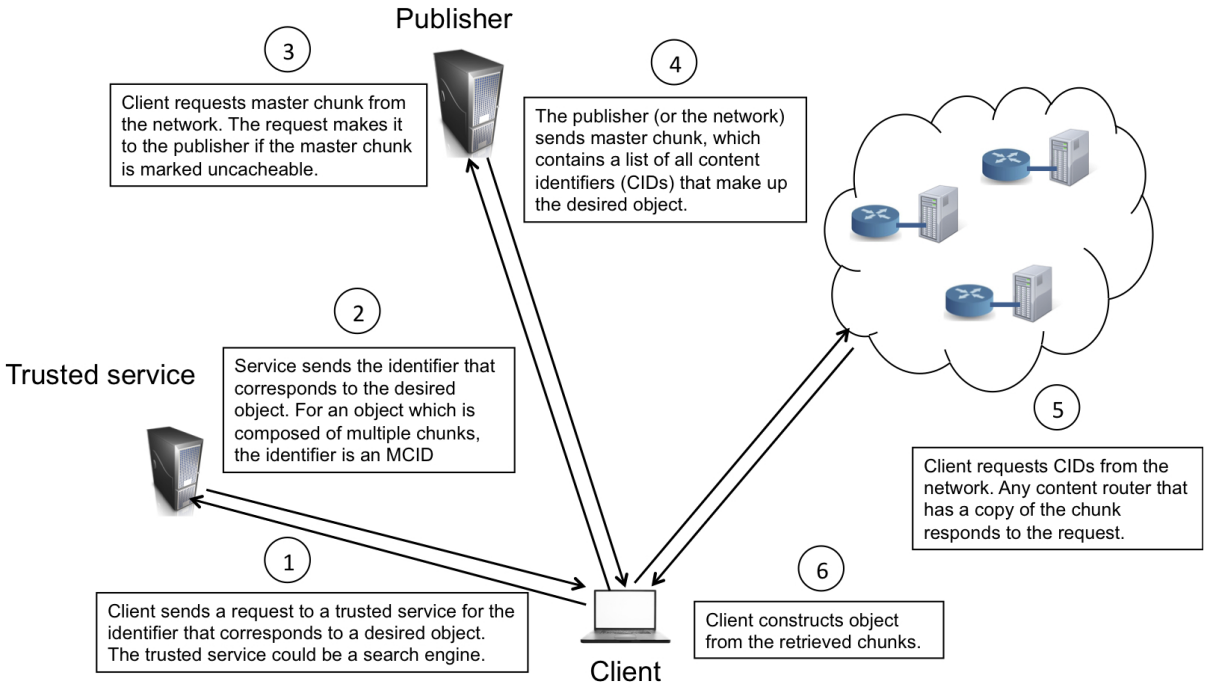
4.5.2 Content discovery, content request and content transfer

We illustrate the process of content discovery, request and transfer for objects that fit in a single chunk in Figure 4.9(a) and for objects that comprise several chunks in Figure 4.9(b). We discuss the stages in more detail in the next paragraphs.

An end-user first uses a trusted service (e.g., a search engine) or another source (e.g., email or link in a document) to obtain the URI for the desired object. In XIA, each client has access to a service, called XChunkP, which performs simple reliable content-retrieval on behalf of the client. As explained previously, content retrieval in our proposed CCN model occurs at the granularity of chunks. A client issues a request to XChunkP to retrieve



(a) Desired object fits in a single chunk



(b) Desired object fits in multiple chunks

Figure 4.9: Content discovery, content request and content delivery in our CCN model.

a particular chunk, passing on the DAG address to XChunkP. Only complete chunks are returned to the client by XChunkP. If the chunk request is not satisfied within a specified

time period, the client reissues the chunk request to XChunkP. Thus, content retrieval and on-time delivery are directly handled by the client in our content delivery model.

We consider a content URI of the form *xid:DAG_address_ for_content* because it enables us to highlight the name-based routing capabilities of XIA. In this case, XChunkP requests the chunk by sending a packet with the DAG address of the content as the destination and the DAG address of the client as the source. We provide an example of a chunk request packet in Figure 4.10. We only focus on the destination field, but a detailed explanation of all fields in the packet header can be found in [74].

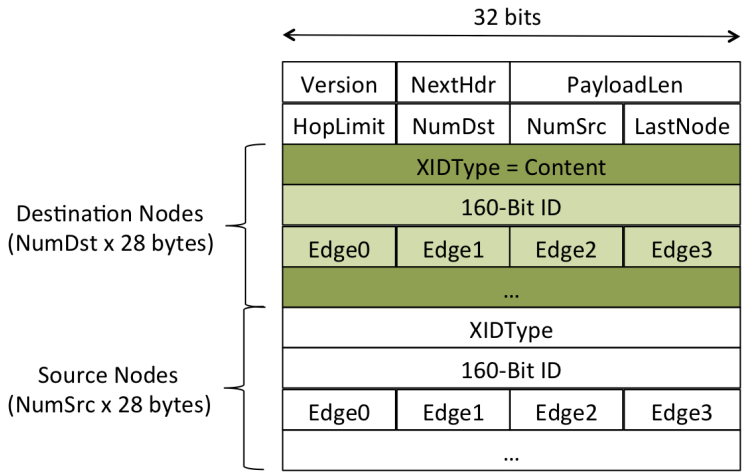


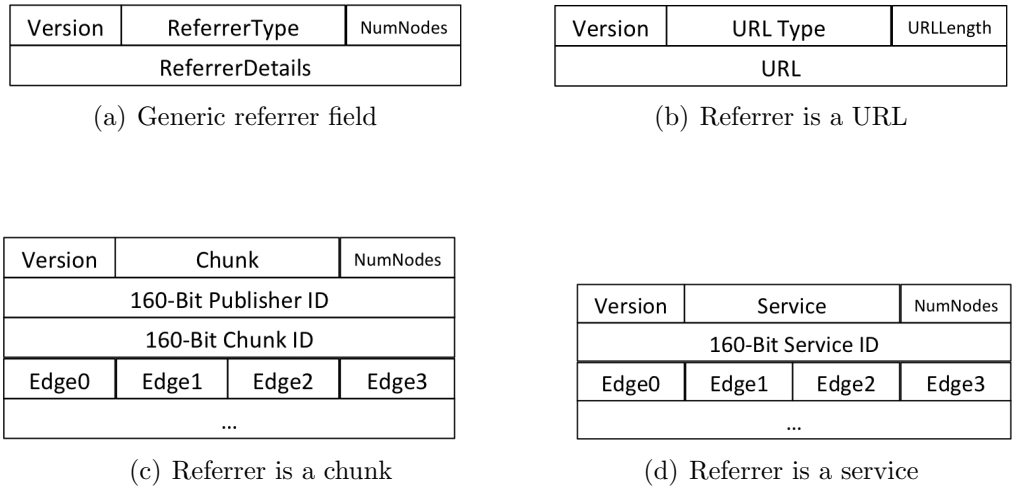
Figure 4.10: Illustration of a chunk request. The client simply sends a packet with a destination address whose original intent is for the desired chunk. Routers that possess content routing capability, also referred to as content routers, perform name-based routing of the request. Routers without content routing capability simply forward the packet on the path to the specified fallback node. Any content router that has a copy of the desired chunk can respond directly to the request. The above figure is borrowed from [74]. See [74] or [83] for a detailed description of all fields in the packet header.

In our model, we include an optional referrer field after the source DAG, which contains information about how the client discovered the URI for the chunk. The referrer field in the request packet in our content delivery model performs a similar function to the referrer field in an HTTP request [24]. Specifically, the referrer field provides information about

how the requester discovered the identifier for the content. This information is useful for several reasons, already discussed in Section 4.3.3. For instance, it enables proper attribution when the same ad is linked from different websites. Similar to what happens with the HTTP referrer field, clients may have incentives to report false information or avoid reporting the referrer altogether. We discuss the privacy implications of the referrer field and the mechanisms provided by our model to incentivize clients to truthfully report the referrer in Chapter 6.

We illustrate the structure of the referrer field in Figure 4.11(a) and provide specific examples for URL referrers, chunk referrers and service referrers in Figures 4.11(b), 4.11(c) & 4.11(d) respectively. The ReferrerType determines how subsequent entries in the referrer field are interpreted. Our model specifies three referrer types namely services, URLs or master chunks but other referrer types could be specified in the future. When the referrer is a chunk, then the publisher of the chunk is specified together with the complete DAG address of the chunk. The router or cache module is able to tell the presence of a referrer field by examining the payload length specified in the request packet. In the absence of a referrer field, the payload length specified in the packet indicates zero bytes.

Upon receipt of the packet, a router first checks the intent in the destination field. In this case, the router learns that the request is for content (i.e., XIDType = Content). This triggers one of two actions depending on the capabilities of the router. A content router, which is a router that is capable of name-based routing, may instruct its cache module to respond directly to the request if it has a copy of the chunk. Otherwise, it will forward the



(a) Generic referrer field

(b) Referrer is a URL

(c) Referrer is a chunk

(d) Referrer is a service

Figure 4.11: Illustration of the optional referrer field in a chunk request packet. The referrer field contains information about how the end-user discovered the identifier for the requested content. The referrer type determines how the contents of the referrer field and how they are interpreted. Our model specifies three referrer types namely services, URLs or master chunks but other referrer types could be specified in the future. The NumNodes field indicates the number of nodes in the DAG, when the referrer contains DAG addresses. In the case of a URL, the NumNodes is replaced by a Length field, which indicates the length of the URL in bytes.

request along the path to the next hop specified in the packet.

On the other hand, a router that is not capable of name-based routing simply forwards the request along the path to the fallback. Any cache module attached to a content router on the path to the fallback responds to the request if it has a copy of the chunk. Otherwise, the cache module of the fallback will ultimately respond to the request with the desired chunk, if it is available. Cache modules that respond to requests log meta-information associated with the chunk (e.g., publisher, CDB, type of service), the request (e.g., source of request, referrer field), the delivery (e.g., time of delivery) and the final destination (e.g., destination DAG).

Routers forward chunk responses on the path to the end-user. Content routers also forward a copy of the chunk to their cache module(s) for potential caching. Routers are

able to identify chunk responses by examining the original intent in the source field of the response packet. Caching in our CCN model is performed at the chunk level. This implies that cache modules need to reassemble any segmented chunks before making caching decisions. Applications can avoid the issues associated with segmentation by limiting the size of chunks.

The caching decision is guided by the local caching policy and makes use of the meta-information contained in the chunk and the local CDB database. In practice, the real policy is the decision on what to purge from the cache when it becomes full, since this decision significantly impacts the performance of the cache [139, 142, 181]. Cache owners may optimize cache replacement algorithms based on factors such as the type of service contracted by the publisher, the monetary reward to the cache owner for serving the object from the cache, the validity periods of the chunk, the size of the chunk and the caching hints specified in the chunk. For example, when a chunk that is related to multiple chunks gets evicted, then it makes sense to evict the related chunks as well.

4.5.3 Meta-information gathering

The structure of routable content in our CCN content delivery model provides an effective way for different network entities to obtain the required meta-information. We summarize the means by which different entities obtain required meta-information in our model in Table 4.6. This table provides comparable information to that specified for the TCP/IP content delivery model in Table 4.2 on page 105.

Table 4.6: Summary of the meta-information required by different network players and the means by which they gather the required meta-information in our proposed content delivery model for XIA.

Network player	Meta-information required	Means to obtain required meta-information
End-user	Trustworthiness of content retrieved	CID, authentication in application-level chunk meta-information
Publisher	Content access information (by whom, when and how)	Cache module logs, CDBs
	How do end-users discover content?	Referrer field in request packet
	Independent content access information to verify bills received from a CDN	Cache control in cache-level chunk meta-information, redirection through a service
CDB	Content delivery information (how much and to whom)	Delivery logs from affiliated cache modules
	Entity to bill for content delivery	Cache-level chunk meta-information
Cache owner	Entity to bill for content delivery	Cache-level chunk meta-information
	Type of caching service to provide	Cache-level chunk meta-information

Naming a chunk after the hash of its content makes it easy for clients to verify the integrity of the chunks they receive. Clients simply hash the contents of the chunk and compare it with the identifier that they requested. In addition, authentication of master chunks allows clients to verify the provenance of all chunks specified in the master chunk,

while obviating the need for authentication in every chunk. Authentication of chunks also allow cache modules to verify the cache-level meta-information carried along with chunks. This ensures that cache modules and their affiliated CDBs account and properly bill for content delivery.

Before we end this section, we identify the key facilitators in our content delivery model and the functionalities they enable in Table 4.7. The table shows that the meta-information contained in the chunk plays a vital role in replicating several functions that are currently performed by HTTP and URLs in the TCP/IP content delivery model (see Table 4.3). In the next section, we discuss accounting and billing for content delivery in more detail.

4.6 Accounting and billing for content delivery

Accounting and properly billing for content delivery in any networking model requires access to trustworthy information. We identify three basic categories of information necessary for these tasks.

The first category of information relates to the properties of the content and it is generally static over a reasonable time period. This category includes the identity of the content publisher, the validity periods for the content, the type of caching service required for the content and the CDB (if any) that handles information and money flow for the content. The second category relates to properties of the content request and includes information about the identity of the requester and information about how the requester obtained the identifier for the object. This information category is dynamic and differs for

Table 4.7: Illustration of some key facilitators in our proposed content delivery model and examples of the functionalities that they enable.

Facilitator	Functionality enabled
Search engine	Content discovery
Uniform resource identifier (URI)	Content discovery Content integrity check
Domain name service (DNS)	DAG resolution
Meta-information in chunk	Content authentication Delegation of content delivery to a third-party Third-party cache control Identifying publisher to bill for content delivery Accounting and billing for content delivery Independent verification of content delivery
Cache module	Gathering content access information
Request packet	Identifying source of request Identifying content discovery agent

each requester.

The final information category relates to the nature of the contractual agreement between the publisher and the CDB and captures details such as the type of service contracted, the duration of the contract, the maximum amount that the publisher is willing to pay during a billing cycle, the valid list of referrers that triggers paid content delivery and the publisher's public key(s). This information category is also fairly static for the duration of the contract, applies to many chunks, and is provided to caches in advance.

We make a design decision to securely bind the first information category to the content. This decision is driven by the fundamental goal of making content self-sufficient, which allows cache modules to make independent caching decisions based solely on the content and previously acquired type three information. This has the potential to reduce communication overhead and the time it takes to make caching decisions. On the other hand, including meta-information with content may increase the content overhead. Nonetheless, we showed in Section 4.5.1 that the fixed field encoding scheme can minimize the chunk overhead.

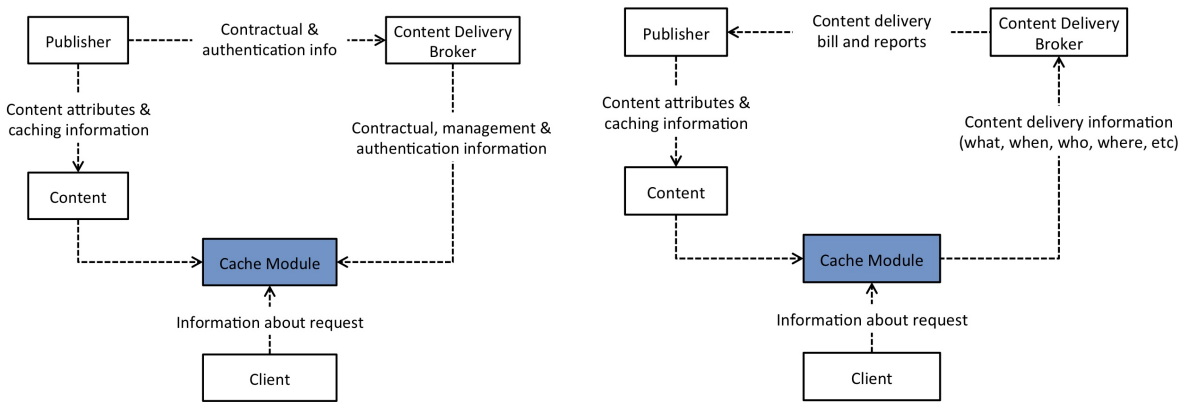
Unlike the first information category, which is bound securely to the content, we collect the second information category through content requests. Thus, in addition to specifying their identity, requesters also provide information in optional fields in the request packet. One such field, which immediately follows the source field in the request packet, is the referrer field, which we already discussed in detail in Sections 4.3.3 & 4.5.2.

The final information category is collected at the time of a contractual agreement and maintained by the CDB in a database. This database (or a portion of it)[8] is shared with cache modules when they become affiliated with the CDB. In addition, the management server at the CDB sends regular updates of the database to affiliated cache modules. The cache modules can also poll the management server for updates, for instance after a period of downtime. In this way, the bulk of the database transfer occurs only once, that is when the cache module first joins the CDB[9].

[8]A publisher may decide to pay for content delivery only in a specific geographic region. In such a situation, the CDB will share the publisher's information only with cache modules located in that region.

[9]This assumes that the database is structured in a way that makes differential updates possible. This

This design decision to locally store the third information category ensures fully decentralized operation of cache modules with minimum communication overhead. We illustrate the information flows in our content delivery model in Figure 4.12. We go through an example in the next section to illustrate how cache modules and CDBs use the different meta-information to account and properly bill for content delivery in XIA.



(a) Information flows to support accounting (b) Information flows to support billing and reporting

Figure 4.12: An illustration of the information flows to support accounting, billing and reporting in our CCN content delivery model. Content attributes and caching information are securely bound to chunks. The CDB keeps a database of contractual and authentication information for publishers, which is shared with affiliated cache modules. Clients provide information about each request, such as the source of the request and the referrer. The cache module uses all sources of information to decide what to cache, what to remove from cache and when to serve an object from cache. Cache modules periodically reports information about content delivery to CDBs. CDBs aggregate delivery information and sends bills and reports to publishers.

4.6.1 Example

We consider a cache module, which is affiliated with a single content delivery broker. In addition, we assume that the cache module has an updated local version of the CDB's database, which contains all the relevant information about content delivery contracts

may entail the use of version numbers and numbered lines for all database entries, for example as practiced with OSPF databases [182].

between the CDB and paying publishers.

For now, we assume that this information includes publishers' ID, the public key(s) used to verify signed chunks of each publisher, the type(s) of service contracted by each publisher and the duration of the content delivery contract between each publisher and the CDB. The database also contains information about the valid list of referrers that triggers paid content delivery for each publisher. In practice, optimizations will be employed to limit the list of valid referrers. For instance, a wildcard can be used after the domain name for URL referrers. The structure of the local version of the CDB's database is illustrated in Figure 4.13.

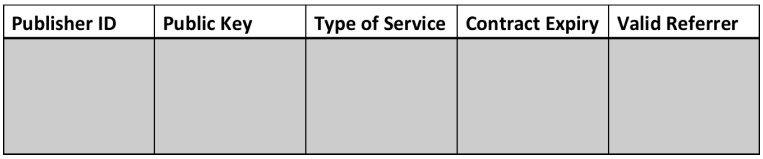


Figure 4.13: An example of the local version of the CDB's database stored at a cache module. This database is obtained from the content delivery broker and is regularly updated by the management server at the CDB. The cache module makes use of the information contained in the database to make caching decisions and to account for content delivered.

The cache module makes paid caching decisions for content of publishers who appear in the database. We only consider paid caching in our discussion because accounting and billing are only relevant in this case. For paid caching, signatures are required to ensure the correct party is billed. Ideally, the cache module verifies the authenticity of the cache-level meta-information before making a caching decision.

However, signature verification for all chunks, using current technology, imposes additional latency, which affects the performance of the cache module. In order to mitigate

this, cache modules probabilistically verify chunks. Cache module can make other optimizations in order to reduce the verification burden. For instance, a separate database could store the CIDs of verified chunks, even when they get evicted from the cache, in order to reduce the verification load, should the chunk be re-presented. We anticipate that all meta-information will be verified in the future when technology matures enough to handle such verification without any performance degradation.

After a decision has been made to cache a chunk, the cache module enters relevant information about the chunk in the cache database. An example of this database is illustrated in Figure 4.14.

Chunk ID	Publisher ID	Size	Type of Service	Caching Hint	Begin Validity	End Validity	Last Access	Number of Accesses

Figure 4.14: An example of the cache database stored at a cache module. This database is populated when a chunk is placed in the cache. It is also edited when a chunk is removed from the cache. The content router checks the cache database to determine if a content request can be served locally.

Any request for content in the cache database may trigger content delivery from the cache. The cache module checks the validity period for the chunk and may check the list of valid referrers for the publisher before making a decision to respond to the content request, depending on the type of service specified in the chunk. All content deliveries from the cache are logged in a content delivery log. An example of this log is shown in Figure 4.15.

At regular intervals, the cache module processes the logs for each CDB and sends the delivery log and other supplemental information (e.g., publisher ID of each chunk, size of each chunk and type of service specified in each chunk) to the aggregation server at the

Chunk ID	CDB	Destination DAG	Referrer Type	Referrer	Time of Delivery

Figure 4.15: An example of the content delivery log kept by the cache module.

CDB. The aggregation server can also poll the cache module for the delivery logs. The aggregation server processes all logs from affiliated cache modules to determine bills for publishers and revenues for cache module owners during a billing cycle.

The above example illustrates that the meta-information contained in the chunk, the information contained in the request and the local information obtained through the CDB management servers provide cache modules with all the information necessary to properly account for content delivery in an environment where all entities participate honestly. We discuss additional mechanisms we introduce to deal with the case where dishonest entities exist in Chapter 6.

4.7 Lessons

Our work in this chapter highlights two main lessons, which could be applied by other CCN architectures to ensure the realization of incentive-compatible systems.

The first lesson focuses on the need to provide all the currently useful functionalities that underpin content delivery. The choice that makes sense depends on the needs of the particular architecture and the mechanisms provided by the architecture to route content requests. Architectures that perform name-based routing of content requests must design

new mechanisms to replace TCP/IP facilitators and the ideas we presented in this chapter could be reused.

Secondly, recent efforts to design and analyze the performance of content routers mostly focus on the forwarding and storage requirements. Very little attention has been paid to the need to support accountability and payment flows. Without such support, however, it is unlikely that network operators will deploy sufficient infrastructure to support CCN. We proposed one means to realize accountability in CCN. Other architectures may make different choices. Nevertheless, designers must make provisions for and incorporate the additional information requirements in their content router designs, in order to develop content routers that are capable of meeting the real-world content delivery needs of different network players.

4.8 Related work

We group related work into two categories, namely proposals on structuring routable content in content-centric networks and design and evaluation of content routers.

4.8.1 Structure of routable content

We borrow the idea of chunks as routable content units from several proposals, such as [91, 183]. Further, the advantages of self-certifying identifiers have been well-documented in the literature ([9, 80, 100]). Our work combines these fundamental ideas to create intrinsically secure routable content units which are flexible enough to meet various application needs

and constraints imposed by network operators and cache owners.

4.8.2 Design of content-centric routers

A few studies identify requirements for a content router and propose and analyze the performance of such routers [95, 166, 174]. Our work is complementary to these efforts. First, we propose a different way to organize the functionality of a content router by splitting the forwarding function from the caching function. Secondly, we identify necessary information that must be maintained by the cache module in order to properly account for content delivery. The need for such information has not been recognized in previous attempts to design content routers. As such, our work provides new insights that can help to improve the design and evaluation of content routers.

4.9 Conclusion

In this chapter, we motivated the need to replicate useful TCP/IP content delivery functionalities that break in a content-centric network. We also demonstrated a concrete way to realize these functionalities in the eXpressive Internet Architecture. Our approach is driven by design decisions in three key areas, namely, the structure of routable content, the content delivery ecosystem and infrastructure and the way we distribute required information among different entities.

First, we make content self-sufficient by using a flexible routable content unit called a chunk, which carries meta-information related to content properties. This information is

securely bound to the content by means of self-certifying identifiers. Secondly, we introduce content delivery brokers as network entities that provide infrastructure to facilitate information and money flow between publishers and cache owners. Finally, we carefully split required meta-information needed to make caching and accounting decisions among three different sources.

In our design, chunks carry all the information related to the content, such as the publisher, the content delivery broker, the type of caching service required and authentication information. In addition, clients provide information related to the chunk request, such as the address of the client and the source of content discovery, which we refer to as the referrer, during the chunk request process. Finally, content delivery brokers maintain all management and contractual information, such as the list of paying publishers, the types of service contracted by each publisher, the duration of contracts, the valid list of referrers and authentication information.

These design decisions enable us to replicate all the functionalities that are currently realized through HTTP and URLs in the TCP/IP content delivery model, while providing enhanced security properties and a more efficient means to disseminate content. We also identified lessons that designers of CCN-based architectures can employ to improve their architectures. In the next chapter, we employ several examples and use cases to demonstrate how our design can be used to support diverse content delivery applications and meet the goals of different network players.

Chapter 5

Applications of the Proposed CCN Content Delivery Model

5.1 Introduction

The CCN content delivery model, which we outlined in Chapter 4, provides new security properties, in addition to new and more efficient ways to accomplish currently useful content delivery functions, while still providing support for the conventional ways of accomplishing different content delivery tasks.

In this chapter, we illustrate the versatility and generality of the proposed content delivery model with the aid of three application scenarios. We first apply the model to web-based content delivery (Section 5.2). We also illustrate how the proposed model can

be extended in novel ways to applications such as email (Section 5.3) and near real-time streaming (Section 5.4). Finally, we extend the model to cover other principals, such as service (Section 5.5).

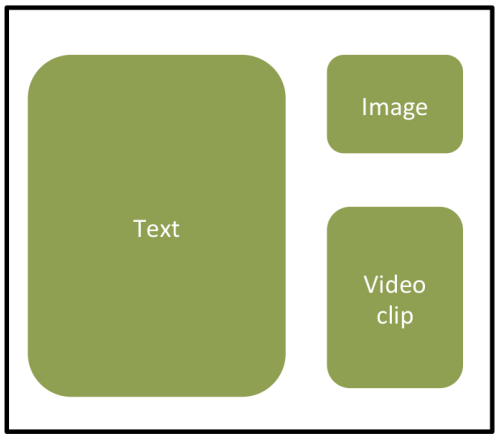
5.2 Web-based content delivery

Web-based content delivery forms a significant proportion of content delivery on the Internet [5]. The delivery paradigm on the web, where the emphasis is on content dissemination to multiple users, rather than pairwise communications, stands to benefit significantly from the CCN content delivery model. Thus, it is important that any proposed CCN architecture deals efficiently with web-based content delivery. We employ three use cases to illustrate the versatility of the content delivery model that we outlined in the previous chapter.

The first is a simple case of a small publisher, who serves a static video, image and text. The publisher is open to the possibility of transparent caching of his content within the network, but he also desires to obtain some content access statistics. The second use case is more complicated in that we have a publisher who serves both static and dynamic content. In addition, the publisher purchases caching services from a content delivery broker and requires a means to verify the bills sent by the CDB. The final use case considers a publisher who serves content through multiple distribution partners (e.g., an advertising agency). In this case, the publisher requires a means to properly account for content delivery by each of its distribution partners in order to compensate them appropriately. We discuss each of these use cases in the next sections.

5.2.1 Publisher does not pay for third-party content delivery

Let us assume that a small publisher wants to serve the web page shown in Figure 5.1(a). The same content is served to all visitors to the web page. The publisher does not purchase caching services from any CDB, but is open to transparent caching of the image and the video clip. At the same time, the publisher would like to obtain an estimate of the number of times the image and the video are viewed. The publisher can accomplish the above goals with our CCN content delivery model as follows.



(a) Web page

```
<!DOCTYPE html>
<html>
<body>
....
text text text
...
<img src="xid:image_DAG;https://www.small_publisher.com/image"
 alt="image" />....
....
text text text
...
<video width="320" height="240" controls="controls" preload = "none">
  <src="xid:master_chunk_DAG;http://www.small_publisher.com/video"
 type="video/mp4" />
</video>
....
</body>
</html>
```

(b) Sample HTML5 document

Figure 5.1: Sample web page and HTML code for the small publisher

First, the publisher creates a chunk for the image. For simplicity, we assume that the image fits in a single chunk, but an extension to multiple chunks is straightforward. In the meta-information for the chunk, the publisher specifies transparent caching as the type of caching service desired. The publisher also converts the video clip into chunks. Here, we assume that the video clip is too large to be efficiently handled by the network and cache modules as a single chunk. Hence, the publisher creates as many chunks as necessary to represent the video clip. The publisher marks all the chunks for transparent caching.

In addition, the publisher creates a master chunk for the video, which contains the identifiers of all the chunks that make up the video and how they are assembled. Here, we assume that the payload in the master chunk is formatted as a DASH MPD file [153]. In order to obtain some access information for the video, the publisher marks the master chunk as uncacheable. The video is named after the hash of the contents of the master chunk. In this way, all requests for the video first fetch the master chunk from the publisher, before fetching the constituent chunks from the network.

After preparing the image and the video clip, the publisher specifies their URIs in the HTML document. Here, the publisher chooses to take advantage of content routing while providing the possibility for traditional host-based routing. This is done by specifying a URI that contains both a DAG address and a URL. Clients choose to request the objects with the DAG address or the URL. A sample HTML document that generates the web page is shown in Figure 5.1(b). Note that the URI may contain URLs of different scheme types.

The final step in the content preparation stage involves making the web page available to end-users. Here, the publisher has three options. In the first, the publisher creates a master chunk for the web page, which clients request using the same techniques that they use to request other chunks. This approach enjoys all the benefits of content-centric networking. In order to obtain page access information, the publisher marks the master chunk as uncacheable. The second approach relies on traditional host-based routing, in the sense that clients use a URL for the page (e.g., *http://www.small_publisher.com*). During

a request, the URL is resolved to the publisher's web server and the publisher returns the HTML document.

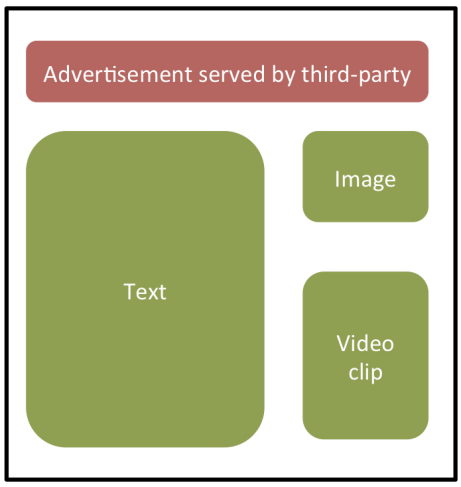
The final option for the publisher is to specify a DAG address for the web service itself, which has a fallback to the web server. We did not previously discuss other XIA principals, but the structure of URIs and the semantics of a request are the same for all principal types in XIA [83]. Furthermore, a request for a service follows the same structure shown in Figure 4.10. The only difference between a service request and a content request is the XIDType specified in the destination intent, which is a service instead of content. We assume the publisher opts to make the web page available as a service, with a fallback to the web server. Thus, the URI for the web page takes the form of *xid:webservice_DAG_address;http://www.small_publisher.com*.

End-users request the web page directly from the web service, which allows the publisher to estimate the number of times the image and the video clip are delivered from the publisher's web page. Once in possession of the HTML document for the page, clients request and obtain the image from anywhere in the network. However, the master chunk for the video is always delivered from the publisher's website, which again allows the publisher to more accurately estimate the number of times the video is downloaded. This is true even when the video is deep linked on other websites. After retrieving the master chunk from the publisher, clients request all the individual chunks for the video from anywhere in the network. Thus, the publisher benefits from transparent caching in the network and at the same time obtains content access information.

Nevertheless, the mechanisms provided by our model do not allow the publisher to both enjoy transparent caching and obtain accurate content access information at the same time. In effect, the publisher will not know how much of the video was watched. This is because any deep linking to cacheable content (e.g., image or individual chunks of the video) results in direct delivery from the network, without any communication with the publisher. Thus, the publisher only obtains information about content access to his own site, as well as to uncacheable master chunks, which may not represent the total number of content accesses in the network. This situation is no different from what currently happens on the web. In order to obtain more accurate statistics while still benefiting from in-network caching, the publisher must purchase caching services through a CDB. We discuss this in the next use case.

5.2.2 Publisher pays a third-party for content delivery

In the second use case, we consider a publisher that serves text, a video and an image to end-users. Unlike the previous use case, the publisher contracts for content delivery with a CDB. Furthermore, the publisher serves advertisements for an advertisement agency. The advertisement agency determines the advertisement shown to an end-user and the advertisement may vary for different end-users. Nevertheless, the publisher desires to be properly compensated for ad requests that originate from his website. We illustrate a sample website for the publisher in Figure 5.2(a).



(a) Web page

```
<!DOCTYPE html>
<html>
<body>
....
/*Banner ad */
<script type = "application/javascript"><!--
publisher_ID = hash(publisher's public key);
ad_width = 468;  ad_height = 60; //--> </script>

<script type = "application/javascript"
src="xid:adservice_DAG;http://www.adagency.com/showimage.js" >
</script>
...
text text text
...
<img src="xid:image_DAG;http://www.cdb.com/publisher.com/image"
 alt="image" />....
....
<video width="320" height="240" controls="controls" preload = "none">
  <src="xid:master_chunk_DAG;http://www.cdb.com/publisher.com/video"
 type="video/mp4" />   </video>
....
</body>
</html>
```

(b) Sample HTML5 document

Figure 5.2: Sample web page and HTML code for the publisher. The text, image and video are created by the publisher. The banner advertisement is chosen and served by an advertisement agency.

The content preparation process for the publisher is similar to the use case described in the previous section. The publisher marks all chunks that require caching with the paid caching service purchased from the CDB. The fundamental difference in the content preparation stage, when compared to the previous use case, is that the publisher inserts the CDB, the type of caching service contracted and a signature (for authentication purposes) as meta-information in every chunk for which he desires paid content delivery services. The type of caching service may be different for each chunk, depending on the specific requirements of the publisher.

The publisher specifies URIs that include DAG addresses and URLs for both the image and the video, as illustrated in the sample HTML code in Figure 5.2(b). The DAG address for the video refers to the master chunk. Finally, the publisher inserts code in the HTML document to display contextual advertisements by an ad agency. This is similar

to what happens today, for example with Google AdSense. Nonetheless, our model adds new features to make the process more flexible and efficient. For instance, ad agencies currently assign unique IDs to distribution partners. Our model eliminates the need for a new publisher identity because the publisher ID provides a unique and secure way to identify publishers. Moreover, the flexibility in XIA allows the publisher to employ different principals in the URI. For instance, the ad agency can run distributed servers in multiple locations which can respond to ad requests based on the end-user's location and provide a DAG address to the service as a URI . This reduces latency and network traffic.

Publishers are compensated for advertisements based on the number of impressions, the number of click-throughs, or a combination of both [184–188]. Usually, advertisement agencies compute the compensation due to each partner. Our model allows publishers to independently estimate their potential revenues based on the number of ad impressions. Publishers can also employ the PING attribute in HTML5 to estimate the number of ad clicks and the subsequent compensation [157]. This is no different from what happens today in the TCP/IP content delivery model [189].

The publisher uses access to the webpage to estimate the volume of images delivered by the CDB and the potential revenues from ad impressions. The publisher also uses access to the video master chunk to estimate the volume of video delivered by the CDB. However, these estimates may not be accurate because of the possibility of deep linking. We discuss a few options to address this issue in Chapter 6.

5.2.3 Keeping track of content delivery from multiple distribution partners

In our final use case, we consider an advertising agency that wants to keep track of the number of advertisements served by each advertising partner. We make the case more interesting by assuming that the advertising agency pays for caching of banner advertisements (images) through a CDB. The situation where the advertising agency serves all advertisements from its servers is trivial. Furthermore, we restrict our discussion to the new mechanisms provided by our model to help the advertising agency to achieve its goals.

As highlighted in the previous use case, advertising partners (publishers) usually insert a code, which is provided by the advertising agency, on their web pages in order to display relevant advertisements to end-users. When executed, this code uses the contextual information contained on the web page and other parameters related to the end-user to provide a relevant advertisement and an appropriate link, which the user can follow to obtain more information. We illustrate this in Figure 5.3.

The advertisement agency provides the code shown in Figure 5.3(a), which the publisher inserts in the HTML document for his web page. The execution of this code results in the HTML equivalent code shown in Figure 5.3(b). We use the equivalent HTML code in Figure 5.3(b) to illustrate the mechanisms provided by our content delivery model to help the advertisement agency to serve the images through a CDB and account for the number of impressions served by each advertisement partner and the number of click-throughs. For simplicity, we assume that all ads yield the same revenue for the advertising partners.

```
<!DOCTYPE html>
<html>
<body>
....
/*Banner ad */
<script type = "application/javascript"><!--
publisher_ID = hash(publisher's public key);
ad_width = 468;  ad_height = 60; //--> </script>

<script type = "application/javascript"
src="xid:adservice_DAG;http://www.adagency.com/showimage.js" >
</script>
...
....
</body>
</html>
```

(a) Sample code provided by advertising agency for insertion in a partner web page.

```
<!DOCTYPE html>
<html>
<body>
....
/*Banner ad */

<a href="xid:caradservice_DAG;http://www.adagency.com/caradredirect" >
<img src="xid:carad_DAG;http://www.cdb.com/carad"
height = "60 " width = "468 " >
</a>
...
....
</body>
</html>
```

(b) Equivalent HTML code generated when client loads web page

Figure 5.3: An example of a script to serve banner advertisements on a partner website and an equivalent HTML code to perform the same function.

During the content preparation stage, the ad agency creates chunks for all advertisements. To simplify our discussion, we assume that all ads fit in a single chunk. An extension to the case where ads are composed of multiple chunks (e.g., video) is straightforward (see e.g., Figure 4.9(b)). The meta-information for each chunk contains the ID of the ad agency, the contracted CDB, the type of caching service contracted by the ad agency and a signature for authentication. The ad agency also specifies the periods during which the advertisements can be delivered by the cache modules affiliated with the CDB. Furthermore, the ad agency sets up an XIA service that redirects clicks on the ad to the relevant web page of the advertiser. For completeness, the ad agency also specifies URLs in the URI to provide support to clients that lack content-centric networking capabilities.

Let us assume that the ad agency serves an advertisement for a car on a partner website. The end-user's client requests the advertisement using the DAG address specified in the HTML document. Any cache module affiliated with the ad agency's CDB may respond to the request. The cache module logs the source of the request and the referrer.

Further, when the end-user clicks on the advertisement, the ad agency receives the address of the end-user together with the referrer information before redirecting the user to the advertiser's web page.

Thus, the ad agency obtains all the relevant information to compute the number of ad impressions served by each partner and the number of clicks originating from each partner. In particular, the ad agency uses the referrer information accumulated by the CDB, through all its affiliated cache modules, to compute the number of ad impressions served by each partner. In addition, the ad agency uses the referrer information it collects from its redirection service to estimate the number of ad clicks on each partner site.

It is possible that the estimate obtained through the CDB may underestimate the number of ad impressions. For example, some cache modules may cache and serve advertisements transparently, which will not be reflected in the statistics collected by the CDB. Also, some clients may choose to enter false or empty referrer information. This could be an issue if the ad agency pays its partners based on the number of ad impressions. To rectify this, the ad agency could trust the access statistics collected by the advertising partner when a discrepancy arises. Alternatively, the pricing models for ad impressions and clicks may be slightly modified to take the likelihood of underestimation into account.

5.3 Email

Electronic mail remains one of the Internet's most popular and important applications [34, Chapter 2]. In this section, we illustrate how our CCN content delivery model facilitates efficient email delivery. We limit our attention to the delivery of an email with several large objects (e.g., images, video, etc) to multiple users, since this use case stands to benefit the most from the CCN content delivery model.

We illustrate how the main ideas from our CCN content delivery model can be easily extended to the email delivery process in Figure 5.4. Let us assume that a sender sends an email with a video and several images to multiple recipients. In the current host-centric paradigm, the sender creates an email message where all the objects (video and images) are placed into one message and sent individually to all recipients. This approach wastes a lot of bandwidth as the number of recipients increases, especially if the objects are very large. Currently, most mail servers limit the size of email attachments in order to address this issue, among others.

With the CCN content delivery model, the sender simply creates an email message that contains URIs to the chunks of the large objects. Once in possession of the email, a recipient's email client uses the URIs in the email body to retrieve all large objects from any network location. This approach significantly decreases the amount of network traffic, since the outgoing mail server only sends the email, without all the large component objects, to other mail servers.

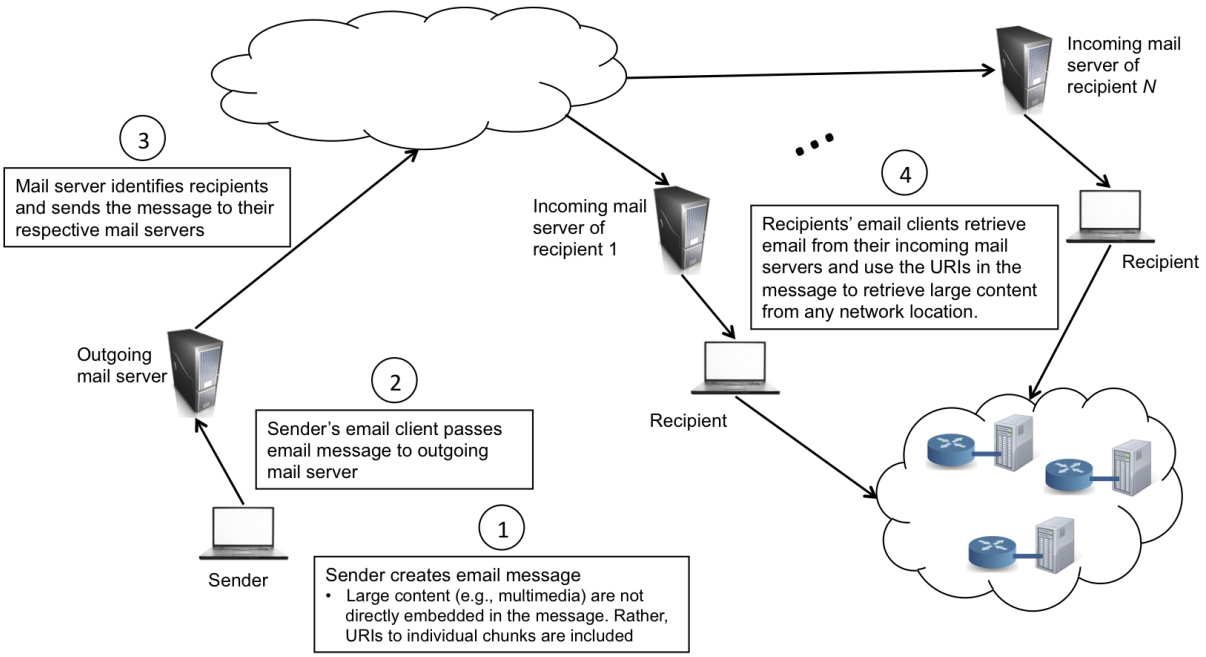


Figure 5.4: The CCN content delivery model can be used to efficiently deliver electronic mail with large attachments to multiple recipients. An email sender simply inserts URIs of large attachments (e.g., multimedia content) in the email body, which can be retrieved from any network location.

In particular, when the links between the mail servers and clients are capacity constrained, our CCN content delivery model provides significant improvements in network traffic reduction and latency, without imposing any limits on the size of email attachments. This is because clients can make use of local high capacity links to retrieve large multimedia components of the email from any network location that has a copy.

Besides the above simple extension, we can envisage more radical extensions of the CCN content delivery model to support email delivery. This involves treating the email itself as a chunk, with its own content identifier, as shown in Figure 5.5. Here, we assume that the sender of the email marks it for caching. The email is sent to the outgoing mail server after creation.

The outgoing mail server performs chunking and (optionally) inserts authentication information before naming chunk(s) of the email. The mail server creates a URI for the chunk(s) that correspond to the email. This URI takes the same form as that already described for web content. Furthermore, the mail server specifies a fall back option in the DAG address to itself. The mail server uses SMTP to push the URI and some meta-information about the email to all incoming mail servers specified in the recipient list of the email. The meta-information consists of the sender, subject, date, priority, recipients and other relevant entries from the email header. In effect, there is no exchange of the actual email message between the mail servers.

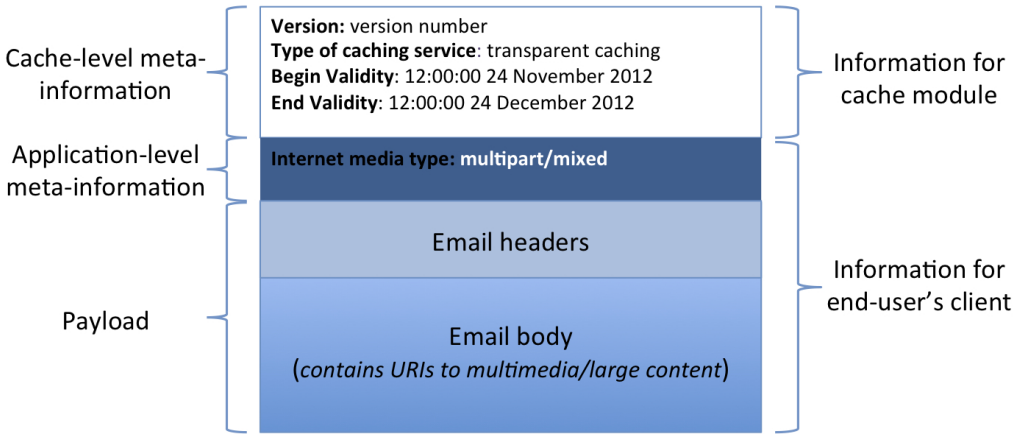


Figure 5.5: Email delivery using the CCN content delivery model. The sender creates an email, which is treated as a chunk in the CCN content delivery model. In such a scenario, the outgoing mail server pushes the URI of the email (perhaps with some meta-information) to mail servers of all recipients. Recipients obtain the URIs and meta-information from their mail servers and retrieve desired emails from nearby caches.

When recipients connect to their mail servers, they obtain the meta-information associated with the email (e.g., sender, subject, date, priority, other recipients, etc) and the URI for the email message. After selecting the emails of interest, the recipient instructs the email client to retrieve the corresponding URIs from any network location using the

same mechanisms already described for web content delivery. The chances of finding the email in a nearby cache module increases as the number of email recipients increase. When the email cannot be retrieved from a nearby cache, a connection is eventually made to the sender's mail server to retrieve the email using an XIA content request. This process is illustrated in Figure 5.6.

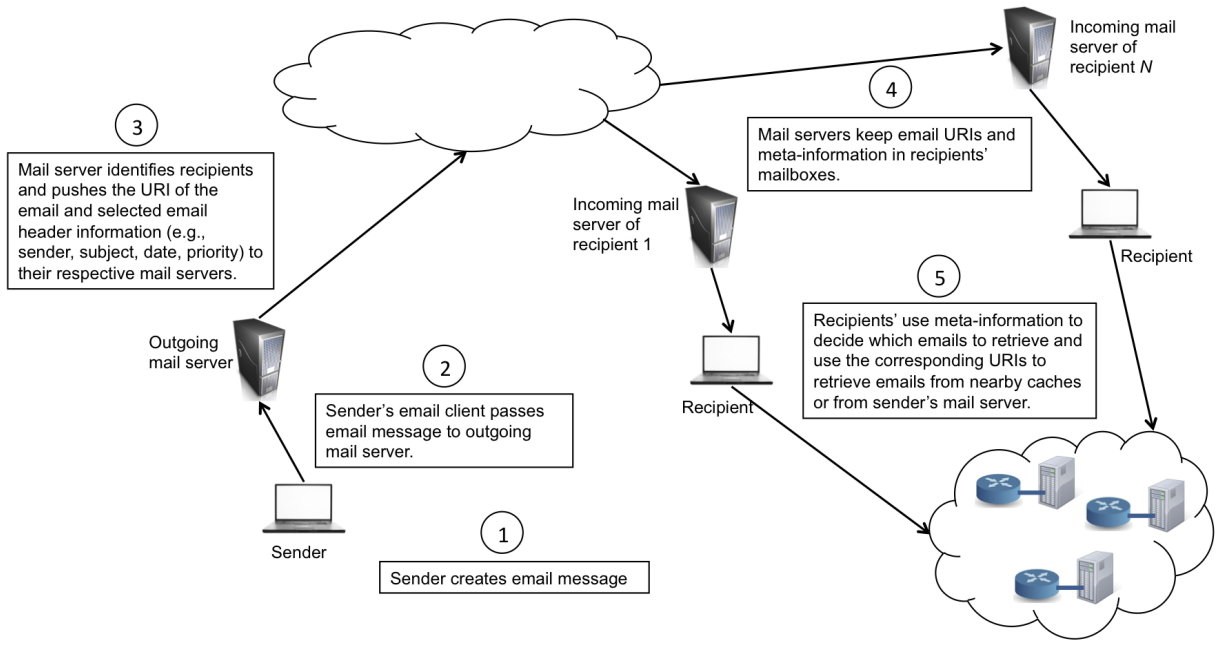


Figure 5.6: An email can be treated as a chunk. Outgoing mail servers create a URI for the email, designating a fallback to the mail server. The mail server uses SMTP to push the URI and other meta-information associated with the email (e.g., sender, subject, date, etc) to all mail servers associated with the recipients of the email. Recipients obtain only the URI and the specified meta-information from their mail servers and instruct their email clients to retrieve desired emails from any network location. When the email is not cached, the email client contacts the sender's mail server to retrieve a copy of the email.

The email delivery model described above has the potential to significantly reduce the network traffic when an email message is sent to several nearby recipients, since actual email transfer occurs only when recipients indicate an interest to read them. Further, popular mass delivered messages are likely to be cached locally, which further reduces the amount of network traffic.

Nevertheless, the new model could lead to increased latency between the time a recipient sees the email header and the time the recipient retrieves the email. This is not a problem since email messages are sent without real-time communication expectations. Especially for bulk email, our model provides significant reductions in network traffic, without substantial degradation of the quality of experience for recipients.

Moreover, the proposed model renders current spam detection and filtering techniques, which rely on analysis of the headers and content of the email by intermediate servers and destination mail servers (see e.g., [190, 191] and references therein), useless since the actual body of the email is not delivered until explicitly requested by the client. Thus, in our model, spam filtering based on content analysis is critical at the outgoing mail server since it is the only entity with access to the contents of the email.

Additional techniques, which rely on inferring trust in the sender are also required (see e.g., [192]). For instance, a mail server can build a database of trusted mail servers and infer (transitive) trust in received email headers based on the originating mail server and the path in the DAG. Developing effecting spam filtering techniques for the new delivery model is an interesting area for future research.

5.4 Near real-time streaming

Video consumption forms a major part of current and anticipated future Internet usage, and must be fully supported in any network architecture [4, 5]. For instance, video traffic

is expected to form more than ninety percent of consumer Internet traffic beyond 2014 [4]. Unlike other multimedia content, such as audio and photos, video files tend to have larger sizes and are therefore ill-suited for delivery approaches that rely on bulk downloads (e.g., FTP, HTTP), especially for applications with low latency requirements.

Various streaming technologies have emerged over the years to deliver large on-demand or live video files on the Internet. These include protocols, which require specialized streaming servers, such as real-time streaming protocol/real-time protocol (RTSP/RTP) [178, 193] and real-time messaging protocol (RMTP) [194]. Other streaming technologies rely on HTTP and, thus, can stream video using the same servers used to deliver web traffic. These include progressive HTTP downloads, Apple HLS [152] and DASH [153].

Because they do not require any additional specialized infrastructure, streaming technologies that employ HTTP currently dominate video delivery on the Internet [195]. Thus, it is imperative for CCN architectures that employ name-based content routing to develop new mechanisms to provide support for streaming services. The CCN content delivery model we presented in Chapter 4 can easily support on-demand streaming (see e.g., Figure 4.9). Hence, we focus our attention in this section on how our CCN model supports real-time streaming applications.

Consistent with the design principles of XIA, we use self-certifying names to identify chunks in our model. Hence, clients can only request objects that have already been created. Thus, our model does not support using in-network caches to provide live streaming services because of the lag between object creation by the publisher and consumption by

the end-user. This contrasts with CCN models that employ hierarchical naming schemes, where an object can be named before it is created (e.g., NDN [3]). With this property, clients can issue requests for objects that do not yet exist, which can theoretically facilitate live streaming via in-network caches.

Because of the lag between object creation and object request, our model only supports near real-time streaming applications, where there is a slight delay between object creation by the publisher and consumption by the client. In such scenarios, we can make use of in-network caching for efficient content delivery. This follows from the natural ability of CCN to multicast content through some degree of caching. Furthermore, the publisher can use the meta-information in chunks to help in-network caches to make optimal caching decisions. We illustrate how our model uses in-network caches to support near real-time one-way streaming in Figure 5.7.

We consider the use case, where a publisher streams a near real-time audio/video presentation to multiple users distributed in several networks. For this application, the publisher creates a service identifier (SID) that corresponds to the particular stream. If the stream occurs at regular intervals (e.g., weekly), the same SID could be used for future streams. The publisher distributes the SID publicly or makes it privately available to eligible clients, perhaps after authentication. In addition, the publisher serves requests to the SID from a single location or pays for replication of the service across the network. We outline how such a service replication ecosystem could function in Section 5.5.

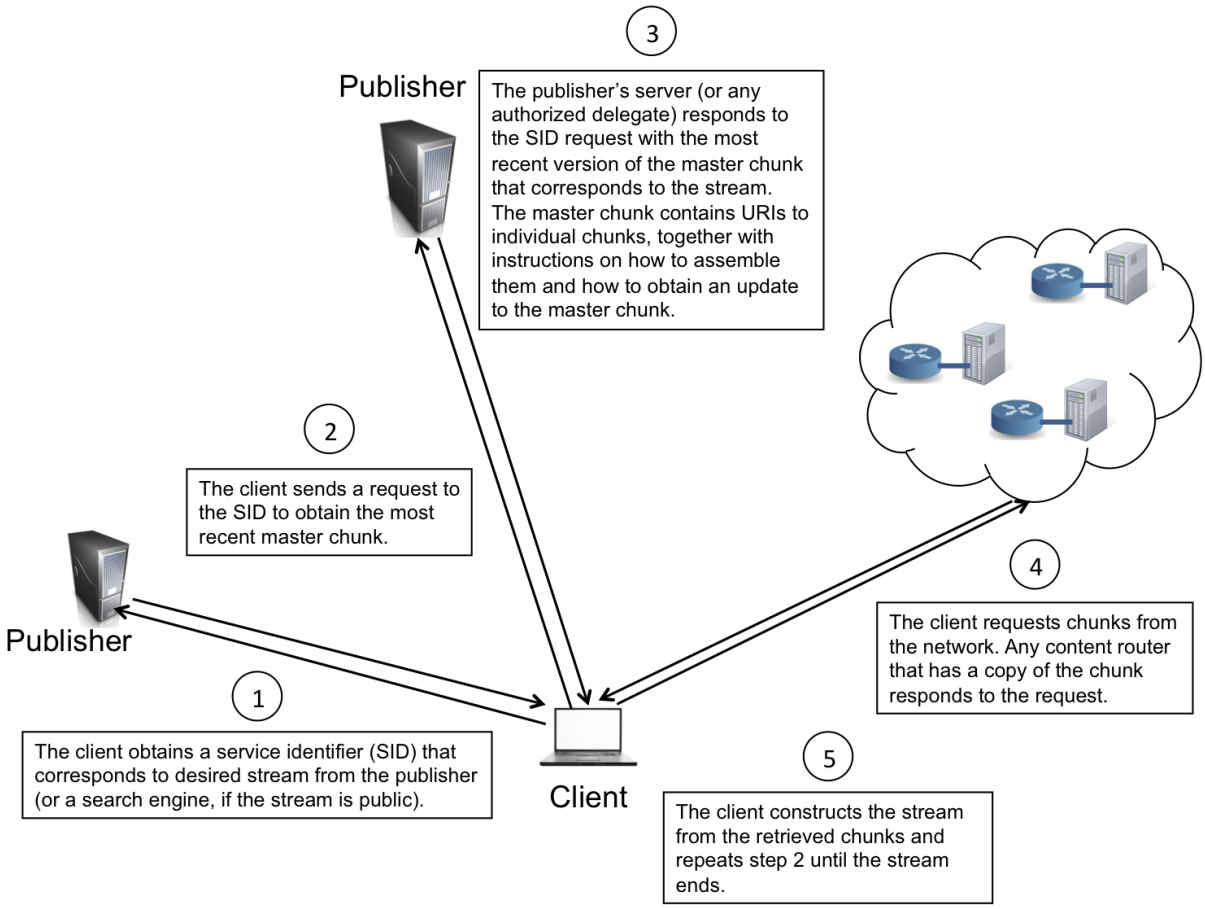


Figure 5.7: Illustration of how we support near real-time streaming in our CCN content delivery model. The use of self-certifying names, generated from the hash of the content, precludes support for caching-enabled live streaming. However, our model uses in-network caches to support near real-time streaming. Clients send a request to a service identifier (SID) that corresponds to the desired stream. The server responds with the most recent master chunk, which contains URIs for all chunks that compose the stream, together with instructions on how to assemble the chunks and how to request an updated master chunk. In order to reduce the size of the master chunk, the latest master chunk includes the identifier of the previous master chunk, together with the identifiers of all chunks generated since the previous master chunk.

In order to join the stream, eligible clients send a request to the SID. Upon receipt of the request, the publisher's server (or any server that can respond to the SID) responds with the current master chunk, which contains URIs of all chunks that compose the stream, together with instructions on how to assemble them and how to request the next update to the master chunk. Existing templates, such as the media presentation description (MPD) file defined for DASH [153] or the manifest file defined for Microsoft Smooth Streaming

[171], already contain specifications to accomplish the above functions. Thus, our model reuses these existing templates.

Several optimizations are made to reduce the size of the master chunk. For instance, the master chunk contains the identifier of the previous master chunk together with CIDs generated since the last master chunk. With this nested approach, clients who have all previous chunks only retrieve the updated CID list, while clients who are just joining the stream use the nested master chunks to retrieve all previous chunks of the stream. Such clients will experience significant latency because they need to go through several nested master chunks before reaching the start of the stream.

In order to reduce latency, the most recent master chunk may also contain URIs for the first few chunks of the video stream referenced in the first (i.e., most deeply nested) master chunk. The client requests the remaining nested master and individual chunks from any network location. Users not interested in watching from the beginning can start viewing using only new chunks referenced in the most recent master chunk.

Because multiple users may be streaming the same presentation, it is very likely that nearby caches will have copies of recent chunks, which facilitates efficient content delivery. The publisher facilitates caching decisions by indicating short validity periods for chunks. The publisher may also include other caching hints in the chunk header. By making use of in-network caches, our CCN content delivery model takes advantage of the multicast capabilities inherent in content-centric networking to reduce the amount of network traffic, as compared to current approaches where multiple simultaneous connections are established

between a streaming server and all eligible clients for the entire duration of the stream.

5.5 Extensions to other principals

Just as the dominant use of the Internet has evolved from point to point communication between two hosts to content dissemination between multiple hosts, it is entirely possible for other principals to dominate Internet usage in the future. For instance, cloud computing, a paradigm where distant servers perform hosting, computation, application delivery and other tasks as a service on behalf of paying customers is seen by some as the potential future of the Internet [77, 196, 197]. So much so that some proposals for a future Internet architecture, such as SCAFFOLD [77] and XIA [74], use service as a networking principal to address the possibility of service-centric networking (SCN).

In this section, we extend the ideas we presented in Chapter 4 to service-centric networking. We show that the content delivery model and ecosystem we presented can be easily generalized to other principals. In particular, we show that we do not require any major changes to our model to support service-centric networking.

5.5.1 Service delivery ecosystem

We illustrate the service delivery ecosystem and infrastructure in Figure 5.8. The basic elements of the ecosystem are similar to those presented in Chapter 4 for CCN content delivery. Specifically, server modules, which interface with routers to deliver services,

replace cache modules. A server module can be a commodity server and it is owned by the network operator or by a third-party. Currently, server modules are mostly owned by third-parties, but this could change in the future.

Unlike content delivery brokers, who only facilitate money and information flow in our CCN content delivery model, service delivery brokers (SDBs) perform two functions in the SCN service delivery model. In addition to facilitating money and information flow between service providers and server module operators, SDBs also perform a matching function by resolving service requests to the nearest server module. Strictly speaking, these two functions can be decoupled and performed by different entities. Thus, we can have integrated SDBs that perform both service resolution and billing, in addition to standalone service resolution or billing providers. For simplicity, we assume that all SDBs perform both functions. All other components of the ecosystem shown in Figure 5.8 perform analogous functions to those introduced in Figure 4.3.

One can see that the CCN ecosystem we presented exhibits a very desirable property in the sense that it can be easily generalized to other principals by simply replacing content with the name of the relevant principal. Hence, one can support multiple principals within our model just by installing all the relevant modules. In other words, the ecosystem we presented is flexible enough to support future evolution.

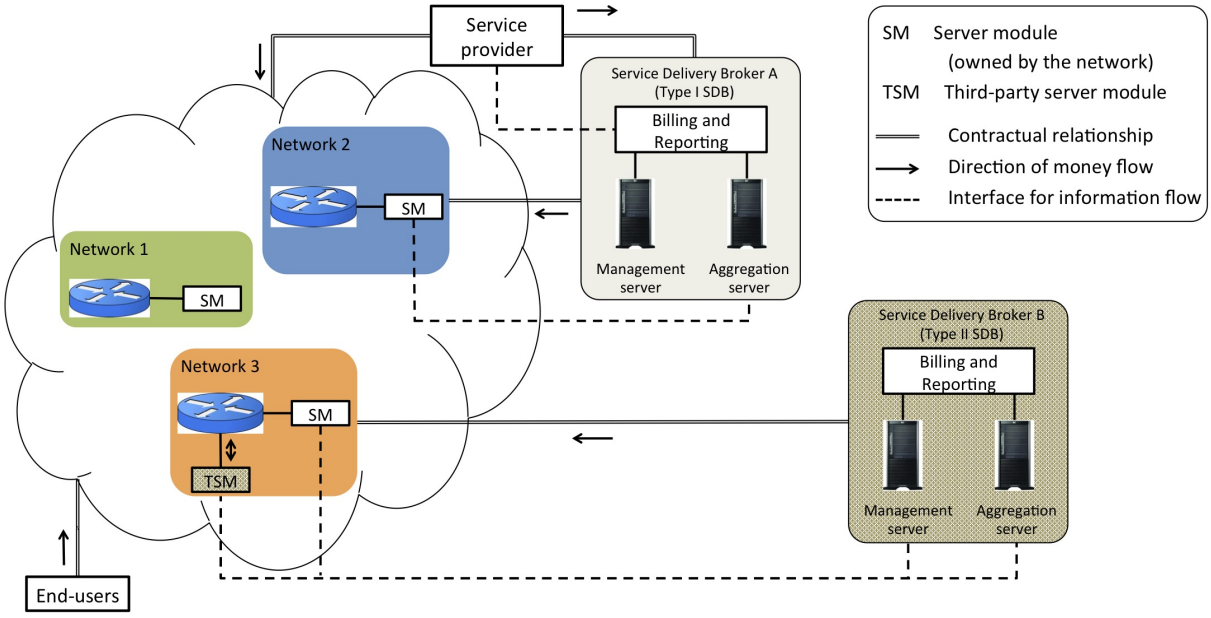


Figure 5.8: A simplified illustration of the service delivery ecosystem and infrastructure, derived directly from our proposed CCN content delivery model. Services are rendered by one or more server modules which interface with the router. There are two categories of server modules - those owned directly by the network and those owned by other entities, which we refer to as third-party server modules. Service delivery brokers (SDBs) serve as transaction brokers between service providers and server owners, in order to minimize transaction costs for service delivery. They provide the infrastructure to facilitate information and money flow between service providers and network operators. They also perform a second function, namely service resolution. Some SDBs, such as SDB B, may also operate their own server modules. Networks contract with one or more SDBs to deliver services on behalf of the SDB. Service providers also contract with one or more SDBs to purchase services. Money flows from service providers to SDBs and from SDBs to networks. In addition, money may flow between third-party server module owners and the network in either direction, depending on the negotiating power of the parties involved.

5.5.2 Service delivery model

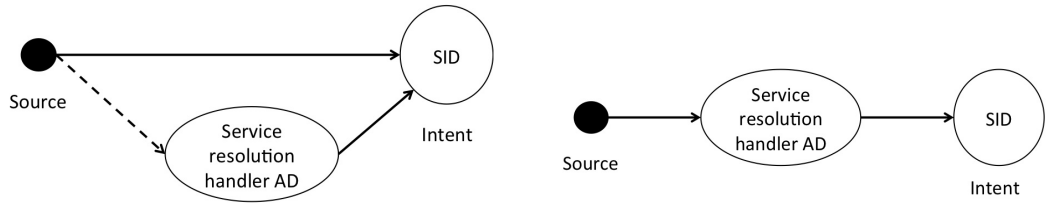
The service delivery model consists of five stages, namely service preparation, service discovery, service request, service delivery and meta-information gathering. We illustrate how the concepts we presented for the content delivery model can be extended to the service delivery model in the eXpressive Interent Architecture (XIA) in the next sections. We focus our attention on service preparation, service request and service delivery.

Service preparation

In the service preparation phase, a service provider instantiates a service on its servers. The service provider can also contract with a service delivery broker to replicate the service across the network. In XIA, a service is named after the hash of a public key. After instantiating a service, the service provider specifies a URI for the service. Similarly to the CCN content delivery, the URI takes the form of *xid:DAG_address_ for_service;service_provider_URL*. As discussed in Chapter 4, this form of URI supports both early binding and late binding and, thus, provides flexibility for different application needs.

Two forms of DAG addressing are used to support service delivery. The service provider can employ a fallback to a service resolution handler, as shown in Figure 5.9(a). In this case, the service resolution handler's role is to refine the DAG address by providing a specific path that leads to the service provider. In XIA terminology, the service resolution handler performs a mapping from AD to AD:sub-AD, where sub-AD represents a more refined AD(s) that lead to the service provider. For instance, the AD could represent one large network provider, whereas sub-ADs represents smaller networks that form part of the large network or are under the large network's control.

Alternatively, the service provider can employ scoping through the service resolution handler, as shown in Figure 5.9(b). Both forms of DAG addressing can be used when the service provider contracts with a service delivery broker. In this case, the SDB's domain (or the domain of a resolution handler that knows how to reach the SDB) serves as the service resolution handler domain in the fallback and the scoped AD.



(a) Supporting service delivery with a fallback

(b) Supporting service delivery with scoping

Figure 5.9: Two forms of DAG addressing can be used to support service delivery. The service provider can employ a fallback to a service resolution handler, as shown in Figure 5.9(a) or scoping through the service resolution handler, as shown in Figure 5.9(b).

In the service delivery model, the DAG address provides the only source of information about how the service provider delivers a service. This contrasts with content delivery, where both the DAG address and the chunk meta-information provide information about how the publisher delivers the content. One could think of the DAG address as the first source of information about how a particular principal (e.g., content or service) is delivered. Depending on the nature of the principal, other sources of information may complement the information contained in the DAG address.

Service request

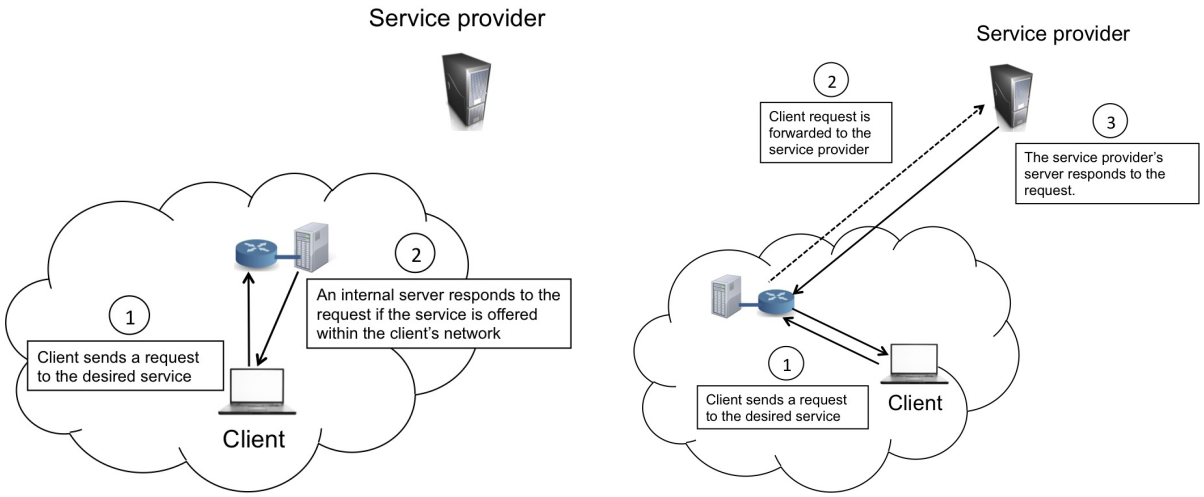
XIA provides for the possiblity for a client to issue a request for a service to a service identifier (more specifically a service URI), instead of a host identifier [74]. The header of this request packet has an identical structure to the header of the chunk request packet shown in Figure 4.10. The only difference is the XIDType. However, unlike a content request where the content identifier tells the cache all it needs to know, service requests need lots of application-specific parameters and data which are carried in the payload of the service request packet.

In XIA, the source of the request can be a host or a service. This capability facilitates interactive services. As an example, a client may send a service request with a service identifier as the source. One could think of the service identifier in the destination address and the service identifier in the source address as representing distinct ports on the destination server and the client respectively, thus facilitating the kind of interactive services already supported through the use of port numbers in a host-centric Internet architecture.

Service delivery

A request for a service can be delivered within the client's network, directly by the service provider or by a server in another network. The delivery mode depends on three factors. First, it depends on the type of service. Stateless services can be replicated, whereas stateful services cannot be replicated. Second, it depends on the popularity of the requested service. Popular services may be replicated in many networks, in order to save on network traffic and improve latency performance. This is analogous to transparent caching in content delivery. Finally, the delivery mode also depends on whether the service provider employs a SDB or delivers the service through its own servers.

We illustrate the delivery process for a service provider who does not use a service delivery broker in Figure 5.10. When the service is replicated in local networks (e.g., because it is stateless and very popular), then requests for the service are handled directly by local servers (Figure 5.10(a)). On the other hand, the request is forwarded to the service provider's domain when the service is not available in the client's local network. This situation is analogous to a cache miss in content delivery.



(a) In-network service delivery

(b) Out-of-network service delivery by a service provider

Figure 5.10: The service delivery process for a service provider who does not use a service delivery broker takes two forms. When the service is replicated in the client's network (e.g., because it is stateless and popular), then requests for the service are handled directly by a server within the client's network (Figure 5.10(a)). This is analogous to transparent caching. On the other hand, when the service is not replicated in the client's network, then the service request is handled directly by the service provider (Figure 5.10(b)). This is analogous to fetching a cache miss directly from the publisher.

Similarly, a request for a service that is offered through a service delivery broker is either delivered locally, when the service is available in the local network, or forwarded to the domain of the service resolution handler, which is assumed to be the domain of the SDB. This is illustrated in Figure 5.11. The SDB uses information about the request (e.g., domain of the client) to determine the closest server that responds to the request. Metrics similar to those used to determine the closest surrogate server in the TCP/IP content delivery model are applied in this case (see e.g., [41]).

Once a server has been identified, the SDB rewrites the destination field in the request to include a refined address, which includes the sub-AD and possibly the HID of the selected server. The selected server responds directly to the client, providing a more refined DAG

address to the client for subsequent communication.

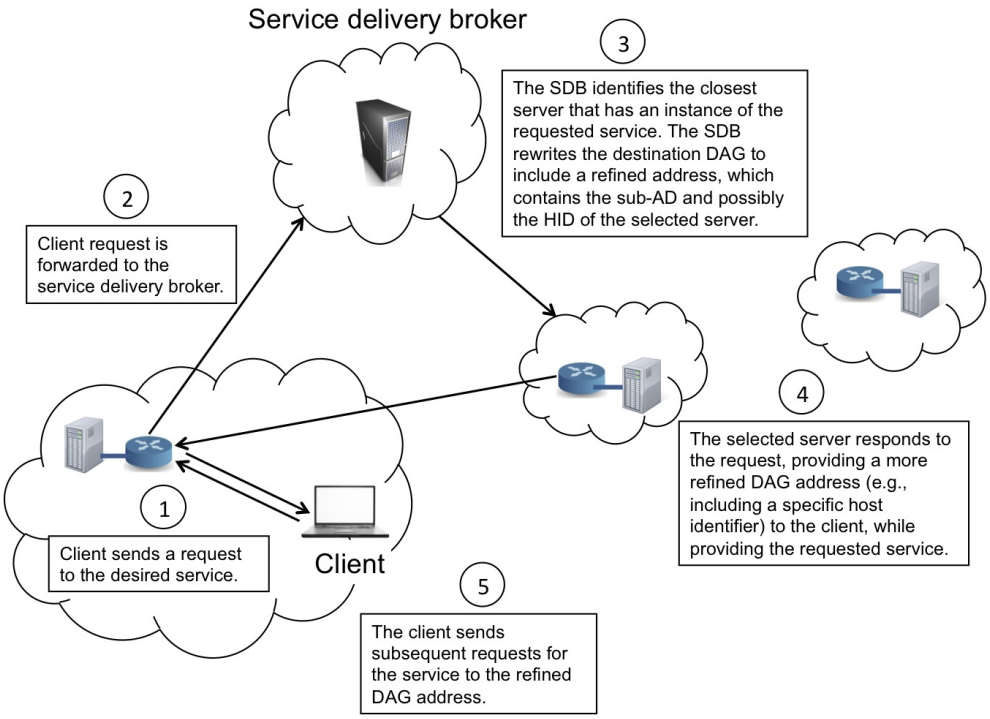


Figure 5.11: When a service provider delivers a service through a service delivery broker (SDB), then a request for the service is either delivered locally, when the service is available in the local network, or forwarded to the domain of the SDB. Using information about the source of the request, the SDB identifies the closest server that responds to the request. The closest server is identified using metrics such as the proximity to the source of the request and the load of the server. Once a server has been identified, the SDB rewrites the destination field in the request to include a refined address, which includes the sub-AD and possibly the HID of the selected server. The selected server responds directly to the client and provides a refined service DAG address for subsequent communication with the client.

Accounting and billing for service delivery

Our service delivery model relies on server modules to periodically report service deliveries to the SDB for accounting and billing purposes. Thus, our model assumes that server modules truthfully report service deliveries to SDBs to ensure that service providers are billed accurately for their usage and server modules are properly compensated for the actual services rendered.

The same concerns we expressed about over-reporting of content apply to over-reporting of services and must be addressed. In the case of service delivery, the SDBs may not always be on the path between the client and the server module (e.g., clients can cache service responses and contact servers directly). Thus, SDBs cannot accurately verify the service deliveries reported by the server modules. This may provide incentives for some server modules to over-report the services they delivered. In the same way, service providers are not always on the path of a service request, which may provide incentives for SDBs to over-report service deliveries. Future research could explore ways to address this issue.

The above discussions show that extending the proposed CCN content delivery model to other principals is straightforward. In particular, support for service delivery in a future Internet architecture can be achieved using the same infrastructure, protocols and mechanisms we have developed for content-centric networking.

5.6 Conclusion

In this chapter, we demonstrated the generality of our proposed CCN content delivery model. Through several use cases, we showed how our CCN content delivery model efficiently supports popular content-based Internet applications such as the web, email and streaming. In particular, we highlighted the potential gains in network traffic reduction from using our proposed model, instead of the current host-centric TCP/IP content delivery model.

Furthermore, we outlined how our proposed content delivery ecosystem can be extended to other principals, such as services. Even though content delivery currently dominates Internet usage, the versatility of our model provides a low-cost means to support other network principals that may dominate future Internet usage. As a result of this versatility, we believe that the proposed CCN content delivery model, including the envisaged ecosystem, provides a viable and better alternative to the current host-centric content delivery ecosystem, which lacks such versatility.

Chapter 6

Dealing with Different Threats in the CCN Content Delivery Model

Threats exist in any content delivery model. For instance, distributed denial of service attacks (DDoS) can seriously hamper the availability of content in the TCP/IP content delivery model [198]. This threat is also present in content-centric network models that employ a name resolution service. Similarly, an end-user's desire for privacy is not guaranteed in the TCP/IP content delivery model, and users must employ additional technologies such as virtual private networks (VPNs) and techniques like onion routing to achieve some degree of privacy (see e.g., [30, 31]). Some CCN-based architectures, such as NDN, overcome privacy threats, but many other CCN-based architectures still pose privacy threats for end-users.

In this chapter, we identify some common threats in both TCP/IP and CCN-based content delivery models and highlight specific threats introduced by the design choices we make in our proposed content delivery model. We focus on threats to publishers and end-users. We also describe the additional mechanisms we introduce in our content delivery model to deal with some specific threats. The rest of the chapter is organized as follows. In Sections 6.1 and 6.2, we identify the threats to publishers and end-users respectively in any content delivery model. In Section 6.3, we describe the mechanisms we introduce to deal with specific threats in our content delivery model. We conclude the chapter in Section 6.4.

6.1 Threats to publishers

The threats to publishers include distributed denial-of-service attacks (DDoS), billing fraud, deep linking and click fraud, as described below.

6.1.1 Denial-of-service attacks

The motivations for denial-of-service attacks vary, but the primary goal of such attacks is to starve some resource(s) in order to limit the availability of content [198, 199]. In the TCP/IP content delivery model, the main resources usually targeted are processing power and bandwidth at the host of the content. Hence, a publisher can mitigate this threat by employing multiple distribution points for content. This is currently accomplished through the use of content delivery networks (CDNs) for content delivery.

By design, content-centric network architectures decouple content from location. Thus, DDoS attacks that focus on starving bandwidth and processing resources at a particular host will, in general, have little effect on the availability of content in the network as a whole. Nevertheless, CCN architectures that employ a name resolution service introduce a vulnerable point in the content delivery model, which can be exploited by DDoS attacks. This is because content availability can be significantly reduced by targeting the resolution service.

On the other hand, architectures that employ name-based routing of content requests enjoy inherent protection against these forms of DDoS attacks due to the sheer number of potential hosts that possess a specific content. Similarly, DDoS attacks that target storage resources in a CCN will have little impact on content availability because of the large number of potential storage locations in the network.

6.1.2 Billing fraud

When payment flows exist between publishers and the network for content delivery, there are always incentives for billing fraud, especially in the absence of effective mechanisms for publishers to estimate the volume of content delivered on their behalf. For instance, in the TCP/IP content delivery model, CDNs may have incentives to over-report the volume of content delivered on behalf of publishers in order to obtain more revenue. This is also true in the CCN content delivery model.

We illustrated some mechanisms available to publishers in the TCP/IP delivery model to counter this threat in Table 4.2 . Together with the competitive nature of the content delivery market, these mechanisms, even though far from foolproof, provide incentives for CDNs to be honest in the TCP/IP content delivery model [189, 200]. Without effective mechanisms for publishers to estimate the volume of content delivered, billing fraud may pose a big risk to them in the CCN content delivery model.

6.1.3 Deep linking

Deep linking results directly from the very nature of the web and, as a result, is present as a threat in all content delivery models. It occurs when a publisher provides a direct link on its website to a specific content (e.g., text, image, audio, video or application) hosted on another publisher's website, bypassing the original publisher's main page or homepage in the process. By employing deep linking, a publisher obtains the benefits of the particular content (e.g., increased website visits) without incurring any costs to serve the content.

The original publisher, meanwhile, incurs two types of costs to serve the deep-linked content. First, the publisher loses advertising revenue because clients that follow deep-links bypass the main page that hosts advertisements. Secondly, the publisher incurs processing and bandwidth costs to serve requests originating from such links. These costs are borne directly as a result of the publisher serving the content from its own servers or indirectly from the publisher paying a third-party CDN to deliver the content on its behalf.

Publishers have employed both technical and legal remedies against deep-linking in TCP/IP content delivery, with varying degrees of success [163, 201]. Technical approaches usually rely on the HTTP referer field to determine whether a request is deep-linked. In rejecting the requests coming from deep-linked sources, the publisher limits the cost of the deep-linked request to just processing costs. Publishers who pay for content delivery also employ cookies, pre-authentication and URL randomization techniques to mitigate the costs associated with deep-linking [39, 134].

In a CCN architecture, distributed caching in the network has the potential to limit the costs associated with deep-linking to only lost advertisement revenue for publishers who opt for transparent caching. Publishers who pay for content delivery, on the other hand, face both types of costs associated with deep linking and require effective mechanisms to address it.

6.1.4 Click fraud

A legitimate click on an online advertisement usually generates value for the advertising agency, the distribution partner, the publisher who paid for the advertisement and the legitimate user who clicked on the advertisement. On the contrary, a fraudulent click, usually referred to as click fraud, occurs when an illegitimate user (e.g., a person not interested in the advertisement or a computer program) clicks on an advert with the intent of generating additional revenue either as an advertising agency or as a distribution partner or causing financial harm to the publisher who paid for the advertisement.

Click fraud poses a threat to online advertising in any content delivery model. Considerable effort has been devoted to address this issue in the TCP/IP web content delivery model and more effort will be required in CCN-based delivery models to address it due to the new challenges introduced by decoupling content from location [189, 200].

6.2 Threats to end-users

We group threats to end-users into two main types, namely, censorship and privacy. We discuss each in the next sections.

6.2.1 Censorship

There are legitimate reasons to censor content. For instance, it is desirable to limit access to inappropriate content for minors. At other times, censorship poses a threat to free speech and expression. This occurs, for example, when a government tries to limit access to unflattering information. The ease of content censorship in the network depends on the degree to which centralized points participate in the content delivery process.

In the TCP/IP delivery model, where content is tightly coupled with location, it is easy to restrict access to content. One simply blocks access to the authoritative host or instructs it to stop delivering the content. This is also true in CCN architectures that employ a name resolution service. In these architectures, censorship can be facilitated through the resolution handlers. Censorship can also be easily facilitated in all content delivery models

that rely on search engines for content discovery - search engines are directed not to return some relevant results. This is particularly relevant in CCN architectures that employ self-certifying identifiers because search engines play a central role in mapping from human-readable names to content identifiers. Due to the sheer size of the web, censorship through search engines is also a serious threat in the TCP/IP model. Censorship facilitated through access networks is also possible in any content delivery model, but it can be circumvented by employing proxy servers and virtual private networks.

6.2.2 Privacy

Regardless of the network architecture, the pursuit of end-user privacy often conflicts with the interests of other stakeholders. For instance, the business models of publishers are often in conflict with user privacy. Publishers desire and often use several tools (e.g., cookies, beacons, HTTP referrers) to collect various kinds of information about users in order to provide more relevant content, including targeted advertisements. For many publishers, the revenues obtained from advertisements funds a substantial part of their operations and helps to provide free services.

By contrast, users may prefer their activities not be tracked with such tools, as demonstrated by recent efforts of the World Wide Web Consortium (W3C) to standardize Do Not Track (DNT) functionality in HTTP [202]. Nevertheless, disabling these tools has several consequences. Publishers lose revenue when they lack the ability to properly target advertisements, which could either reduce the quality of experience they offer end-users or

cause them to charge end-users for their services. Furthermore, disabling tools like cookies may make it cumbersome and more expensive to realize several desirable web functionalities, and diminish the quality of experience for users. For instance, many e-commerce web sites rely on cookies to personalize their services for end-users and to offer shopping cart functionalities. As a result, there is commercial pressure to push back on efforts to safeguard end-user privacy in the TCP/IP content delivery model.

Moreover, the pursuit of privacy goals may also affect efficient network operations. Take content delivery for example. An intelligent cache could store and transparently deliver frequently requested content, thereby improving network latency and reducing network traffic. The reduction in network traffic leads to savings in backhaul costs, which could be translated to lower prices for end-users. Nevertheless, the realization of these savings requires the network to be aware of the content requested by and delivered to end-users, information which, if not used properly, could have serious privacy implications.

A user concerned about privacy may encrypt his communications to hide the contents from the network operator. However, this action hinders the ability of the network to serve content from local caches, which increases the overall network traffic. An architecture like NDN makes it easy to identify the content requested, but harder to identify the user and thus addresses such privacy fears for the user and also encourages the network operator to deploy the architecture. Nevertheless, such an architecture introduces new problems for law enforcement because it makes it harder to track users who perform illegal activities.

The preceding discussion highlights that privacy is often in conflict with other desirable economic or policy goals and will therefore continue to be a threat for end-users in any content delivery model. We summarize the threats inherent in different content delivery models in Table 6.1. In the table, both NDN and generic XIA do not face the threat of billing fraud because they assume only transparent caching in the network. In the presence of payment flows for content delivery, all delivery models face the threat of billing fraud.

Table 6.1: Examples of threats in different content delivery models.

Threat	TCP/IP delivery model	Name resolution CCN architectures	Name-based routing CCN architectures		
			NDN [3]	XIA [74]	XIA with proposed content delivery model
DDoS	✓	✓			
Billing fraud	✓	✓			✓
Deep-linking	✓	✓	✓	✓	✓
Falsified content	✓	✓	✓		
Click fraud	✓	✓	✓	✓	✓
Censorship	✓	✓	✓	✓	✓
Privacy	✓	✓		✓	✓

6.3 Dealing with specific threats in our model

In an ideal world without malicious or greedy entities, the model we described in Chapter 4 provides sufficient mechanisms to support the content delivery needs of different stakeholders. However, the preceding paragraphs illustrate that we need additional mechanisms to safeguard the interests of all stakeholders in the presence of malicious entities. In this section, we describe specific mechanisms we introduce in our model to overcome billing fraud, deep-linking, censorship and privacy threats. We also discuss specific threats that we mitigate in our model by relying on the fundamental building blocks of XIA.

6.3.1 Billing fraud

Billing fraud could occur in two ways. In the first, the cache module deliberately over-reports the volume of content delivered in order to obtain more revenue. In the second, a non-paying publisher uses the identity of a paying publisher to obtain paid content delivery for free. Signature verification in the cache-level meta-information is enough to deal with the latter. We introduce a publisher pre-authorization mechanism to enable publishers to deal with the first form of billing fraud. A CDB may charge different premiums to offer the above mechanism, since it imposes additional burdens on cache modules and the CDB.

Publisher pre-authorization

Publisher pre-authorization allows a publisher to authorize individual content requests before they may be delivered from a cache module. Unlike referrer-based blocking where

cache modules use pre-agreed criteria to make content delivery decisions on behalf of the publisher, pre-authorization shifts the content delivery decision to the publisher. Thus, the publisher can employ a richer set of criteria to make decisions in real-time for each request.

In general, the publisher and the CDB share a secret key for publisher pre-authorization, which the publisher uses to encrypt a token for clients to use when retrieving content from any network location. However, using a single key to encrypt multiple messages over long periods of time may introduce security risks [203]. To implement this service in a more secure way, the CDB provides the publisher with a one-way hash chain, with enough keys to cover the duration of the contract[1]. In particular, for a contract duration that consists of T mini-periods, the CDB provides the publisher with a one-way hash chain $r_0, r_1, r_2, \ldots, r_{T-2}, r_{T-1}$, where r_0 is a random seed and each element in the chain is generated by $r_i = hash(r_{i-1})$. Thus, if we consider a day as a mini-period, then a contract that lasts for a year will have 365 elements in the hash chain. The main security property of the one-way chain is that without knowledge of the random seed, it is impossible to predict the elements, when used in the reverse order (i.e., $r_{T-1}, r_{T-2}, \ldots, r_2, r_1, r_0$).

The publisher marks all content that requires pre-authorization with a type of service code that implies this functionality. In addition, the publisher provides mechanisms that always direct clients to the publisher's servers for each request for such content. For instance, the URI provided for such content could be a service identifier. When a client establishes a connection with the publisher, the publisher issues an authorization token, together with

[1]See e.g., [203] for an in-depth treatment of one-way hash chains.

a DAG address for the chunk; the client then retrieves the chunk from any network location. The token is computed as $\left[\text{publisher ID}, \{\text{HID of client, CID, timestamp, expiry, other data}\}_{r_i}\right]$, where $\{\cdot\}_{r_i}$ denotes encryption with key r_i and r_i is the appropriate key from the one-way hash chain. For the first mini-period, the appropriate key is r_{T-1}, whereas the appropriate key for the last mini-period is r_0.

The client sends a request for the CID and includes the authorization token as part of the request in the referrer field. The referrer type refers to a token and the referrer details contain the token (see Figure 4.11(a)). A cache module first checks the type of service specified for the chunk and expects an authorization token for the request in the referrer field. Cache modules have knowledge of the appropriate key for each mini-period for each publisher (e.g., the CDB may distribute the hash chain to cache modules). The cache module does not honor the content request when either the token is absent or cannot be decrypted.

Even when the token is decrypted successfully, the cache module checks if the data in the token matches the request before delivering the content, in order to avoid replay attacks. It should be mentioned that successful operation of publisher pre-authentication based on periodic encryption keys requires network-wide clock synchronization so that all parties agree on when a mini-period ends. This is trivial to achieve for the kind of time periods envisaged. Additionally, publisher pre-authentication could also be employed as a means to implement paid access by users.

6.3.2 Deep linking

Our content delivery model reduces the impact of deep linking on the resources of publishers who opt for transparent caching, since the network undertakes widespread caching on their behalf. Such publishers only lose advertising revenue as a result of deep-linking. We provide a referrer-based blocking mechanism for publishers who pay for content delivery to limit the financial impact of deep linking. We describe this next.

Referrer-based blocking

In referrer-based blocking, a publisher specifies a list of referrers that trigger paid content delivery, as part of the contract for that type of caching service. The list of referrers may include specific publisher IDs, service IDs or URLs. The CDB propagates this information to affiliated cache modules through the database sent from the management server to the cache modules.

Before delivering a chunk marked with a type of service that requires referrer-based blocking, the cache module checks the referrer field of the request packet. The cache module only delivers the content for payment if the referrer field matches an entry in the valid referrer list specified by the publisher. Referrer-based blocking is not foolproof. It requires cache modules to always verify the authenticity of chunks they store for paid content delivery. It also requires clients to verify the publisher information contained in master chunks. Without authentication, it is trivial for any entity to insert a publisher ID in a master chunk, which can trigger paid content delivery. It also assumes clients are not

forging referrer fields. This is a reasonable assumption when the threat we are countering is merely deep linking.

6.3.3 Censorship

Unique content identifiers make it possible to monitor access to content. It also makes it easy to censor content in the network. Nevertheless, the use of self-certifying identifiers over both the meta-information and content payload makes it easy to get around censorship in our content delivery model, since the content identifier can be easily changed by changing a single bit in the header or payload. This makes it extremely difficult for any intermediary to censor content based on the content identifier for any reasonably long period of time. Censorship based on the publisher identifier is possible in our delivery model.

However, it is trivial for publishers to overcome this threat, since publishers are required to provide their identity in a chunk header only for paid content delivery. In cases where a publisher suspects the threat of censorship based on its identity, the publisher marks all content for transparent caching and omits the publisher identifier in the chunk header.

6.3.4 Privacy

Our content delivery model makes several design choices which facilitate the realization of different stakeholder goals. Sometimes, these design choices trade-off end-user privacy in order to achieve efficient network operation and other desirable network goals such as

security. For instance, our model defines a referrer field, which performs a similar function to the HTTP referrer field, as part of a content request.

The referrer field provides an efficient means to deal with some threats, such as deep-linking, and also provides publishers and their delegates with a means to obtain desirable information which could be used for beneficial purposes for both the publisher and the end-user. Nevertheless, similarly to the TCP/IP HTTP content delivery model, it also allows the activities of users to be tracked in a way which end-users may not desire.

To address this conflict, our model allows users to trade off performance with privacy. Users concerned about privacy could omit the referrer field when sending out requests. However, this could mean increased latency during content retrieval. This is because cache modules that offer paid caching services may not deliver content without a legitimate referrer field in the request, which causes requests to go all the way to the publisher. By allowing users to make this trade-off decision, users can trade privacy for lower latency.

Furthermore, in XIA, all hosts are authenticated prior to joining the network. This makes it easier to deal with threats such as denial of service attacks, falsification of meta-information and click fraud, among others. However, this also makes it easy to associate end-users with the content they access. This could be beneficial for law enforcement purposes. At the same time, this information could be used for malicious purposes, such as spying and easy identification of dissidents.

One way to mitigate this issue is to employ approaches like the one proposed in [96], which exploits computational asymmetry to force a potential adversary to undertake significant computation in order to monitor a request. Even though such an approach does not provide ideal privacy, it can make it much more difficult for an adversary to monitor requests of a large number of users.

6.3.5 Other threats

In addition to the specific mechanisms we introduce to overcome some of the threats identified in Table 6.1, the XIA architecture provides additional mechanisms which enable us to address other threats such as click-fraud and falsification of meta-information.

Click fraud

Content delivery brokers can employ current techniques to detect click fraud in our content delivery model [189, 204]. For instance, the CDB can identify hosts that exhibit suspicious content request behavior. In XIA, a host is identified by a host identifier (HID), which is the hash of a public key. All hosts are authenticated prior to joining the network [83]. Thus, HIDs in XIA provide a foolproof means to identify all hosts that participate in a communication session. This property allows us to deal more effectively with click fraud.

Unlike IP addresses, which can be easily spoofed, HIDs in XIA are authenticated, which eliminates the possibility of identifying the wrong host. Once a host has been identified, secondary action can be taken by the host's home network. For instance, a malicious host

can be taken offline, whereas a compromised host can be cleaned and brought back online.

Falsification of meta-information

Intrinsic security properties in XIA provide a means to deal with falsification of meta-information. Specifically, chunk integrity checks and signature verification enable cache modules to detect modified chunks and chunks with false signatures respectively. Further, it is possible to identify the source of such chunks in XIA because hosts are authenticated before joining the network. Once a source (HID) has been identified, secondary action can be taken on the identified host by the home network to prevent future occurrences.

6.4 Conclusion

In this chapter, we identified common threats in any content delivery model and outlined the mechanisms provided by our CCN content delivery model to address these threats. We showed that our proposed CCN content delivery model offers several mechanisms to address threats such as billing fraud, deep-linking, click fraud and censorship.

We also discussed the inherent conflicts between achieving end-user privacy and meeting other desirable goals, such as efficient network operation, enabling different business models and fulfilling the needs of law enforcement, and identified the means provided by our model to allow users to make the trade-off between achieving privacy on one hand and achieving superior performance on the other hand.

Chapter 7

Implications and Future Research Directions

In this dissertation, we have explored both socio-economic issues related to content-centric networks and technical issues related to content delivery support in a specific example of a content-centric network. We employed a generic network model to investigate the incentives of different network players to deploy and use CCN-based architectures (Chapter 2) and the social welfare implications of different cache deployment scenarios (Chapter 3), and obtained some interesting insights. Furthermore, we used those insights as a basis, and the eXpressive Internet Architecture (XIA) as a reference CCN architecture, to design a suitable content delivery ecosystem, routable content structure and a content delivery model to support content delivery in a content-centric network (Chapters 4 - 6).

Even though our results were obtained in the context of a specific model and reference CCN architecture, the insights we obtained have several important implications for CCN research in general. In this chapter, we highlight the implications of our results for content-centric networks, future Internet research and public policy. We also provide some directions for future research.

7.1 Content-centric networking

The implications of the results presented in this dissertation for content-centric network-based architectures are briefly discussed in this section.

7.1.1 Support for payment mechanisms

Our main result from Chapter 2 showed that, without some explicit monetary compensation from publishers, networks will fail to deploy the socially optimal number of caches to support a CCN-based architecture. This result is fundamental and suggests that the transparent caching assumptions utilized in most CCN architectures will lead to sub-optimal deployment outcomes. All CCN architectures must make provisions for payment flows for content delivery and the information flows required to support such payments, in order to ensure optimal deployment of caches in the network.

7.1.2 Support for transaction brokers and other intermediaries

In a CCN ecosystem, where caches are deployed by many different networks, obtaining information about content delivery becomes a burdensome task for a publisher. Each pub-

lisher has to establish a relationship with all cache owners in order to obtain accurate information about content delivery. Furthermore, direct payment flows between publishers and cache operators for content delivery lead to significant transaction costs which subtracts from the benefits of low-latency content delivery. We showed in Chapter 3 that the overall social welfare from deploying caches diminishes rapidly when a publisher deals with multiple networks for content delivery.

To mitigate this issue and improve overall social welfare, transaction brokers acting as intermediaries between cache owners and publishers can serve to minimize transaction costs for payment and information flows. This is particularly important in CCN architectures, such as NDN, which employ name-based routing of content requests. These architectures must recognize the need for transaction brokers and make provisions to support such intermediaries. Architectures that perform name resolution of content requests can utilize the resolution service as an intermediary for information and money flow to realize similar benefits. Furthermore, it is important to provide support for multiple intermediaries in order to eliminate rent-seeking behavior.

7.1.3 Minimum chunk sizes

In a CCN ecosystem where payment flows exist, cache owners require reliable information about the publisher of a piece of content and the publisher's desired level of caching service in order to make caching and cache replacement decisions, and to get properly compensated for content delivery. In Chapter 4, for CCN architectures that employ self-certifying identifiers, we identified several important pieces of information that are required in each

piece of routable content in order to facilitate this task.

Moreover, architectures that employ hierarchical or human-readable names still require a means to prove provenance of content. These requirements imply that the minimum size of routable content is limited by the minimum amount of information required to facilitate caching decisions. Thus, the design of CCN protocols must take such content size restrictions and the associated overhead considerations into account.

7.1.4 Content router design

Recent efforts to design and analyze the performance of content routers mostly focus on the forwarding and storage requirements. Very little attention has been paid to the need to support accountability and payment flows for content delivery in a content-centric network. Without such support, however, it is unlikely that network operators will deploy sufficient infrastructure to support CCN. We proposed one means to realize accountability in CCN in Chapter 4. Other architectures may make different choices. Nevertheless, designers must make provisions for and incorporate the additional information requirements in their content router designs, in order to develop content routers that are capable of meeting the real-world content delivery needs of different network players.

7.2 Future Internet architecture design

The majority of CCN architecture proposals in the literature are specifically designed to address the technical problem of efficient content dissemination. Very little effort is spent

to incorporate socio-economic considerations in the early stages of the design of these architectures. As we have pointed out in Chapters 2 and 3, the resulting architectures lack the necessary mechanisms to address the concerns of all stakeholders and fail to realize optimal deployment outcomes from an overall social welfare point of view.

In this dissertation, we have illustrated another paradigm of network architecture design, which is motivated to an equal degree by economic and stakeholder considerations on one hand and by technical aspects on the other hand. Specifically, this dissertation shows that investing some effort to obtain an in-depth understanding of the economic incentives of various stakeholders can inform content-centric network architecture design in ways that are not obvious by just taking into account technical considerations. The design of future Internet architectures in general could significantly benefit from the approach introduced in this dissertation.

Recent history suggests that the dominant usage of the Internet can evolve over time in ways that we cannot currently envisage. Future Internet architectures must be flexible enough to easily support such evolution. Using an in-depth understanding of the economic incentives of different players, we were able to design a delivery model that is flexible enough to seamlessly support different dominant uses, be it content or service delivery, as demonstrated in Chapter 5. This illustrates that an in-depth understanding of economic incentives can lead to more flexible architectures that can address fundamental stakeholder needs, even in the midst of change in network usage.

7.3 Public policy

Our results have several implications for public policy. First, we showed in Chapter 2 that interconnection and co-location create one avenue where eyeball networks could obtain an advantage in the provisioning of caching services in a content-centric network. Exploiting this advantage, however, leads to a deadweight loss for the system as a whole. Transit pricing serves as a check on the extent to which an eyeball network can exploit its terminating access monopoly in such a manner and, thus, could limit the extent of the deadweight loss.

Still, regulators must monitor this space and be ready to address interconnection and co-location issues in a content-centric network if the market fails to yield desirable outcomes. Posting uniform interconnection and co-location rates could be one way to address this issue in case of a market failure. For this task, regulators may use one result from Chapter 3, which showed, under certain assumptions about caching costs, that in the presence of third-party cache modules it is welfare-maximizing for eyeball networks to pay cache-module owners to co-locate and deliver traffic.

Secondly, caching infrastructure provides one avenue for networks to circumvent transport-focused network neutrality regulations. In particular, it is reasonable to expect networks to use cache modules as strategic assets to open up new revenue streams from differential quality of service business models. For instance, networks can unfairly favor a publisher's content through the algorithms employed at the cache module. Hence, policy measures taken to address network neutrality in a content-centric network must also take into account caching infrastructure.

Finally, we showed in Chapter 6 that architectures that aim to facilitate privacy make it difficult for law enforcement to identify entities that engage in illegal activities. Such architectures also increase network costs and decrease efficiency, which dampens incentives for deployment. In effect, there is an inherent trade-off between privacy goals and other desirable socio-economic goals. Regulators must therefore avoid imposing rigorous rules that fail to strike an appropriate balance between desirable goals of different network stakeholders. We showed in Chapter 6 that providing technical tools to enable stakeholders to make their own trade-off decisions strikes a good balance in realizing the goals of different network stakeholders.

7.4 Future research

The research presented in this thesis raises several questions which future research could address. First, the model we used to analyze the incentives of different network players to deploy cache modules assumed that only one cache module participates in the delivery of a specific content. In practice, it is possible for multiple cache modules to participate in the delivery of any content. For instance, even with the presence of a transit network that provides caching, eyeball networks may still find it desirable to deploy transparent caching to save the metropolitan bandwidth required to transport files across distant locations within their networks. Thus, future research could investigate the incentives of different network players to deploy caches and the competitive dynamics at play under these circumstances.

Secondly, we did not explicitly model end-user behavior in response to poor quality of content delivery experience. However, the short, medium and long term decisions users take in reaction to poor quality of experience may have different impacts on the incentives of different types of network players to deploy caching infrastructure. It will be extremely useful to understand how the results presented in this thesis change when such dynamic end-user behavior is explicitly taken into account.

Thirdly, we assumed in our analysis that cache modules will be designed to achieve reasonably high cache hit ratios. However, this is not a trivial task. Publishers encode the same content for different network conditions, device types and geographic regions. Thus, a single piece of content could have on the order of 120 or more versions to cater to different scenarios [205]. When each piece of content is broken into chunks, the number of chunks could multiply very quickly. Obtaining high cache hit ratios is very challenging under these circumstances. Future research could investigate different schemes to realize high cache hit ratios under such circumstances.

Fourthly, we have illustrated how the ideas we presented for content delivery could be seamlessly extended to other principals such as services. However, we have not outlined the detailed mechanisms needed to support the information flows required for such a service delivery ecosystem. Future work could shed more light on the details of the mechanisms necessary to support other principals in a future Internet. In addition, a more in-depth and systematic analysis of threats introduced in ecosystems centered on different principals (besides hosts) and effective means to address the threats would be very beneficial.

Moreover, it will be desirable to implement our proposed model in a realistic testbed environment in order to characterize its performance. Even though such a testbed implementation may not reflect the eventual deployment environment (in ten to twenty years), the insights obtained will be valuable in improving the design to address realistic deployment challenges. It will also provide useful insights for the design and characterization of content routers.

Finally, future research must investigate architectural options that provide a good balance between the needs of various stakeholders. One important issue in this direction is to investigate architectures that provide a reasonable balance between network performance, commercial needs of publishers, end-user privacy and the needs for law enforcement.

References

[1] David D. Clark. The Design Philosophy of the DARPA Internet Protocols. In *Proceedings of ACM SIGCOMM Computer Communications Review Vol 18, No 1*, 1988. 1.1, 1, 1.1.2, 1.2

[2] David D. Clark, John Wroclawski, Karen R. Sollins, and Robert Braden. Tussle in Cyberspace: Defining Tomorrows Internet. In *Proceedings of ACM SIGCOMM*, pages 347–356, August 2002. 2.1

[3] Van Jacobson, Diana Smetters, James Thornton, Michael Plass, Nicholas Briggs, and Rebecca Braynard. Networking Named Content. *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies, Rome, Italy*, pages 1–12, December 2009. 1.1, 1.2, 1.1, 1.3, 1.3, 2.1, 2.2, 2.3.3, 3.2.2, 3.3, 4.1, 4.3, 4.4.2, 5.4, 6.1

[4] CISCO. Cisco Visual Networking Index - Forecast and Methodology 2009 - 2014. Technical report, CISCO Systems, San Jose, CA, June 2010. 1.1, 2.2, 5.4

[5] Sandvine Intelligent Broadband Networks. Fall 2011 Global Internet Phenomena Report. Technical report, Sandvine Incorporated ULC, October 2011. 1.1, 1.1.3, 1.1.3, 5.2, 5.4

[6] S. M. Bellovin. Security Problems in the TCP/IP Protocol Suite. *SIGCOMM Computer Communications Review*, 19:32–48, April 1989. 1, 4.2.1

[7] Jennifer Rexford and Constantine Dovrolis. Future Internet Architecture: Clean-Slate Versus Evolutionary Research. *Commun. ACM*, 53:36–40, September 2010. 1, 1.2

[8] Xin Zhang, Hsu-Chun Hsiao, Geoffrey Hasker, Haowen Chan, Adrian Perrig, and David Andersen. SCION: Scalability, Control, and Isolation On Next-Generation Networks. Technical Report CMU-CyLab-10-020, CyLab, Carnegie Mellon University, March 2011. 1, 1.2, 1.1

[9] David Andersen, Hari Balakrishnan, Nick Feamster, Teemu Koponen, Daekyeong Moon, and Scott Shenker. Accountable Internet Protocol (AIP). *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*, pages 339–350, August 2008. 1, 4.8.1

[10] Teemu Koponen, Scott Shenker, Hari Balakrishnan, Nick Feamster, Igor Ganichev, Ali Ghodsi, P. Brighten Godfrey, Nick McKeown, Guru Parulkar, Barath Raghavan,

Jennifer Rexford, Somaya Arianfar, and Dmitriy Kuptsov. Architecting for Innovation. *SIGCOMM Computer Commununications Review.*, 41:24–36. 1, 1.2, 1.1

[11] RFC 1112. Host Extensions for IP Multicasting, August 1989. 1.1.1, 2

[12] A. Striegel and G. Manimaran. A Survey of QoS Multicasting Issues. *Communications Magazine, IEEE*, 40(6):82 –87, June 2002. 1.1.1, 2

[13] RFC 1546. Host Anycasting Service, December 1993. 1.1.1

[14] RFC 4786. Operation of Anycasting Services, December 2006. 1.1.1

[15] Chris Metz. IP Anycast: Point-to-(Any) Point Communication. *IEEE Internet Computing*, 6(2):94–98, 2002. 1.1.1

[16] S. Weber and Liang Cheng. A Survey of Anycast in IPv6 Networks. *Communications Magazine, IEEE*, 42(1):127 – 132, January 2004. 1.1.1

[17] Jerome H. Saltzer, David P. Reed, and David Clark. End-to-end Arguments in System Design. *ACM Transactions on Computer Systems*, 2(4):277–288, November 1984. 1.1.2

[18] Michael Walfish, Jeremy Stribling, Maxwell Krohn, Hari Balakrishnan, Robert Morris, and Scott Shenker. Middleboxes no Longer Considered Harmful. In *Proceedings of the 6th Conference on Symposium on Operating Systems Design & Implementation - Volume 6*, Berkeley, CA, USA, 2004. USENIX Association. 1.1.2, 1.1

[19] Dave Clark, Bill Lehr, Steve Bauer, Peyman Faratin, Rahul Sami, and John Wroclawski. Overlay Networks and the Future of the Internet. *Communications and Strategies*, 63(3):1 – 21, 2006. 1.1.2, 1.1.3, 1.1.3, 4.4.2

[20] David D. Clark and Marjory S. Blumentahl. The End-to-End Argument and Application Design: The Role of Trust. In *TPRC '07 : Proceedings of the Telecommunications Policy Research Conference*, 2007. 1.1.2

[21] RFC 3303. Middlebox Communication Architecture and Framework, August 2002. 1.1.2

[22] Anindya Datta, Kaushik Dutta, Helen Thomas, and Debra Van der Meer. World Wide Wait: A Study of Scalability and Cache-Based Approaches to Alleviate It. *Management Science*, 49(10):1425–1444, October 2003. 1.1.2

[23] Jia Wang. A Survey of Web Caching Schemes for the Internet. *ACM SIGCOMM Computer Communication Review*, 29(5):36–46, October 1999. 1.1.2

[24] RFC 2616. Hypertext Transfer Protocol - HTTP/1.1, June 1999. 1.1.2, 1, 4.2.3, 4.2.4, 4.2.5, 4.3.2, 4.4.4, 4.5.2

[25] Michael Rabinovich and Oliver Spatscheck. *Web Caching and Replication*. Addison - Wesley, 2002. 1.1.2

[26] G. Barish and K. Obraczke. World Wide Web Caching: Trends and Techniques. *Communications Magazine, IEEE*, 38(5):178 –184, May 2000. 1.1.2

[27] Joan Engebretson. Frontier Interview: Caching Reduces Internet Backhaul Traffic by 25%. `http://www.telecompetitor.com/`, March 2012, accessed on September

6, 2012. 1.1.2

[28] Ashok Anand, Archit Gupta, Aditya Akella, Srinivasan Seshan, and Scott Shenker. Packet Caches on Routers: The Implications of Universal Redundant Traffic Elimination. *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*, pages 219–230, August 2008. 1.1.2

[29] David Andersen, Hari Balakrishnan, Frans Kaashoek, and Robert Morris. Resilient Overlay Networks. *SIGOPS Operating Systems Review*, 35:131–145, October 2001. 1.1.3

[30] M.G. Reed, P.F. Syverson, and D.M. Goldschlag. Anonymous Connections and Onion Routing. *Selected Areas in Communications, IEEE Journal on*, 16(4):482 –494, May 1998. 1.1.3, 6

[31] David Goldschlag, Michael Reed, and Paul Syverson. Onion Routing. *Commun. ACM*, 42(2):39–41, 1999. 6

[32] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: the Second-Generation Onion Router. In *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, SSYM'04, Berkeley, CA, USA, 2004. USENIX Association.

[33] Tor. `https://www.torproject.org/`. 1.1.3

[34] Jim Kurose and Keith Ross. *Computer Networking: A Top-Down Approach*. Addison-Wesley, 5th edition, 2010. 1.1.3, 5.3

[35] R. Kumar and K.W. Ross. Peer-Assisted File Distribution: The Minimum Distribution Time. In *Hot Topics in Web Systems and Technologies, 2006. HOTWEB '06. 1st IEEE Workshop on*, pages 1 –11, November 2006. 1.1.3

[36] Joe Stewart. BitTorrent and the Legitimate Use of P2P. `http://www.joestewart.org/p2p.html`, February 2004, accessed on September 6, 2012. 1.1.3

[37] Stephanos Androutsellis-Theotokis and Diomidis Spinellis. A Survey of Peer-to-Peer Content Distribution Technologies. *ACM Computer Surveys*, 36(4):335–371, 2004. 3

[38] Eng Keong Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. A survey and comparison of peer-to-peer overlay network schemes. *Communications Surveys Tutorials, IEEE*, 7(2):72 – 93, 2005. 3, 1.2

[39] Erik Nygren, Ramesh K. Sitaraman, and Jennifer Sun. The Akamai Network: a Platform for High-Performance Internet Applications. *SIGOPS Oper. Syst. Rev.*, 44(3):2–19, 2010. 1.1.3, 4.2.1, 6.1.3

[40] Gang Peng. CDN: Content Distribution Network. Technical Report TR-125, State University of New York at Stony Brook, Stony Brook, NY, February 2008. 1.1.3, 4.2.1

[41] John Dilley, Bruce Maggs, Jay Parikh, Harald Prokop, Ramesh Sitaraman, and Bill Weihl. Globally Distributed Content Delivery. *IEEE Internet Computing*, 6(5):50–58, Sep-Oct 2002. 1.1.3, 4.2.1, 4.2.3, 4.3.1, 4.4.2, 5.5.2

[42] Athena Vakali and George Pallis. Content Delivery Networks: Status and Trends.

IEEE Internet Computing, 7(6):68–74, Nov-Dec 2003. 1.1.3, 4.4.2

[43] Swaminathan Sivasubramanian, Guillaume Pierre, and Maarten van Steen. Analysis of Caching and Replication Strategies for Web Applications. *IEEE Internet Computing*, 11(1):60–66, Jan-Feb 2007. 1.1.3, 1.2, 4.4.2

[44] IDATE Consulting and Research. The Online Content Distribution Market - Opportunities for CDN. Technical report, 2010. 1.1.3

[45] Thomas Eisenmann. Akamai Technologies. Technical Report 9-804-158, Harvard Business School, April 2004. 1.1.3

[46] Dan Rayburn. Updated List Of Vendors In The Content Delivery Ecosystem. `http:cdnlist.com`, August 2012, accessed on September 6, 2012. 1.1.3

[47] 2011 Akamai Annual Report. `http://www.akamai.com/dl/investors/akamai_annual_report_11.pdf`, accessed on September 6, 2012. 1.1.3

[48] Craig Labovitz. CDN and Over-the-Top Traffic Data. Content Delivery Summit, May 2012. 1.1.3

[49] Craig Labovitz, Scott Iekel-Johnson, Danny McPherson, Jon Oberheide, and Farnam Jahanian. Internet Inter-Domain Traffic. *SIGCOMM Computer Communications Review*, 40:75–86, August 2010. 1.1.3, 2.2, 4.1

[50] NSF Future Internet Architectures Project. `http://www.nsf.gov/pubs/2010/nsf10528/nsf10528.htm`. 4

[51] Scalable and Adaptive Internet Solutions (SAIL). `http://www.sail-project.eu/`. 4

[52] Subhabrata Sen, Jennifer Rexford, and Don Towsley. Proxy Prefix Caching for Multimedia Streams. *Proceedings of IEEE INFOCOM 1999*, pages 1311 – 1319, March 1999. 1.2

[53] Terence P Kelly. *Optimization in Web Caching: Cache Management, Capacity Planning and Content Naming*. PhD thesis, The University of Michigan Computer Science Department, 2002.

[54] R. Rejaie, Haobo Yu, M. Handley, and D. Estrin. Multimedia Proxy Caching Mechanism for Quality Adaptive Streaming Applications in the Internet. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 980 –989 vol.2, 2000.

[55] Yan Chen, Lili Qui, Weiyu Chen, Luan Nguyen, and Randy Katz. Efficient and Adaptive Web Replication Using Content Clustering. *IEEE Journal on Selected Areas in Communications*, 21(6):979–994, August 2003. 2.2

[56] Norihito Fujita, Yuichi Ishikawa, Atushi Iwata, and Rauf Izmailov. Coarse-grain Replica Management Strategies for Dynamic Replication of Web Contents. *Computer Networks*, 45(1):19–34, May 2004.

[57] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications.

SIGCOMM Computer Communications Review, 31(4):149–160, August 2001.

[58] Antony Rowstron and Peter Druschel. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In Rachid Guerraoui, editor, *Middleware 2001*, volume 2218 of *Lecture Notes in Computer Science*, pages 329–350. Springer Berlin / Heidelberg, 2001.

[59] Xinyan Zhang, Jiangchuan Liu, Bo Li, and Y.-S.P. Yum. CoolStreaming/DONet: a Data-Driven Overlay Network for Peer-to-Peer Live Media Streaming. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 3, pages 2102 – 2111 vol. 3, March 2005. 1.2

[60] RFC 4301. Security Architecture for the Internet Protocol, December 2005. 1.2

[61] RFC 4309. Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP), December 2005. 1.2

[62] RFC 5246. The Transport Layer Security (TLS) Protocol Version 1.2, August 2008. 1.2, 4.2, 4.2.5

[63] S. Kent, C. Lynn, and K. Seo. Secure Border Gateway Protocol (S-BGP). *Selected Areas in Communications, IEEE Journal on*, 18(4):582 –592, April 2000. 1.2

[64] RFC 4033. DNS Security Introduction and Requirements, March 2005. 1.2

[65] RFC 4034. Resource Records for the DNS Security Extensions, March 2005.

[66] RFC 4033. Protocol Modifications for the DNS Security Extensions, March 2005. 1.2

[67] David R. Cheriton and Mark Gritter. TRIAD: A New Next-Generation Internet Architecture. Technical report, Stanford University, January 2000. 1.2, 1.1, 1.3, 1.3, 2.1, 4.3

[68] Mark Gritter and David R. Cheriton. An Architecture for Content Routing Support in the Internet. In *Proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems - Volume 3*, USITS'01, Berkeley, CA, USA, 2001. USENIX Association. 1.1, 1.3

[69] Hari Balakrishnan, Karthik Lakshminarayanan, Sylvia Ratnasamy, Scott Shenker, Ion Stoica, and Michael Walfish. A Layered Naming Architecture for the Internet. *SIGCOMM Computer Communications Review*, 34:343–352, August 2004. 1.1, 1.3, 1.3, 2.1, 2.6.2, 4.3

[70] Teemu Koponen, Mohit Chawla, Byung-Gon Chun, Andrey Ermolinskiy, Kye Hyun Kim, Scott Shenker, and Ion Stoica. A Data-Oriented (and Beyond) Network Architecture. *SIGCOMM Computer Communications Review*, 37:181–192, August 2007. 1.1, 1.3, 2.1, 2.6.2, 4.3, 4.4.3

[71] Christian Dannewitz, Kostas Pentikousis, René Rembarz, Éric Renault, Ove Strandberg, and Javier Ubillos. Scenarios and Research Issues for a Network of Information. In *Proceedings of the 4th International Mobile Multimedia Communications Conference*. ACM, July 2008. 1.1

[72] Bengt Ahlgren, Matteo D'Ambrosio, Marco Marchisio, Ian Marsh, Christian Dannewitz, Börje Ohlman, Kostas Pentikousis, Ove Strandberg, René Rembarz, and Vinicio Vercellone. Design Considerations for a Network of Information. In *Proceedings of the ACM CoNEXT Conference*, pages 66:1–66:6. ACM, December 2008. 1.1, 1.3, 1.3, 2.1, 2.6.2, 4.3, 4.4.3

[73] Dirk Trossen, Mikko Särelä, and Karen Sollins. Arguments for an Information-Centric Internetworking Architecture. *ACM SIGCOMM Computer Communication Review*, 40(2):27 – 33, April 2010. 2.1, 2.6.2

[74] Ashok Anand, Fahad Dogar, Dongsu Han, Boyan Li, Hyeontaek Lim, Michel Machado, Wenfei Wu, Aditya Akella, David G. Andersen, John W. Byers, Srinivasan Seshan, and Peter Steenkiste. XIA: An Architecture for an Evolvable and Trustworthy Internet. In *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, HotNets-X, New York, NY, USA, 2011. ACM. 1.2, 1.3, 4.1, 4.3, 4.4, 4.4.1, 4, 4.5.2, 4.10, 5.5, 5.5.2, 6.1

[75] Jaeyoung Choi, Jinyoung Han, Eunsang Cho, Ted Kwon, and Yanghee Choi. A Survey on Content-Oriented Networking for Efficient Content Delivery. *IEEE Communications Magazine*, 49(3):121–127, March 2011. 2.1, 2.2

[76] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman. A Survey of Information-Centric Networking. *Communications Magazine, IEEE*, 50(7):26 –36, July 2012. 1.2, 1.3, 1.3, 2.1, 4.4.1, 4.4.3

[77] Michael Freedman, Matvey Arye, Prem Gopalan, Steven Y. Ko, Erik Nordström, Jennifer Rexford, and David Shue. Service-Centric Networking with SCAFFOLD. Technical Report TR-885-10, Princeton University Department of Computer Science, September 2010. 1.2, 1.1, 5.5

[78] X. Yang, D. Clark, and A.W. Berger. NIRA: A New Inter-Domain Routing Architecture. *Networking, IEEE/ACM Transactions on*, 15(4):775 –788, August 2007. 1.2, 1.1

[79] Diana Smetters and Van Jacobson. Securing Network Content. Technical report, Palo Alto Research Center, Palo Alto, CA, October 2009. 1.2, 1.3, 1.3, 1.2, 2.1, 4.1

[80] Ali Ghodsi, Teemu Koponen, Jarno Rajahalme, Pasi Sarolahti, and Scott Shenker. Naming in Content-Oriented Architectures. In *Proceedings of the ACM SIGCOMM Workshop on Information-Centric Networking*, ICN '11, pages 1–6, New York, NY, USA, 2011. ACM. 1.2, 1.3, 4.4.1, 4.4.3, 4.8.1

[81] Ivan Seskar, Kiran Nagaraja, Sam Nelson, and Dipankar Raychaudhuri. Mobilityfirst future internet architecture project. In *Proceedings of the 7th Asian Internet Engineering Conference*, AINTEC '11, pages 1–3. ACM, 2011. 1.2

[82] Jad Naous, Arun Seehra, Michael Walfish, David Mazieres, Antonio Nicolosi, and Scott Shenker. The Design and Implementation of a Policy Framework for the Future Internet. Technical Report TR-09-28, The University of Texas at Austin, September 2009. 1.1

[83] Ashok Anand, Fahad Dogar, Dongsu Han, Boyan Li, Hyeontaek Lim, Michel Machado, Wenfei Wu, Aditya Akella, David Andersen, John Byers, Srinivasan Seshan, and Peter Steenkiste. XIA: An Architecture for an Evolvable and Trustworthy Internet. Technical Report CMU-CS-11-100, Carnegie Mellon University, January 2011. 1.1, 1.3, 2.1, 2.6.2, 4.4, 4, 4.10, 5.2.1, 6.3.5

[84] Carl Ellison and Bruce Schneier. Ten Risks of PKI: What You're not Being Told about Public Key Infrastructure. *Computer Security Journal*, 16(1):1 – 8, 2000. 1.3

[85] Don Davis. Compliance Defects in Public-Key Cryptography. In *In Proceedings of the 6th USENIX Security Symposium*, pages 171–178, 1996. 1.3

[86] Mikko Särelä, Teemu Rinta-aho, and Sasu Tarkoma. RTFM: Publish/Subscribe Internetworking Architecture. In Paul Cunnigham and Miriam Cunnigham, editors, *Proceedings of ICT-MobileSummit 2008 Conference*. IIMC, 2008. 1.3, 2.3.3, 4.3

[87] Wei Koong Chai, Ning Wang, I. Psaras, G. Pavlou, Chaojiong Wang, G.G. de Blas, F.J. Ramon-Salguero, Lei Liang, S. Spirou, A. Beben, and E. Hadjioannou. CURLING: Content-Ubiquitous Resolution and Delivery Infrastructure for Next-Generation Services. *Communications Magazine, IEEE*, 49(3):112 –120, march 2011. 1.3, 4.3

[88] Matteo D'Ambrosio, Christian Dannewitz, Holger Karl, and Vinicio Vercellone. MDHT: A Hierarchical Name Resolution Service for Information-Centric Networks. In *Proceedings of the ACM SIGCOMM Workshop on Information-Centric Networking*, ICN '11, pages 7–12, New York, NY, USA, 2011. ACM. 1.3

[89] Matthew Caesar, Tyson Condie, Jayanthkumar Kannan, Karthik Lakshminarayanan, and Ion Stoica. ROFL: Routing on Flat Labels. *SIGCOMM Computer Communications Review*, 36(4):363–374, August 2006. 1.3

[90] Ioannis Psaras, Richard G. Clegg, Raul Landa, Wei Koong Chai, and George Pavlou. Modelling and Evaluation of CCN-Caching Trees. In *Proceedings of the 10th international IFIP TC 6 conference on Networking - Volume Part I*, NETWORKING'11, pages 78–91, Berlin, Heidelberg, 2011. Springer-Verlag. 1.3, 2.2, 4.1, 4.4.4

[91] Giovanna Carofiglio, Massimo Gallo, Luca Muscariello, and Diego Perino. Modeling Data Transfer in Content-Centric Networking. In *Proceedings of the 23rd International Teletraffic Congress*, ITC '11, pages 111–118. ITCP, 2011. 4.8.1

[92] C. Fricker, P. Robert, J. Roberts, and N. Sbihi. Impact of Traffic Mix on Caching Performance in a Content-Centric Network. In *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*, pages 310 –315, March 2012. 1.3, 2.2, 4.1, 4.4.4

[93] Van Jacobson, Diana K. Smetters, Nicholas H. Briggs, Michael F. Plass, Paul Stewart, James D. Thornton, and Rebecca L. Braynard. VoCCN: Voice-Over Content-Centric Networks. In *Proceedings of the 2009 Workshop on Re-architecting the Internet*, ReArch '09, pages 1–6, New York, NY, USA, 2009. ACM. 1.3

[94] Zhenkai Zhu, Sen Wang, Xu Yang, Van Jacobson, and Lixia Zhang. ACT: Audio

Conference Tool over Named Data Networking. In *Proceedings of the ACM SIGCOMM Workshop on Information-Centric Networking*, ICN '11, pages 68–73, New York, NY, USA, 2011. ACM. 1.3

[95] Diego Perino and Matteo Varvello. A Reality Check for Content Centric Networking. In *Proceedings of the ACM SIGCOMM Workshop on Information-Centric Networking*, ICN '11, pages 44–49, New York, NY, USA, 2011. ACM. 1.3, 4.4.4, 4.8.2

[96] Somaya Arianfar, Teemu Koponen, Barath Raghavan, and Scott Shenker. On Preserving Privacy in Content-Oriented Networks. In *Proceedings of the ACM SIGCOMM Workshop on Information-Centric Networking*, ICN '11, pages 19–24, New York, NY, USA, 2011. ACM. 1.3, 6.3.4

[97] Steven DiBenedetto, Christos Papadopoulos, and Daniel Massey. Routing Policies in Named Data Networking. In *Proceedings of the ACM SIGCOMM Workshop on Information-Centric Networking*, ICN '11, pages 38–43, New York, NY, USA, 2011. ACM. 1.3

[98] Jarno Rajahalme, Mikko Särelä, Pekka Nikander, and Sasu Tarkoma. Incentive-Compatible Caching and Peering in Data-Oriented Networks. In *Proceedings of the 2008 ACM CoNEXT Conference*, pages 62:1–62:6. ACM, December 2008. 2.1

[99] Byung-Gon Chun, Kamalika Chaudhuri, Hoeteck Wee, Marco Barreno, Christos H. Papadimitriou, and John Kubiatowicz. Selfish Caching in Distributed Systems: A Game-Theoretic Analysis. In *Proceedings of the 23rd Annual ACM Symposium on Principles of Distributed Computing*, PODC '04, pages 21–30, New York, NY, USA, 2004. ACM. 2.1, 4.4.1

[100] Ali Ghodsi, Scott Shenker, Teemu Koponen, Ankit Singla, Barath Raghavan, and James Wilcox. Information-Centric Networking: Seeing the Forest for the Trees. In *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, HotNets-X, pages 1:1–1:6, New York, NY, USA, 2011. ACM. 2.2, 4.1, 4.4.1, 4.4.3, 4.8.1

[101] William B. Norton. The Evolution of the U.S. Internet Peering Ecosystem. White paper, DrPeering.net, 2003. 2.2

[102] T.G. Griffin and G. Wilfong. Analysis of the MED Oscillation Problem in BGP. In *Proceedings of the 10th IEEE International Conference on Network Protocols.*, pages 90 – 99, November 2002. 2.2

[103] Renata Teixeira, Aman Shaikh, Tim Griffin, and Jennifer Rexford. Dynamics of Hot-Potato Routing in IP Networks. *SIGMETRICS Performance Evaluation Review*, 32:307–319, June 2004.

[104] Ramesh Johari and John N. Tsitsiklis. Routing and Peering in a Competitive Internet. In *Proceedings of the 43rd IEEE Conference on Decision and Control*, volume 2, pages 1556 – 1561 Vol.2, December 2004. 2.2

[105] M. Caesar and J. Rexford. BGP Routing Policies in ISP Networks. *Network, IEEE*, 19(6):5 – 11, Nov.-Dec. 2005. 2.2

[106] Comcast Corporation. Letter to FCC Re: Preserving the Open Internet, GN

Docket 09-191. http://www.comcast.com/MediaLibrary/1/1/About/PressRoom/Documents/Comcastexparte1130.pdf, November 2010. 2.2, 2.3.2, 2.3.3, 2.4.1, 4.4.2

[107] Dan Rayburn. CDN Market Improving, But Latest Pricing Data Shows Challenges Still Lie Ahead. http://slidesha.re/PCleu, October 2009. 2.2

[108] Personal communications with Dan Rayburn of Streamingmedia.com, July 2011. 2.2

[109] Pablo Rodriguez, Keith W. Ross, and Ernst W. Biersack. Distributing Frequently-Changing Documents in the Web: Multicasting or Hierarchical Caching? In *Computer Networks and ISDN Systems. Selected Papers of the 3rd International Caching Workshop*, pages 2223–2245, 1998. 2.2

[110] Alec Wolman, Geoff Voelker, Nitin Sharma, Neal Cardwell, Molly Brown, Tashana Landray, Denise Pinnel, Anna Karlin, and Henry Levy. Organization-Based Analysis of Web-Object Sharing and Caching. In *Proceedings of the 2nd USENIX Symposium on Internet Technologies and Systems*, volume 2, Berkeley, CA, USA, 1999. USENIX Association. 2.2, 3.3

[111] Meeyoung Cha, Haewoon Kwak, Pablo Rodriguez, Yong-Yeol Ahn, and Sue Moon. I Tube, You tube, Everybody Tubes: Analyzing the World's Largest User Generated Content Video System. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, IMC '07, pages 1–14, New York, NY, USA, 2007. ACM. 2.2

[112] Asit Dan and Don Towsley. An Approximate Analysis of the LRU and FIFO Buffer Replacement Schemes. *SIGMETRICS Performance Evaluation Review*, 18:143–152, April 1990. 2.2

[113] Terence Kelly and Douglas Reeves. Optimal Web Cache Sizing: Scalable Methods for Exact Solutions. *Computer Communications*, 24(2):163–173, February 2001.

[114] Hao Che, Ye Tung, and Zhijun Wang. Hierarchical Web Caching Systems: Modeling, Design and Experimental Results. *IEEE Journal on Selected Areas in Communications*, 20(7):1305 – 1314, September 2002. 2.2

[115] Anja Feldmann, Ramn Cceres, Fred Douglis, Gideon Glass, and Michael Rabinovich. Performance of Web Proxy Caching in Heterogeneous Bandwidth Environments. In *Proceedings of IEEE INFOCOM*, pages 107–116, 1999. 2.2, 3.3

[116] Zhaoming Zhu, Yonggen Mao, and Weisong Shi. Workload Characterization of Uncacheable HTTP Content. In Nora Koch, Piero Fraternali, and Martin Wirsing, editors, *Web Engineering*, volume 3140/2004 of *Lecture Notes in Computer Science*, pages 391 – 395. Springer Berlin / Heidelberg, 2004. 2.2, 3.3

[117] Sandvine Intelligent Broadband Networks. Fall 2010 Global Internet Phenomena Report. Technical report, Sandvine Incorporated ULC, October 2010. 2.2

[118] Mark Tsimelzon, Bill Weihl, Joseph Chung, Dan Frantz, John Basso, Chris Newton, Mark Hale, Larry Jacobs, and Conleth O'Connell. ESI Language Specification 1.0. Technical Report 04, World Wide Web Consortium (W3C), August 2001. 2.2

[119] Alec Wolman, Geoffrey M. Voelker, Nitin Sharma, Neal Cardwell, Anna Karlin, and Henry M. Levy. On the Scale and Performance of Cooperative Web Proxy Caching.

Operating Systems Review, 34(5):16 – 31, December 1999. 2.3.2

[120] Patrick Agyapong and Marvin Sirbu. Economic Incentives in Content-Centric Networking: Implications for Protocol Design and Public Policy. In *TPRC '11: Proceedings of Telecommunication Policy Research Conference*, 2011. 2.3.4, 2.3.4, 4.2.1, 4.3.5, 4.4.1

[121] John McCallum. Disk Drive Storage Price Decreasing with Time (1955-2012). `http://www.jcmit.com/disk2012.htm`, accessed on August 27, 2012. 2.3.4

[122] William B. Norton. Internet Transit Prices - Historical and Projected. White paper, DrPeering.net, 2010. 2.3.4, 2.4.1

[123] Dan Rayburn. MLB.com Now Using Level 3's CDN For Video Delivery, Akamai No Longer Sole CDN. blog.streamingmedia.com, May 4 2011. 2.4.1

[124] Dan Rayburn. Wall Street Questioning Akamai's CDN Business After Company's Q1 Earnings Call. blog.streamingmedia.com, May 3 2011.

[125] Ray Willington. Netflix Drops Akamai, Selects Level 3 For Streaming Provider. hothardware.com, November 12 2010. 2.4.1

[126] Dan Rayburn. An Overview Of Transparent Caching and Its Role In The CDN Market. `http://bit.ly/drX0ti`, October 2010. 4, 3.2.2, 1

[127] David Clark, William Lehr, and Steven Bauer. Interconnection in the Internet: The Policy Challenge. In *TPRC '11: Proceedings of Telecommunication Policy Research Conference*, 2011. 2.4.1

[128] Dan Rayburn. Telcos And Carriers Forming New Federated CDN Group Called OCX (Operator Carrier Exchange). blog.streamingmedia.com, June 27 2011. 2.5, 3.2.3

[129] Edgecast Licensed CDN Offering. `http://www.edgecast.com/solutions/licensed-cdn/`, accessed on April 6th, 2013. 2.5

[130] Dan Rayburn. Akamai Developing a Licensed CDN Offering for Telcos and Carriers. `http://bit.ly/lVNFNM`, June 2011, accessed on July 25th, 2011.

[131] Dan Rayburn. Limelight Launches Managed CDN Offering For Carriers, Is A Deal With F5 Next? `http://bit.ly/At3g6p`, February 2012, accessed on April 6th, 2013.

[132] Dan Rayburn. AT&T Building Out Their Content Delivery Network Using EdgeCast's Software. `http://bit.ly/fCG07N`, February 2011, accessed on April 19, 2011. 2.5

[133] Federal Communications Commission. In the Matter of Preserving the Open Internet; Broadband Industry Practices. Report and Order FCC 10-201, FCC, December 2010. 2.5

[134] Akamai Product Offerings. `http://www.akamai.com/html/solutions/index.html`, accessed on July 26th, 2011. 2.5, 6.1.3

[135] Limelight Networks Product Offerings. `http://www.limelightnetworks.com/CDN-services/`, accessed on July 26th, 2011.

[136] 2010 Akamai Annual Report. http://www.akamai.com/dl/investors/akamai_annual_report_10.pdf, accessed on July 26th, 2011. 2.5

[137] Dan Rayburn. A Detailed Look At Akamai's Application Delivery Product - Part 1. http://bit.ly/15GApD, April 2008, accessed on July 26th, 2011. 2.5

[138] Markus Hofmann, Eugene T. S. Ng, Katherine Guo, Sanjoy Paul, and Hui Zhang. Caching Techniques for Multimedia over the Internet. Technical report, Bell Labs Technical Memorandum, April 1999. 2.6.2

[139] Martin Arlitt, Rich Friedrich, and Tai Jin. *Lecture Notes in Computer Science: Computer Performance Evaluation*, volume 1469/1998, chapter Performance Evaluation of Web Proxy Cache Replacement Policies, pages 193–206. Springer-Verlag, Berlin/Heidelberg, January 1998. 2.6.2, 4.4.3, 4.5.2

[140] I. Chang, R. Engel, D. Kandlur, D. Pendarakis, and D. Saha. Key Management for Secure lnternet Multicast using Boolean Function Minimization Techniques. In *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 689 –698 vol.2, March 1999. 2.6.2

[141] S. Setia, S. Koussih, S. Jajodia, and E. Harder. Kronos: a Scalable Group Rekeying Approach for Secure Multicast . In *Security and Privacy, 2000. S P 2000. Proceedings. 2000 IEEE Symposium on*, pages 215 –228, 2000. 2.6.2

[142] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker. Web Caching and Zipf-like Distributions: Evidence and Implications. In *Proceedings of INFOCOM 1999*, pages 126–134, 1999. 3.2.1, 4.5.2

[143] Anirban Mahanti, Carey Williamson, and Derek Eager. Traffic Analysis of a Web Proxy Caching Hierarchy. *IEEE Network*, 14(3):16 – 23, May/June 2000.

[144] Ludmila Cherkasova and Minaxi Gupta. Analysis of Enterprise Media Server Workloads: Access Patterns, Locality, Content Evolution, and Rates of Change. *IEEE/ACM Transactions on Networking*, 12:781–794, October 2004.

[145] Hongliang Yu, Dongdong Zheng, Ben Zhao, and Weimin Zheng. Understanding User Behavior in Large-scale Video-on-Demand Systems. In *Proceedings of ACM EuroSys*, pages 333–344, 2006.

[146] Cheng Huang, Jin Li, and Keith W. Ross. Can Internet Video-on-Demand be Profitable? In *SIGCOMM '07: Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 133–144, New York, NY, USA, August 2007. ACM.

[147] Phillipa Gill, Martin Arlitt, Zongpeng Li, and Anirban Mahnti. YouTube Traffic Characterization: A View From the Edge. In *IMC '07: Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, pages 15 – 24, October 2007. 3.2.1

[148] Dan Rayburn. A Summary Of Transparent Caching Architectures and Associated Performance. blog.streamingmedia.com, June 1 2011. 3.2.2, 1

[149] CDN Strategy Blog. Transparent Internet Caching. http://www.jet-stream.com/

blog/transparent-internet-caching/, June 03 2011. 1, 3.2.2, 3.3, 2

[150] Patrick Agyapong and Marvin Sirbu. Economic Incentives in Information-Centric Networking: Implications for Protocol Design and Public Policy. *Communications Magazine, IEEE*, 50(12):18 –26, December 2012. 4.1, 4.3.5, 4.4.2

[151] Peyman Faratin, Dave Clark, P. Gilmore, Steve Bauer, A. Berger, and William Lehr. Complexity of Internet Interconnections: Technology, Incentives and Implications for Policy. In *TPRC '07: Proceedings of Telecommunication Policy Research Conference*, 2007. 4.1

[152] R. Pantos and W. May. HTTP Live Streaming - Draft Version 8. Internet draft, Apple Inc, March 2012. 4.2.1, 4.4.3, 5.4

[153] Thomas Stockhammer. Dynamic Adaptive Streaming over HTTP –: Standards and Design Principles. In *Proceedings of the 2nd Annual ACM Conference on Multimedia Systems*, MMSys '11, pages 133–144, New York, NY, USA, 2011. ACM. 4.2.1, 4.4.3, 5.2.1, 5.4, 5.4

[154] RFC 1738. Uniform Resource Locators (URL), December 1998. 4.2.1

[155] RFC 3986. Uniform Resource Identifier (URI): Generic Syntax, Jan. 2005. 4.2.1, 7

[156] RFC 3568. Known Content Network (CN) Request-Routing Mechanisms, July 2003. 4.2.1, 4.2.3, 4.3.1, 4.3.5

[157] W3C. HTML 5.1 Editor's Draft. `http://www.w3.org/html/wg/drafts/html/master/Overview.html`, January 2013. 4.2.1, 5.2.2

[158] Kristen Purcell, Joanna Brenner, and Lee Rainie. Search Engine Use 2012. Technical report, Pew Research Center, Washington, D.C., March 2012. 4.2.2

[159] P. Mockapetris and K. J. Dunlap. Development of the Domain Name System. *SIGCOMM Computer Communications Review.*, 18(4):123–133, 1988. 4.2.3

[160] Yue Zhang, Serge Egelman, Lorrie Cranor, and Jason Hong. Phinding Phish: Evaluating Anti-Phishing Tools. In *Proceedings of the 14th Annual Network and Distributed System Security Symposium (NDSS 2007)*, February 2007. 4.2.5

[161] Electronic Frontier Foundation. HTTP Everywhere Browser Extension. `https://www.eff.org/https-everywhere/`. 4.2.5

[162] Mark Sableman. Link Law Revisited: Internet Linking Law at Five Years. *Berkeley Technology Law Journal*, 16(3):1273–1344, 2001. 4.3.3

[163] Robert M. Scott. Deep Linking - Policy and Rule Considerations for Safeguarding Open Internet Navigation. *Telecommunications and High Technology Law*, 1:355–380, 2002. 4.3.3, 6.1.3

[164] Patrick Agyapong and Marvin Sirbu. Social Welfare Implications of Different Cache Deployment Scenarios. *SSRN Working Paper Series*, July 2012. 4.3.5, 4.4.1, 4.4.2

[165] Peyman Faratin. Economics of Overlay Networks: An Industrial Organization Perspective on Network Economics. In *Proceedings of NetEcon 2007*, 2007. 4.4.2

[166] Somaya Arianfar, Pekka Nikander, and Jörg Ott. On Content-Centric Router De-

sign and Implications. In *Proceedings of the Re-Architecting the Internet Workshop*, ReARCH '10, pages 5:1–5:6, New York, NY, USA, 2010. ACM. 4.4.2, 4.4.4, 4.8.2

[167] George Pallis and Athena Vakali. Insight and Perspectives for Content Delivery Networks. *Communications of the ACM*, 49(1):101–106, January 2006. 4.4.2

[168] Jayson Sakata and Robert Peters. Systems and Methods for Invoking Commands Across a Federation. *U.S. Patent 8117276B1*, February 2012. 6

[169] John Chuang and Marvin Sirbu. Distributed Network Storage Service with Quality-of-Service Guarantees. In *Proceedings of the Internet Soceity INET'99 Conference*, San Jose, CA, June 1999. 4.4.2

[170] Elias Balafoutis, Antonis Panagakis, Nikolaos Laoutaris, and Ioannis Stavrakakis. The Impact of Replacement Granularity on Video Caching. In Enrico Gregori, Marco Conti, Andrew Campbell, Guy Omidyar, and Moshe Zukerman, editors, *NETWORKING 2002: Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications*, volume 2345 of *Lecture Notes in Computer Science*, pages 214–225. Springer Berlin / Heidelberg, 2006. 4.4.3

[171] Microsoft Corporation. IIS Smooth Streaming Transport Protocol. Documentation, Microsoft Corporation, 2009. 4.4.3, 5.4

[172] Stacey Higginbotham. We will Soon Live in a 100Gbps World. Gigaom `http://gigaom.com/broadband/we-will-soon-live-in-a-100-gbps-world/`, February 2011, accessed on August 11, 2012. 4.4.4

[173] Petri Jokela, András Zahemszky, Christian Esteve Rothenberg, Somaya Arianfar, and Pekka Nikander. LIPSIN: Line Speed Publish/Subscribe Inter-Networking. *SIGCOMM Computer Communications Review*, 39(4):195–206, 2009. 4.4.4

[174] Matteo Varvello, Diego Perino, and Jairo Esteban. Caesar: a Content Router for High Speed Forwarding. In *Proceedings of the ACM SIGCOMM Workshop on Information-Centric Networking*, ICN '12, pages 73 – 78, New York, NY, USA, 2012. ACM. 4.4.4, 4.8.2

[175] RFC 2046. Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types, November 1996. 4.4.4, 4.8

[176] RFC 4288. Media Type Specifications and Registration Procedures, December 2005. 4.4.4, 4.8

[177] RFC 5321. Simple Mail Transfer Protocol, October 2008. 4.4.4

[178] RFC 3550. RTP: A Transport Protocol for Real-Time Applications, July 2003. 4.4.4, 5.4

[179] RFC 3261. SIP: Session Initiation Protocol, June 2002. 4.4.4

[180] RFC 4648. The Base16, Base32, and Base64 Data Encodings, October 2006. 4.5.1

[181] Carey Williamson. On Filter Effects in Web Caching Hierarchies. *ACM Transactions on Internet Technology*, 2(1):47 – 77, February 2002. 4.5.2

[182] RFC 2328. OSPF Version 2, April 1998. 9

[183] Niraj Tolia, Michael Kaminsky, David Andersen, and Swapnil Patil. An Architecture for Internet Data Transfer. *Proceedings of the 3rd Symposium on Networked Systems Design and Implementation*, May 2006. 4.8.1

[184] Donna Hoffman and Thomas Novak. Advertising Pricing Models for the World Wide Web. In Deborah Hurley, Brian Kahin, and Hal Varian, editors, *Internet Publishing and Beyond: The Economics of Digital Information and Intellectual Property*. 2000. 5.2.2

[185] Jeffrey Yu Hu. Performance-based Pricing Models in Online Advertising. *Available at SSRN* `http://ssrn.com/abstract=501082`, 2004.

[186] Mohammad Mahdian and Kerem Tomak. Pay-per-Action Model for Online Advertising. In *Proceedings of the 1st International Workshop on Data Mining and Audience Intelligence for Advertising*, ADKDD '07, pages 1–6. ACM.

[187] David Evans. The Economics of the Online Advertising Industry. *Review of Network Economics*, 7(3), September 2008.

[188] David Evans. The Online Advertising Industry: Economics, Evolution and Privacy. *Journal of Economic Perspectives*, 23(3):37–60, 2009. 5.2.2

[189] Nir Kshetri. The Economics of Click Fraud. *Security Privacy, IEEE*, 8(3):45 –53, May-June 2010. 5.2.2, 6.1.2, 6.1.4, 6.3.5

[190] Carlos Castillo, Debora Donato, Aristides Gionis, Vanessa Murdock, and Fabrizio Silvestri. Know Your Neighbors: Web Spam Detection Using the Web Topology. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 423–430, New York, NY, USA, 2007. ACM. 5.3

[191] Zoltan Gyongyi, Pavel Berkhin, Hector Garcia-Molina, and Jan Pedersen. Link Spam Detection Based on Mass Estimation. In *Proceedings of the 32nd International Conference on Very Large Data Bases*, pages 439–450, 2006. 5.3

[192] Paul-Alexandru Chirita, Jörg Diederich, and Wolfgang Nejdl. MailRank: Using Ranking for Spam Detection. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, pages 373–380, 2005. 5.3

[193] RFC 2326. Real Time Streaming Protocol, April 1998. 5.4

[194] Adobe Systems Incorporated. Real Time Messaging Protocol Chunk Stream, June 2009. 5.4

[195] Saamer Akhshabi, Ali Begen, and Constantine Dovrolis. An Experimental Evaluation of Rate-Adaptation Algorithms in Adaptive Streaming over HTTP. In *Proceedings of the 2nd Annual ACM Conference on Multimedia Systems*, MMSys '11, New York, NY, USA, 2011. ACM. 5.4

[196] Thomas Ristenpart, Eran Tromer, Hovav Shacham, and Stefan Savage. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *CCS '09: Proceedings of the 16th ACM Conference on Computer and Communications*

Security, pages 199–212, New York, NY, USA, November 2009. ACM. 5.5

[197] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A View of Cloud Computing. *Commun. ACM*, 53(4):50–58, April 2010. 5.5

[198] Jaeyeon Jung, Balachander Krishnamurthy, and Michael Rabinovich. Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web Sites. In *Proceedings of the 11th International Conference on World Wide Web*, WWW '02, pages 293–304. ACM, 2002. 6, 6.1.1

[199] David Moore, Colleen Shannon, Douglas J. Brown, Geoffrey M. Voelker, and Stefan Savage. Inferring Internet Denial-of-Service Activity. *ACM Transactions on Computer Systems*, 24(2):115–139, May 2006. 6.1.1

[200] Kenneth C. Wilbur and Yi Zhu. Click Fraud. *Marketing Science*, 28(2):293–308, March 2009. 6.1.2, 6.1.4

[201] Brian Wassom. Copyright Implications of "Unconventional Linking" on the World Wide Web: Framing, Deep Linking and Inlining. *Case Western Reserve Law Review*, 49(1):181–256, 1998. 6.1.3

[202] W3C TPWG. `http://www.w3.org/2011/tracking-protection/`. 6.2.2

[203] William Stallings and Lawrie Brown. *Computer Security: Principles and Practice*. Pearson Prentice Hall, 2008. 6.3.1, 1

[204] Bernard Jansen. Click Fraud. *Computer*, 40(7):85 –86, July 2007. 6.3.5

[205] Janko Roettgers. To Stream Everywhere, Netflix Encodes Each Movie 120 Times. http://gigaom.com/2012/12/18/netflix-encoding/, December 18 2012. 7.4