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.

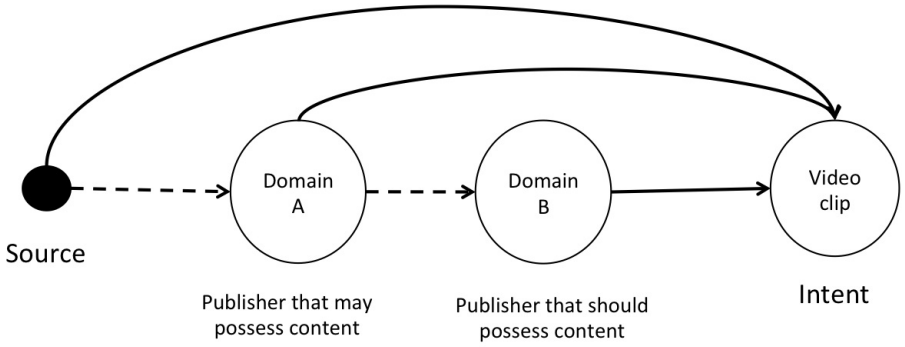

Figure 4.1: Illustration of the DAG addressing style in XIA. The user is primarily interested in obtaining a video clip from any location that has a copy. Nevertheless, the user knows that in the absence of nearby copies, the video may be obtained from publisher A or publisher B. A directed acyclic graph, 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-

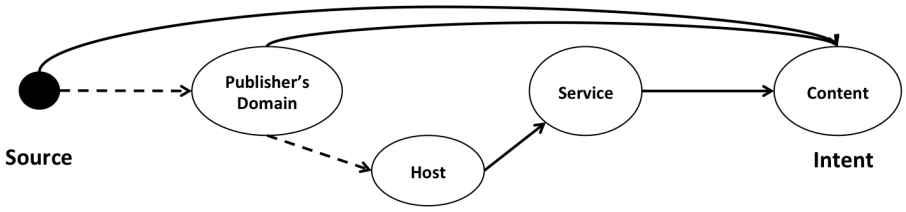

Figure 4.2: Illustration of the DAG addressing style in XIA. The publisher of the video clip provides users with a DAG address that captures the following intent. First, users can obtain the video clip from any content router in the network that has a copy. In the event that a router does not have a 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

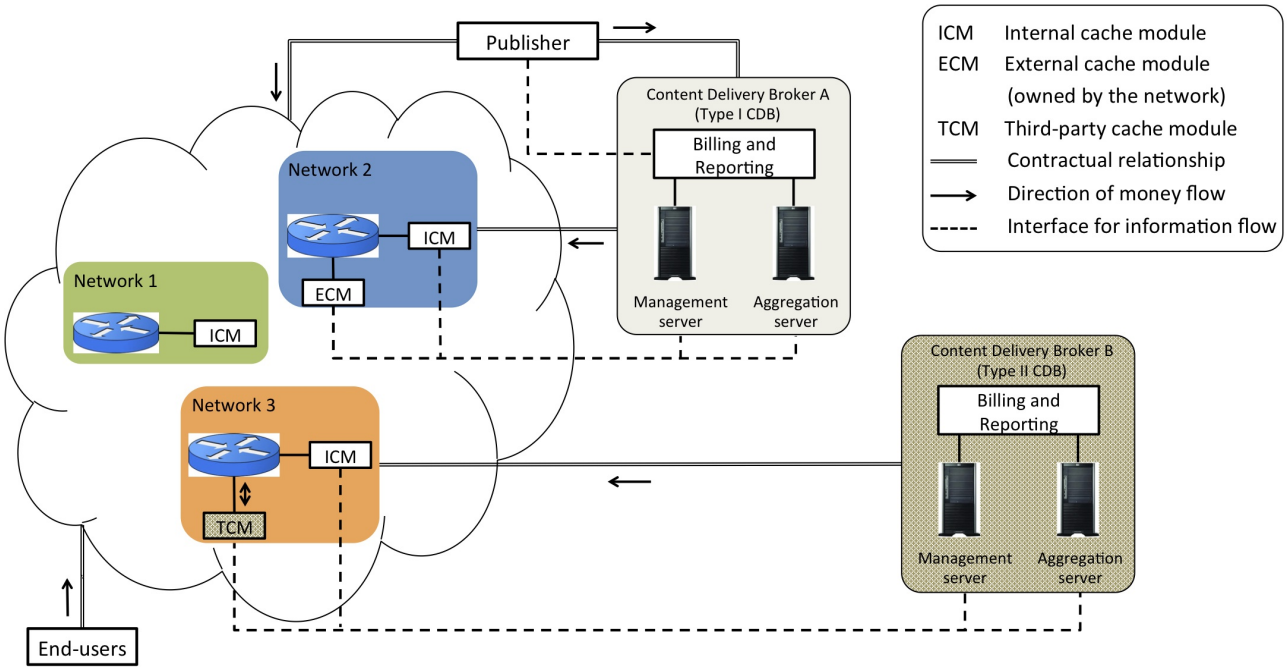

Figure 4.3: A simplified illustration of the CCN content delivery ecosystem and infrastructure. Caching and content delivery are performed by one or more cache modules, which interface with the router either internally or externally. There are two categories of external cache modules - those owned directly 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

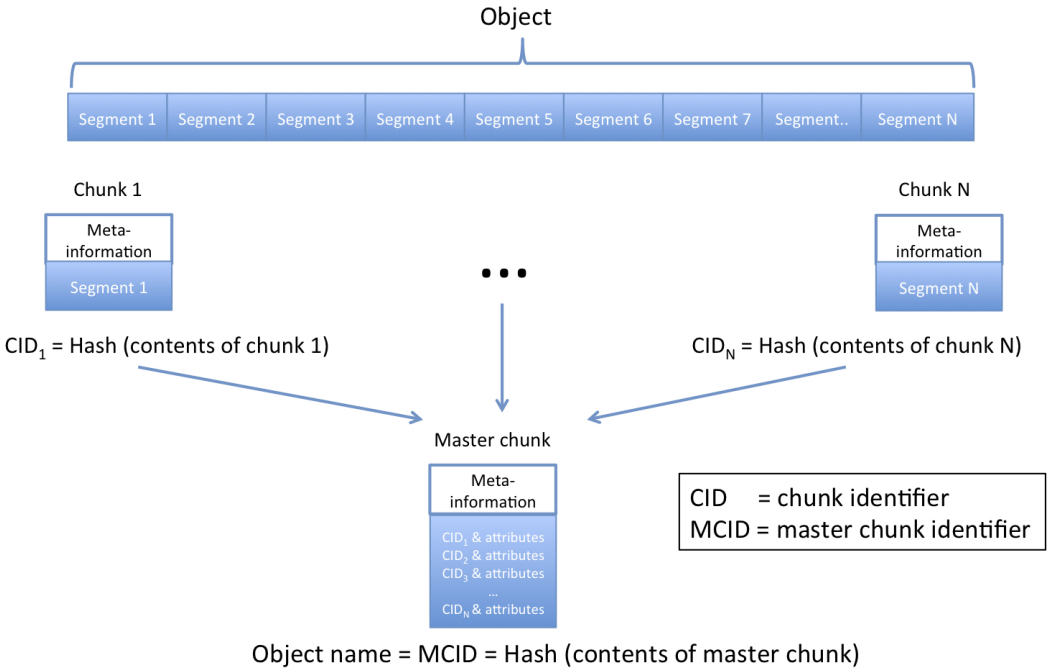

Figure 4.4: Relationship between chunks, master chunks and objects. A chunk describes the basic routable content unit. An object may be composed of several chunks. A master chunk contains a list of all the chunks that compose an object. When an object is composed of a single chunk, then the object is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

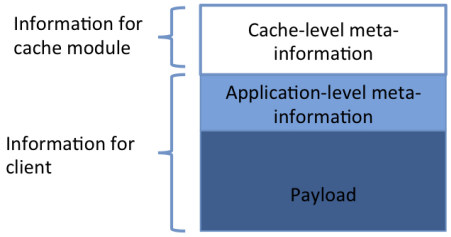

Figure 4.5: Structure of a chunk. All chunks contain two layers of meta-information. The first layer is used by cache modules to make caching decisions whereas the second layer is used by applications to know how to process the payload.

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

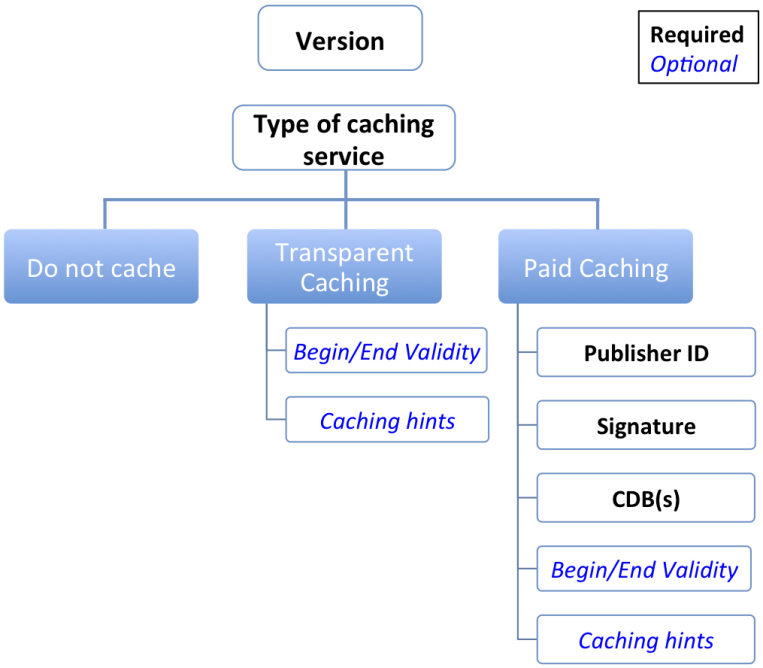

Figure 4.6: Cache-level meta-information include all information necessary for the cache module to make a caching decision. The type of caching service specified determines whether or not Publisher ID and Signature are present in the cache-level meta-information.

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.

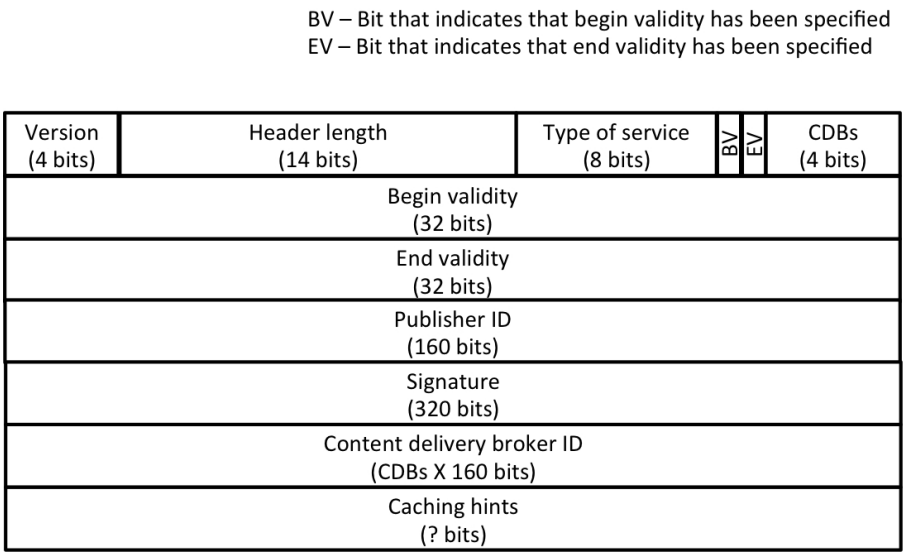

Figure 4.7: Encoding cache-level meta-information using a fixed field scheme. The header length is the number of bits contained in the header. The type of service indicates the type of caching service desired by the publisher. Possible values include: do not cache, transparent caching and other values 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,

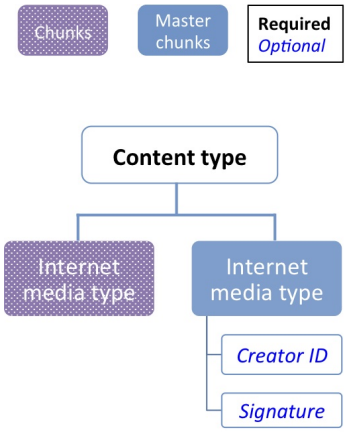

Figure 4.8: Application-level meta-information includes all information necessary for the client to process the payload. The content type specifies the Internet media type (see [175, 176]) to which the payload belongs. Optionally, the application-level meta-information may also specify authentication 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]).