Practical Accounting in Content-Centric Networking

Cesar Ghali* Gene Tsudik* Christopher A. Wood+ Edmund Yeh
University of California, Irvine Northeastern University
{cghali, gene.tsudik, woodc1}@uci.edu eyeh@ece.neu.edu

***Abstract*— Content-Centric Networking (CCN) is a recent network paradigm designed to address some key limitations of the current IP-based Internet. One of its main features is in-network content caching which allows requests for content to be served by routers. Despite the benefits of improved bandwidth utilization and lower latency of retrieving popular content, in-network caching inhibits producers from collecting information about content that is requested and later served from network caches. Such information is often needed for accounting and popularity purposes. In this paper, we address accounting in CCN by varying the degree of consumer, router, and producer involvement. We also identify and analyze inherent performance and security tradeoffs. We show that fine-grained accounting is infeasible with router caches and without explicit application support. We then recommend accounting strategies that entail a few simple requirements for CCN architectures. Finally, we show, via experimental results, that network-layer CCN accounting is viable and incurs low overhead for all parties involved. approaches.**

I. Introduction

Content-Centric Networking (CCN) is a recent inter-networking paradigm exemplified by two well-known research efforts: (1) the CCNx project at the Palo Alto Research Center (PARC) [1], and (2) the NSF-funded Named-Data Networking (NDN) project [2]. In IP-based networking, a user requests *content* by addressing the host at which it is stored. In contrast, in CCN, *content* is assigned a unique name and is addressed directly. Any entity can become a content *producer* as long as it is authorized for a certain part of the global content namespace. A user, called a *consumer*, obtains content by issuing an *interest* carrying the former's name. An interest can be satisfied by any entity (host or router) that either creates or caches the requested content. Once found, content follows, in reverse, the exact path of the preceding interest towards the consumer. Any routers on this reverse path may cache content to satisfy future interests.

This in-network caching facilitates efficient content distribution by reducing end-to-end latency and bandwidth consumption, especially, for popular content. Because of it many interests might not reach the original content producer. Consequently, a producer might only receive a small fraction of all interests for a given piece of content. However, the number, sources, and timing of interests represent information important to the producer for accounting purposes. Even if the timing and the number of interests were somehow communicated to the producer, interest sources would remain unknown since CCN inherently lacks consumer information, e.g., source addresses, in interests.

Furthermore, if and when CCN is deployed in the real world, router cache space will likely be treated as a valuable (and even premium) resource. Thus, a mechanism is needed for reporting cache hits to content producers and router owners in order to inform them about content usage. To be viable, such a mechanism must incur minimal bandwidth, computation, and storage overhead. Finally, to mitigate false cache usage reporting and other attacks, it also must be secure. In this paper, we design a lightweight network-layer accounting mechanism for CCN, with security features. It is applicable to both CCNx and NDN[1]. Our contribution is three-fold:

- Identification and motivation for features needed for CCN accounting and its security.
- First comprehensive technique for content and cache usage accounting, with varying levels of consumer, router, and producer involvement.
- Analysis of performance and security tradeoffs.

In the rest of this paper, we use the term CCN to refer to both CCNx and NDN.

II. CCN Overview

This section overviews CCN in the context of CCNx. Certain features such as packet formats, message fields, and forwarding particulars are slightly different in NDN.

Unlike IP, which focuses on end-points of communication and their names and addresses, CCN makes *content* named, addressable, and routable [1]. A content name is represented with a standard URI-like string. For example, a BBC news home page for April 15, 2015 might be named `/uk/bbc/news/2015APR15/index.htm`. CCN communication adheres to the *pull* model whereby content is delivered to consumers only upon explicit request. There are two basic types of packets (messages) in CCN: interest and content[2]. A consumer requests content by issuing an *interest* message. If an entity can "satisfy" a given interest, it returns the corresponding *content object*. Content is said to satisfy a given interest if their names match exactly, i.e., an interest for `/snapchat/bob/video-749.avi` can only be satisfied by a content object named `/snapchat/bob/video-749.avi`.

*Supported by NSF award: "CNS-1040802: FIA: Collaborative Research: Named Data Networking (NDN)".
+Supported by NSF Graduate Research Fellowship DGE-1321846.

[1]Support for NDN requires minor packet format and protocol changes.
[2]We use the terms *content* and content object interchangeably.

CCN interests include the name of the requested content. They may also carry a payload that enables consumers to push data to producers along with the content request [3].[3] CCN content objects include a number of fields, of which only four are relevant to our discussion:

- `Name` – URI-compliant sequence of name components.
- `Payload` – Actual data of the content object.
- `Validation` – Validation algorithm information (e.g., the signature algorithm used, its parameters, and a link to the public verification key), and validation payload, e.g., the signature. For simplicity, we use the term "signature" to refer to this field.
- `ExpiryTime` – Producer-recommended time to cache this content.

There are three types of CCN entities or roles:

- *Consumer* – an entity that issues an interest for content.
- *Producer* – an entity that produces and publishes content.
- *Router (Forwarder)* – an entity that forwards interest and content packets based on routing information.

Both consumers and producers have forwarders to move packets across the network. Internally, each forwarder maintains three components [4]:

- *Content Store* (CS) – a cache used for content. Its size is determined by local resource availability. Each router may unilaterally determine whether to cache specific content and for how long.
- *Forwarding Interest Base* (FIB) – a table of name prefixes and corresponding outgoing interfaces. It is used to route interests based on longest-prefix-match of the name.
- *Pending Interest Table* (PIT) – a table of outstanding (pending) interests and a set of corresponding incoming interfaces. A PIT entry might indicate multiple incoming interfaces reflecting the possibility of multiple interests arriving at, or near, the same time.

Upon receiving an interest with the name N a router first checks its cache for a local copy of content with the same name. If no local copy is found and there are no pending interests for N, the router forwards the interest to the next hop(s) according to its FIB and forwarding strategy. For each forwarded interest, a router creates a new PIT entry with state information, including the name and the interface on which the interest arrived. Moreover, if an interest with the name N arrives while there is already an entry for the same name in the PIT, the router collapses the new interest and only stores the interface on which it was received. When content is returned, the router forwards it to all recorded incoming interfaces and flushes the corresponding PIT entry. Since no additional information is needed to deliver content, an interest does not carry a source address.

III. Accounting in CCN

As mentioned earlier, router caches present a major challenge for accounting in CCN. In particular, *if interests are satisfied by caches, how can a content producer collect information about the popularity of, or demand for, its content?*

In this section, we discuss design elements for accounting in CCN. For the time being, we do not take into account security or privacy considerations. Specifically, assuming benign (well-behaved) consumers, routers, and producers, our *initial* goal is to determine the minimal functionality needed by all CCN entities to facilitate *correct* accounting. In doing so, we consider three types of accounting information:

- *Individual* information is tied directly to a specific consumer. An example might be the number of times a particular consumer requested a particular content. It provides linkability between consumers and content they obtain. Moreover, it requires revealing consumer identities, at least to the producer.
- *Distinct* information is functionally equivalent to individual information except that consumer identities are not revealed.
- *Aggregate* information represents collective or combined values over a set of consumers. For example, it might include the number of times a particular piece of content was requested from a specific geographic location or from a particular ISP. Aggregate information enables some degree of consumer privacy.

We believe that these three types correspond to most accounting information needed in any real-world CCN application and focus on them in the remainder of this work. Also, we recognize that accounting should not be mandatory for all content. Some producers might not care about the popularity of *any* of their content while others might need accounting information only for *some* of their content. We refer to content for which producers desire such information as *accountable* content.

Another important design dimension is whether accounting information is reported in real time (on-line) or off-line. In the latter case, a network management protocol can be envisaged whereby an AS-level accounting server periodically collects cache hit logs from its routers and reports the results to producers. This kind of accounting seems viable. However, it involves a potentially significant delay in notifying producers about demand for their content. This might be unacceptable for content for which real-time demand information is needed. Since real-time accounting presents a more difficult challenge, we concentrate on it in the rest of this paper.

A. Counting Cache Hits vs. Content Requests

Another variable in supporting CCN accounting is exactly what is being counted: instances of cache hits or instances of requested content being served to the consumer? A cache hit occurs when a router finds requested content in its cache. We assume that accounting for cache hits is only relevant for routers, i.e., network elements.[4] An instance of content being served occurs when a cache hit takes place *and* the content is actually delivered to a *single* consumer.

[3]NDN interests currently do not include a payload field. Thus, NDN can not support our accounting extension.

[4]Accounting for cache hits in content stores or at producers themselves is out of scope.

It might seem that these two types of events are the same, i.e., a content is served once for every cache hit and vice-versa. However, this is not the case in CCN. Whenever a router receives an interest, it may choose to multicast (forward) it out on multiple interfaces. This behavior is officially allowed since a router's FIB can express multiple next hops for a given name prefix. One practical reason for allowing it is to facilitate fast(er) content retrieval. However, it also complicates accounting. Consider the following scenario:

Suppose that a consumer issues an interest for content CO is received by router R_1. The latter forwards it to two upstream routers R_2 and R_3, based on the FIB. Both R_2 and R_3 have CO in their respective caches and each replies to R_1 with its cached version. Assuming that R_2's copy of CO is the first to reach R_1, the latter forwards CO downstream and flushes the appropriate PIT entry. When R_3's copy of CO arrives, R_1 discards it since it does not refer to an existing PIT entry. If both R_2 and R_3 inform CO's producer P about a cache hit, P would incorrectly assume that CO was requested twice. Even though CO is served twice by two distinct routers, a single consumer received one copy of CO.

In this example, the number of cache hits might not match that of content requests. This occurs because there is no way to distinguish among multiple interests issued for the same content.[5] In other words, if consumers Cr_1 and Cr_2 issue interests for CO at different times, their interests would be identical. Moreover, even if CO is not cached, i.e., interests for it reach P, and if Cr_1 and Cr_2 issue interests for CO at roughly the same time, P would be unable to distinguish between this case – when two consumers ask for CO – and the case in the scenario above – when one consumer asking for CO and R_1 decides to multicast the interest upstream. Note that the number of cache hits is two in both cases, while the number of contents served is two and one, respectively.

The reason for supporting both types of accounting is intuitive: a producer might need to know the exact demand for its content, whether on aggregate, distinct, or individual basis. Separately, a producer might need to know which routers experience cache hits for its content. The latter could be used to reconcile billing by the producer for cache usage.

Finally, even though accounting for cache hits and content requests is not the same thing, we naturally would like to use the same mechanism as much as possible to provide both. Therefore, in the rest of the paper, and unless otherwise mentioned, we use the term *accounting* to refer to both accounting for cache hits and content requests.

B. Accounting via Content Access Control

One potential accounting approach is to use encryption-based access control for content. Suppose that producers encrypt all accountable content, and decryption keys – which, in CCN, are represented as content objects with well-defined names – are configured not to be cached (i.e., their `ExpiryTime` values are 0). Even if interests requesting such content are satisfied from router caches, consumers would need to separately issue interests requesting decryption key(s). Such interests would bypass router caches and reach the producers, thereby enabling per-request accounting.

With content encryption, the desired type of accounting might dictate how key-requesting interests should be generated. For example, in case of individual accounting, consumers can include some kind of consumer-specific data in interests when keys are requested. Such data allows producers to link these interests to specific consumers. However, if only aggregate accounting is required, interests requesting keys do not have to carry any consumer-specific data. Such interests need to carry some kind of a nonce so that the producer can distinguish between the cases of (1) receiving two interests from two different consumers, and (2) receiving duplicates that stemmed from a single interest (issued by one consumer) which was multicasted by some downstream router.

Accounting via content encryption has two primary advantages: (1) it is transparent to the network and (2) it does not require any new features or message types. However, despite its apparent simplicity, it is inefficient. All accountable content needs to be encrypted and keys need to be requested and distributed separately. Thus, content is obtained by issuing at least two interests – one for the content and one for the key(s).

We believe that an ideal accounting mechanism should efficiently work for all accountable content. That is, it should not require a consumer to issue more than a single interest for accountable content. Also, content accounting should be distinct from content access control.

C. Accounting via Push Interests

The accounting approach proposed in this paper is based on real-time reporting. Its key element is a new message type called a *push* interest, denoted as *pInt*. Its main purpose is to inform the producer that its content has been requested and a cache hit occurred. Structurally, a *pInt* carries a name similar to a regular interest. However, the most important distinguishing feature of a *pInt* is that, unlike a regular interest, it does not leave behind any state in routers. Specifically, a *pInt* referencing CO is forwarded until it reaches the producer P, and no information about *pInt* is retained by any intervening router. A router forwards a *pInt* just as it forwards a regular interest with the exception that *pInt* messages are not multicast. This restriction is necessary to prevent producers from receiving duplicate copies of the same *pInt*.

A router R generates a *pInt* in two cases:

1) A regular interest for CO is satisfied from R's cache. R generates a *pInt* referencing CO and forwards it upstream towards P.
2) R receives a content CO corresponding to a PIT entry. R forwards that it on all downstream interfaces listed in that PIT entry. However, before flushing the entry, R generates a *pInt* that aggregates all collapsed interests.

[5]NDN interests carry a random nonce used for interest loop detection, which can be helpful in this distinction. However, CCNx interests do not carry nonces.

Algorithm 1 *pInt*-Generation

```
1: Input: CO[N, ACCT], Int[N, PL], R_id
2: pInt.Name := CO.N
3: pInt.Type := CO.ACCT
4: pInt.Origin := R_id
5: if CO from local cache then
6:     pInt.Cdata := Int.PL
7:     pInt.Count := 1
8: else
9:     e = FindPITEntry(CO.N)
10:    for each i in e/{Int} do
11:        pInt.Cdata := pInt.Cdata || i.PL
12:        pInt.Count := pInt.Count + 1
13:    end for
14: end if
15: Forward pInt according to the FIB
```

(These aggregation details are discussed in Section III-D.) Note that *collapsed* refers only to those interests that were not originally forwarded upstream. This is because the interest forwarded upstream presumably already (1) reached P, or (2) triggered its own *pInt* via case 1 above.[6]

The above is summarized in Algorithm 1. In order for P to inform routers about what content requires accounting, we also introduce a new content header flag `ACCT`, which reflects one of the following:

1) `NONE`: no accounting information.
2) `AGGREGATE`: aggregate accounting information.
3) `DISTINCT`: distinct accounting information.
4) `INDIVIDUAL`: individual interest-level accounting.

Whenever a cache hit occurs, routers behave the same in cases 2, 3, and 4. The only difference is when a content arrives and a router has a number of previously collapsed interests for that content. In case 2, a router generates a *pInt* with the count of collapsed interests for a given content. In cases 3 and 4, a router reports the *actual* interests, which can optionally be bundled into a single *pInt*.

D. pInt *Format and Features*

We now describe *pInt* message format which is very similar to CCNx 1.0 interests [5]:

- `Name`: copied entirely from `Name` field in the interest (or PIT entry) that triggers a *pInt*.
- `Type`: flag indicating whether this *pInt* is for aggregate, distinct, or individual accounting.
- `Origin`: identifies the router that generates the *pInt*, e.g., the router's prefix (if available) or public key digest.
- `Count`: set to 1 in the case of a cache hit, or the number of interfaces *minus one* on which the content object was forwarded downstream, if interest collapsing occurred.
- `Cdata`: a random nonce or consumer-specific data used by producer for different purposes based on the accounting type required (i.e., distinct or individual). If `Count`>1, this is a sequence of `Count` consumer-specific data values culled from corresponding interests. Such data can be carried in the interest `Payload` field.

Semantics of the `Cdata` field depend on the type of required accounting information. Below we discuss consumer-specific data requirements for each accounting type. As stated above, aggregate accounting for cache hits does not require `Cdata` to be present.

Aggregate: The problem in this type of accounting is that producers do not have the means to distinguish between the cases where received interests (or *pInt* messages) with the same name are multicast by routers or generated by several distinct consumers. However, if consumers include random nonces and timestamps as consumer-specific data, this distinction can be achieved.

Distinct: `Cdata` needs to reflect the uniqueness of interests. This can be achieved via consumer-provided nonce and timestamp combination. The nonce format is application-specific and can range from a random number to the hash of the content name and the timestamp. Note that the same knowledge provided to the producer in the distinct accounting case can also be attained using aggregate accounting type if `Cdata` reflects the uniqueness of interests. However, we keep the distinction between these two types for ease of classification.

Individual: `Cdata` needs to reflect identities of consumers that issued interests. This can take the form of:

1) Consumer public keys or their digests. Note that this form reveals consumer identities to all network entities – not only to producers.
2) Group public keys or their digests. A group can be an organization, autonomous system (AS), or a geographical region. In this case, the group identify is revealed rather than that of the individual consumer.
3) Unique consumer identifiers (i.e., pseudonyms). Although this does not violate consumer anonymity, such identifiers need to be assigned to consumers by producers or a trusted third party before any interests are issued. This form of `Cdata` also allows interest linkability.[7]
4) Consumer identity (using any of the three previous forms) with nonces and timestamps. This form of `Cdata` allows producers to know which consumers request what content, as well as how many times such requests are made.

Each of the above incurs different overhead for consumers and producers. However, router overhead is only very slightly affected. This is because routers simply populate `Cdata` of generated *pInt* messages using information contained in the `Payload` field of the corresponding interests, regardless of how consumer-specific data is generated. In other words, routers are oblivious to the accounting type used. Also, note that the choice of which form to use is an application-specific issue.

[6]For example, suppose that R receives an interest for CO on interfaces: 2, 3, 5, and 6. Regularly, only the first one is forwarded, say, on interface 9. Others are collapsed into the same PIT entry. Now, when CO arrives on interface 9, it is forwarded on all 4 incoming interfaces. However, R generates a *pInt* that reflects only three interfaces: 3, 5 and 6.

[7]Interest linkability is defined as the ability of an eavesdropper (observer or adversary) to reveal that fact whether two captured interests are issued by the same consumer.

E. Accounting Correctness

Definition 1. *An accounting technique is* correct *if it accurately reports cache hit and content request information to the producer, assuming that all participants faithfully follow the rules (i.e., no malicious behavior) and there are no transmission errors, no packet loss, and no node failures that affect accounting-relevant traffic.*

Definition 2. *An accounting technique is* probabilistically correct *if it is correct with a negligible probability of error, i.e., inaccurate or false information is reported.*

We now informally demonstrate correctness of each proposed accounting technique (individual, distinct, and aggregate) for cache-hit and content-request cases.

Cache Hit: A router R generates a *pInt* for every cache hit on an accountable content object. Since all routers (including R) on the path to producer P forward *pInt* messages according to their FIB entries, all such messages are guaranteed to be delivered to P. This provides accurate cache-hit counts for accountable content objects. This argument holds for all three types of accounting. The only difference is that `Cdata` fields of *pInt* must contain appropriate consumer-specific data in some accounting types.

Content Request: Although *pInt* messages provide *correct* individual, distinct, and aggregate accounting for cache hits, they only offer *probabilistically correct* accounting for content requests. As stated above, consumers can include nonces and timestamps in `Payload`. This information would allow producers to distinguish between cases where received interests (or *pInt* messages) with the same name are multicast by routers or generated by several distinct consumers.

IV. Security Considerations

Thus far, we assumed that all entities involved in accounting are benign. However, this assumption is clearly unrealistic in practice. In this section, we identify requirements for *secure* CCN accounting. We also demonstrate that some attacks can not be prevented or even detected without additional and non-negligible cryptographic overhead. Furthermore, secure accounting involves a trade-off between security and overhead for consumers and producers; as we show in this section, routers are unaffected.

A. Adversary Model

The anticipated adversary $\mathcal{A}$dv is a *malicious router* generating *pInt* messages for bogus interests when individual accounting is required. In other words, $\mathcal{A}$dv tries to inflate individual accounting information for both cache hits and content requests. For now, we assume that consumers behave honestly. We consider malicious consumers later in Section V.

For completeness, we identify certain other attacks and justify their exclusion from the discussion below.

- A router that (1) does not generate *pInt* messages when necessary, or (2) generates *pInt* messages without forwarding content downstream. Both cases can be reduced to packet loss. We do not address these attacks since this kind of misbehavior is very difficult to detect.
- A *consumer* that continuously generates interests to inflate accounting information. If aggregate or distinct accounting is required, the producer would be unable to detect such malicious behavior. On the other hand, if individual accounting is required, consumer-specific data can be used to detect continuous requests. However, this scenario can be reduced to Interest Flooding Attacks [6], which is outside the scope of this paper. A similar argument applies to distinct accounting information.
- An external attacker controlling the network can eavesdrop on, drop, or replay packets, including *pInt* messages. This attack is largely irrelevant if links are encrypted, which is a realistic assumption for adjacent routers. In most cases, consumers and producers communicate to edge routers over secure link-layer channels.
- An adversary tries to inflate aggregate or distinct accounting information. This cannot be prevented deterministically due to the likely usage of multicast forwarding strategies.

Based on these adversarial features, we only consider security of individual accounting information. We now define a secure accounting technique as follows.

Definition 3. *An accounting technique is* secure *with respect to Adv if it is correct and all Adv malicious behavior can be detected.*

Strategies and requirements to combat this adversary are discussed in the following section.

B. Mitigating Forgeries and Replay Attacks

Section III-C mentioned several options for generating consumer-specific data. However, in order to prevent inflation attacks, such data must be unforgeable and resistant to replay attacks. We define *secure* consumer-specific data as follows.

Definition 4. *Consumer specific data is* secure *if it can be authenticated by at least the producer, and is neither forgeable nor subject to replay attacks.*

Providing replay resistance can be accomplished if consumer-specific data carries a nonce r and a timestamp t. Thus, secure consumer-specific data Sec-CrSD assumes the following format:

$$\textsf{Sec-CrSD} = \left[\textsf{CrSD}||r||t, \mathsf{f}_k\left(\textsf{CrSD}||r||t||Int.N\right)\right]$$

where CrSD is consumer-specific data formatted as described in Section III-D[8], $\mathsf{f}_k(\cdot)$ is a function that computes an authenticated integrity check using a key k. The interest name in the $\mathsf{f}_k(\cdot)$ computation binds Sec-CrSD to the interest to which it is appended. This prevents $\mathcal{A}$dv from using the same Sec-CrSD for generating multiple *pInt* messages with different

[8]Note that CrSD and `Cdata` are different. The former is generated by consumers and assigned to `Payload` field of the interest, while the latter is a field in a *pInt* and may contain none or many CrSD-s.

names. $\mathsf{f}_k(\cdot)$ can be realized as a Message Authentication Code such as HMAC [7] (if consumers share keys with producers), or a digital signature function. Each alternative has well-known advantages and drawbacks. In addition to verifying $\mathsf{f}_k(\cdot)$, producers need to maintain a list of all received nonces within the current time window, for each accountable content.

Based on this discussion, we conclude that unforgeability and replay resistance can not be achieved unless *secure* consumer-specific data is used. This is not possible with aggregate or distinct accounting since consumer-specific data is not provided. One way to fix it is to include Sec-CrSD in all interests regardless of accounting type required. However, this introduces unnecessary overhead for both consumers and producers.

C. Preserving Consumer Anonymity

We now consider privacy issues. Ideally, Sec-CrSD needs to be opaque to all network entities, except producers. We start by defining consumer-specific data indistinguishability, which is necessary to maintain anonymity among an arbitrary set of consumers.

Definition 5. *Let Cr^a and Cr^b be consumers who each generate an interest for the same CO and let CrSD^a and CrSD^b be consumer-specific data values for their respective interests. Let Adv be an eavesdropper (except CO's producer) not directly connected to either Cr^a or Cr^b. Let the event of Adv learning the source of CrSD^a and CrSD^b be denoted as: $Adv_{\text{rev}}\left(\mathsf{CrSD}^a, \mathsf{CrSD}^b\right) = 1$. These two interests are* **indistinguishable** *if the probability of $Adv_{\text{rev}}\left(\mathsf{CrSD}^a, \mathsf{CrSD}^b\right) = 1$ is no better than a random guess. That is,*

$$\Pr\left[Adv_{\text{rev}}\left(\mathsf{CrSD}^a, \mathsf{CrSD}^b\right) = 1\right] \leq \frac{1}{2} + \epsilon(n),$$

for any negligible function ϵ and security parameter n.

We also assume that consumers know the producer's public key pk before requesting content (see Section V). Let A-CrSD denote an anonymous consumer-specific data: $\mathsf{A\text{-}CrSD} = \mathsf{Enc}_{pk}\left(\mathsf{Sec\text{-}CrSD}\right)$ where $\mathsf{Enc}_{pk}(\cdot)$ is a public key encryption function using pk, and Sec-CrSD is formed as defined in Section IV-B.

To prevent Adv from learning that multiple interests are generated by the same consumer, their A-CrSD values should be indistinguishable. This can only be achieved if $\mathsf{Enc}_{pk}(\cdot)$ is a CPA-secure public key encryption scheme, i.e., secure against Chosen Plaintext Attacks [8]. In some encryption schemes, this is done by mixing in a random number (nonce) with the every plaintext before encryption.

Theorem 1. *Assuming a CPA-secure public key encryption scheme $\mathsf{Enc}_{pk}(\cdot)$, A-CrSD format shown in Equation (1) guarantees* indistinguishability *of consumer-specific data with overwhelming probability.*

The proof can be found in [9].

Since public key operations are relatively expensive, symmetric cryptography is the natural alternative, albeit, with its own problems. A-CrSD can be formed as: $\mathsf{enc}_k(\mathsf{Sec\text{-}CrSD})$, where $\mathsf{enc}_k(\cdot)$ is a CPA-secure symmetric encryption function and k is a consumer-producer shared key. This would require additional operations for managing such keys. In order for producers to quickly determine the key that correctly decrypts a given A-CrSD-s field, consumers should include a cleartext key identifier (e.g., a key label), which clearly violates interest indistinguishability. One way to avoid this exposure is for multiple consumers to share the same key with the producer, e.g., based on location or time. An extreme option is to maintain unique per-consumer keys without labelling and require the server to discover the correct key by "brute force". This poses some obvious issues, e.g., new and exciting Denial-of-Service opportunities.

V. Individual Accounting in Practice

So far, we made some assumptions in the context of individual accounting:

1) Consumers know *what* accounting information is needed for a desired content object.
2) Consumers know the producer's public key pk used to encrypt Sec-CrSD.
3) Consumers behave honestly.

The first assumption seems to be particularly problematic, especially, if a consumer has no prior relationship with a producer. However, there are at least two ways for consumers to learn what a producer expects in an interest. First, recall that CCN network-layer trust management [10] requires the consumer to know the public key of the producer before requesting content. This, in turn, means that the consumer must pre-fetch the producer's public key. It is easy to extend the producer's public key certificate pk to include accounting requirements for constructing interests for that producer's namespace. An alternative is for a consumer to "blindly" issue a trial interest for some random content in the namespace of a given producer. This interest would likely not adhere to the producer's accounting rules. In this case, the producer simply replies with a public key certificate that includes its accounting requirements.

Without such techniques, a consumer cannot be expected to provide specific information in an interest. We therefore conclude that individual accounting necessitates an initial phase whereby consumers learn producer's CrSD requirements and pk for generating Sec-CrSD or A-CrSD values in interests. This initial phase would address assumptions (1) and (2) above. However, a misbehaving consumer can just ignore producer's requirements and pull content from router caches without providing correct accounting information to the producer, thus bypassing the accounting mechanism.

The main problem is that routers have no means to validate CrSD fields that arrive in interests referencing cached content. Allowing routers to do so is undesirable, because: (1) at least one cryptographic operation per interest would be needed, and (2) producers would need to inform routers about key(s) needed to validate CrSD, for each individually accountable content. The former is a new expense and a Denial-of-Service

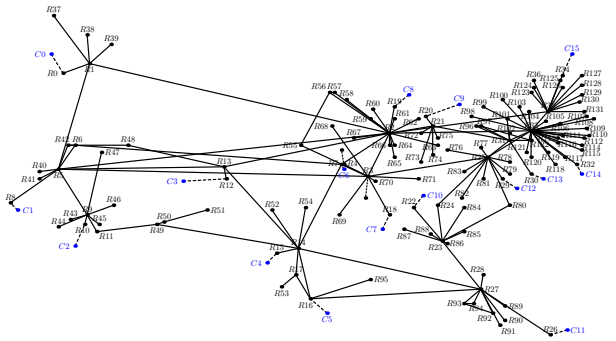


Fig. 1. AT&T topology - each edge router above is connected to a group consumers.

opportunity, while the latter is a key management nightmare. Also, replay prevention would add further complexity.

We therefore conclude that assumption (3) is unrealistic in the presence of dishonest consumers. Consequently, individual accounting should be handled at the application layer.

Recommendations: Based on the above discussion, we now present some recommendations for CCN accounting. First, if individual accounting information is needed, producers must simply set all content cache time to zero (0). This will force all interests to be routed to the producer. If an interest for content that requires individual accounting is received and the required accounting information is missing, producers should reply with a NACK [11] indicating consumer-specific data requirements for obtaining that content. The consumer can then re-issue an interest with the correct information. Since producers process all interests before responding with content, they can determine if a given interest for individual accountable content is valid and thus detect consumer misbehavior.

For aggregate and distinct accountable content, consumers should always include a random nonce in CrSD. If a router caches some content for which `ACCT` flag is `AGGREGATE`, CrSD can be simply dropped when *pInt* messages are generated. Otherwise, if `ACCT` flag is `DISTINCT`, the nonce must be copied into the *pInt*. This is a simple modification to the router *pInt* generation procedure described in Algorithm 1 which yields insignificant overhead for consumers and routers.

This simple policy can be extended to all interests. Since consumers are not generally expected to know what type of accounting information is required, they can blindly generate a nonce for each interest they issue. Routers would correctly propagate these nonces in *pInt* messages to the producer according to the rule above. As previously noted, NDN already supports default nonce generation in interests (for the purpose of interest loop detection); CCNx needs to be extended to satisfy this requirement.

VI. Analysis and Experimental Assessment

We now analyze performance of proposed accounting techniques and discuss some experimental results. Further experimental and analytical results can be found in [9].

Consider a scenario with k consumers $Cr_1, \ldots, Cr_k$ and a single producer P. Let $[Cr_i, R_1, \ldots, R_l, P]$ be the path traversed by interests issued by Cr_i for accountable content CO. Let R_c, $1 \leq c \leq l$ be the router nearest to Cr_i where CO is cached. Let p_l be the number of messages traversing $R_1 - R_c$ path in one direction, and let p_r be the number of messages traversing the $R_c - P$ path in one direction. Finally, let γ be the number of interests issued by all Cr_i, $i = 1, \ldots, k$, along the $R_1 - P$ path. Recall that encryption-based accounting requires consumers to issue at least two interests: one – for the content itself, and (at least) one – for decryption keys. The former traverses $R_1 - R_c$ path and the latter – $R_1 - P$ path. Thus, $p_l = 4\gamma$ and $p_r = 2\gamma$. In the *pInt*-based approach, a single interest is issued for CO on $Cr - R_c$ path, then a *pInt* is generated at R_c and forwarded along $R_c - P$ path. In this case, $p_l = 2\gamma$ and $p_r = \gamma$. Note that $R_c = P$ is identical to the scenario where there are no router caches and thus no *pInt* messages. This case performs worse than the *pInt*-based variant since $p_l = 2\gamma$ and $p_r = 2\gamma$; a single RTT from the Cr_i to P for CO.

Differences in p_l occur because, unlike interests, *pInt* messages elicit no response from the producer. In fact, overhead, in terms of the number of messages, of the encryption-based technique is exactly twice that of the *pInt*-based one. Overhead of the cache-less variant is also higher than that of the *pInt*-based technique. Furthermore, producers, and consumers incur additional overhead due to en-/de-cryption. Therefore, we focus on the *pInt*-based technique.

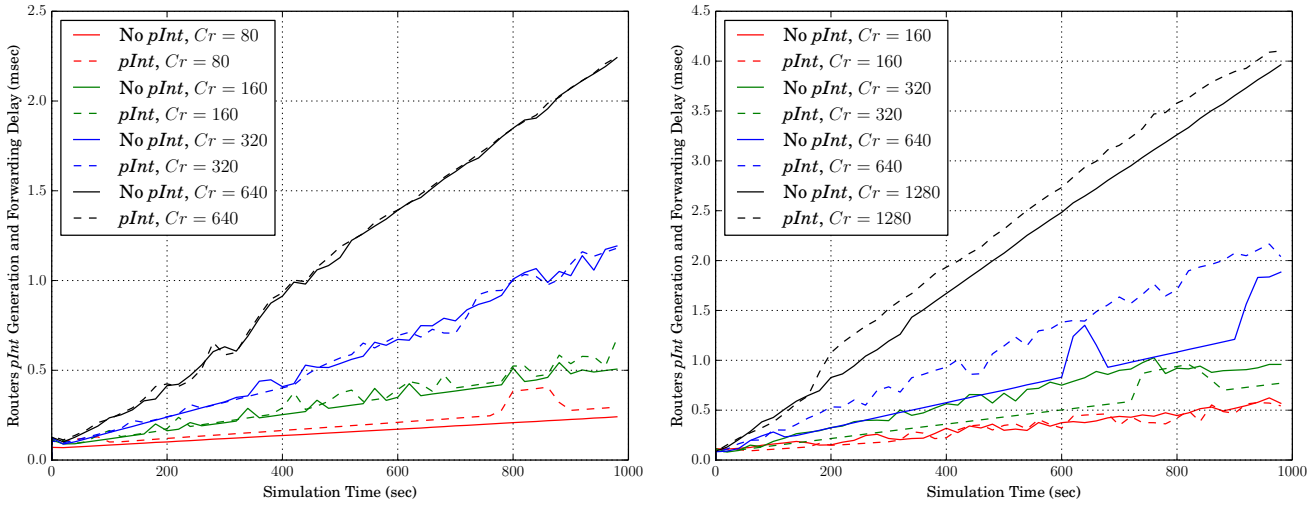
A. Router Overhead

To measure router overhead due to generating and forwarding *pInt* messages we extended ndnSIM 2.0 [12] – an implementation of NDN architecture as a NS-3 [13] module for simulation purposes – to support *pInt* messages. Using this modified architecture we ran two sets of experiments using the following topologies:

- The DFN network, Deutsches ForschungsNetz (German Research Network) [14], [15]: a German network developed for research and education which includes 30 routers positioned in different areas of the country.
- The AT&T backbone network [16]: as shown in Figure 1, in includes over 130 routers. Each logical consumer represents multiple physical consumers connected to an edge router.

In all experiments, consumers issue interests at a rate of 10/sec for the same content with the name `/prefix/A/00`. To capture the worst-case scenario, with the maximum number of *pInt* messages, we disable interest collapsing and set the `ExpiryTime` of the content to the simulation time to ensure that this content is cached at routers throughout the whole duration of the simulation. This forces routers to generate one *pInt* for every cache hit, resulting in the maximum number of *pInt* messages.

We measure router overhead as compared to the baseline case where *pInt* messages are not generated. Figure 2(a) shows this overhead in the DFN topology parameterized by the number of consumers connected to edge routers: 80, 160, 320, and 640. Even with 640 consumers, the overhead of an average



(a) DFN topology.

(b) AT&T topology.

Fig. 2. *pInt* messages generation overhead at routers.

router is negligible. Figure 2(b) shows the same overhead in the AT&T topology. In case of $1,280$ consumers, routers incur 15% overhead, which we consider to be tolerably low.

VII. Related Work

Network-layer accounting in CCN and related interest-based ICN architectures remains an open topic in the literature [17]. However, certain economic aspects, such as *how* to set and enforce prices, has been widely discussed [18], [19]. These results imply an application-layer strategy whereby payment (not usage) information is willingly sent on behalf of the consumer. This conflicts with the approach advocated by Agyapong et al. [20], wherein only ISP-related entities are involved in payment coordination. [20] considers payment as an application layer concern.

Patané et al. [21] study a similar problem in the context of IP-compatible architectures. Specifically, they focus on ones with dedicated router caches like Content Distribution Networks (CDNs) and transit networks that chauffeur traffic between different ISP provider networks. Payment policies proposed in [21] are identical, though. All parties pay for the resources which were used to deliver their content. Patané et al. also neglect to discuss means by which this payment and usage information can be propagated. Similar to [19], Kocak et al. [22] discuss methods where content providers can coordinate price information and contracts between ISPs. They also opt for an open, unfederated approach, which fits with our model of autonomous accounting information propagation.

[23] proposes the Secure E-Commerce Transactions for Multicast Services (SETMS) framework. It serves as overlay functionality on top of the core network and supports significantly more functionality than our accounting scheme, e.g., dynamic subscription to content, secure e-payments, identity authentication, etc. Our work is much more low-level in that it can serve as a *building block* for such a framework but does not replace it outright. Also in the context of multicast communication, [24] present a distributed management architecture for IP multicast services. Agents (e.g., routers) individually collate information about multicast groups and traffic that is later used directly for billing purposes. This information is not propagated to producers in real-time. Our accounting mechanism could be *enhanced* by this architecture (or SETMS) to support billing.

Another important element of this work is the generation of secure consumer-specific data in *pInt* messages. There is rich literature of packet-level authentication in the IP-based Internet, much of which is contained in [25]. However, techniques such as digital signatures and symmetric-key MACs require some possibly unrealistic assumptions, such as shared keys amongst all pairs of routers and trusted third parties for key generation and management. Using improved public-key cryptographic algorithms based on elliptic curves can help improve the signature scheme efficiency (see DNSCurve [26]). However, the sheer volume of interests in CCN and related ICNs will very likely be substantially larger than DNS queries in IP networks, leading to only modest performance gains.

VIII. Conclusion

This paper represents the first attempt to tackle the accounting issue in CCN. We presented a simple and lightweight network-layer accounting technique and showed how to securely extend it to the application-layer. We assessed performance of the proposed technique and demonstrated that CCN accounting is both possible and practical.

References

[1] "Content centric networking (CCNx) project," http://www.ccnx.org.
[2] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *CoNEXT*, 2009.
[3] "CCNx 1.0 protocol specications roadmap," http://www.ccnx.org/113/ccn-publications-papers/ccn-1-0-protocol-roadmap-available/.
[4] "CCNx protocol: CCNx node model," http://www.ccnx.org/releases/latest/doc/technical/CCNxProtocol.html.

[5] "CCNx messages in TLV format," https://tools.ietf.org/html/draft-mosko-icnrg-ccnxmessages-01.

[6] P. Gasti, G. Tsudik, E. Uzun, and L. Zhang, "DoS and DDoS in named data networking," in *ICCCN*, 2013.

[7] H. Krawczyk, M. Bellare, and R. Canetti, "RFC 2104: HMAC: Keyed-hashing for message authentication," 1997.

[8] J. Katz and Y. Lindell, *Introduction to modern cryptography: principles and protocols*. CRC Press, 2007.

[9] C. Ghali, G. Tsudik, C. A. Wood, and E. Yeh, "Practical accounting in content-centric networking (extended version)," in *arXiv*, 2015.

[10] C. Ghali, G. Tsudik, and E. Uzun, "Network-layer trust in named-data networking," *ACM CCR*, vol. 44, no. 5, 2014.

[11] A. Compagno, M. Conti, C. Ghali, and G. Tsudik, "To NACK or not to NACK? negative acknowledgments in information-centric networking," in *ICCCN*, 2015.

[12] S. Mastorakis, A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnSIM 2.0: A new version of the NDN simulator for NS-3," NDN, Technical Report NDN-0028, 2015.

[13] "Network simulator 3 (NS-3)," http://www.nsnam.org/.

[14] "DFN-Verein," http://www.dfn.de/.

[15] "DFN-Verein: DFN-NOC," http://www.dfn.de/dienstleistungen/dfninternet/noc/.

[16] A. Compagno, M. Conti, P. Gasti, and G. Tsudik, "Poseidon: Mitigating interest flooding DDoS attacks in named data networking," in *LCN*, 2013.

[17] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos, "A survey of information-centric networking research," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 2, 2014.

[18] A. Araldo, D. Rossi, and F. Martignon, "Design and evaluation of cost-aware information centric routers," in *ICN*, 2014.

[19] T.-M. Pham, S. Fdida, and P. Antoniadis, "Pricing in information-centric network interconnection," in *IFIP Networking Conference*, 2013.

[20] P. K. Agyapong and M. Sirbu, "Economic incentives in information-centric networking: Implications for protocol design and public policy," *IEEE Communications Magazine*, vol. 50, no. 12, 2012.

[21] R. Patané and J. Remond, "Economics of information-centric networks," *Internet Economics VIII*, 2014.

[22] F. Kocak, G. Kesidis, T.-M. Pham, and S. Fdida, "The effect of caching on a model of content and access provider revenues in information-centric networks," in *SocialCom*, 2013.

[23] A. K. Venkataiahgari, J. W. Atwood, and M. Debbabi, "Secure e-commerce transactions for multicast services," in *E-Commerce Technology, 2006. The 8th IEEE International Conference on and Enterprise Computing, E-Commerce, and E-Services, The 3rd IEEE International Conference on*, 2006.

[24] H. Sallay and O. Festor, "A highly distributed dynamic IP multicast accounting and management framework," in *IFIP/IEEE IM*, 2003.

[25] D. Lagutin, "Redesigning internet-the packet level authentication architecture," *Licentiates Thesis-Helsinki University of Technology*, 2008.

[26] D. J. Bernstein, "DNSCurve: Usable security for DNS," 2009, http://dnscurve.org/.