APCN: A Scalable Architecture for Balancing Accountability and Privacy in Large-scale Content-based Networks

Yuxiang Ma[a,b], Yulei Wu[c,*], Jun Li[a,*], Jingguo Ge[d,e]

[a] *Computer Network Information Center, Chinese Academy of Sciences, Beijing, 100190, China*
[b] *School of Computer and Information Engineering, Henan University, Kaifeng, 475004, China*
[c] *College of Engineering, Mathematics and Physical Sciences, University of Exeter, Exeter, EX4 4QF, UK*
[d] *Institute of Information Engineering, Chinese Academy of Sciences, Beijing, 100093, China*
[e] *School of Cyber Security, University of Chinese Academy of Sciences, Beijing, 100049, China*

Abstract

Balancing accountability and privacy has become extremely important in cyberspace, and the Internet has evolved to be dominated by content transmission. Several research efforts have been devoted to contributing to either accountability or privacy protection, but none of them has managed to consider both factors in content-based networks. An efficient solution is therefore urgently demanded by service and content providers. However, proposing such a solution is very challenging, because the following questions need to be considered simultaneously: 1) How can the conflict between privacy and accountability be avoided? 2) How is content identified and accountability performed based on packets belonging to that content? 3) How can the scalability issue be alleviated on massive content accountability in large-scale networks? To address these questions, we propose the first scalable architecture for balancing Accountability and Privacy in large-scale Content-based Networks (APCN). In particular, an innovative method for identifying content is proposed to effectively distinguish the content issued by different senders and from different flows, enabling the accountability of a content based on any of its packets. Furthermore, a new idea with double-delegate (i.e., source and local delegates) is proposed to improve the performance and alleviate the scalability issue on content accountability in large-scale networks. Extensive NS-3 experiments with real trace are conducted to validate the efficiency of the proposed APCN. The results demonstrate that APCN outperforms existing related solutions in terms of lower round-trip time and higher cache hit rate under different network configurations.

Keywords: Accountability, Privacy, Content Networks, Performance Analysis.

1. Introduction

Accountability is one of the key issues that concerns service providers (SPs) and content providers (CPs) most where the user traffic is required to be accountable (e.g., obtaining the packet source address) to stop in-progress attacks and prevent future harmful actions [4, 12]. In other words, when receiving messages, the requester needs to know who should receive sanctions when needed. However, Internet users always attempt to hide their true identity (e.g., source address) to protect their privacy in cyberspace [12, 17, 47] because most popular websites collect, store and share vast amounts of personal data regardless of the users' disapproval [25, 35, 50]. The current Internet allows some legitimate anonymous usages, e.g., visiting shopping websites, anonymous donations, secret ballots and anonymous reporting, owing to the support of several popular protocols such as Tor, Crowds and Address Hiding Protocol (AHP) [43]. This anonymity conflicts with the expectations of SPs or CPs, and, even the agreement in the cyberspace community - "one needs to be held accountable for one's actions" [12].

Privacy protection and accountability are two critical factors to ensure cybersecurity, and neither is dispensable. However, the existing IP source address acting as both the sender address and the return address triggers a series of contradictions between accountability and privacy [24, 45]. How to achieve accountability and privacy in the network has been a challenging task and has received tremendous research efforts [3, 14, 20, 22, 23, 30]. These efforts have focused on either accountability or privacy.

Content access is the dominant service in today's Internet. The authoritative Cisco Visual Networking Index (VNI) [11] has recently predicted that by 2021, every second, a million minutes of video content will cross the network. To support effective and scalable content distribution over the network, the current Internet is shifting towards a content-centric mode from a traditional host-oriented mode. The content-centric mode, e.g., peer-to-peer (P2P) networks and content delivery networks (CDNs), has been widely adopted by SPs or CPs for efficient file

*Corresponding author
Email addresses: y.l.wu@exeter.ac.uk (Yulei Wu), lijun@cnic.cn (Jun Li)

sharing and video transmission; this is evidenced by the prediction that the CDN traffic will carry 71% of all Internet traffic by 2021 [11]. On the other hand, the state-of-the-art studies on the future Internet have focused on information-centric networking (ICN), which is a clean-slate architecture centered on content distribution, leveraging in-network storage for caching and introducing new interaction models that decouple the content requesters and senders (also known as consumers and providers in ICN) [1, 15].

The rapid growth of the current Internet has given rise to new requirements and challenges in balancing accountability and privacy, especially in networks focused on content delivery. A distinct challenge is that, with the one-to-many communication mode widely used in content-based networks, the same content issued by the same sender needs to have the same identifier [1, 8]. However, this fact is not consistent with the existing packet-based architecture [39], since it was designed for one-to-one communication mode, i.e., accountability was performed based on the information of each packet; the cost of network bandwidth and storage will then increase significantly if applied to content-based networks. In addition, the characteristics of Internet traffic at the packet level are notoriously complex and extremely variable [13, 16]. Moreover, an intolerable verification (for accountability) delay on a packet will be produced if the verifier is located far away from the sender. How to protect the privacy of Internet users, while monitoring their behaviours to perform accountability when necessary is extremely important and a very challenging task in large-scale content-based networks [1].

To the best of our knowledge, no prior work has been reported to address this challenging issue. To fill in this research gap, we propose a new scalable architecture for balancing Accountability and Privacy in Content-based Networks (APCN). In this regard, we aim to design an architecture with respect to the unique characteristics of the content and content-based networks while accommodating the case of flow-based one-to-one communication mode. The design philosophy is to reduce the storage required for verification information and improve the network efficiency and scalability in large-scale content-based networks. Since the content-based network is driven by requesters [2] [1, 36], the proposed APCN is designed from the requester's point of view. In other words, the content requesters should be protected on privacy, and punished (on accountability) if issuing malicious requests. APCN also accounts for the balance of accountability and privacy for content senders. However, for public repositories (e.g., data centers and cache servers), the repository or its operating company should be accountable for its data based on some agreement, and its privacy can be protected by hiding the true identity of the servers. This could facilitate the mitigation of distributed denial-of-service (DDoS) attacks on public repositories. Furthermore, this architecture is designed as a protocol at the network layer because, in many applications, security at the network layer has many advantages over that provided in the other layers of the protocol stack [5], e.g., content-based distribution, access control, quality-of-service (QoS) guarantee, and security management [33, 34]. This is because the network layer forwards packets without dealing with the payload, and intermediate nodes can check or verify (but not modify) the packet at any hop, which results in fast detection of malicious packets. In summary, the main contributions of this paper are as follows:

- A new identifier, content ID (CID) is proposed to make APCN fit for networks focusing on content delivery. CID can uniquely identify content through hashing, and it can also be used in content self-certifying. With this newly introduced flexible content identifier, the same content issued from the same sender to different requesters would have the identical identity, and the same content issued by different senders would have different identities. In addition, with the proposed CID, a content can be verified using any of its packets.
- A double-delegate paradigm with one remote delegate in the source network domain and one delegate in the local network domain is proposed in APCN to make full use of in-network cache in content-based networks to obtain the verification information from a nearby delegate. In contrast to the existing work focusing only on the protection of sender privacy, this design allows the proposed architecture to include built-in support on the privacy protection for both content senders and requesters. In addition, this design is able to enhance network performance by verifying packets via the delegate at the local network, reducing the packet transmission delay, lowering bandwidth consumption, and decreasing the access burden to the original remote delegate.
- The potential security issues of the proposed APCN architecture in real-world deployment are analysed, and the corresponding solutions are provided and discussed.
- A theoretical analysis of key accountability processes of the proposed APCN is conducted through developing an elegant analytical model.
- We collect a real trace from an Internet service provider to evaluate the effectiveness and the performance of APCN. The performance of the proposed architecture is compared with that of a packet-based architecture in terms of two key metrics, round trip time

[1] The *content-based network* considered in this paper is an abstract network focusing on content delivery, so the proposed solution can be applied for both the networks under a content-centric mode and the ICN.

[2] The terms *requester* and *receiver* are used interchangeably in this paper, because the receiver usually acts as the content requester in content-based networks[8, 15].

(RTT) and cache hit rate. The results demonstrate that the proposed APCN has a higher efficiency in achieving the balance of privacy and accountability in large-scale content-based networks. The reason for having such a high efficiency is because the verification information is generated based on content rather than per packet in the proposed architecture; since a verified content can be reused for future verification, the verify process can, therefore, be completed locally.

The remainder of the paper is organized as follows. After introducing the related work in Section 2, we provide the problem description for this study in Section 3. Section 4 describes the design of the proposed APCN. The details of the accountability process along with the theoretical analysis are shown in Section 5. We analyse the potential security issues in the real-world deployment of APCN in Section 6. A performance analysis is carried out in Section 7 through extensive experiments. Finally, Section 8 concludes this study.

2. Related Work

In this section, we provide details of the existing work related to our study in terms of accountability and privacy and self-certifying address.

2.1. Accountability and Privacy

Accountability and privacy are considered two important factors for ensuring the success of the current Internet. Several existing works have been reported to address the problems raised by either accountability only or privacy protection only. For example, Accountable Internet Protocol (AIP) [3] was the most famous solution whose primary objective is accountability. In AIP, each host maintains a small cache to store the hashes of very recently sent packets. The first-hop router verified the packet by challenging the source with the hash of a packet it sent. If not successfully verified, the packet is dropped by the router, and a verification packet is sent to the source. When a packet traverses the boundary of an accountability domain, the previous accountability domain where the packets originate must determine whether the source address was valid. AIP can stop an attack by using the shutoff protocol, where a victim host sends an explicit shutoff instruction to the host who generates such traffic.

In addition to the related work considering accountability only, some existing solutions addressed privacy protection only. For instance, Tor [14] is one of the most popular projects for protecting user privacy. In its onion routing, instead of making socket connections directly to a responding machine, Tor makes connections through a sequence of machines called onion routers. Onion routing allows anonymous connections for both the requester and the sender. The Tor protocol (or Tor network) can effectively protect user privacy. The main drawback is very obvious in that we might never determine who is responsible for illegal behaviour. Furthermore, Lightweight Anonymity and Privacy (LAP) [22] was another popular contribution to privacy protection. It attempts to enhance anonymity by obscuring the topological location of an end-user based on two building blocks: packet-carried forwarding state and forwarding-state encryption. LAP allows each packet to carry its own forwarding state, and thus, the accountability domain can determine the next hop from the packet instead of retaining the per-flow state locally. In each accountability domain, a private key was used to encrypt/decrypt forwarding information in the packet header, preventing other accountability domains from obtaining the packet forwarding information. Some IPv6-based communication, as envisioned in the Internet-of-Things (IoT) scenarios, allows an individual host to be multi-addressed or to change addresses over time for privacy protection [23].

Efforts were seldom made to consider the effects of both accountability and privacy, until very recently when APIP [39] was proposed to solve the balancing problem between these two factors in packet-based networks under a one-to-one communication mode. In APIP, the sender issues a packet with an accountability delegate's address instead of the source address. At the same time, a message is forwarded to the accountability delegate to notify what has been issued by the sender. Routers (and the requesters) along the packet transmission path can verify the received packets by sending a verify request to the accountability delegate. Requesters can send messages to the accountability delegate to report malicious behaviours if they find the received packet was malicious or illegal.

Table 1 summarizes the differences between these various schemes and makes a comparison with the proposed APCN architecture. Tor is an application layer tool that can be used to protect user privacy. LAP, AIP, APIP and the proposed APCN all work at the network layer. Among them, LAP focuses on privacy protection, and AIP focuses on accountability. Although APIP can balance privacy and accountability, its implementation requires operations on all data packets, and the third party might disclose the user's information. The main objective of this paper is to provide a scalable architecture to balance accountability and privacy in large-scale networks focusing on content delivery (i.e., one-to-many communication). In addition, we address the reliability and security issues of third parties in this study.

2.2. Self-certifying Address

It is worth noting that, in recent years, many research efforts have been made on new types of network addresses, most of which emphasize self-certifying [37]; this has been considered the trend in the development of communication networks [18, 37, 46]. The self-certifying identifiers have been widely adopted in a variety of distributed systems. For example, in [18], self-certifying was established by binding three different entities: Real-World Identity

Table 1: The Comparison of Different Schemes

Scheme	Layer	Accountability	Privacy	Level
Tor [14]	Application Layer	Yes	No	Session
LAP [22]	Network Layer	No	Yes	Session
AIP [3]	Network Layer	No	Yes	Packet
APIP [39]	Network Layer	Yes	Yes	Packet
APCN	Network Layer	Yes	Yes	Content

(RWI), name, and public key. To achieve the goal of self-certifying name in the AIP [3], the name of an object is considered the public key or the hash of the public key that corresponds to that object. In the NSF-funded MobilityFirst project [46], a self-certifying Globally Unique Identifier (GUID) was derived simply as a one-way hash of a public key, which was then used to support the authentication and security throughout the project. In Host Identity Protocol (HIP) [40], which was standardised by the IETF, the host identifier was also derived by a fixed-size hash of the public key.

3. Problem Description

To adapt to the trend of pervasiveness of content access in today's Internet, a new architecture is proposed for balancing privacy and accountability. As introduced in Section 1, the proposed APCN is designed as a protocol at the network layer to address the problem of balancing accountability and privacy presenting in the current content-based networks with one-to-many communication. In this study, the definitions of privacy and accountability are given below, which are consistent with many existing studies [12, 17, 25, 28].

Definition 1. *Privacy protection ensures that when users send packets, intermediate nodes (e.g., routers) cannot determine who the sender is according to the received packets.*

Definition 2. *Accountability means that when malicious behaviours occur we can find the sender who should be held responsible for this behaviour.*

According to RFC 3697 [41], a *network flow* is a sequence of packets sent from a particular source (sender) to a particular destination (receiver) that the source desires to label as a flow. We follow this definition in this study.

It is worth noting that the security of data transmission has traditionally been entrusted to key-based cryptographic techniques. However, data encryption can only protect content from being known; our work protects the identity of the communication entity, i.e., preventing intermediate routers or attackers from knowing who is communicating.

Scenario. In this study, we consider balancing accountability and privacy in a one-to-many communication mode (transmission scenario). That is, multiple requesters request a content, i.e., a content is sent to multiple receivers. In the following, we provide two specific scenarios as examples.

We consider a scenario where requesters want to send anonymous requests to obtain desired contents from the sender which might be a CP. To protect the requester's privacy, the CP (possibly a sensitive site) hides the requester's real address when sending the content back to the requester. In this process, we need to prevent malicious anonymous accesses to the CP and also the malicious CP (e.g., its servers are used to attack other network nodes). Consequently, the behaviours of both the requesters and the CP need to be checked. Considering a requester is not located in the same accountability domain of the CP, if other requesters request the same content from the same domain as this requester, they can verify the validity of the content locally (e.g., verify whether the content is from an honest CP), without having to go to the CP's accountability domain for verification.

We consider another simple scenario. Today's mobile devices are typically equipped with integrated cameras, enabling people to create multimedia content simply, and thus remarkably enlarging the population of multimedia producers. When an individual wants to send multimedia content to multiple people (e.g., his/her friends), we need a new architecture that can provide more efficient and greener services to balance accountability and privacy; for example, generating verification information based on content rather than flow. In other words, we only need to generate one brief message per content, rather than generating one brief message for each receiver.

Security Assumption. We assume that the third party (i.e., delegates) is honest-but-curious. This assumption has been widely used in many related studies [10, 29, 31]. The delegate executes the task specified by the protocol but is curious about any information that may be disclosed in this process. In addition, the delegate may be attacked and controlled by attackers. We will provide a method in Section 6.1 to reduce the negative effects of the third party when it is no longer trusted or is attacked.

Threat Model. In addition to the risk of malicious delegates, the architecture also faces a variety of security threats. Threats may come from the sender, the verifier (intermediate routers or the receiver), and the content requester. In particular, the sender might send malicious

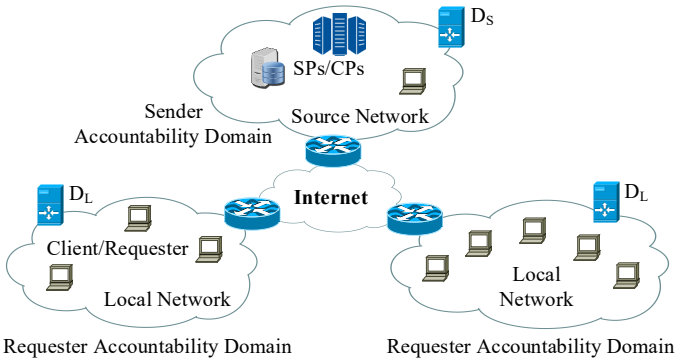

Figure 1: The proposed APCN architecture with double delegates.

content to the requester or attack other network nodes. The verifier may send a large number of invalid and malicious verify requests. For example, when no corresponding content needs to be verified, the verifier still issues a large number of verification requests to the delegate. The requester may attack other network nodes through issuing a large number of malicious requests or sending a large number of invalid brief messages to the delegate. In addition, intermediate nodes (e.g., routers) may steal the CID, add it to the header of the packet, and pretend to be content from a secure sender. The intermediate node may also initiate a replay attack using some packets of a content and the corresponding CID. Therefore, security threats may appear in the brief, verify and shutoff processes. The attack type includes DDoS attacks, flood attacks, replay attacks, and identity theft. In the proposed architecture, we address these threats.

4. The Proposed APCN Architecture

The meaning of IP address has become overloaded; the source address represents too many roles in today's Internet [44]. To balance privacy and accountability, it is necessary to separate the packet accountability address and the return address originally integrated in the source address. A recent work [39] contributed to separating these two types of addresses in flow-based networks (i.e., one-to-one communication mode). However, its address format (including NID, HID and SID) is fundamentally unsuitable for content-based networks because it can only identify a flow, but cannot identify the same content sent by the same sender in different flows. In addition, to achieve the goal of scalable accountability for the contents in large-scale networks such as the Internet, a delegate adopted in the source domain only is insufficient and could degrade the network performance if the distance between the source domain and the verifier who wants to challenge the received packet is relatively far. In what follows, the proposed APCN with a new address format and a newly introduced local delegate is presented.

The proposed APCN includes the double accountability delegates shown in Fig. 1: a source delegate (D_S) in the sender accountability domain and a local delegate (D_L) in the requester accountability domain. D_S is responsible for the senders and hides the true identity of the senders if needed. This delegate cannot guarantee that the packet issued by the sender is correct or non-malicious but merely works as a third party to acknowledge whether the packet was truly issued by the sender. Attacks from senders can be prevented by the Shutoff() procedure (see Section 5.3). To protect the privacy of the requesters and achieve the goal of enhancing network performance in terms of reduced packet delay, lowered bandwidth consumption, and decreased access burden to the source delegate, D_L is proposed in the APCN architecture. D_L plays an important role in balancing accountability and privacy for requesters, and it caches the required information used to respond to the requests from local verifiers.

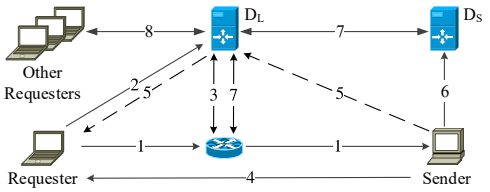

Figure 2: Lifecycle of the request process in the proposed APCN architecture.

The lifecycle of the request process in APCN is shown in Fig. 2, including the following steps:

1. The requester issues a request to the sender who has the content, with an accountability address to prevent intermediate routers along the path from knowing the requester source address. If required, the requester source address can be masked or encrypted in the content payload.
2. The requester sends the brief message to D_L, where the brief message is used to verify what the sender has issued (more details on brief messaging is discussed in Section 5.1).
3. A verifier can challenge any request sent by requesters. The sender and any intermediate routers along the path can be the verifier. More details on the verify process are discussed in Section 5.2. If the sender determines that the request it received is malicious or illegal, it can stop verifying this request by notifying the delegate, and the delegate will punish the requester if required (more details on the shutoff process are discussed in Section 5.3).
4. If the requester allows the sender to know its true identity, the sender can use the requester source address, which has been masked or encrypted, in the payload for response or connection purposes. In this process, content producers (e.g., users who generate the content) or CPs (e.g., date centers) can act as the sender. Since the content has been encrypted, intermediate routers still never know what the requester requested.

5. If the requester does not provide its true identity in the payload, the sender responds to the request with the requester's accountability address (the address of D_L). After receiving the content, D_L forwards it to the requester. This approach is an alternative option to Step (4) and can be used to mitigate the traffic analysis attack.
6. Similar to Step (2), the sender issues the brief message to D_S.
7. A verifier can challenge a content issued by the sender through verifying any packet of that content. Similar to Step (3), the verify and shutoff (stop verifying) processes may occur during content transmission. In general, the verification request is directed to D_L and then forwarded to D_S only when the verification cannot be performed at D_L. More details on the verify and shutoff procedures are discussed in Sections 5.2 and 5.3.
8. For other requesters, if the cached verification information is still available in D_L, the verification can be performed at the local network similar to the corresponding procedures in Step (3). For a special case, if D_L or the local network caches the content, we can still follow steps (1) - (5) to take the accountability of the original sender at the local network.

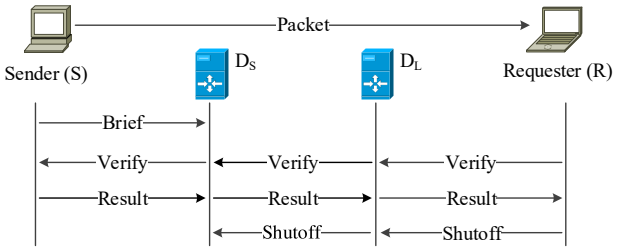

Figure 3: An overview of the request process.

Fig. 3 shows an overview of the request process. Given that the Internet is divided into many geographical network domains, the design of the APCN architecture can efficiently achieve a scalable balance of accountability and privacy in large-scale content-based networks.

4.1. Addresses

APCN adopts self-certifying address, which is similar to the address used in the popular AIP [3] and XIP in XIA (a typical future Internet architecture focusing on network security) [19]. Each packet has three addresses: destination address, accountability address, and return address. Table 2 shows different types of addresses with their formats.

Each address consists of three elements: 1) a network ID (NID) used to identify a network domain, 2) a host ID (HID) used to represent a host, and 3) a content ID (CID) used for the identification of different contents. The NID and HID are generated by the hash of a public key, and

Table 2: Addresses in the Packet Header

Address Type	**Format**	**Note**
Destination Address	$NID_D : HID_D : CID_D$	Mandatory
Accountability Address	$NID_A : HID_A : CID_A$	Mandatory
Return Address	$NID_R : HID_R : CID_R$	Optional

the CID is explained below. It is worth noting that, although NID:HID can be used to locate a unique host, the CID is adopted for finer-grained positioning in a host. In other words, the CID can be used to distinguish different contents in the same host. If the application requires ports for communication, the socket ID (SID) can be used to replace the CID in the destination address and the return address. The SID is used at the host to demultiplex packets to sockets [42]. These logical IDs may be separate header fields or a combined field (e.g., an IP address that encodes both an NID and an HID; the content can be represented by the CID, or the port number serves as an SID). In the following, we describe the CID in detail.

In APCN, the CID is proposed as a key element for overcoming the limitations of the packet-based approach. There are three goals of using the CID: 1) it is capable of self-certifying; 2) it can effectively identify a content, where the same content issued from a sender to different requesters will have the identical identity, and the same content issued by different senders will have different identities; 3) it can meet the requirements of verifying a content based on any of its packets at any time instead of performing the verification after receiving all the packets of a content.

In XIA, the content identifier is the cryptographic hash of the associated content [19]. This does not fit the proposed architecture because the same content sent by different senders will have the same CID if we generate the CID as recommended by XIA. Thus, we cannot distinguish who should be held accountable if needed. Therefore, in APCN the CID is more than just a simple hash of the content but also binds the public key of the sender and the address of the accountability delegate to the content, i.e.,

$$\text{CID} = H(Content \| K^+_{sender} \| (NID_A : HID_A))$$

where H is a cryptographically secure hash function, and $\|$ represents concatenation. The accountability delegate address $(NID_A : HID_A)$ is used to distinguish the same content sent from different accountability domains. K^+_{sender} $/K^-_{sender}$ denotes the public/private key pair of the sender; K^+_{sender} is included in the CID to distinguish the same content issued by different senders in the same accountability domain. In addition, K^+_{sender} plays an important role in the self-certifying process [19, 46]. The security analysis of CID and how to implement self-certifying is presented

in Section 6.2.

In APCN, the public/private key pair of the sender is allowed to be generated by the sender, and then the public key is sent to the accountability delegate. The key pair does not require frequent replacement, and the sender can decide when to regenerate a new key pair. If delegates find different senders having the same public key in the same accountability domain, they can ask these senders to regenerate a new key pair. Otherwise, the same content issued by different senders in this accountability domain would have the same CID.

The CID is generated when the content is produced, and it can uniquely identify a content. When a content is divided into several packets for transmission, the same CID carried in these packet headers can be used to verify the given content at any time, without waiting for the entire content to be received.

In APCN, a pair of NID:HID (i.e., $NID_A : HID_A$ and $NID_D : HID_D$) can be used to identify a network flow (the definition of flow was given in Section 3), and the CID can be used to distinguish different contents in a flow. When a sender sends the same content to different receivers, the CID might appear in different flows. This allows APCN to locate the problem more accurately. For example, when we want to stop forwarding malicious or illegal content, we can shutoff a content transmission rather than a flow, unless the sender is an attacker (more details on shutoff will be discussed in Section 5.3).

4.2. The Necessity of Using Self-Certifying Identifiers

Why is the CID designed to be self-certifying? As we introduced in Section 2.2, the self-certifying identifier has been considered the trend in the development of communication networks [18, 37, 46]. Therefore, in addition to NID and HID, CID as part of the address should also be self-certifying. Another important reason is that, although the delegate acting as a third party can vouch for a sender or a requester, i.e., the delegate can confirm that a requester issues a request or a sender sends a content. However, it is not possible for the delegate to verify that the sender owns the content. If someone needs to confirm that the sender is the owner of the content, the CID should have the ability of self-certifying. We analyse the security of CID and introduce how to implement the self-certifying CID in Section 6.2.

4.3. Local Delegate (D_L)

D_L plays an important role in balancing the privacy and accountability of the requester, which has been described in the *lifecycle of the request process* (see Fig. 2).

Table 3: Cached Information in D_L Used for Local Packet Verifying

CID	**Result (T/F)**	**Expiration**
CID1	T	30 s
CID2	F	20 s

In addition, D_L is another key element in the proposed APCN architecture that is used for reducing the delay of packet verification through local verifying. The verification information stored in D_L includes the CID, the validation results and the expiration time. A typical example of what is stored in D_L can be found in Table 3. "T/F" can be used as the validation result to denote whether a delegate vouches for a packet (content). Since we do not add a synchronous or push mechanism between D_L and D_S, for security considerations, we set the expiration time for each entry in D_L [49] (setting the expiration time is discussed in Section 7.1). The expiration time can also help decrease the negative effects when the requester would not detect the malicious problem in time because, after the expiration time, the verification information needs to be reacquired from D_S (this is elaborated in Section 5.2). The storage occupied by the caching verification information is another factor required in the proposed design. If there is insufficient space, the entry will be removed by a suitable replacement policy, e.g., Least Recently Used (LRU).

5. The Accountability Process in APCN

This section presents how accountability is performed in APCN. In the following, let $\{x\}_{KEY}$ indicate a signature on x with a KEY. The KEY can be a private key or a symmetric key. For example, $\{x\}_{K_A^-}$ denotes a signature on x with *A's* private key and $\{x\}_{K_{AB}}$ represents a signature on x with a symmetric key shared by A and B. Let $\rightarrow$ be the packet transmission path, e.g., $A \rightarrow B$ indicates packets sent from A to B. For the sake of clarity of illustration, the following abbreviations will be used throughout this section: S denotes the Sender, R represents the Requester, P is the Packet, H is the cryptographically secured hash function, and MAC denotes the Message Authentication Code.

5.1. Brief()

Brief() is the process in which S sends a brief message to D_S to notify what it has issued. Brief needs to contain enough information to allow D_S to respond to any verification request. Each brief message includes a ClientID, a fingerprint and a MAC using the key K_{SD_S}:

$$\text{Brief}(P) = ClientID \| F(P) \| MAC_{K_{SD_S}}(ClientID \| F(P))$$

where *ClientID* is used to notify D_S who sent the packet. It could be either the sender's HID or other identifiers only known by D_S. MAC is a tool used to ensure message integrity. The fingerprint is used as evidence to respond to the verification request. D_S stores fingerprints and determines whether the content was truly sent by S through querying whether the corresponding fingerprint exists. The packet fingerprint *F(P)* is given by

$$F(P) = H(K_{SD_S} \| CID \| H(P_{body}))$$

where K_{SD_S} is a symmetric key which is only known by S and D_S; it is created if D_S agrees to be responsible for S and can be replaced periodically. K_{SD_S} is included in the fingerprint to prevent others from establishing the association between P and *F(P)*. It is also a measure to protect the privacy of the senders. P_{body} represents the payload of a packet. It is included in the *F(P)* to meet the requirement of verifying a content at any time based on any packet belonging to that content.

Instead of sending full-size fingerprints, a bloom filter can be used. Specifically, S can periodically send a bloom filter to D_S after collecting fingerprints. In this case, K_{SD_S} is no longer needed in the fingerprint, because if we use a bloom filter, it is difficult for an observer to know what is contained in the fingerprint. The use of a bloom filter can reduce the storage usage of D_S and the network overhead between S and D_S, but at the same time, the false positive rate of the bloom filter is inevitable. In addition, the expiration time of the brief message is not limited by D_S. It is worth noting that, regardless of whether the direct transmission of fingerprints or bloom filters is used, the delegate will not know the payload of the content issued by the sender.

How to detect malicious behaviours in the brief process. Malicious senders may send a flood of Briefs to their delegate. The delegate can issue some policies to prevent Brief flooding, such as using bloom filters or setting a maximum number of Briefs received per second from the same host. This flooding behaviour can be detected. For example, a delegate receives numerous brief messages from a sender, but there is no verification request sent to the delegate to verify whether the sender has sent something.

5.2. Verify()

The purpose of Verify() is to determine whether a delegate is responsible for a content. A verifier can check any received packets belonging to a content. The delegate will check two things to achieve this purpose: 1) whether the content was truly sent by S through checking whether the corresponding fingerprint exists in D_S or the corresponding CID exists in D_L, and 2) the transmission of this content has not been shutoff. If both checks are passed, the delegate replies with a VERIFIED message. It is worth noting that only one packet of the same content is required to be verified because all packets of the same content carry the same CID in their headers. Once validated, the content is vouched for by some delegate. The verifier can maintain a whitelist locally for recording which CID (content) has recently been verified. We can set a duration to keep this whitelist alive, after which the verifier needs to perform the process of Verify() again to avoid some cases, e.g., the sender is attacked during this period of time.

In our proposal, a verifier only needs to verify a single packet from a piece of content, and then it whitelists all other packets with the same CID. Under this case, what stops an attacker from sending arbitrary packets using that CID? Because in the brief process (see Section 5.1) the hash of packets is sent to the delegate; the verifier can, therefore, challenge a sender using any packet of the content. If an attacker steals the CID, the payload of that fake packet will be incorrect. If the fake packet is used in the verify process, it will not pass the verify. If the CID is stolen, but the fake packet is not issued to the delegate for verification, even if the requester receives the packet, it still cannot be used to assemble the content. The requester will ask to retransmit the packet, similar to the current TCP/IP network.

In the Verify() process, the following three cases may occur.

5.2.1. No validation results found at the local delegate

A verifier first enquires whether D_L has a record of verification results of a content. If no result could be found because that content has not been requested or verified locally or the content verification result stored locally has expired, D_L will forward the request to D_S.

5.2.2. Validation results found at the local delegate

When the verification response by D_S returns, D_L caches the result. If the same content is requested for verification by other verifiers in the same accountability domain, the verification is performed directly by D_L. This greatly improves the efficiency of verification and reduces the consumption of network resources. If verifiers use the same delegate with senders, the verifier can directly verify from D_S.

5.2.3. No verification information in the delegate (neither D_S nor D_L)

If information used for verification cannot be found in all delegates (including D_S and D_L), because the verification information is expired, or the Briefs and validation results are lost during transmission, we can obtain the results from the sender. In this case, D_S can send a verification request to the sender. The sender verifies whether the content has been sent. If the sender does not support recursive verification, or cannot confirm whether the content was sent by itself, because there might be no record in the sender or the sender has re-generated the corresponding CID, the content is not vouched for by any delegate. In this case, the content needs to be dropped and the requester can re-request it.

When a verifier sends the verify request to the delegate, it needs to include the CID and the hash of the packet. MAC with K_V is used to ensure that the response from the delegate is what V requests, where K_V is the private key only known by the verifier V. This process can be expressed as

$$V \to D_S,\ V \to D_L \text{ and } D_L \to D_S :$$
$$\text{Verify}(P) = CID \| H(P_{body}) \| MAC_{K_V}(CID \| H(P_{body}))$$

If there is no verification information in the delegate, D_S sends the verification to the sender, i.e.,

$$D_S \rightarrow S : \{ \mathit{VERIFICATION} \| CID \}_{K_{SD_S}}$$

where the *VERIFICATION* is the request sent from D_S to S. Once the verification result is obtained, the delegate responds to the verifier via the following process.

$$D_S \rightarrow D_L \text{ and } D_L \rightarrow V :$$
$$\text{Result}(P) = \{ \mathit{VERIFIED} \| \text{Verify}(P) \}_{K^-_{D_S}}$$

where $K^+_{D_S}$ / $K^-_{D_S}$ is the public/private key pair of D_S.

How to deal with a large number of verification requests in which attacks may occur. In this process, delegates can count where the verification requests originated. If a verifier sends a large number of duplicate verify messages (regardless of the verification interval) over a period of time, the delegate will temporarily stop the verification request from that verifier. In addition, if a verifier sends a large number of invalid verification requests, e.g., the received verification cannot find the corresponding sender in the records, delegates will check whether the verifier is malicious or there are malicious nodes posing as a sender in the accountability domain.

5.3. Shutoff()

When a receiver (i.e., senders or requesters) identifies a malicious flow or content, they can stop the traffic using Shutoff(). The shutoff message will first be sent to D_L, such as Verify(), then forwarded to D_S, i.e.,

$$V \rightarrow D_L :$$
$$\text{Shutoff}(P)_{\text{D}_\text{L}} = \{ CID \| H(P_{body}) \| duration \}_{K^-_R}$$
$$D_L \rightarrow D_S :$$
$$\text{Shutoff}(P)_{\text{D}_\text{S}} = \{ CID \| H(P_{body}) \| duration \| HID_V \}_{K^-_{D_L}}$$

where the *duration* is the length of time that the delegate stops verifying the content sent from the malicious sender; the HID_V is the HID of the verifier who sends the shutoff, which should be included in the shutoff message when being forwarded from D_L to D_S, because if malicious shutoffs are detected, the one who sends the shutoff message could be accountable through D_L. D_L also records the shutoff request for subsequent actions.

When a sender's delegate receives a shutoff for a particular CID, does it block all future transmissions of that piece of content or all future transmissions from that sender? Our suggestion is to make different decisions based on different circumstances. Delegates can record and count the frequency and the number of valid shutoffs. For an occasional case, e.g., only one requester sends a shutoff against the content, we can stop the content being forwarded to the requester who sent the shutoff message. If a content is reported as malicious by many requesters, we can stop forwarding that content in the network. When a sender issues considerable malicious contents, it is necessary to prevent the sender from sending packets for a period of time and even address the sender's legal responsibilities.

Whether to respond to the shutoff instruction. It is optional for the delegate to respond to the shutoff instruction. Our suggestion is that if the shutoff is successful, the delegate needs to respond. Otherwise, the delegate can ignore such an instruction, or punish the verifier who sent a malicious shutoff. If the verifier does not receive a response within a pre-defined threshold, for example, due to the loss of shutoff messages during transmission, it can re-send the shutoff to the delegate.

5.4. Theoretical Analysis

In this section, an elegant analytical model is derived to analyse the efficiency of APCN compared with a packet-based (PKT) architecture. In APCN, we allow the same content acquired by multiple requesters to use the same validation message. This means that we can detect problems early, and then perform shutoff or accountability actions. Fig. 4 shows the difference in the accountability processes between APCN and the PKT architecture.

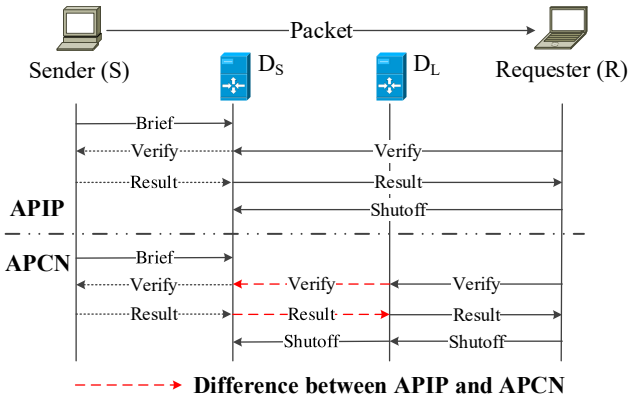

Figure 4: The comparison of accountability process between APCN and the PKT architecture.

5.4.1. The delay in the accountability process

The delay of the verify process in the PTK architecture can be expressed as

$$D^{PKT} = \sum_{i \in V} R_{D_S,i} P_{D_S}$$

where $R_{D_S,i}$ denotes the RTT between the i-th verifier and D_S, and P_{D_S} is the probability of finding the verification information at D_S. V represents the set of verifiers.

In APCN, if the verification information is cached in D_L, the delay can be given by

$$D^{APCN}_{D_L} = \sum_{i \in V} R_{D_L,i} P_{D_L}$$

where $R_{D_L,i}$ denotes the RTT between the i-th verifier and D_L, and P_{D_L} is the probability of finding the verification

information at D_L. If no verification information is found in D_L, the delay will be:

$$D_{D_S}^{APCN} = \sum_{i \in V} R_{D_S,i}(1 - P_{D_L})P_{D_S}, (R_{D_S,i} \geq R_{D_L,i})$$

and the reduced RTT can be given by

$$\Delta_{D^{APCN}} = \sum_{i \in V} R_{D_L,i}P_{D_L} - \sum_{i \in V} R_{D_S,i}P_{D_L}P_{D_S}$$

5.4.2. Reduced access burden to the source delegate

Let N_{total} denotes the total number of requests (including verify and shutoff requests), and N_{D_L} represents the number of requests received at D_L. With the proposed APCN, the cache hit rate of D_S can be reduced by

$$\varphi = \frac{N_{total} - N_{D_L}}{N_{total}} = \frac{\sum_{i \in V} Req_i(1 - P_{D_L})}{\sum_{i \in V} Req_i} = 1 - P_{D_L}$$

where Req_i denotes the request issued by the i-th verifier.

Through theoretical analysis, we find that APCN can reduce the RTT of packet verification and relieve the access burden to the delegate D_S.

5.4.3. Analysis of Space Complexity and Time Complexity

In this section, we analyse the space complexity and the time complexity of the proposed APCN.

Space complexity. Using the proposed APCN architecture, the maximum communication overhead, which involves verification information delivery, is no more than $Total_{message}(\Delta t)$:

$$Total_{message}(\Delta t) = m \times n \times (\delta + \delta) = 2mn\delta$$

where m represents the number of requesters (network users), and n denotes the request rate of a requester. δ denotes the cost of storing a piece of verification information in a given time window (Δt). Since there are two delegates in the accountability process (i.e., D_L and D_S), and each of them needs to store one copy, therefore two δ are needed. The space complexity of the PKT architecture is $mn\delta$ because the verification information in the PKT architecture only needs to be stored in the D_S.

Time complexity. In the proposed APCN architecture, the worst scenario is that there is no verification information in both D_L and D_S, and the verify message needs to be sent to the sender for verification (more details of this process can be found in Section 5.2.3). Therefore, the total time complexity of the accountability process in a given time window (Δt) is no more than $Total_{time}(\Delta t)$:

$$Total_{time}(\Delta t) = O(n^3)$$

However, in most cases, verifiers can obtain the verification results from the D_L (Section 7 addresses this argument). Even if the verifier needs to obtain the verification result from D_S, the time complexity is $O(n^2)$. The time complexity of the PKT architecture is $O(n^2)$. Because in the PKT architecture, verification information is sent directly to the D_S, and then sent to the sender if there is no result in the D_S.

6. Security Analysis in Real-world Deployment

In this section, a list of possible security problems that may be encountered in the real-world deployment of the proposed architecture and their countermeasures are analysed.

6.1. The Duty of Accountability Delegates

The delegate has three basic responsibilities in APCN: 1) protecting the privacy of their clients, 2) verifying packets via fingerprints or cached verification information, and realizing the accountability to their clients, and 3) dropping (stopping verifying) invalid or malicious packets if needed.

Recall that the delegate is assumed to be honest-but-curious in the above-mentioned design. However, what is harmful if the delegate is no longer trusted? Client privacy may be released, and the verification function may not be performed, or the incorrect verification results may be sent. Thus, the delegate cannot take the role of accountability, and the clients can decide to replace the delegate and take legal measures to punish the delegates [6, 39].

In APCN, multiple delegates can be adopted in each accountability domain; the client can then randomly select a delegate or choose one based on the credibility of the delegates to reduce the risk of victimization. In addition, a *Registry* can be used in our architecture to be responsible for generating, distributing, and managing the client's real identity. A client ID can be generated by the hash of the sender's public key and a random number (i.e., nonce): Client ID $= H(K^{+}_{sender} \| nonce)$.

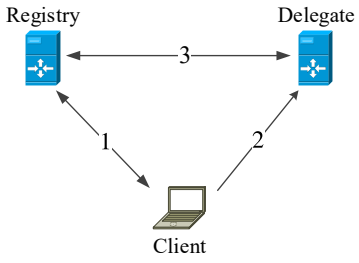

Figure 5: The relationships between Client, Delegate, and Registry.

The relationships between the Client (e.g., requesters, senders), the Delegate, and the Registry are shown in Fig. 5, and their interactions can be described as follows:

1. A client registers a client ID with the registry. Only the registry knows the mapping between the client ID and the client's identity. Since the registry does not participate in the accountability process (see Section 5), it does not know the behaviour of the client.
2. The clients send brief messages to the delegate as we described in step (2) of Fig. 2. More details of the brief process can be found in Section 5.1.
3. When a client becomes a malicious node and needs to be punished, delegates need to cooperate with the registry to identify the real identity of a client.

Thus, delegates do not learn the real identity of a client, because the client ID is only a flexible and replaceable alias of a client. It can replace the HID in the packet header when a client sends brief messages to the delegate. Even if the delegate is attacked, it will not result in the disclosure of client privacy. Only when a delegate cooperates with a registry, can the delegate completely learn exactly which client has communicated with whom. However, even the delegate and the registry can be attacked, in which case the attacker still cannot obtain what (packet payloads) the client has issued, as the delegate only stores client behaviours rather than the packet payloads. Therefore, the design of the registry decentralizes the delegate's rights, and this decentralized design helps to improve the security and reliability of the proposed architecture [2, 32].

6.2. The Security of the Content ID (CID)

We described why CID is designed to be self-certifying in Section 4.2. Self-certifying means that an entity (e.g., requester or sender) claims to have an identity (e.g., CID), and other entities can verify whether the identity belongs to the claimant without third-party participation. Because the sender's public key and the address of the delegate are available, anyone can verify the authenticity of the CID after receiving the content. In addition, if a node disguises the sender, it can also be discovered by the following steps:

1. The verifier sends a random nonce N to the one that declares itself to be the sender (requester or content owner).
2. The sender encrypts the random nonce N using its private key, and then sends the encrypted N along with its public key to the verifier if the verifier does not have the public key.
3. The verifier decrypts the encrypted random number N using the public key and determines whether the random number N is what it sent to the sender.

The verifier can then calculate whether the hash of the content, the sender's public key and the delegate's address are equal to the CID (i.e., $H(Content \| K^{+}_{sender} \| (NID_A : HID_A)) = \text{CID}$). In this process, the content is what the verifier receives, the accountability address (i.e., $NID_A : HID_A$) is in the packet header, and the public key is just verified.

6.3. Leaking Privacy by Logs

In the proposed APCN architecture, when a requester sends a request to a sender (i.e., the content owner), the source address in the packet header is the requester's delegate address (i.e., requester's accountability address). Therefore, the log analysis can only learn that there exists a communication between two accountability domains, without learning which node makes/receives the communication unless there is only one node in each domain.

6.4. How to Deal with a Replay Attack

In the existing packet-based architecture, a fingerprint is a function of the sender, the receiver, and the data, while in APCN it only depends on the sender and the data. Once a sender sends a brief to its delegate for a particular piece of content, an attacker might replay any packet from that content to other nodes in the network. In this case, the victim sends a shutoff message to inform the local delegate which CID has been used by the attacker. Because the shutoff message is sent to the local delegate in the first instance (then forwarded to the source delegate), when subsequent packets of the malicious content (used for attack) enter the local accountability domain, the delegates stop vouching for that content, and the routers stop forwarding that content; the attack will eventually be blocked. It is worth noting that shutoff does not affect the normal communication of both sides, i.e., the sender can still transmit other contents to the requester. However, in the existing packet-based architecture, if the receiver (content requester) sends a shutoff message to the sender's delegate, the other packets from the sender to the receiver (i.e., the specific flow) will be blocked. As a result, the sender will not be able to send other contents to the receiver for a period of time. In other words, it will affect the normal transmission of other contents, especially when a flow contains multiple contents.

7. Performance Evaluation and Analysis

In this section, extensive simulation experiments are conducted to validate the effectiveness and evaluate the performance of the proposed APCN architecture. In particular, we compare the performance of APCN with that of a packet-based architecture (PKT) [39] to show its relative merits. To achieve this purpose, we develop a discrete-event simulator based on the NS-3 simulation framework [21] and implement both the APCN and the PKT architectures. Table 4 shows the key system parameter settings in the simulation environment.

To evaluate the effect of different parameters on performance, we change the number of content from 10,000 to 60,000 in Section 7.2 and set α (Zipf distribution parameter) from 0.7 to 1.3 in Section 7.3. In Section 7.4, we vary the cache size in delegates. We increase the request rate of the requesters from 25 to 150 in Section 7.5. In Section 7.6, we vary the number of requesters. In Section 7.7, we change the distance between D_S and D_L by changing the link delay. We introduce the changes in the parameters in each subsection. If there is no particular highlight, the default parameter settings in Table 4 will be used.

The same topology used in Fig. 2 is adopted in the simulation environment. The topology of the China Science and Technology Network (CSTNet), a small Internet service provider with more than 400 nodes is used to simulate the local network. A core router of CSTNet is selected as D_L, and the 345 leaf routers act as clients/requesters. We

collect the network data from a border router of CSTNet to investigate the performance of the APCN and the PKT architectures. This one-hour trace was taken on August 13, 2013 at noon, from 11:00 to 12:00. It contains 463.22 million flows. The system parameters are set based on the existing studies [15, 26] and real-world measurements. In the local network, the link delay is set to 1 ms because according to our measurement, the average delay from the requesters to D_L is 2.57 ms and the average number of hops from the requesters to D_L is 2.48 hops. The request rate is set to 50 requests/s [26]. The dataset comes from a SIGCOMM paper [15], which contains more than 100,000 objects, and the request rate distribution follows Zipf's law [7]. Given that the exponent parameter of the Zipf distribution is 0.99 in the United States, 0.92 in Europe, and 1.04 in Asia [15], in this paper, we chose 1.0 as the default value.

Table 4: The System Parameters Setting

Parameter	Value
Number of requesters (clients)	345
Link delay between D_S and D_L	3 ms/hop × 10 hops = 30 ms
Forwarding strategy	Best Route
Replacement	Least Recently Used (LRU)
Payload	1,024 Bytes
Link bandwidth at local networks	100 Mbps
Link bandwidth between D_S and D_L	1,000 Mbps
Max packets for transmission queue on the link (both directions)	10,000 packets
Number of content	100,000
Cache size in delegates	100,000 packets
Link delay of connections at local networks (link delay between clients)	1 ms
The request rate	50 requests/s
Request packet size	31 Bytes
Zipf (q, α)	(0, 1.0)
Simulation time	600 s

The hop count and link delay between D_S and D_L can reflect the distance between them. To have reasonable settings of link variables (e.g., delay), we made some real-world measurements. A measurement result reveals that the RTT between Beijing and a site in a university in Kaifeng that is approximately 600 kilometres away from Beijing is 32 ms. Another measurement result shows that the RTT between Beijing and a farther site in Hong Kong is 169 ms. Based on these results, the delay between D_S and D_L is initially set to 30 ms. The effects of varying distances between D_S and D_L are evaluated in Section 7.7.

We conducted experiments to investigate the effects of different settings of the expiration time of the verification information stored in the delegate on network performance, and found that with the expiration time exceeding 30 s, the network performance will not have significant improvement (see Section 7.1). Therefore, the expiration time of verification information in D_L is set to 30 s in the simulation.

Two key performance metrics are adopted to evaluate the performance of these architectures: reduced RTT ratio and cache hit rate. The *reduced RTT ratio* represents the effect of the reduced RTT in the accountability process when using the proposed APCN compared with that when using the PKT architecture. The *cache hit rate* is the probability of successfully finding the verification information at D_L, reflecting the effect of relieving the burden of accessing D_S and reducing the bandwidth consumption between the two delegates - D_L and D_S. These two performance metrics can be expressed as follows:

$$Reduced\ RTT\ Ratio = \frac{\overline{RTT}_{pkt} - \overline{RTT}_{apcn}}{\overline{RTT}_{pkt}}$$

$$Cache\ Hit\ Rate = \frac{N_{hit}}{N_{hit} + N_{miss}}$$

where $\overline{RTT}_{pkt}$ and $\overline{RTT}_{apcn}$ are the average RTTs for the accountability process under the PKT and the APCN architectures, respectively. N_{hit} is the number of packets successfully verified at D_L, and N_{miss} is the number of packets that do not find the verification information at D_L.

7.1. Effects of the Expiration Time of Verification Information Cached in the Delegates

To investigate the effects of the expiration time of the cache entries in the delegate, Fig. 6 depicts the performance results between APCN and the PKT architecture against the varying expiration time under different numbers of contents. Fig. 6(a) shows the reduced RTT ratio (recall that the reduced RTT ratio reflects the performance improvement due to the use of APCN compared with the PKT architecture), and Fig. 6(b) depicts the cache hit rate of D_L, which reflects the effect of relieving the access burden on D_S.

From the figure, we find that with the increase in the expiration time, network performance improves, since both the reduced RTT ratio and the cache hit rate increase. The increase in the cache hit rate means that more verify() procedures are performed at the local delegate, D_L, resulting in reduced access to the source delegate, D_S. As the number of content increases, network performance degrades. We will explain this phenomenon and discuss the

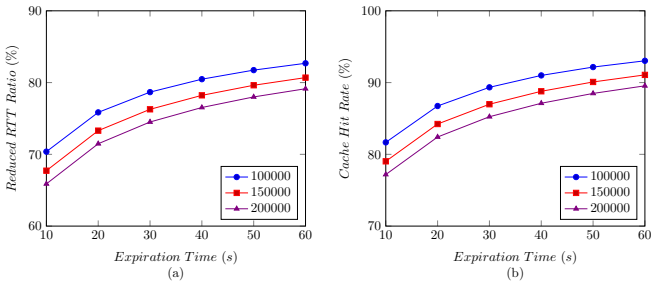

Figure 6: Effects of expiration time of cache entry in delegates: (a) reduced RTT ratio and (b) cache hit rate.

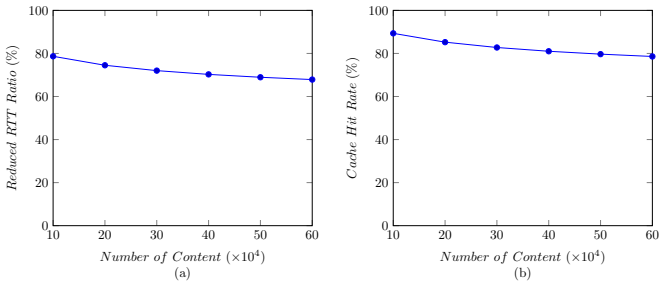

Figure 7: Effects of the number of content: (a) reduced RTT ratio and (b) cache hit rate.

effects of the number of content in Section 7.2. From Fig. 6, we also find that, when the expiration time increases, the performance improvement due to the use of APCN slows down, especially when the expiration time exceeds 30 s because, with the growth of the expiration time, the frequency of retrieving verification information from D_S is reduced, i.e., the negative effects of the expiration time are gradually reduced. This investigation demonstrates that the expiration time of the verification cache entry can be appropriately pre-determined with the given system settings.

7.2. Effects of the Number of Content

In Section 7.1, we compared the effects of the expiration time of the cache entry under different numbers of content (i.e., 100,000, 150,000 and 200,000). It can be observed that there is a slight decrease in the network performance when the number of content increases. To evaluate the effect of the number of content, Fig. 7 depicts the performance metrics against the varying number of content from 100,000 to 600,000. The other parameters are unchanged and are the default values, i.e., the expiration time of the cache entry is 30 s, the parameter of the Zipf distribution is 1.0, and the client request rate is 50 requests/s. We can see that with the increase in the number of content, the performance of APCN compared with the PKT architecture has significant improvement. For example, when the number of content is 300,000, the reduced RTT ratio is 72.02%, and the cache hit rate in the D_L is 82.76%. In addition, the effect of performance improvement due to the use of APCN is gradually reduced. Specifically, when the number of content increases from 300,000 to 600,000, the reduced RTT ratio drops from 72.02% to 67.87%, and the cache hit rate drops from 82.76% to 78.59% because, with the increase in the number of content, the number of times accessing D_S to obtain the verification information also increases.

These results mean that if the number of clients (requesters) and the request rate of each client remains constant, the performance of the proposed APCN relative to the PKT architecture has significant improvement, but the effect of performance improvement is gradually reduced if the number of content increases. In reality, with the increase in the number of content, the number of clients and the request rate should grow rapidly [11]. The merits of the proposed APCN architecture are more apparent if the client's request rate or the number of clients increases (see Section 7.5 and Section 7.6).

7.3. Effects of the Content Popularity

In this section, we consider the Zipf distribution for content popularity (i.e., different distributions of the content request) [38], as it has been widely used in related studies for this purpose. In the Zipf distribution, q denotes the parameter of rank, and α represents the value of the exponent characterizing the distribution. To investigate the effects of content popularity, we set the parameters of the Zipf distribution as follows: $q = 0$ and $\alpha = 0.7$, 0.8, 0.9, 1.0, 1.1, 1.2 and 1.3. With $q = 0$, the increase in α represents the increase in the concentration of the request.

Fig. 8 shows the improvement of APCN compared with the PKT architecture against the varying α when the request rate is 50, and the number of content is 100,000. From the figure, we find that with the increase in the parameter α, the performance of the proposed architecture improves, as the reduced RTT ratio and cache hit rate increase significantly. Specifically, as the value of α increases from 0.7 to 1.3, the reduced RTT ratio increases from 77.57% to 83.39%, and the cache hit rate in D_L increases from 85.21% to 95.52% because more verify() requests can be performed at D_L in the local network before the verification information expires. In other words, the accountability for those contents is promptly verified locally. In the real world, the higher the popularity of the content, the higher the frequency and the larger number of verification requests that are processed at D_L.

7.4. Effects of the Delegate Cache Ability

In this section, we change the cache size in the delegates to assess the impact of caching ability on network performance. The cache ability is defined as follows:

$$Cache\ Ability = \frac{S_{cache}}{N_{content}}$$

where S_{cache} represents the cache size in the delegates, and $N_{content}$ denotes the number of content.

From Fig. 9(a) we see that when the cache ability is more than 50%, the growth of the cache ability does

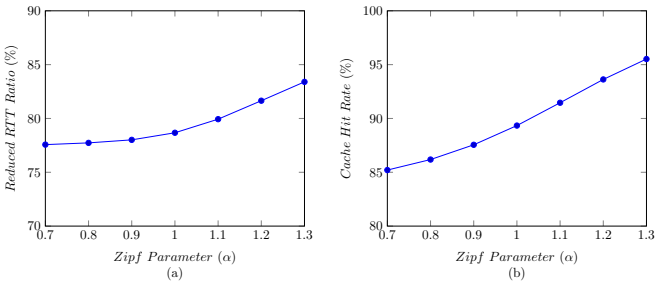

Figure 8: Effects of content popularity: (a) reduced RTT ratio and (b) cache hit rate.

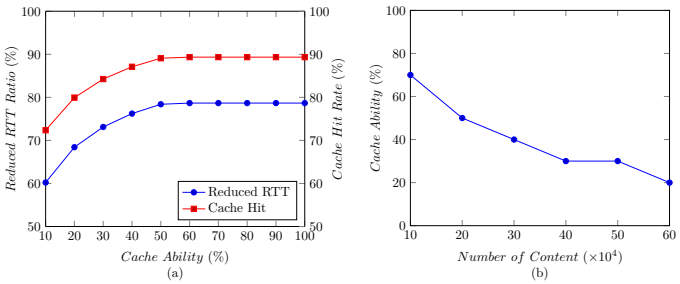

Figure 9: Effects of delegate cache ability: (a) reduced RTT ratio and cache hit rate and (b) requirement of cache ability.

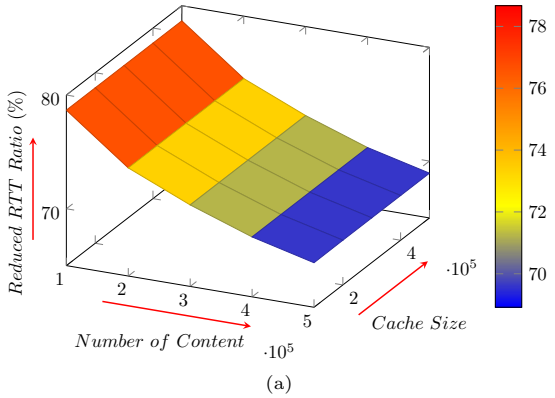

Figure 10: Effects of the number of content and cache size in delegates: (a) reduced RTT ratio and (b) cache hit rate.

not affect the performance improvement. When the cache ability is set as 70% (e.g., the cache size is 70,000 packets) or higher, increasing the cache size no longer affects the results - the reduced RTT ratio is 78.67% and the cache hit rate is 89.34% - because under the current parameter settings, if the cache capacity is less than 70%, the delegates will use the LRU replacement policy to remove some cached verification information to meet the storage limitation. However, when the cache ability of the delegate reaches 70%, the storage space is not a constraint, because the cached verification information will be invalid (expiration) before needing to be deleted for new entries.

In addition, we evaluate the cache ability requirements to deal with the growth of content. Fig. 9(b) depicts the cache ability against the varying number of content. The results show that when the number of content increases, the need for cache ability decreases gradually. This means that the delegate's cache ability growth rate should be lower than the growth rate of the number of content. Reflecting real-world deployment, when the number of content increases dramatically, the storage capacity of the delegates does not need to have the same growth rate.

We also evaluate the effect of the delegate cache ability by simultaneously changing the number of content and the cache size in the delegates. In this evaluation, the number of content, and the cache size in delegates increase from 100,000 to 500,000, respectively. Fig. 10(a) and Fig. 10(b) show that when the cache size is at a high level, continuing to increase the cache size does not significantly increase the reduced RTT ratio and the cache hit rate because in this case, the cache size is sufficient for storing verification information for popular content (content popularity follows the Zipf distribution). For example, when the number of content is 500,000, and the cache size is 100,000, the reduced RTT ratio is 68.92%, and the cache hit rate is 79.65%. When the cache size increases to 200,000 and the number of content remains at 500,000, the reduced RTT ratio is 68.94%, and the cache hit rate is 79.68%. The result reflects the same conclusion as Fig. 9(b), i.e., with the increase in the number of content, the cache ability of the delegates is not required to have the same growth rate to meet the needs of caching verification information.

In addition, from Fig. 10 we also see that along with the increase in the number of content, both the reduced RTT ratio, and the cache hit rate decrease slowly, which is also consistent with the results reflected in Fig. 7.

7.5. Effects of the Request Rate

To evaluate the effect of the request rate, Fig. 11 depicts the reduced RTT ratio and the cache hit rate against the varying request rates of each requester from 25 to 150. We observe that when the request rate increases, the efficiency and advantage of our architecture is more obvious. Specifically, when the request rate is 25 requests/s, the reduced RTT ratio is 74.4%, and the cache hit rate is 84.98%. If the request rate increases to 150 requests/s, the reduced RTT ratio increases to 83.76%, and the cache

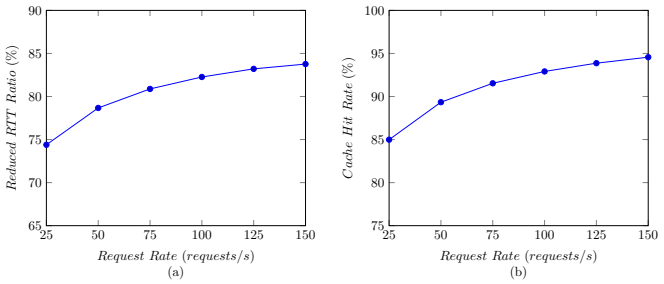

Figure 11: Effects of request rate of requesters: (a) reduced RTT ratio and (b) cache hit rate.

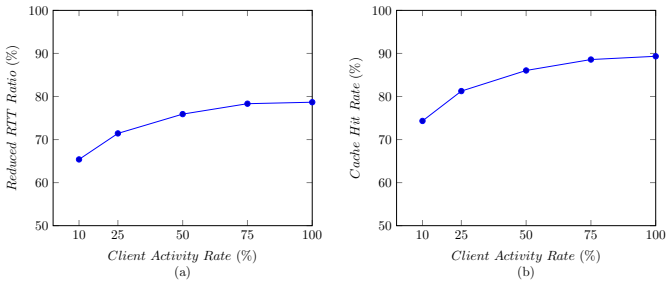

Figure 12: Effects of the number of requesters: (a) reduced RTT and (b) cache hit rate.

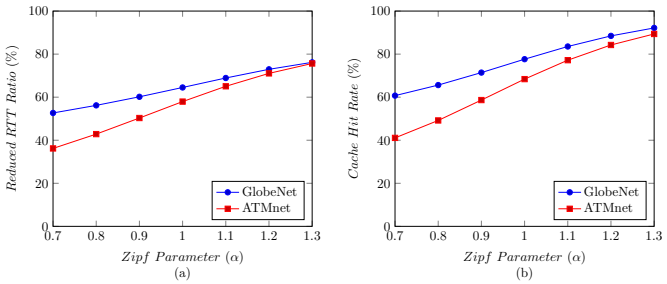

Figure 13: Effects of different topologies and different content popularity: (a) reduced RTT ratio and (b) cache hit rate.

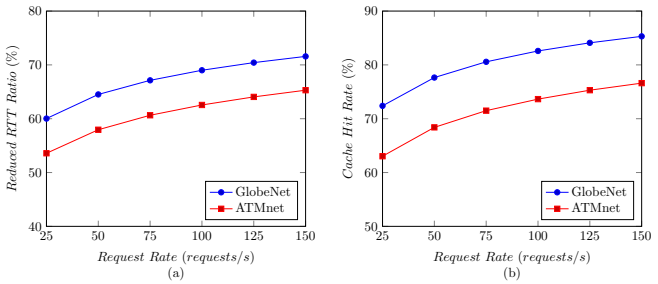

Figure 14: Effects of different topologies and different request rates: (a) reduced RTT ratio and (b) cache hit rate.

hit rate reaches 94.57%. In addition, with the continuously increasing request rate, the performance improvement will stabilize (i.e., the growth becomes slower) because the more frequent the client requests content, the more verifies that will be done in the local delegate. In reality, with the rapid development of the Internet, the frequency of clients requesting content also rapidly increases [11].

7.6. Effects of the Number of Requesters

To evaluate the impact of the number of requesters, in this section we randomly select some requesters from all 345 clients acting as active clients to request contents. At the same time, the other clients do not request content temporarily. This will ensure that we can do this assessment without changing the network topology. The number of active clients divided by the total number of clients is called the client activity rate, i.e.,

$$Client\ Activity\ Rate = \frac{N_{active}}{N_{client}}$$

where N_{active} represents the number of active clients, and N_{client} denotes the number of clients in the network.

In this section, we randomly select 34, 86, 172, 259 and 345 active clients, representing the client activity rate as 10%, 25%, 50%, 75% and 100%, respectively. Fig. 12 shows the performance results. The results demonstrate that with the same network topology, the more requesters that are in the system, the better our architecture performs (compared with the PKT architecture). For example, if the client activity rate is 10%, i.e., there are only 34 nodes requesting content, the reduced RTT ratio is 65.38%, and the cache hit rate is 74.32%. When the client activity rate increases to 50%, i.e., 172 requesters in the network, the reduced RTT ratio increases to 75.91%, and the cache hit rate reaches 89.34%. That means, with the increase in the number of requesters, the performance of the proposed APCN architecture is superior to that of the PKT architecture. In addition, with the constantly increasing number of requesters, the performance improvement of APCN slows down, since the reduced RTT ratio and cache hit rate increase smoothly because along with the increase in the number of requesters, more requests, therefore, target new content that local requesters have not recently requested. As a result, the access to the source delegate (D_S) to obtain the verify results will increase.

We also evaluate the effect of changing network topologies on network performance. We choose two topologies (GlobeNet and ATMnet) from The Internet Topology Zoo [27], where GlobeNet includes 67 nodes, and ATMnet has 21 nodes. The two topologies are used as local networks. Figs. 13 and 14 show the effect of varying content popularities and request rates of the requesters on network performance. From these two figures, we see that as the number of nodes (requesters) increases in the topology, our architecture is more efficient than the PKT architecture (reflected by a higher reduced RTT ratio and increased cache hit rate). This conclusion is consistent with the results shown in Fig. 12. In that evaluation, we did not change the topology and only changed the number of users by randomly selecting different numbers of requesters to act as active clients. In addition, Fig. 13(a) shows that as the value of α increases from 0.7 to 1.3, the reduced RTT ratios for both topologies (i.e., GlobeNet and ATMnet) tend to be identical. The cache hit rate depicted in Fig. 13(b) shows a similar phenomenon. This is because

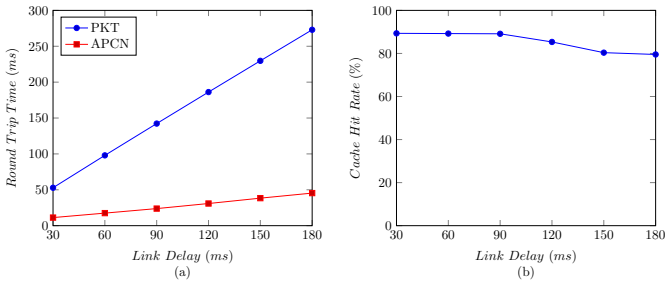

Figure 15: Effects of different distances between D_S and D_L: (a) RTT and (b) cache hit rate.

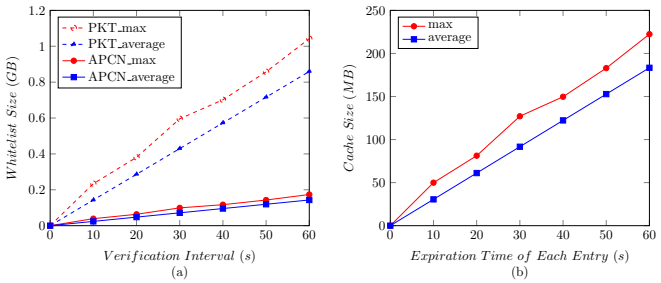

Figure 16: (a) Effects of whitelist size in PKT and APCN, and (b) effects of cache size in D_L.

as α increases, the content requested by the requester will be more concentrated, i.e., more requests are initiated for popular content. As a result, more verification is performed at the local network. Therefore, the scale of the topology has less influence on network performance when α reaches a certain threshold.

7.7. Effects of the Distance Between D_S and D_L

In this section, we investigate the effect of the distance between D_S and D_L on network performance by changing the link delay between D_S and D_L from 30 ms to 180 ms to represent different distances in reality. Fig. 15 depicts the performance metrics against the varying distances between D_S and D_L. Fig. 15(a) shows that with the increase in the distance, the performance improvement due to the use of APCN is more obvious since the difference of RTT between the PKT architecture and the APCN increases significantly. However, with the increase in the distance, the cache hit rate slightly decreases, which is shown in Fig. 15(b), because if the verification information cached in the local delegate expires, it requires a longer time to reacquire the verification information from the remote D_S. During this period, D_L could not provide such a verification. Thus, as the distance between delegates grows, this effect becomes more apparent. For example, if the link delay between the sender and the requester accountability domains is set to 90 ms, the RTT in APCN is reduced by 83.27% compared with that in the PKT architecture. At the same time, 89.13% verification requests will be performed at the local network (i.e., requester accountability domain).

7.8. Effects of the Whitelist Size and Cache Size

For the sake of comparing with the PKT architecture, in this section, we assume one flow transmits one content in the network [9]. This consideration reflects the worst case on the overhead of the proposed APCN and the general case of the PKT architecture (i.e., different flows generate different verification information), since in realistic scenarios multiple flows may transmit the same content. In addition, because both the PKT architecture and the APCN generate fingerprints based on packets, this consideration makes the cost of storage and bandwidth consumption in the Brief() process the same. Recall that each ID (e.g., CID) is 20 bytes since SHA-1 (Secure Hash Algorithm 1) produces 20-byte digests. The proposed APCN only needs to record the CID in the whitelist, but the PKT architecture needs to record two addresses, i.e., the destination address and accountability address (120 bytes) [39]. Thus, the whitelist used in APCN is much smaller than that of the PKT architecture. Fig. 16(a) quantitatively shows the storage cost of the whitelist in the PKT architecture and the APCN where the whitelist size is calculated based on both the maximum and the average number of flows against the given verification interval. From the figure, we can find that when the verification interval is 30 s, the proposed APCN only requires 101 MB to store the whitelist, while the PKT architecture needs 610 MB. When the interval is set to 60 s, 178 MB is sufficient to store the whitelist in APCN, but the PKT architecture requires 1.04 GB.

Caching verification information in D_L is the overhead introduced in APCN compared with the PKT architecture. The costs required to store the verification information in D_L will be investigated as follows. Because CID requires up to 20 bytes, and the fields for verification information and the expiration time require up to 5 bytes [48], each entry cached in D_L requires 25 bytes in storage space. Fig. 16(b) depicts the required cache size against the expiration time of each entry. From the figure, we find that even when the expiration time is set to 60 s, it requires only 222 MB storage space to cache the verification information. If the expiration time is set to 30 s, 127 MB of storage space is sufficient to store all verification information.

In summary, the results in this section have shown that APCN outperforms the PKT architecture in terms of lower RTT and a higher cache hit rate under different network configurations. In our experiments, the RTT in APCN could be reduced by 78.67% compared with the packet-based architecture; meanwhile, 89.34% of the verification could be performed at the local network. In addition, the proposed APCN architecture can reduce the overhead of network storage. In the evaluation, our experiments suggested that when the expiration time of the cache entry is set to 30 s, the size of a key design element, the whitelist, maintained in the verifier dramatically decreased in APCN (101 MB) in contrast to the one (610 MB) needed in the packet-based architecture; moreover, the cache size in the local delegate was maintained at an acceptable level of ap-

proximately 127 MB.

8. Conclusions and Discussions

In this paper, we proposed the first scalable architecture for balancing Accountability and Privacy in large-scale Content-based Networks, called APCN. A separate delegate was adopted to be responsible for the accountability of packets, and meanwhile, protect the privacy of both senders and requesters. In addition, a double-delegate scheme was proposed to improve the performance and alleviate the scalability issue in large-scale networks, and the newly introduced local delegate was used for the privacy protection of requesters. Furthermore, a new content identification was proposed to identify and trace a content. The security of the proposed APCN architecture in the real-world deployment was analysed. The proposed APCN was designed at the network layer, which makes it more effective than designs at other layers.

We developed a discrete-event simulator based on the NS-3 framework and collected the real trace from an Internet service provider to validate the effectiveness and evaluate the performance of the proposed APCN. Extensive simulation experiments were conducted to compare the performance of the proposed APCN with that of the packet-based architecture. It is worth noting that deployment of the APCN does not mean that all network users should use the proposed architecture whenever they use the network. When users need to protect their privacy in content delivery scenarios, they can choose to use the APCN architecture. The proposed architecture can be installed as a protocol in routers and can be incrementally deployed. We will explore other architectures for balancing accountability and privacy based on flow level and service level, respectively. Users can, therefore, choose to use different protocols such as flow-based, content-based, and service-based architectures in different scenarios.

Acknowledgments

This work is partially supported by the National Key R&D Program of China (No. 2017YFB1401500 and No. 2017YFB0801801), the National Science and Technology Major Project of the Ministry of Science and Technology of China (No. 2017ZX03001019), and the National Natural Science Foundation of China (No. 61672490 and No. 61303241).

References

[1] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, B. Ohlman, A survey of information-centric networking, IEEE Communications Magazine 50 (7) (2012) 26–36.

[2] M. Ali, R. Dhamotharan, E. Khan, S. U. Khan, A. V. Vasilakos, K. Li, A. Y. Zomaya, SeDaSC: secure data sharing in clouds, IEEE Systems Journal 11 (2) (2017) 395–404.

[3] D. G. Andersen, H. Balakrishnan, N. Feamster, T. Koponen, D. Moon, S. Shenker, Accountable Internet Protocol (AIP), in: Proc. ACM SIGCOMM, ACM, 339–350, 2008.

[4] A. Bender, N. Spring, D. Levin, B. Bhattacharjee, Accountability as a Service, in: Proc. USENIX SRUTI, USENIX, 5:1–5:6, 2007.

[5] M. Blaze, J. Ioannidis, A. D. Keromytis, Trust Management and Network Layer Security Protocols, in: International Workshop on Security Protocols, Springer, 103–108, 1999.

[6] A. Boukerche, Y. Ren, A trust-based security system for ubiquitous and pervasive computing environments, Computer Communications 31 (18) (2008) 4343–4351.

[7] L. Breslau, P. Cao, L. Fan, G. Phillips, S. Shenker, Web caching and Zipf-like distributions: Evidence and implications, in: Proc. IEEE INFOCOM, IEEE, 126–134, 1999.

[8] A. Carzaniga, M. J. Rutherford, A. L. Wolf, A routing scheme for content-based networking, in: Proc. IEEE INFOCOM, IEEE, 918–928, 2004.

[9] A. Chanda, C. Westphal, D. Raychaudhuri, Content based traffic engineering in software defined information centric networks, in: Prof. IEEE INFOCOM WKSHPS, IEEE, 357–362, 2013.

[10] Y.-R. Chen, A. Rezapour, W.-G. Tzeng, Privacy-preserving ridge regression on distributed data, Information Sciences 451 (2018) 34–49.

[11] Cisco, Cisco Visual Networking Index: Forecast and Methodology, 2016-2021, 2017.

[12] D. D. Clark, J. Wroclawski, K. R. Sollins, R. Braden, Tussle in cyberspace: defining tomorrow's Internet, IEEE/ACM Transactions on Networking (TON) 13 (3) (2005) 462–475.

[13] H.-N. Dai, R. C.-W. Wong, H. Wang, On capacity and delay of Multichannel wireless networks with infrastructure support, IEEE Transactions on Vehicular Technology 66 (2) (2017) 1589–1604.

[14] R. Dingledine, N. Mathewson, P. Syverson, Tor: The second-generation onion router, in: Proc. USENIX SSYM, USENIX, 12–21, 2004.

[15] S. K. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. Maggs, K. Ng, V. Sekar, S. Shenker, Less pain, most of the gain: Incrementally deployable ICN, in: Prof. ACM SIGCOMM, ACM, 147–158, 2013.

[16] S. B. Fred, T. Bonald, A. Proutiere, G. Régnié, J. W. Roberts, Statistical bandwidth sharing: a study of congestion at flow level, in: Proc. ACM SIGCOMM, ACM, 111–122, 2001.

[17] J. Ghaderi, R. Srikant, Towards a theory of anonymous networking, in: Proc. IEEE INFOCOM, IEEE, 686–649, 2010.

[18] A. Ghodsi, T. Koponen, J. Rajahalme, P. Sarolahti, S. Shenker, Naming in content-oriented architectures, in: Proc. ACM ICN, ACM, 1–6, 2011.

[19] D. Han, A. Anand, F. Dogar, B. Li, H. Lim, M. Machado, A. Mukundan, W. Wu, A. Akella, D. G. Andersen, et al., XIA: Efficient support for evolvable internetworking, in: Proc. USENIX NSDI, USENIX, 309–322, 2012.

[20] F. Hao, S. Li, G. Min, H.-C. Kim, S. S. Yau, L. T. Yang, An efficient approach to generating location-sensitive recommendations in ad-hoc social network environments, IEEE Transactions on Services Computing 8 (3) (2015) 520–533.

[21] T. R. Henderson, M. Lacage, G. F. Riley, C. Dowell, J. Kopena, Network Simulations with the ns-3 Simulator, SIGCOMM demonstration .

[22] H.-C. Hsiao, T. H.-J. Kim, A. Perrig, A. Yamada, S. C. Nelson, M. Gruteser, W. Meng, LAP: Lightweight anonymity and privacy, in: Proc. IEEE S&P, IEEE, 506–520, 2012.

[23] R. Hummen, H. Wirtz, J. H. Ziegeldorf, J. Hiller, K. Wehrle, Tailoring end-to-end IP security protocols to the Internet of Things, in: Proc. IEEE ICNP, IEEE, 1–10, 2013.

[24] M. U. Ilyas, M. Z. Shafiq, A. X. Liu, H. Radha, Who are you talking to? Breaching privacy in encrypted IM networks, in: Proc. IEEE ICNP, IEEE, 1–10, 2013.

[25] T. Isdal, M. Piatek, A. Krishnamurthy, T. Anderson, Privacy-preserving P2P data sharing with OneSwarm, in: Proc. ACM SIGCOMM, ACM, 111–122, 2010.

[26] X. Jiang, J. Bi, nCDN: CDN Enhanced with NDN, in: Prof. IEEE INFOCOM WKSHPS, IEEE, 440–445, 2014.
[27] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, M. Roughan, The Internet Topology Zoo, IEEE Journal on Selected Areas in Communications 29 (9) (2011) 1765–1775.
[28] R. Küsters, T. Truderung, A. Vogt, Accountability: definition and relationship to verifiability, in: Proc. ACM CCS, ACM, 526–535, 2010.
[29] J. Lai, Y. Mu, F. Guo, P. Jiang, W. Susilo, Privacy-enhanced attribute-based private information retrieval, Information Sciences 454 (2018) 275–291.
[30] X. Li, Q. Wang, H.-N. Dai, H. Wang, A Novel Friendly Jamming Scheme in Industrial Crowdsensing Networks against Eavesdropping Attack, Sensors 18 (6).
[31] C. Liu, S. Zhou, H. Hu, Y. Tang, J. Guan, Y. Ma, CPP: Towards comprehensive privacy preserving for query processing in information networks, Information Sciences 467 (2018) 296–311.
[32] W. Lou, K. Ren, Security, privacy, and accountability in wireless access networks, IEEE Wireless Communications 16 (4) (2009) 80–87.
[33] K. Lu, Y. Qian, H.-H. Chen, A secure and service-oriented network control framework for WiMAX networks, IEEE Communications Magazine 45 (5) (2007) 124–130.
[34] H. Luo, J. Kong, P. Zerfos, S. Lu, L. Zhang, URSA: ubiquitous and robust access control for mobile ad hoc networks, IEEE/ACM Transactions on Networking (TON) 12 (6) (2004) 1049–1063.
[35] X. Ma, L. T. Yang, Y. Xiang, W. K. Zeng, D. Zou, H. Jin, Fully reversible privacy region protection for cloud video surveillance, IEEE Transactions on Cloud Computing 5 (3) (2017) 510–522.
[36] A. Majumder, N. Shrivastava, R. Rastogi, A. Srinivasan, Scalable content-based routing in pub/sub systems, in: Proc. IEEE INFOCOM, IEEE, 567–575, 2009.
[37] D. Mazieres, M. Kaminsky, M. F. Kaashoek, E. Witchel, Separating key management from file system security, in: Proc. ACM SOSP, ACM, 124–139, 1999.
[38] L. Muscariello, G. Carofiglio, M. Gallo, Bandwidth and storage sharing performance in information centric networking, in: Proceedings of the ACM SIGCOMM workshop on Information-centric networking, ACM, 26–31, 2011.
[39] D. Naylor, M. K. Mukerjee, P. Steenkiste, Balancing accountability and privacy in the network, in: Proc. ACM SIGCOMM, ACM, 75–86, 2014.
[40] P. Nikander, R. Moskowitz, Host Identity Protocol (HIP) Architecture, RFC 4423, 2006.
[41] J. Rajahalme, A. Conta, B. Carpenter, S. Deering, IPv6 Flow Label Specification, RFC 3697, 2004.
[42] G. Regnier, S. Makineni, I. Illikkal, R. Iyer, D. Minturn, R. Huggahalli, D. Newell, L. Cline, A. Foong, TCP onloading for data center servers, Computer 37 (11) (2004) 48–58.
[43] J. Ren, J. Wu, Survey on anonymous communications in computer networks, Computer Communications 33 (4) (2010) 420–431.
[44] S. Sevilla, J. Garcia-Luna-Aceves, Freeing the IP Internet architecture from fixed IP addresses, in: Proc. IEEE ICNP, IEEE, 345–355, 2015.
[45] M. Sung, J. Xu, J. Li, L. Li, Large-Scale IP Traceback in High-Speed Internet: Practical Techniques and Information-Theoretic Foundation, IEEE/ACM Transactions on Networking (TON) 16 (6) (2008) 1253–1266.
[46] A. Venkataramani, J. F. Kurose, D. Raychaudhuri, K. Nagaraja, M. Mao, S. Banerjee, Mobilityfirst: a mobility-centric and trustworthy Internet architecture, ACM SIGCOMM Computer Communication Review 44 (3) (2014) 74–80.
[47] N. Wang, J. Fu, J. Zeng, B. K. Bhargava, Source-location privacy full protection in wireless sensor networks, Information Sciences 444 (2018) 105–121.
[48] L. Wu, R. J. Barker, M. A. Kim, K. A. Ross, Hardware partitioning for big data analytics, IEEE Micro 34 (3) (2014) 109–119.
[49] X. Yang, D. Wetherall, T. Anderson, A DoS-limiting network architecture, in: Prof. ACM SIGCOMM, ACM, 241–252, 2005.
[50] G. Zhang, Y. Yang, J. Chen, A historical probability based noise generation strategy for privacy protection in cloud computing, Journal of Computer and System Sciences 78 (5) (2012) 1374–1381.