Yaoqi Jia*, Guangdong Bai, Prateek Saxena, and Zhenkai Liang

Anonymity in Peer-assisted CDNs: Inference Attacks and Mitigation

Abstract: The peer-assisted CDN is a new content distribution paradigm supported by CDNs (e.g., Akamai), which enables clients to cache and distribute web content on behalf of a website. Peer-assisted CDNs bring significant bandwidth savings to website operators and reduce network latency for users. In this work, we show that the current designs of peer-assisted CDNs expose clients to privacy-invasive attacks, enabling one client to infer the set of browsed resources of another client. To alleviate this, we propose an *anonymous peer-assisted CDN (*APAC*)*, which employs content delivery while providing *initiator anonymity* (i.e., hiding who sends the resource request) and *responder anonymity* (i.e., hiding who responds to the request) for peers. APAC can be a web service, compatible with current browsers and requiring no client-side changes. Our anonymity analysis shows that our APAC design can preserve a higher level of anonymity than state-of-the-art peer-assisted CDNs. In addition, our evaluation demonstrates that APAC can achieve desired performance gains.

Keywords: Peer-assisted CDNs, Anonymity, Inference Attacks

DOI 10.1515/popets-2016-0041
Received 2016-02-29; revised 2016-06-02; accepted 2016-06-02.

1 Introduction

Content Delivery Networks (CDNs) were introduced a decade ago. To distribute data to end users in a fast way, CDN operators (e.g., Akamai [1] and CloudFlare [4]) assist sites to deliver content to end users with their multiple data centers across the world. Complementary to these infrastructure-based approaches, numerous CDNs have adopted peer-to-peer techniques to distribute content, e.g., Swarmify [20], PeerCDN [17] Akamai NetSession [2], Squirrel [50], CoralCDN [44, 45], FlowerCDN [43] and Maygh [76].

On one hand, involving end users as peers (client-side CDNs) to distribute data in peer-assisted CDNs reduces the fetching time of resources and saves the bandwidth of CDNs' servers. For instance, Akamai NetSession can offload over 70% of the traffic to peers with high reliability [77]. On the other hand, in contrast to infrastructure-based systems in which trusted centralized servers are deployed, untrusted peers in peer-assisted CDNs can easily join the system. The compromised or malicious peers can then a) modify content and inject unauthorized content; b) delay or deny content delivery to other peers; c) misreport their contributions to manipulate the accounting for commercial services [29, 77]; d) infer which peer they deliver/fetch a resource to/from, and what content the peers have requested.

In practice, peer-assisted CDNs introduce various measures to tackle these security issues. For example, to protect the authenticity and integrity of the content in peers, FireCoral introduces the *signing service* to authenticate content, and peers can verify the content with the hash information supplied by the *tracker* [69]. In addition, Aditya *et al.* proposed RCA, a reliable client accounting system for NetSession to discover all misreportings and protocol violations by faulty or malicious clients and quarantine these potentially colluding clients [29]. However, privacy leakage in peer-assisted CDNs has not been actively studied yet, and anonymity is also seldom considered by existing peer-assisted CDNs.

Although the majority of resources (e.g., images) served on CDNs are publicly accessible for all users, a user's online activities (e.g., visiting personalized web pages and fetching relevant resources) have been proven to be privacy-sensitive. For example, a user's digital identity can be revealed when visiting social network websites [72]; visiting map service/political websites reflects a user's geolocation/political orientation [52]. Revealing a user's browsing history will significantly leak the user's privacy. To demonstrate this threat, we present *inference attacks* on peer-assisted CDNs: by placing controlled peers and observing the requested contents in the system, the adversary can effectively infer content-access activities of benign peers.

To demonstrate the effectiveness of the proposed inference attacks, we conduct inference attacks against widely used peer-assisted CDNs, including Swarmify, BemTV and P2PSP.

***Corresponding Author: Yaoqi Jia:** National University of Singapore, E-mail: jiayaoqi@comp.nus.edu.sg
Guangdong Bai: National University of Singapore, E-mail: baiguangdong@comp.nus.edu.sg
Prateek Saxena: National University of Singapore, E-mail: prateeks@comp.nus.edu.sg
Zhenkai Liang: National University of Singapore, E-mail: liangzk@comp.nus.edu.sg

From our experiments, we find that in any of these systems, when a peer (i.e., an initiator) sends a request for a specific resource, the server assists the initiator to directly fetch the content from another nearby peer (i.e., the responder) based on locality, without any mechanism to conceal the initiator's or the responder's identity. Therefore, the adversary who controls a number of peers is capable of mounting inference attacks on existing peer-assisted CDNs to identify the initiator or responder of a forwarded request. Through the observed communication, the adversary can infer the user's browsing history and preferences. The adversary can then use this information for spear phishing, personal targeted advertisements, and social engineering attacks [54].

The primary goal of peer-assisted CDNs is to save the server's bandwidth and reduce client-side network latency. It is quite challenging to conceal peers' identities (IP addresses) to mitigate inference attacks as well as to preserve reasonable responsiveness for the deployed site. On the other hand, although anonymous mechanisms have been well studied in communication systems [8, 11, 21, 38, 41, 46, 57, 60, 64, 65], their previous applications rarely impose stringent performance demands such as in a real-time caching system. Their primary goal (e.g., onion routing [64], Tor [41], Mixminion [38] and Crowds [65]) is to preserve high level of anonymity. Directly adopting these approaches may introduce non-negligible performance overhead for clients, e.g., the client-side communication overhead for circuit setup in onion routing-based and Crowds-based systems. For instance, Mixminion can introduce several-hour latency for message transmission [38].

To address these challenges, we develop an *Anonymous Peer-assisted CDN (*APAC*)* for web applications. APAC involves client-side browsers as peers to distribute content without requiring any software installation for users. Once users visit the deployed site, APAC's client-side code will assist their browsers to join APAC as peers and serve resources to other users. APAC is compatible with mainstream browsers and requires negligible modification on the deployed websites. In practice, the adversarial peers can exist uniformly in the system or densely surround the victim. To achieve an adequate level of anonymity in different scenarios, we introduce a new *region-based circuit selection algorithm* to construct a path (denoted as *circuit*) consisting of peers. Our APAC encapsulates the request with standard layered encryption and the initiator can fetch the content via the constructed circuit.

Our goal is not to build a perfect anonymity-preserving system, as it may introduce huge performance overhead, but we aim to enable the system to make trade-off between performance and anonymity, which is significantly beyond that of current peer-assisted CDNs. Using a standard measure of *degree of anonymity* [40], we show that APAC can preserve a high degree of initiator/responder anonymity (i.e., at 0.8 degree of anonymity recommended for anonymous communication systems by Diaz *et al.* [40]), with only two intermediate peers for a circuit even if 35% of all peers are under the adversary's control. Our *locality-aware* APAC is tunable to select intermediate nodes nearby the initiator/responder, which reduces the resource fetching time (network latency) for peers. Our evaluation in a city-wide network shows that with two intermediate peers for a circuit, APAC can reduce 44.1% client-side network latency and save 97.3% server-side bandwidth when fetching 2 MB[1] resources via peers, with 0.8 degree of anonymity.

Contributions. Compared to regular CDN services, APAC saves the bandwidth of the CDN's edge server, and reduces the latency for peer when the edge server and peers are not in the same city. In contrast to single-hop peer-assisted CDNs such as Swarmify, APAC provides an adequate level of anonymity for peers, but it trades off the performance, e.g., network latency. In summary, we provide the following contributions:

- We systematically analyze inference attacks on real-world services, i.e., Swarmify, BemTV and P2PSP.
- We develop an anonymous peer-assisted CDN (APAC) for web applications, which involves browsers as peers to distribute content. From our analysis, APAC can preserve high level of initiator/responder anonymity even if 35% peers are compromised.
- APAC is compatible with current browsers, and requires no client-side installation. Our evaluation demonstrates that APAC can bring desired network latency reduction for peers and bandwidth savings for deployed sites. APAC can customize and balance between three considerations: anonymity, performance and compatibility with browsers.

2 Motivation & Problem Statement

2.1 Background of Peer-assisted CDNs

Conventional CDN operators, e.g., Akamai [1] and CloudFlare [4], distribute website contents around the world-wide *edge* servers for faster loading of content resources. CDN operators can increase the geo-density of edge servers up to a point (e.g., placing edge servers in major cities of various

1 Since the averaged total size of transferred data when loading a site is 2268 KB [22], we use the fetching time of 2 MB resources as the representative.

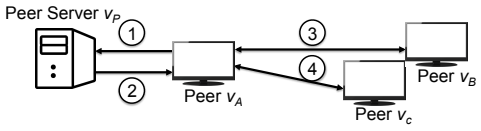

Fig. 1. Illustration of content delivery in current peer-assisted CDNs. Peer v_A sends requests to the peer server for resources R_1 and R_2 (in ①). The server responds to v_A with a list of peers, e.g., v_B and v_C having R_1 and R_2 respectively (in ②). v_A connects with v_B/v_C, and fetches resources from them (in ③ ④).

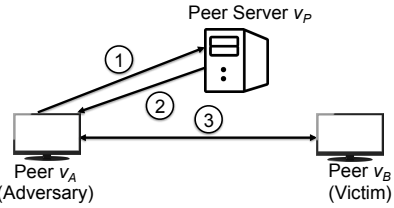

Fig. 2. Illustration of an inference attack to infer the responder of a request. The adversarial peer v_A sends a request for a resource R to the peer server (in ①). Based on locality, the server replies v_A with the nearby victim v_B having R (in ②). v_A fetches R from v_B (in ③), thus the adversary infers that the victim has viewed R recently.

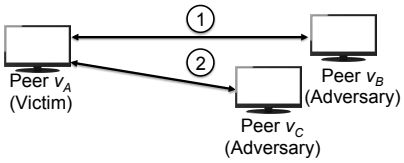

Fig. 3. Illustration of an inference attack to infer the initiator of a request. The user v_A fetches resources from peer v_B and v_C which are controlled by the adversary (in ① ②). Therefore, by passively observing the requests from the victim, the adversary can determine that the victim is looking for (interested in) the requested content.

countries), but fetching resources for users far away from an edge server has major latency.

To address this challenge, various peer-assisted CDNs have been proposed utilizing proxies or client-side software (e.g., browsers) as peers [2, 15, 17, 20, 25, 29, 43–45, 50, 69, 70, 73, 76]. Previous studies have shown advantages of peer-assisted CDNs: they can significantly reduce the cost of all parties including Internet Service Providers (ISPs) [48, 53]. For example, NetSession has over 25 million users in 239 countries and territories, and offloads 70−80% of the traffic to peers without the trade-off of reliability [77]. In peer-assisted CDNs, the peers are the clients that fetch and distribute content, e.g., browsers in Maygh [76] and client programs in NetSession [2]. In such design, a peer server coordinates peers to distribute content. As Figure 1 illustrates, after communicating with the peer server, peers can fetch content from other nodes.To reduce the network latency, the peer server typically assigns the initiator a peer who has the requested resource as the responder, based on the distance between their IP addresses or geolocations (called a *locality-aware* peer selection algorithm). Alternatively, the server directly responds with a list of peers having the requested content, the initiator will contact these peers in parallel. After receiving the content, the initiator can serve it to other peers.

2.2 Inference Attacks & Real-world Examples

As shown in Figure 1, a typical peer-assisted CDN does not conceal the initiator's/responder's identity (IP address) for each request. By observing the forwarded requests from the controlled peers in the system, the adversary can effectively infer the initiator/responder of each request, and further infer the victim's browsing history and preferences (as shown in Figure 2 and 3). We term such attacks as *inference attacks*. In an inference attack, when any of the adversarial peers is the responder/initiator of a request, the adversary can definitely determine which peer is the initiator/responder of the request. By profiling a user's browsing history and preferences with the inference attack, the adversary can infer the victim's digital identity [72] and precise geolocation [52], as well as further abuse the sensitive information for spear phishing, personally targeted advertisements, or even social engineering attacks [54].

Inference attacks in peer-assisted CDNs have not been carefully studied. We analyze inference attacks on three real-world services, including Swarmify, BemTV and P2PSP, to show the prevalence and effectiveness of such attacks[2].

Swarmify. Swarmify [20] assists the deployed site to deliver content to users from other peers based on locality. It requires sites to include a service-specific library based on WebRTC [26] and deploy optional changes on resources. Although all communications between peers in Swarmify are encrypted, our study demonstrates that it does not guarantee anonymity for peers.

We have mounted an inference attack as follows. We first deployed Swarmify on a website and set 10 images and 2 videos as the targeted resources. For the first time, we launched a Chrome browser as the victim's peer. The victim's peer randomly fetched part of resources (unknown to the attacker) from the remote server. To infer what content the victim has requested, we located the adversarial node nearby the victim's peer in the same local area network (LAN), and used Wireshark [28] to eavesdrop on the network traffic from/to the con-

2 Previous work [76, 77] mentions that peers can learn the IP addresses of the connect peers in numerous peer-assisted CDNs, e.g., NetSession, FireCoral, FlowerCDN and Maygh. Thus, these services are conceptually vulnerable to inference attacks too.

trolled node. When the malicious peer requested all resources, we clearly observed that the peer server replied with the victim's IP address and instructed the malicious node to fetch particular resources from the victim's peer. Therefore, the attacker can infer the responder's identity (the victim's IP address) and what content the responder holds. For the second time, the adversarial peer first fetched and buffered all content from the site. When the victim's peer requested several resources, the controlled peer was instructed to distribute the particular resources to the victim's peer. Observing the traffic from the controlled peer, the adversary identified the victim's IP address and the requested resources. Hence we conclude that Swarmify is vulnerable to inference attacks.

BemTV & P2PSP. BemTV [3] is a hybrid CDN and P2P infrastructure for streaming live videos over HTTP, which is also built upon WebRTC with the aim of utilizing clients' web browsers to relay the streamed media files. BemTV requires additional setup on the server side to manage connections between peers, but this is completely transparent to peers.

We have tested BemTV to live-stream a sample video that had been accessed by several computers located within a university network infrastructure. While streaming out the media files, we observed the incoming and outgoing traffic of requests from a computer that acted as an adversary. For each media file, the attacker was able to figure out which specific users (determined by the mapping of IP address) fetched/delivered the media content from/to the adversarial peer. Based on this information, the adversary is capable of determining who is the initiator/responder and further infer the exact video that the victim is watching. In addition, it turns out that BemTV does not guarantee the content integrity between peers in the network, which opens up possibilities for content pollution attacks [47, 51, 61]. Besides BemTV, we have also carried out inference attacks on another video streaming service called P2PSP [16]. We find that P2PSP suffers the same anonymity issues as BemTV does.

Key findings. Our analysis (in Appendix B) reveals that existing peer-assisted CDNs (including Swarmify, BemTV and P2PSP) cannot preserve an adequate level of anonymity when over 20% peers in the network are malicious. Our analysis is conservative, as we assume that the peer-assisted CDN's server *randomly* assigns a peer having the requested resource as the responder for the request. In reality, if the CDN is locality-aware and instructs the peers to fetch content from nearby peers, it is even easier for the adversary to identify the peers near her controlled peers. Hence the existing peer-assisted CDNs provide much worse anonymity for users than that in our conservative analysis.

2.3 Problem Statement

In this paper, for a particular message (i.e., resource request), we call the peer who initiates the request the *initiator*, and call the peer who responds the request the *responder*. We define *initiator/responder anonymity* to mean that the adversary cannot identify the initiator/responder among peers for a resource request. The assumptions on the adversary are:

- *Internal*: The adversary controls some of peers, which are part of the system and can observe the information about forwarded packets.
- *Partial*: The adversary controls a limited number of peers (e.g., a fraction f), and cannot perform any traffic analysis on the rest of the system.
- *Non-adaptive*: The adversary places nodes first, and then the system constructs a circuit for the request. Once the request is in progress, the adversary cannot alter the placement of peers.

We consider an honest-but-curious adversary, which follows the protocol and places nodes randomly in the network or nearby a victim to increase their chance to determine whether the victim initiates/responds a request. We assume that existing accounting mechanisms [29] can be deployed to discover protocol-violating peers, e.g., massively sending or delaying/-denying content delivery requests [29]. Sybil attacks [39, 42, 74] and denial-of-service attacks (DOS) [63] are out of scope in this paper.

The adversary's goal is to identify the initiator/responder of a request with high probability. At the beginning of the request, any benign peer is indistinguishable as the initiator/responder. When the request is in progress, the adversarial peers may be chosen as hops for the request. Hence the adversary can make some observations of the request and gain more knowledge to infer the initiator/responder with higher probability. An anonymous system is required to guarantee that the adversary's observations give minimal *advantage* to determine the initiator/responder. Such an adversary's advantage can be quantified as an entropy metric — the amount of uncertainty in determining the initiator/responder of a request to be a specific victim node. Considering the initiator anonymity, we define the system's entropy as:

Definition 2.1. *Given a request in the system, where Ψ is the set of peers, and p_u is the probability that the peer u is the initiator of the request, the entropy $H(I)$ for the system is defined by:*

$$H(I) = -\sum_{u \in \Psi} p_u \log_2 (p_u) \tag{1}$$

If the adversary has no a priori information on the request, the system preserves the maximum entropy H_M. With some

observations, e.g., the responder is adversarial and monitors the forwarded packets, the adversary has higher probability to infer the initiator, as well as the system's entropy decreases. Let O be the set of all observations for the adversary. When an observation $o \in O$ occurs with the probability $P(o)$, the corresponding entropy is $H(I|o)$.

Definition 2.2. *The conditional entropy of the system on observing O is defined by:*

$$H(I|O) = \sum_{o \in O} P(o) \cdot H(I|o) \tag{2}$$

The advantage gained by the adversary with observation O is the difference in the entropy before and after O, that is: $H_M - H(I|O)$. The degree of anonymity is defined as the normalized value of this difference in certainty of the adversary's guesses about a victim being the initiator:

Definition 2.3. *The degree of initiator anonymity provided by a system is defined by:*

$$D(I|O) = 1 - \frac{H_M - H(I|O)}{H_M} = \frac{H(I|O)}{H_M} \tag{3}$$

The larger $D(I|O)$ is, the higher level of anonymity a system provides. $D(I|O) = 0$ or $D(R|O) = 0$ means absolutely no anonymity, i.e., the adversary knows with 100% the initiator or responder of a request; When the initiator or responder is not identifiable among all peers, $D(I|O) = D(R|O) = 1$. To preserve an adequate level of anonymity, the degree of anonymity for a system is at least ϵ ($\epsilon = min\{D(I|O)\}$). In this paper, we set ϵ as 0.8, which is suggested by a previous study [40]. Hence, if $D(I|O)$ and $D(R|O)$ in a system are over 0.8, we consider the system preserve an adequate level of initiator/responder anonymity. Analogous to $H(I|O)$ and $D(I|O)$, we define $H(R|O)$ and $D(R|O)$ to quantify the responder anonymity in a system.

3 Anonymous Peer-assisted CDN

In this section, we present our anonymous CDN system APAC that provides protections against inference attacks as well as balances the performance overhead. To build practical anonymous peer-assisted CDNs, we have three goals below.

Anonymity: In a peer-assisted CDN, the adversary may control a fraction f of peers. The adversarial peers can be uniformly scattered in the network or densely surround the victim. When a benign peer sends/responds a request to another peer to fetch/deliver a resource, our system should conceal its identity and the linkability to the requested resource. Therefore, other peers do not know who is the initiator/responder of the request.

Performance: Peer-assisted CDNs are designed to assist customers (e.g., sites) to save bandwidth of servers and reduce latency of delivering content to users. Our system should not sacrifice this merit. Therefore, one important goal for us is to balance performance and anonymity.

Compatibility: Our design should introduce no (or minor) changes on websites and clients, such that our system can be easily deployed on various web applications and is user-friendly. Compared with other peer-assisted CDNs (e.g., NetSession) that require the end users to install standalone software, our system can attract more users to join as peers for content delivery.

3.1 Design of APAC

Overview. APAC consists of two primary components: a peer server run by the site operator, and the client-side code implemented in JavaScript and executed in each peer, i.e., a user's web browser. Independent of the site's content server (or the CDN's edge server), the peer server does not directly deliver content, but maintains the connections with peers and coordinates peers to fetch data either from the content server or other peers. At the beginning, when peers request resources, the peer server instructs them to retrieve the content from the content server. The peers who have retrieved the content will store it and be ready to distribute it. When the total number of joined peers and the number of peers having resources are sufficient to preserve the required anonymity based on proper configurations, the peer server will construct circuits for new requests. The server chooses nodes based on our selection algorithm, and arranges them into a path (or circuit), through which the request and content will be transmitted. Then the initiators of requests fetch content from other peers via the circuits. After receiving the content from other peers, peers first verify its integrity, then store and serve it to other peers. We detail the functionalities and implementation in Section 4. As Figure 4 illustrates, analogous to current peer-assisted CDNs, APAC can be deployed for different edge (content) servers, and serves users far from the nearest edge server. We show the performance of APAC in a city-wide scale in Section 5. Next, we demonstrate how APAC achieves the three design goals.

Anonymity: We introduce a new *region-based circuit selection algorithm* that APAC's peer server uses to construct circuits for requests. A circuit consists of three categories of nodes: nodes nearby the initiator and the responder, and nodes chosen globally (not included in the first two categories). To provide initiator/responder anonymity, APAC's peer server

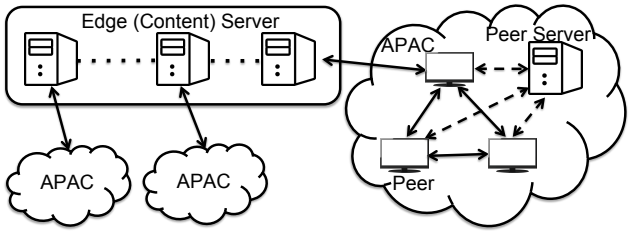

Fig. 4. The overview for the deployment of APAC.

communicates with peers and constructs a circuit for each requests, instead of letting the initiator to construct the circuit in conventional onion routing-based approaches, which only preserve initiator anonymity. Each request in APAC is encrypted as a layered encryption packet with the keys of selected nodes for the circuit in the peer server, and is transferred through the circuit. In this way, the initiator and responder know what resource is requested, but cannot identify each other. Any intermediate node only knows its predecessor and successor, but does not know the transmitted content, or determine which peer is the initiator or responder. In contrast to the current peer-assisted CDNs, in which controlling one node in a request is enough to identify the initiator/responder, it is difficult for the adversary to infer the initiator when only controlling the responder or several intermediate peers. To preserve higher level of anonymity, APAC can set a longer length for the circuit. APAC can also be adjusted to select all intermediate nodes globally to avoid choosing disproportionate number of adversarial peers.

For a given circuit, by controlling the first relay, the responder and at least half of the intermediate peers alternatively in the circuit, the adversary can determine the initiator [75]. We show that an adversarial placement of peers (e.g., placing more peers nearby the victim) does not give significant advantage over a randomized placement by an honest-but-curious adversary (details in Appendix A). Even in the analysis of the worst case that the adversary can learn the distance between any two controlled peers, our system can still preserve an adequate level of anonymity with proper configurations (details in Section 3.4.1).

Performance: We design APAC to be *locality-aware* to achieve good performance. With APAC's peer server, we can avoid the non-negligible overhead for the circuit setup on the client. Furthermore, based on the locality information of all peers maintained by the server, we utilize the *region-based circuit selection algorithm* to balance the anonymity and performance. Instead of randomly choosing intermediate peers, our algorithm can reduce the network latency by selecting peers nearby the initiator/responder. By adjusting the maximum length of the circuit and the distribution factors (controlling the number of intermediate nodes in each region), APAC can provide significant network latency reduction and bandwidth savings as well as preserve a high degree of anonymity. We discuss the design of circuit construction in the next section, and evaluate the performance in Section 5.

Algorithm 1: Region-based Circuit Selection Algorithm in APAC

input : $N-$ number of peers, $v_{init}-$ initiator, $v_{res}-$ responder, $R-$ requested resource, $L_{max}-$ the maximum number of intermediate nodes (or relays), α_{init}, $\alpha_{res}(0 \leq \alpha_{init} + \alpha_{res} \leq 1)-$ distribution factors, $N_{init}-$ number of peers nearest to v_{init}, $N_{res}-$ number of peers nearest to v_{res}, $\mathbb{S}_R-$ set of peers having R, $N_R = |\mathbb{S}_R|$, $G-$ topology graph of peers

output : $cir-$selected circuit

$v_{res} \leftarrow randomSelect(\mathbb{S}_R)$
$l \leftarrow random(1, L_{max})$
$cir_{init} \leftarrow \langle\rangle$; $cir_{im} \leftarrow \langle\rangle$; $cir_{res} \leftarrow \langle\rangle$
if $\lfloor \alpha_{init} l \rfloor \geq 1$ **then**
  $G' \leftarrow removeFrom(G, \{v_{res}\})$
  $\mathbb{S}_{init} \leftarrow selectNearest(G', v_{init}, N_{init})$
  $cir_{init} \leftarrow randomSelectNodes(\mathbb{S}_{init}, \lfloor \alpha_{init} l \rfloor)$
if $\lfloor \alpha_{res} l \rfloor \geq 1$ **then**
  $G' \leftarrow removeFrom(G, \mathbb{S}_{init})$
  $\mathbb{S}_{res} \leftarrow selectNearest(G', v_{res}, N_{res})$
  $cir_{res} \leftarrow randomSelectNodes(\mathbb{S}_{res}, \lfloor \alpha_{res} l \rfloor)$
if $l - \lfloor \alpha_{init} l \rfloor - \lfloor \alpha_{res} l \rfloor \geq 1$ **then**
  $G' \leftarrow removeFrom(G, \mathbb{S}_{init} \cup \mathbb{S}_{res})$
  $\mathbb{S}_{im} \leftarrow getNodes(G')$
  $cir_{im} \leftarrow$
    $randomSelectNodes(\mathbb{S}_{im}, l - \lfloor \alpha_{init} l \rfloor - \lfloor \alpha_{res} l \rfloor)$
$cir \leftarrow concatenate(v_{init}, cir_{init}, cir_{im}, cir_{res}, v_{res})$
return cir

Compatibility: We provide a web overlay with WebRTC [26] supported by mainstream browsers (e.g., Firefox and Chrome) to achieve the peer-to-peer communication and data transmission among different peers (browsers). Thus when visiting a deployed website as usual, the user's browser will automatically join APAC as peers to fetch/distribute content from/to other peers in a transparent manner. Meanwhile, APAC's client-side code only requires the retrofitted website to specify the targeted resources. No specification means all static resources on the page can be distributed via peers. We demonstrate the details in Section 4.

3.2 Circuit Construction

The key technique of our approach is the locality-aware circuit selection algorithm. Given a set of parameters of the anonymity requirements and threat levels, the algorithm selects a circuit with optimized performance. In this section we

Notation	Description
l_{cir}	The circuit's length (number of nodes including the initiator and responder in the circuit - 1)
l, L_{max}	Number of intermediate peers or relays ($l = l_{cir} - 1$), the maximum l
$\mathbb{S}_{init}$	The set of N_{init} peers nearest to the initiator
$\mathbb{S}_{res}$	The set of N_{res} peers nearest to the responder
$\mathbb{S}, \mathbb{S}_R, \mathbb{S}_{im}$	The set of all peers in the system, the set of peers having a resource R, $\mathbb{S} - \mathbb{S}_{init} - \mathbb{S}_{res}$
N, N_{init}, N_{res}, N_R	Number of peers in $\mathbb{S}$, $\mathbb{S}_{init}$, $\mathbb{S}_{res}$, $\mathbb{S}_R$
α_{init}, α_{res}	Fraction of nodes for the circuit in $\mathbb{S}_{init}$, $\mathbb{S}_{res}$
f, f_R, f_{init}, f_{res}, f_{im}	Fraction of peers are compromised in the system, $\mathbb{S}_R$, $\mathbb{S}_{init}$, $\mathbb{S}_{res}$, $\mathbb{S}_{im}$

Table 1. Notations for anonymity analysis.

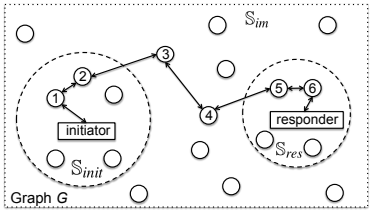

Fig. 5. A 8-node (6-relay) circuit from the initiator to the responder in APAC.

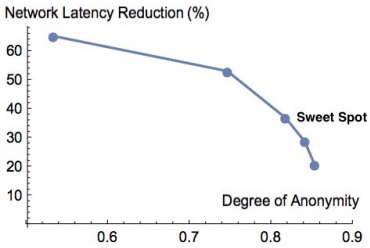

Fig. 6. The network latency reduction (based on the loading time without APAC) decreases when increasing the degree of anonymity of the system.

discuss the algorithm, and discuss parameter selection in the next section.

When other peers have already cached the requested resources, a peer can fetch content from them. We propose the circuit selection algorithm for APAC to choose intermediate peers from three categories: near-initiator, globally random (not nearby the initiator/responder) and near-responder nodes as a circuit for each request. Every request in APAC is encapsulated in layers of encryption (like onions) and transferred via the circuit.

Algorithm 1 describes our region-based circuit selection algorithm. In APAC, the peer server maintains certain information about the current network (e.g., the total number of peers). The input parameters L_{max}, α_{init} , α_{res}, N, N_R, N_{init}, and N_{res} are decided by the peer server based on the trade-off between anonymity and performance, defined in Table 1.

– The peer server randomly chooses one peer as the responder from the set of peers having the requested resource R (line 1),
– Based on the range from 1 to L_{max}, the number of intermediate nodes (l) is determined (line 2).
– According to the distribution factor α_{init}, the peer server randomly picks nodes nearby the initiator as the first part of the circuit[3] (line 4 - 7).
– According to another distribution factor α_{res}, the peer server selects nodes nearby the responder as the last part of the circuit (line 8 - 11).
– The remaining nodes of the circuit are randomly chosen from the rest of all peers (line 12 - 15).
– The peer server concatenates all the nodes (including the initiator, intermediate nodes and responder) in sequence as the circuit for the request (line 16).

Figure 5 demonstrates a 8-node circuit constructed by the peer server in APAC.

3.3 Parameters Selection

Depending on the requirement of anonymity/performance for the deployed web application, the input parameters for Algorithm 1 can be adjusted based on the analysis in Section 3.4 and 5. In this section, we briefly illustrate the primary factors to select these parameters.

The maximum number of intermediate nodes L_{max}. The distribution of peers in the network, the required anonymity, the fraction of adversarial peers and other factors affect the selection of L_{max}. As an in-advance conclusion, to achieve better anonymity, the system is supposed to select larger L_{max} (as shown in Figure 7, 8 & 9). On the other hand, to achieve better performance of the system, the smaller L_{max} is preferred (as shown in Figure 14). As Figure 6 shows, by adjusting the setting of APAC at the sweet spot, the system can preserve an adequate level of anonymity (i.e., 0.8 degree of anonymity)

3 The distances between the chosen peers and the initiator/responder are measured based on the peers' geographical coordinates, which can be obtained via *navigator.geolocation* [9] or using the GeoIP service to map network addresses to physical locations.

with minor performance overhead[4]. In our experiment, setting two intermediate nodes for the circuit makes the system at the sweet spot.

Distribution factors α_{init} & α_{res}. To provide higher level of anonymity for peers even when adversarial peers are densely near the initiator/responder, the system can select smaller $\alpha_{init}/\alpha_{res}$ and N_{init}/N_{res} to diversify the intermediate nodes on the circuit. Thus the probability to choose an adversarial node as the intermediate node is approximate to f, which does not give the adversary significant advantage. APAC can enlarge α_{init} & α_{res} to reduce client-side network latency for peers (as shown in Figure 16 & 17).

The other parameters N, N_R, N_{init} & N_{res}. By increasing the total number of peers N and the number of peers having requested resources N_R, the system can preserve higher level of anonymity, but it requires more users to join the system to start the content delivery via peers. To reduce the client-side network latency, the system can set small N_{init} & N_{res} to let intermediate nodes nearby the initiator/responder.

The security parameters such as the maximum number of intermediate nodes L_{max} and distribution factors α_{init} & α_{res} can be tuned to adjust the anonymity level of APAC. To achieve higher level of anonymity for peers, the developer can select larger L_{max} to increase the length of a circuit and larger N/N_R to increase the threshold of starting using APAC, as well as choose smaller $\alpha_{init}/\alpha_{res}$ and N_{init}/N_{res} to diversify the intermediate nodes on the circuit. To preserve initiator/responder anonymity for peers, currently peers cannot overrule the security parameters set by the developer; otherwise, adversarial nodes may choose low level of anonymity to easily infer the identity of the initiator/responder for a circuit. In the extreme scenario where a peer seeks an even higher privacy guarantee, it can opt out APAC to directly fetch resources from the resource server as fallback.

3.4 Anonymity Analysis

In this section, we analyze the anonymity of APAC in a mathematical way and demonstrate the way to choose proper parameters for the circuit selection algorithm based on the requirement of anonymity.

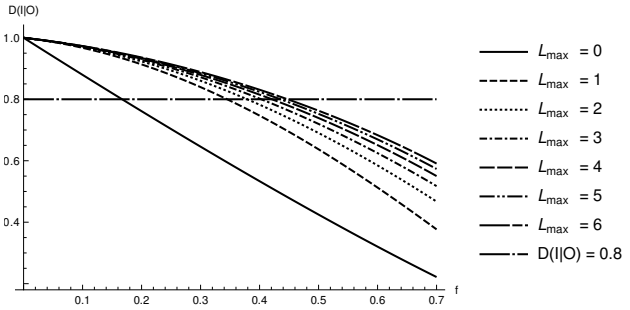

Fig. 7. Effect of varying L_{max}: Increasing the maximum number of intermediate nodes increases the degree of initiator anonymity.

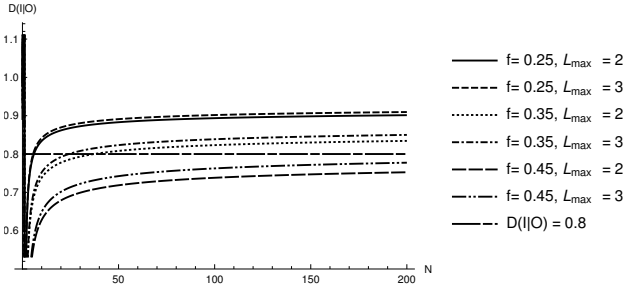

Fig. 8. Effect of varying N: Increasing the total number of joined peers increases the degree of initiator anonymity.

3.4.1 Analysis of Initiator Anonymity

Considering the initiator v_{init} issues a request for a resource R, the peer server sets up a $(l+2)$-node circuit based on the selection algorithm (in Algorithm 1), and the responder v_{res} delivers R. Our analysis follows the definitions in Section 2.3 and the notations in Table 1.

In APAC, each intermediate node in a circuit only knows its predecessor and successor. To identify the initiator of a circuit and link the requested resource to it, the adversary has to infer the circuit's length, control the responder (i.e., to know which resource is requested) and the first relay (i.e., to identify the predecessor as the initiator). The padding for each encryption layer makes the adversary difficult to infer the relative positions of the controlled peers in a circuit. Thus to confirm one of the controlled nodes is the first relay (only when the adversary knows the length), the adversary has to control the responder and some intermediate nodes in the circuit to reconstruct the circuit. For a given circuit, by controlling the first relay, the responder and at least half of the intermediate peers alternatively in the circuit, the adversary can determine the initiator [75]. However, we consider the worst case that the adversary can learn the distance between any two controlled peers by passively logging forwarded requests and transmit-

4 The figure is based on our anonymity analysis in Section 3.4 (Figure 7) and performance analysis in Section 5 (Figure 14). We consider that 100 peers (35% are adversarial) join APAC in one city, increasing the circuit's length can provide higher level of anonymity to benign peers but introduce more network latency.

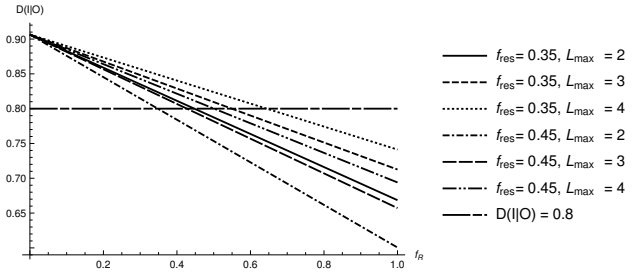

Fig. 9. Effect of varying f_R: Increasing the number of adversarial peers having the requested resource decreases the degree of initiator anonymity.

ted data as well as timing attacks[5]. Thus by determining the correct circuit's length as well as compromising the first relay and the responder, the adversary can infer that the first relay's predecessor is the initiator.

We assume that O_I is the observation that the adversary can identify the initiator for a circuit with the probability $P(O_I)$ that O_I occurs. Based on the derivation in Appendix A, the degree of initiator anonymity can be computed as:

$$D(I|O) = \frac{H(I|O)}{H_M} = (1 - P(O_I)) \frac{\log_2 ((1-f)N)}{\log_2 N} \quad (4)$$

In practice, the adversarial peers can exist uniformly in the system or densely around the initiator/responder. Next, we discuss the degree of initiator anonymity in these two scenarios.

Randomly placing peers in the system. In this case, the fractions of adversarial peers are equal in three different regions, i.e., $f_{init} = f_{res} = f$. If $f_R = f$, the probability for the adversary to identify the initiator of the circuit (proved in Appendix A) $P(O_I) = \frac{1}{L_{max}} \sum_{l=1}^{L_{max}} \frac{f^2}{L_{max}-l+1}$. Therefore, the degree of initiator anonymity can be represented as:

$$D(I|O) = \left(1 - \frac{1}{L_{max}} \sum_{l=1}^{L_{max}} \frac{f^2}{L_{max} - l + 1}\right) \frac{\log_2 ((1-f)N)}{\log_2 N} \quad (5)$$

To quantify the effects of L_{max} and N to the degree of initiator anonymity, we plot Equation 15 as Figure 7 and 8. Let $N = 100$, $L_{max} = 0, 1, 2, 3, 4, 5, 6$, Figure 7 shows that increasing the maximum circuit length can increase the degree of initiator anonymity. The first plot ($L_{max} = 0$) represents the degree of anonymity for current peer-assisted CDNs (derivation in Appendix B). We can see that the system does not hold 0.8 degree of anonymity even when $f < 0.2$. On the contrary, APAC preserves 0.8 degree of anonymity when $L_{max} = 2$ and over 35% peers are compromised. As shown in Figure 8, increasing the total number of joined peers can slowly increase the initiator anonymity. From Figure 7 and 8, we can see that when $L_{max} \geq 2$ and $N \geq 100$, APAC can preserve the suggested degree of anonymity (i.e., 0.8), even if 35% of all peers are under the adversary's control.

Placing peers nearby the initiator/responder. For a targeted peer, the adversary may increase f_R by storing the requested resource in more controlled peers and enlarge f_{init}/f_{res} by locating controlled peers nearby the initiator/responder. As we discussed in Section 3.1, without considering the worst case, the placement of peers nearby the initiator/responder does not help the adversary to gain significant advantage. Next we discuss the influence of this placement of peers in the worst case. If the first relay is in the initiator's region or the responder's region ($l_1 > 0$ or $l_1 = l_2 = 0$ & $l_3 > 0$ as shown in Equation 9), then $P(O_I) = \frac{1}{L_{max}} \cdot \sum_{l=1}^{L_{max}} \frac{f_{init} f_R}{L_{max}-l+1}$ or $P(O_I) = \frac{1}{L_{max}} \sum_{l=1}^{L_{max}} \frac{f_{res} f_R}{L_{max}-l+1}$. The degree of initiator anonymity can be computed as (using f_{res} as the representative):

$$D(I|O) = \left(1 - \frac{1}{L_{max}} \sum_{l=1}^{L_{max}} \frac{f_{res} f_R}{L_{max} - l + 1}\right) \frac{\log_2 ((1-f)N)}{\log_2 N} \quad (6)$$

To quantify the effects of f_R and L_{max} to the degree of initiator anonymity, we plot Equation 6 as Figure 9. Let $N = 100, f = 0.35$, as shown in Figure 9, increasing f_R and f_{res} can decrease the initiator anonymity. However, if we set α_{init} and α_{res} properly, which makes $l_1 = 0$ and $l_3 = 0$, then $P(O_I) = \frac{1}{L_{max}} \cdot \sum_{l=1}^{L_{max}} \frac{f_{im} f_R}{L_{max}-l+1}$. If we also set $N_{init} = N_{res} = 0$, then $f_{im} = f$. Hence increasing f_{res} does not directly decrease $D(I|O)$. On the contrary, when f_{res} or f_{init} increases, f_{im} may decrease accordingly as f is the same. In this case, if $f_{im} = 0.35$, the first, second and fourth plots in Figure 9 can represent the degree of initiator anonymity accordingly. Thus when $L_{max} = 2$, APAC can preserve over 0.8 degree of the initiator anonymity if f_R is around 0.45.

3.4.2 Analysis of Responder Anonymity

Analogous to initiator anonymity, in order to identify the responder of a given circuit, the adversary has to determine the circuit's length as well as control the last relay and the initiator. We assume that O_R is the observation that the adversary can identify the responder for a circuit with the probability $P(O_R)$ that O_R occurs. The degree of responder anonymity

5 To precisely determine the distance between two peers is difficult but possible for the adversary with timing attacks [31, 36]. To show the effectiveness of APAC against the adversary, we analyze the initiator anonymity in this worst case.

can be computed as (details of the derivation in Appendix A):

$$D(R|O) = \frac{H(R|O)}{H_M} = (1 - P(O_R)) \frac{\log_2 ((1-f)N)}{\log_2 N} \tag{7}$$

The analysis for the two scenarios: adversarial peers uniformly exist in the system and more adversarial peers near the initiator/responder, is similar to the analysis for the degree of initiator anonymity. As a result, if we properly tune the parameters, e.g., $L_{max} = 2, N = 100, \lfloor \alpha_{init} l \rfloor = 0$ and $\lfloor \alpha_{res} l \rfloor = 0$, APAC can preserve 0.8 degree of initiator/responder anonymity even if 35% of all peers are adversarial and have different distributions.

4 Implementation of APAC

In this section, we discuss the implementation details of APAC. We implement APAC with 2600+ lines of code as well as several libraries and frameworks. APAC's client-side code is written in JavaScript with 1000+ lines of code, apart from three libraries [5, 18] The peer server is also written in JavaScript with 1600+ lines of code based on Node.js platform and PeerServer [18]. APAC is compatible with mainstream browsers (e.g., Chrome, Firefox and Opera) and various operating systems (e.g., Mac OS, Linux and Windows). Similar to other peer-assisted CDNs (e.g., Swarmify), APAC is designed not to violate the same-origin policy — APAC is deployed "per website" along with the client-side scripts.

4.1 Components in APAC

In APAC, the communications between the peer server and peers are over HTTPS (e.g., TLS protocol [23]), and the data transmission among peers are over WebRTC which uses DTLS protocol [7] as shown in Figure 10. TLS and DTLS protocols ensure that communication channels are secure and the adversary cannot eavesdrop or tamper with any message in the communications among peers and servers. APAC utilizes Session Traversal Utilities for NAT (STUN) [19] to deal with Network Address Translator (NAT) [14] traversal. In this way, even behind NAT a peer can communicate with another peer with a public IP address or also behind NAT.

Resources in APAC**.** Similar to other CDNs, e.g., Swarmify and NetSession, only static resources (e.g., files, images and videos) are delivered in APAC. To ensure content integrity, we use flat naming mechanism for resources, i.e., naming the resource with its hash value. Alternatively, we can attach the hash value for each resource request, then the peer can verify the content with the value at the client side. Since the content server is trusted, the peer only needs to verify the integrity of the resource fetched from other peers. To preserve authenticity, peers are limited to fetch resources provided/authenticated by the content server from other peers. The content server in APAC is the website's server or the CDN's edge server, which stores the website's pages and resources. To retrofit a website to deploy APAC, developers only need to append APAC's client-side code on pages. Optionally, developers can explicitly specify several shared resources instead of sharing all resources in the page.

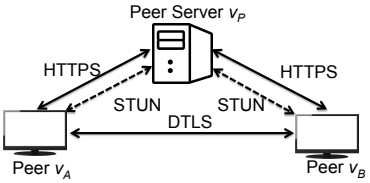

Fig. 10. Overview of the communication channels in APAC.

Peer Server in APAC**.** The peer server is in charge of maintaining the connections with peers, the properties of each peer (e.g., identity, IP address, peer key, locality and names of stored resources) and the properties of each resource (e.g., location in the content server and identities of peers having the resource). Meanwhile, when a peer requests a resource, the peer server either assists the peer to fetch the content from the server or constructs a circuit for the peer to retrieve the resource from another peer. Key management and path choosing are done by the peer server, and peer servers can be distributed to improve fault tolerance.

Peers in APAC**.** To achieve our third goal (compatibility), APAC is designed to support a plug-and-play style joining/leaving without any solicited action. When a user visits the retrofitted site, APAC's client-side code automatically assists the user's browser to join APAC as peers to fetch/deliver content from/to other peers, without any additional installation of extensions or software. Users leave APAC's network when closing tabs, like Swarmify.[6] The client-side code is purely based on JavaScript and compatible with all mainstream browsers. With WebRTC [26], the code enables browser-based real-time communication and data transmission for peers. It also helps peers to store the requested contents in indexedDB [13] (i.e., a high-performance client-side storage), which enables peers to directly load the same resources from indexedDB without issuing new requests. With data URI

6 To support opt-out in APAC for users, the website can host two copies, i.e., the original one and the retrofitted one. A user can explicitly choose which one to visit via setting preferences in the site's first page.

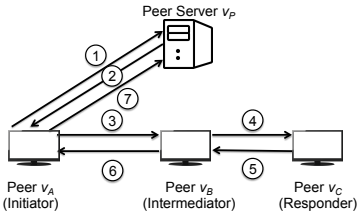

Fig. 11. Overview of how a peer fetches content from another peer via three nodes in APAC.

scheme [6] [7], the client-side code appends the content fetched from the server, another peer or indexedDB to the specific location in the visiting page.

4.2 Content Delivery in APAC

In this section, we illustrate how peers fetch content from the content server, other peers and indexedDB in APAC.

Initiation of peers. When a user first visits the site deployed with APAC, the client-side code will negotiate with the peer server to assist the user's browser to join the system as a peer. The user's browser issues an initiation request to the peer server with the current IP address or coordinates obtained from *navigator.geolocation* [9]. After receiving the request, the server recognizes the user's browser as a new peer, and responds the peer with an identity and a peer key (symmetric key). Meanwhile, the server records the peer with its properties, i.e., identity, IP address, peer key and locality (derived from IP addresses or coordinates). As for the peer key, after a period of time, each peer will communicate with the peer server to update a new key. After the initiation, the new peer can take advantage of the service provided by APAC to fetch/deliver resources from/to other peers.

Fetching content from the content server. When the total number of peers and the number of peers having the requested resource do not meet the requirement for anonymity (the input parameters in the circuit selection algorithm), the peer server will instruct peers to retrieve resources directly from the content server. In this case, the peer server responds the peer with the requested resource's original URL on the content server. Then the peer issues another request to the content server to retrieve the resource. After receiving the resource, the peer stores it in indexedDB, appends it to the page and updates the caching status to the peer server.

Fetching content from other peers. Content delivery via peers can start to operate when the system is sufficient to preserve the required anonymity as discussed in Section 3.4. We illustrate how peer v_A fetches a resource R from v_C via a 3-node circuit (as shown in Figure 11) step by step.

- ①: Peer v_A issues a request for a resource R to the peer server v_P;
- ②: After receiving the request, v_P first searches for online peers having R. Then v_P sets up a 3-node circuit based on the configuration, constructs the packet below with layered encryption, and responds v_A with the packet.

$$\begin{aligned}&\Big\{ID_{v_B}, K_R, \Big\{ID_{v_C}, Nonce_{v_B},\\ &\Big\{R_{name}, K_R, Nonce_{v_C}\Big\}_{K_{v_C}}\Big\}_{K_{v_B}}\Big\}_{K_{v_A}}\end{aligned} \tag{8}$$

  K_{v_A}, K_{v_B} and K_{v_C} are peer keys for v_A, v_B and v_C respectively. ID_{v_B} and ID_{v_C} are the identities for v_B and v_C. K_R is the key generated by v_P for v_A and v_C to encrypt/decrypt R. R_{name} is the requested resource's name. $Nonce_{v_B}$ and $Nonce_{v_C}$ contain the timestamp and padding, which make any intermediate node difficult to determine its relative position on the circuit. With Advanced Encryption Standard (AES) [37], the data (i.e., R_{name}, K_R and $Nonce_{v_C}$) is encrypted by K_{v_C} (denoted by $\{...\}_{K_{v_C}}$), K_{v_B} and K_{v_A} layer by layer.
- ③: Peer v_A receives the packet, decrypts it with K_{v_A}, and obtains ID_{v_B} (i.e., the next peer to contact), K_R and the remaining cipher text. Then v_A forwards the remaining data to v_B.
- ④: While receiving the packet from v_A, v_B obtains ID_{v_C} and $Nonce_{v_B}$ by decrypting it with K_{v_B}. After verifying the timestamp in $Nonce_{v_B}$, v_B forwards the remaining packet to v_C.
- ⑤: Peer v_C receives the packet from v_B, strips off the layer with K_{v_C}, and obtains R_{name}, K_R and $Nonce_{v_{res}}$. v_C verifies the timestamp in $Nonce_{v_{res}}$ and searches R_n in its indexedDB. If the timestamp is not out of date and R exists in the indexedDB, v_C responds to v_B with $\big\{R, Nonce_R\big\}_{K_R}$. Otherwise, v_C responds to v_B with an empty packet.
- ⑥: After receiving the response from v_C, v_B directly forwards the packet to v_A.
- ⑦: When receiving $\big\{R, Nonce_R\big\}_{K_R}$ from v_B, v_A decrypts it with K_R and obtains R. v_A verifies the integrity of R. If the obtained packet is empty or R is bogus, v_A issues another request for R to v_P. Otherwise, v_A stores R in its indexedDB and issues a request to v_P to update the status that v_A has R.

Fetching content from indexedDB. Once a peer successfully fetches a resource from the content server or another peer (with verifying the integrity) and stores it in the indexedDB, the peer will directly load the resource from its indexedDB instead of

7 The data URI scheme encodes data with base64, which introduces certain overhead. We use other serialization techniques for data transmission to reduce the overhead, and only use URI scheme when appending the data to web pages.

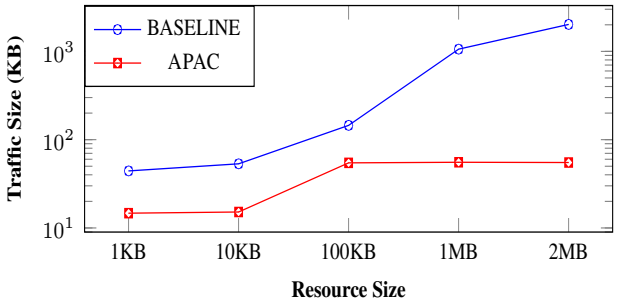

Fig. 12. Total outgoing network traffic size of a server in response to a request in APAC setting ($APAC$) and in the client-server setting ($BASELINE$), measured in KB.

issuing a new request afterwards. APAC is compatible with the current browser cache, i.e., the default cache and HTML5 application cache. The APAC's client-side code typically appends resources using data URI scheme. Since these cache mechanisms support to cache resources with data URI scheme, they can help to speed up the loading time for these resources.

5 Performance Evaluation

To demonstrate the effectiveness of APAC, we evaluate the *network throughput* of the deployed site's server, and the *network latency* of fetching resources in APAC. We investigate the following research questions in this section.

– What is the bandwidth saving of APAC compared with non-peer-to-peer client/server system?
– What is the latency reduction of APAC with various configurations (e.g., circuit's length and distribution factors)?

5.1 Measurement Setup

We have deployed APAC on a website that provides images of different sizes in a range of 1 KB - 2 MB. APAC supports all standard text and media formats, e.g., html, JavaScript, jpg, ogg and mp4. Since the majority of requests in popular websites are for images [76], we use images as the representatives of distributed resources in our experiment. Both the site's content server (i.e., edge server) and the peer server are located in City A[8]. Considering a typical scenario that peers and the content/peer server are in different cities, we place peers in City B different from City A. We launch over 100 different browser instances with located across City B. When browsing the same site deployed with APAC, these browsers join the system as peers.

To measure the network throughput, we deploy a performance measurement tool called `iftop` on the content server as well as the systems hosting browser instances. To measure the client-side network latency in various situations, we adjust the settings in the circuit selection algorithm. We vary the number of nodes on the circuit from 2 (no intermediate node) to 6 by changing L_{max}, and diversify the locations of intermediate nodes (tuning distribution factors α_{init} and α_{res}) in City B. We demonstrate how these settings for circuit construction affect the performance of APAC.

Comparison to other CDNs. CDN operators can increase the geo-density of edge servers up to a point (e.g., placing edge servers in major cities of various countries), but fetching resources for users far away from an edge server has major latency. Considering this typical scenario, we place peers and the peer/content server (or CDN's edge server) in different cities in our experiment setting. In this setting, compared to regular CDN services that clients directly fetch resources from the CDN's edge server, APAC can reduce network bandwidth and network latency. We show the details of bandwidth savings in Figure 12 and 13, and network latency reduction in Figure 14. For example, when every circuit has up to 4 nodes, APAC can reduce 97.3% bandwidth for the CDN's edge server and reduce 27.4% to 44.1 network latency of fetching 2 MB resources for peers. Naturally, a single-hop peer-assisted CDN, such as Swarmify, can achieve higher network latency reduction (e.g., 69.4% for Swarmify case) than APAC does. However, it does not preserve the anonymity guarantee which APAC aims to provide. We provide the detailed comparison on latency reduction in Figure 14.

5.2 Bandwidth Saving

First, we measure how much bandwidth can be saved by deploying APAC on the server side. We record the total size of network packets that go out from the content/peer server after the circuit ($L_{max} = 2$ for this experiment) has been established and the initiator starts fetching a specific resource (labeled as $APAC$). As a baseline, we also measure the size of outgoing network traffic from the content server in the typical client-server environment (labeled as $BASELINE$). We define the bandwidth saving (labeled as BS) as the percentage reduction from $BASELINE$.

Figure 12 shows the total size of outgoing network traffic from the server in client-server and APAC settings. The total size of outgoing packets from the server in APAC ($APAC$)

8 In this paper, City A represents New York City, and B represents Singapore. In our evaluation, we deployed one peer server, which is adequate for our experiments of 100 peers. For a large scale of users, e.g., 1 million, multiple peer servers can be deployed to handle requests in parallel.

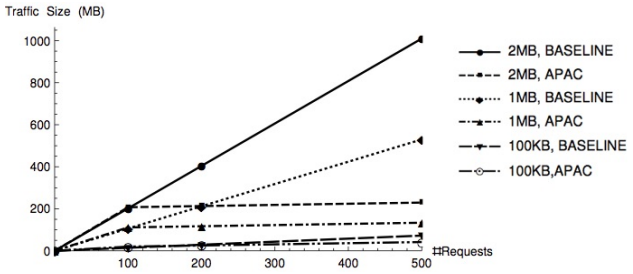

Fig. 13. When the number of requests increases, the total outgoing network traffic for $BASELINE$ significantly increases, but the traffic for $APAC$ only slowly increases.

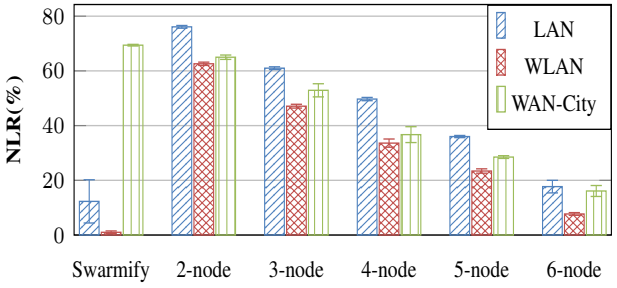

Fig. 14. Network latency reduction (NLR) of an initiator peer in Swarmify and APAC under three network configurations, as compared to the baseline. All data are based on a resource with size of 2 MB and averaged over 30 runs with 95% confidence intervals for each of them.

is much smaller than the size of network traffic in the normal client-server setting ($BASELINE$), and this applies to all resources with different sizes. The reason is that peers in APAC can fetch a large fraction of resources from other peers instead of the server. When the initiator fetches a resource of 2 MB[1], APAC can save 97.3% of the server's bandwidth[9]. When the total number of requests increases, APAC can significantly save the bandwidth as shown in Figure 13[10].

5.3 Network Latency

Next, we measure the network latency when fetching content in APAC with different peer placements. According to the statistics until 15th February 2016 [22], the averaged total size of transferred data when loading a site is 2268 KB. Thus in our experiment, we use the loading time of 2 MB resources (i.e., images) as the representative. With fixing the size of the resource to 2 MB, we evaluate APAC under the settings where the initiator, responder and intermediate nodes are located in: 1) the same LAN, 2) the same WLAN and 3) different LANs but in the same city (i.e., City B, labeled as "WAN-City"). We then vary the length of circuit as well as distribution factors and compare those results with a baseline: the network latency for a browser in City B to directly fetch the resource from the content server without APAC[11]. Analogous to BS, we define the client-side network latency reduction NLR as the percentage reduction from $BASELINE$. The higher the percentage of NLR is, the better is the performance (0% = $BASELINE$). All our results are the average of 30 runs with 95% confidence intervals for each of them.

Varying the circuit's length. In Figure 14, "Swarmify" represents the Swarmify system, and "2-node" represents the non-anonymous peer-to-peer setting in APAC, i.e., 2 nodes without any intermediate nodes. Since they do not have any intermediate nodes to route data, they have the largest network latency reduction, i.e., 69.4% for Swarmify in WAN and 76.1% for APAC in LAN respectively. Due to the implementation issue, Swarmify does not properly support P2P data transmission in LAN and WLAN, so the NLR for Swarmify in LAN and WLAN are quite small in Figure 14. We have reported this issue to their developer team. Since APAC involves more intermediate nodes than a single-hop non-anonymous system like Swarmify, APAC naturally shows a less latency improvement than Swarmify. This is illustrated in Figure 14, for a 4-node circuit where APAC provides a latency reduction (49.7%) lower than the performance obtained for Swarmify (69.4%) and non-anonymous setting (76.1%). Notably, APAC still outperforms the baseline. In general, with the increase of the circuit's length, the network latency reduction decreases, as more hops are required to route the transmitted data. In the three configurations, the latency in LAN is the smallest, as all the peers in the circuit are nearby. Since the bandwidth in WLAN is limited, the latency in WLAN is larger than the other two settings.

We also evaluate the setup latency of a circuit for various number of intermediate nodes. As Figure 15 shows, the setup latency of a circuit is only a small fraction of the network latency. For example, the setup latency of a circuit for 2 to 6 nodes is 11.6 ms, 83.8 ms, 152.7 ms, 227.3 ms, and 308.5 ms respectively in the LAN setting. The percentage of the network latency for each of them is 0.52%, 2.28%, 3.22%, 3.77%, and 3.98% respectively.

Varying the distribution factors α_{init} and α_{res}. In addition to adjusting the circuit's length, we can also tune the distribution factors α_{init} and α_{res} to reduce the latency in APAC.

9 $BASELINE$ = 2020 KB, $APAC$ = 55.1 KB, and BS = 97.3%.

10 We set the requirement for the total number of peers as 100, thus the content delivery via peers starts to operate when 100 peer join the system.

11 During the initial investigation, we observed the baseline latency ($BASELINE$) to be 9420 ms.

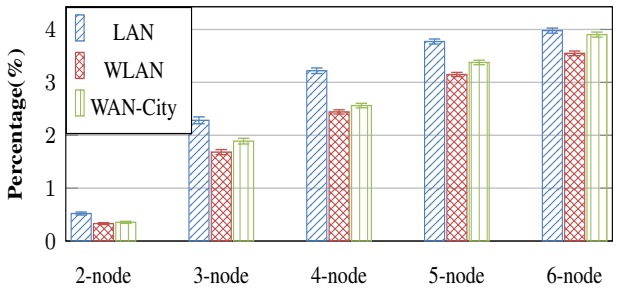

Fig. 15. The percentage of the network latency for setup overhead of a circuit in three configurations.

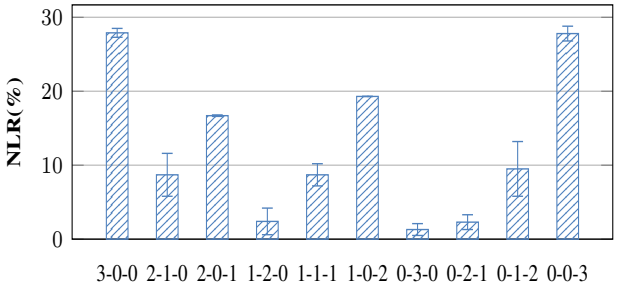

Fig. 16. Network latency reduction (NLR) varies when intermediate nodes are in different regions. "a-b-c" means the number of nodes in the initiator's region ($a = \lfloor \alpha_{init} l \rfloor$), nodes randomly chosen ($b = l - \lfloor \alpha_{init} l \rfloor - \lfloor \alpha_{res} l \rfloor$) and nodes in the responder's region ($c = \lfloor \alpha_{res} l \rfloor$).

For a 5-node circuit, we place the initiator and responder in two different regions (different LANs across City B), and we diversify the locations of intermediate peers to change α_{init} and α_{res}. As Figure 16 illustrates, setting more intermediate nodes around the initiator/responder (increasing $\alpha_{init}/\alpha_{res}$) for a circuit, the system further reduces the network latency. Thus for the circuit construction, the system can select larger α_{init} and α_{res} to reduce performance overhead. For the best case in our experiment, the system can reduce 27.8% latency only by adjusting the distribution factors. The system can be set as not "locality-aware" and randomly chooses all intermediate nodes (i.e., 0-3-0 setting) for the circuit. This configuration causes the worst performance, but it enables the system to preserve the best level of anonymity comparing to other settings.

Overall performance. We evaluate the overall performance in APAC for the deployed site, when 100 peers (browsers) have joined APAC and are available for content delivery. We set the maximum number of intermediate nodes on a circuit as 2 ($L_{max} = 2$), and record each 50 different resource requests in APAC when all intermediate nodes are randomly chosen (0-2-0/0-1-0 for $\lfloor \alpha_{init} l \rfloor = \lfloor \alpha_{res} l \rfloor = 0$) and all relays are nearby the initiator (2-0-0/1-0-0 for $\lfloor \alpha_{init} l \rfloor = l$) respectively. The

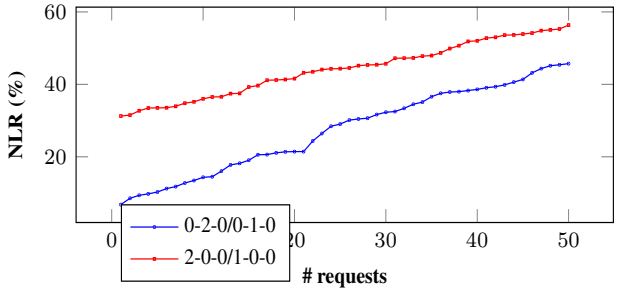

Fig. 17. The network latency reduction (NLR) for 50 requests(sorted in ascending order) when 100 joined peers are in APAC.

average NLR for the 0-2-0/0-1-0 and 2-0-0/1-0-0 settings are 27.4% and 44.1% respectively. The 0-2-0/0-1-0 setting has the considerable network latency reduction, though it holds the worst performance comparing to other settings analogous to Figure 16. The 0-2-0/0-1-0 setting provides the higher level of anonymity than the 2-0-0/1-0-0 one, which makes the adversary that places peers near the victim gain limited advantage over the honest-but-curious adversary as discussed in Section 3.4.1. As Figure 17 demonstrates, the latency reduction in the 2-0-0/1-0-0 setting is much larger than the other one, which reflects that tuning distribution factors α_{init} and α_{res} can assist APAC to provide reasonable performance.

Key findings. As we discussed in Section 3.4, when the circuit has up to two relays ($L_{max} = 2$), and adversarial peers uniformly exist in the network ($f = f_{init} = f_{res} = f_R$), APAC can preserve the adequate degree of initiator/responder anonymity (i.e., 0.8). As we show in this section, with intermediate peers up to 2 ($L_{max} = 2$) and proper setting for distribution factors ($\alpha_{init}/\alpha_{res}$), APAC can save 97.3% bandwidth for the site's server and reduce 27.4%/44.1% network latency of fetching 2 MB resources for peers within one city. Therefore, APAC can preserve initiator/responder anonymity with considerable bandwidth saving and latency reduction.

5.4 Performance under Churn

By default, APAC is deployed without any browser extensions, so once a user closes the tab for the retrofitted site, her browser (or peer) will leave APAC's network. If the peer happens to be a node in a circuit, its departure will break down the circuit, and the initiator has to issue another request. Therefore, as a part of the dynamics of peer participation or *churn* [68], the departure of peers in the duration of a request/-circuit (i.e., from the creation of the circuit to its teardown) influences the success of data transmission through the circuit. To quantify the success rate of circuits under Churn, we math-

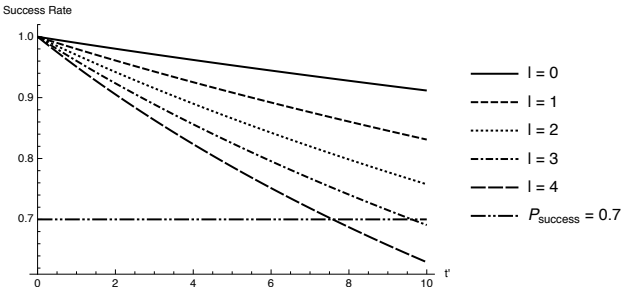

Fig. 18. The success rate decreases when the length of the circuit increases.

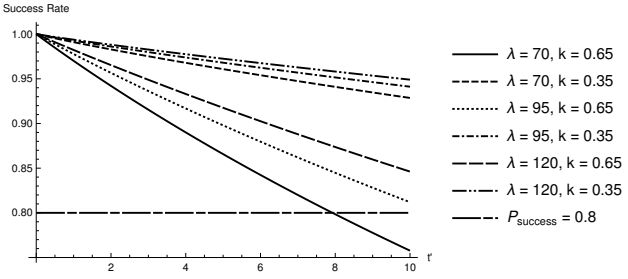

Fig. 19. The success rate increases when the stay time follows the Weibull distribution with larger λ and smaller k. (The circuit has 2 intermediate nodes.)

ematically analyze this issue based on a previous study conducted by Liu *et al.* [55]. It shows that the time users spend on a web page follows Weibull distribution [27]. We assume that the stay time of users on the deployed site's pages also follows Weibull distribution, and we detail our analysis in Appendix C.

As an in-advance conclusion, we show that the success rate of a circuit decreases when its length l or its duration t' increases in Figure 18. We can see that if data transmission via a circuit is finished quickly (i.e., the circuit has short duration), e.g., a small-size resource transferred in a high-speed network, the success rate of the circuit is quite high, e.g., over 90% when the median stay time is 39.8 s and $t' = 2$. This indicates that over 90% requested resources can be successfully transmitted via circuits for the first time without issuing additional requests. In some cases, e.g., fetching 2 MB resources via a 4-node circuit, the duration of the circuit is similar to the fetching time directly from server (9.42 s), the success rate is still around 80%. If the site can incentivize users to stay longer on the page, the success rate can increase a lot, e.g., over 95% with 42.1 s median stay time (as plotted in Figure 19). Alternatively, the site operator can create several backup circuits for one request, and the complete data transmission through any circuit is counted as the success for the request. This can drastically increase the success rate, e.g., over 2 backup circuits can make the success rate over 99% (details in Appendix C).

5.5 Load on Peers

We consider CPU and bandwidth for APAC to evaluate the load on peers. The size of the client side code of APAC is 9.5 KB, and 68KB when including all additional libraries. This is significantly smaller than the average-transfer data (2268 KB [22]) when loading websites. APAC would not affect the client-side performance much. Notably, most of the JavaScript files are static and cached.

We instrumented the htop and iftop tools [10, 12] to measure the average CPU usage and bandwidth for peers in a 3-node circuit for 10 times. Each client runs on a machine with a typical PC configuration — Intel i7-2600 CPU of 3.4 GHz and 8 GB memory. We find that APAC only incurs a reasonable 0.88% of CPU overhead and 4.21MB bandwidth per circuit for the clients. Further, the site operator can also limit the maximum number (e.g., 10) of joining circuits for one peer in one cycle of completing data transmission in a circuit, to avoid exhausting the peer's available bandwidth.

6 Related Work

In this section, we discuss recent work related to security & privacy in peer-assisted CDNs and anonymous communication systems.

Security & Privacy in Peer-assisted CDNs. Numerous peer-assisted CDNs have been proposed in recent years [2, 17, 20, 29, 43–45, 50, 69, 70, 73, 76]. In contrast to traditional infrastructure-based CDNs, peer-assisted CDNs offload content delivery tasks on clients (peers) to save the bandwidth of servers [48, 53], and reduce the latency of fetching content at the client side. For instance, NetSession can offload $70-80\%$ of the traffic to the peers [77]. For the thorough evaluation on Etsy, Maygh is able to reduce the 95th-percentile bandwidth due to image content at the operator by over 75% [76].

Meanwhile, researchers also propose solutions to preserve the integrity and authenticity of content [29, 69, 76]. For example, FireCoral introduces *signing service* and *tracker* components to authenticate and verify content [69]. On the other hand, no systematic studies of inference attacks on peer-assisted CDNs have been conducted yet and few defenses against such attacks are deployed on peer-assisted CDNs. In this work, we systematically analyze inference attacks on peer-assisted CDNs, and have mounted attacks on three popular systems, i.e., Swarmify, BemTV and P2PSP. Furthermore, we propose APAC to mitigate this class of attacks with minor performance overhead. We raise the bar significantly beyond what the current peer-assisted CDNs have.

Anonymous Communication Systems. To achieve user

Init. Anony. : Initiator Anonymity, Resp. Anony.: Responder Anonymity.

System's Name	No Installation	Init. Anony.	Resp. Anony.	Locality-aware
Onion Routing/Tor-based Systems [30, 41, 64, 71]	✗	✓	✗	Partially
Crowds [65], & Morphmix [66], etc. [46, 56–59, 62]	✗	✓	✗	✗
Hidden Service [21], I2P [11] & Freenet [8], etc. [24, 32, 49, 67]	✗	✓	✓	✗
APAC	✓	✓	✓	✓

Table 2. Comparison with Low-latency Anonymous Communication System

anonymity, anonymous communication systems are designed to hide user identity from third parties. Anonymous communication systems can be classified into high-latency and low-latency systems. High-latency anonymous communication systems e.g., Mixminion [38] and Mixmaster [60] are designed to be against a global passive adversary who can observe all traffic in the network. However, the high-latency message transmission (e.g., several hours) makes them unsuitable to be implemented in peer-assisted CDNs.

As Table 2 shows, there are four design considerations for low-latency communication systems from the system's perspective. Onion routing/Tor-based systems [30, 41, 64, 71], semi-centralized systems [33, 34], and p2p low-latency anonymous systems [8, 11, 21, 24, 32, 46, 49, 56–59, 62, 65–67] provide initiator anonymity. For instance, MorphMix creates paths on an unstructured overlay to forward communications; ShadowWalker is based on a random walk over redundant structured topologies and Pisces uses social networks to achieve anonymous communications. Further, Hidden Service [21], I2P [11], Freenet [8], and other approaches [24, 32, 35, 49, 67] also preserve responder anonymity.

Nevertheless, the primary goal for these approaches is to preserve high level of anonymity. They typically require users to install client-side software and are not locality-aware by default[12]. In Tor-based approaches, typically the initiator creates/constructs the circuit and selects the intermediate nodes for each request. Therefore these approaches preserve initiator anonymity, but cannot hide the responder's identity. Furthermore, together with the peer-to-peer anonymous communication systems, these systems introduce non-negligible circuit setup latency when the initiator indirectly communicates with intermediate nodes to set up the circuit. For example, the circuit setup latency for a 4-hop circuit in ShadowWalker is 1820 ms [57]. Therefore, we cannot directly apply previous mechanisms (as shown in Table 2) to achieve our predefined three goals. In APAC, we utilize the peer server (instead of peers) to construct the circuit for each request, which avoids the non-negligible overhead and preserves responder anonymity. Further, based on the locality information of all peers maintained by the server, we optimize the onion routing's circuit selection algorithm in APAC by introducing distribution factors, which can be tuned to locate intermediate peers nearby the initiator/responder to reduce network latency. Comparing to previous low-latency approaches, APAC can achieve all the primitives listed in the table.

12 The relay selection algorithm in Tor can be adjusted to be locality-aware [30].

7 Conclusion

In this paper, we systematically study inference attacks on peer-assisted CDNs. Further, we developer an anonymous peer-assisted CDN called APAC, which preserves a high degree of anonymity that is significantly beyond what current peer-assisted CDNs have. Our performance evaluation shows that the locality-aware APAC can bring desired network latency reduction for content fetching and bandwidth savings for deployed sites.

8 Acknowledgements

We thank the anonymous reviewers of this paper for their helpful feedback, and our shepherd Mohamed Ali (Dali) Kaafar for his insightful comments and suggestions for preparing the final version of the paper. We thank Enrico Budianto for his help on our experiments and Loi Luu for his constructive feedback on this paper. This work is supported by the Ministry of Education, Singapore under Grant No. R-252-000-560-112 and the National Science Foundation of US under Grant No. CNS-1318872. All opinions expressed in this work are solely those of the authors.

References

[1] Akamai. http://www.akamai.com/. Accessed: 2015.
[2] Akamai netsession interface. http://www.akamai.com/client. Accessed: 2015.
[3] Bemtv. http://bem.tv/. Accessed: 2015.

[4] Cloudflare. https://www.cloudflare.com/. Accessed: 2015.

[5] crypto-js: Javascript implementations of standard and secure cryptographic algorithms. https://code.google.com/p/crypto-js/. Accessed: 2015.

[6] The "data" url scheme. http://tools.ietf.org/html/rfc2397. Accessed: 2015.

[7] Datagram transport layer security. https://tools.ietf.org/html/rfc4347. Accessed: 2015.

[8] Freenet: The free network. https://freenetproject.org. Accessed: 2015.

[9] Geolocation api specification. http://www.w3.org/TR/geolocation-API/. Accessed: 2015.

[10] htop - an interactive process viewer for unix. http://hisham.hm/htop/. Accessed: 2016.

[11] I2p: The invisible internet project. https://geti2p.net/en/. Accessed: 2015.

[12] iftop: display bandwidth usage on an interface. http://www.ex-parrot.com/pdw/iftop/. Accessed: 2016.

[13] Indexed database api. http://www.w3.org/TR/IndexedDB/. Accessed: 2015.

[14] Network address translation. http://en.wikipedia.org/wiki/Network_address_translation. Accessed: 2015.

[15] Octoshape. http://www.octoshape.com/. Accessed: 2015.

[16] P2psp. http://www.p2psp.org/webrtc-streaming/. Accessed: 2015.

[17] Peercdn. https://peercdn.com/.

[18] The peerjs library. http://peerjs.com/. Accessed: 2015.

[19] Session traversal utilities for nat (stun). https://tools.ietf.org/html/rfc5389. Accessed: 2015.

[20] Swarmify. http://www.swarmify.com/. Accessed: 2015.

[21] Tor: Hidden service protocol. https://www.torproject.org/docs/hidden-services.html.en. Accessed: 2015.

[22] Total transfer size & total requests. http://httparchive.org/trends.php. Accessed: 2016.

[23] The transport layer security (tls) protocol. https://tools.ietf.org/html/rfc5246. Accessed: 2015.

[24] Tribler. http://www.tribler.org/. Accessed: 2015.

[25] Velocix. http://www.velocix.com/. Accessed: 2014.

[26] Webrtc. http://www.webrtc.org/. Accessed: 2015.

[27] The weibull distribution. http://reliawiki.org/index.php/The_Weibull_Distribution. Accessed: 2015.

[28] Wireshark. https://www.wireshark.org/. Accessed: 2015.

[29] P. Aditya, M. Zhao, Y. Lin, A. Haeberlen, P. Druschel, B. M. Maggs, and B. Wishon. Reliable client accounting for p2p-infrastructure hybrids. In *NSDI*, 2012.

[30] M. Akhoondi, C. Yu, and H. V. Madhyastha. Lastor: A low-latency as-aware tor client. In *IEEE S&P*, 2012.

[31] R. Annessi and M. Schmiedecker. Navigator: Finding faster paths to anonymity. In *IEEE Euro S&P*, 2016.

[32] K. Bauer, D. McCoy, D. Grunwald, and D. Sicker. Bitblender: Light-weight anonymity for bittorrent. In *AIPACa*, 2008.

[33] P. Boucher, A. Shostack, and I. Goldberg. Freedom systems 2.0 architecture. *Zero Knowledge Systems, Inc*, 2000.

[34] J. Boyan. The anonymizer: Protecting user privacy on the web. *Computer-Mediated Communication Magazine*, 1997.

[35] F. Burgstaller, A. Derler, S. Kern, G. Schanner, and A. Reiter. Anonymous communication in the browser via onion-routing.

[36] F. Cangialosi, D. Levin, and N. Spring. Ting: Measuring and exploiting latencies between all tor nodes. In *IMC*, 2015.

[37] J. Daemen and V. Rijmen. *The design of Rijndael: AES-the advanced encryption standard*. 2002.

[38] G. Danezis, R. Dingledine, and N. Mathewson. Mixminion: Design of a type iii anonymous remailer protocol. In *IEEE S&P*, 2003.

[39] G. Danezis, C. Lesniewski-Laas, M. F. Kaashoek, and R. Anderson. Sybil-resistant dht routing. In *ESORICS*. 2005.

[40] C. Diaz, S. Seys, J. Claessens, and B. Preneel. Towards measuring anonymity. In *PET*, 2003.

[41] R. Dingledine, N. Mathewson, and P. F. Syverson. Tor: The second-generation onion router. In *USENIX Security*, 2004.

[42] J. R. Douceur. The sybil attack. In *Peer-to-peer Systems*. 2002.

[43] M. El Dick, E. Pacitti, and B. Kemme. Flower-cdn: a hybrid p2p overlay for efficient query processing in cdn. In *EDBT*, 2009.

[44] M. J. Freedman. Experiences with coralcdn: A five-year operational view. In *NSDI*, 2010.

[45] M. J. Freedman, E. Freudenthal, and D. Mazières. Democratizing content publication with coral. In *NSDI*, 2004.

[46] M. J. Freedman and R. Morris. Tarzan: A peer-to-peer anonymizing network layer. In *CCS*, 2002.

[47] Y. Gao, L. Deng, A. Kuzmanovic, and Y. Chen. Internet cache pollution attacks and countermeasures. In *ICNP*, 2006.

[48] C. Huang, A. Wang, J. Li, and K. W. Ross. Understanding hybrid cdn-p2p: why limelight needs its own red swoosh. In *NOSSDAV*, 2008.

[49] T. Isdal, M. Piatek, A. Krishnamurthy, and T. Anderson. Privacy-preserving p2p data sharing with oneswarm. In *CCR*, 2010.

[50] S. Iyer, A. Rowstron, and P. Druschel. Squirrel: A decentralized peer-to-peer web cache. In *PODC*, 2002.

[51] Y. Jia, Y. Chen, X. Dong, P. Saxena, J. Mao, and Z. Liang. Man-in-the-browser-cache: Persisting https attacks via browser cache poisoning. *Computers & Security*, 2015.

[52] Y. Jia, X. Dong, Z. Liang, and P. Saxena. I know where you've been: Geo-inference attacks via the browser cache. *IEEE Internet Computing*, 2014.

[53] T. Karagiannis, P. Rodriguez, and K. Papagiannaki. Should internet service providers fear peer-assisted content distribution? In *SIGCOMM*, 2005.

[54] S. Le Blond, A. Uritesc, C. Gilbert, Z. L. Chua, P. Saxena, and E. Kirda. A look at targeted attacks through the lense of an ngo. In *USENIX Security*, 2014.

[55] C. Liu, R. W. White, and S. Dumais. Understanding web browsing behaviors through weibull analysis of dwell time. In *SIGIR*, 2010.

[56] A. Mislove, G. Oberoi, A. Post, C. Reis, P. Druschel, and D. S. Wallach. Ap3: Cooperative, decentralized anonymous communication. In *SIGOPS European Workshop*, 2004.

[57] P. Mittal and N. Borisov. Shadowwalker: peer-to-peer anonymous communication using redundant structured topologies. In *CCS*, 2009.

[58] P. Mittal, F. Olumofin, C. Troncoso, N. Borisov, and I. Goldberg. Pir-tor: scalable anonymous communication using private information retrieval. In *USENIX Security*, 2011.

[59] P. Mittal, M. Wright, and N. Borisov. Pisces: Anonymous communication using social networks. In *NDSS*, 2012.

[60] U. Möller, L. Cottrell, P. Palfrader, and L. Sassaman. Mixmaster protocol-version 2. 2003.

[61] G. Nakibly, J. Schcolnik, and Y. Rubin. Website-targeted false content injection by network operators. *arXiv preprint arXiv:1602.07128*, 2016.
[62] A. Nambiar and M. Wright. Salsa: a structured approach to large-scale anonymity. In *CCS*, 2006.
[63] T. Peng, C. Leckie, and K. Ramamohanarao. Survey of network-based defense mechanisms countering the dos and ddos problems. *CSUR*, 2007.
[64] M. G. Reed, P. F. Syverson, and D. M. Goldschlag. Anonymous connections and onion routing. *J-SAC*, 1998.
[65] M. K. Reiter and A. D. Rubin. Crowds: Anonymity for web transactions. *TISSEC*, 1998.
[66] M. Rennhard and B. Plattner. Introducing morphmix: peer-to-peer based anonymous internet usage with collusion detection. In *WPES*, 2002.
[67] V. Scarlata, B. N. Levine, and C. Shields. Responder anonymity and anonymous peer-to-peer file sharing. In *ICNP*, 2001.
[68] D. Stutzbach and R. Rejaie. Understanding churn in peer-to-peer networks. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 189–202. ACM, 2006.
[69] J. Terrace, H. Laidlaw, H. E. Liu, S. Stern, and M. J. Freedman. Bringing p2p to the web: Security and privacy in the firecoral network. In *IPTPS*, 2009.
[70] L. Vu, I. Gupta, K. Nahrstedt, and J. Liang. Understanding overlay characteristics of a large-scale peer-to-peer iptv system. *TOMCCAP*, 2010.
[71] T. Wang, K. Bauer, C. Forero, and I. Goldberg. Congestion-aware path selection for tor. In *FC*. 2012.
[72] G. Wondracek, T. Holz, E. Kirda, and C. Kruegel. A practical attack to de-anonymize social network users. In *IEEE S&P*, 2010.
[73] H. Yin, X. Liu, T. Zhan, V. Sekar, F. Qiu, C. Lin, H. Zhang, and B. Li. Design and deployment of a hybrid cdn-p2p system for live video streaming: experiences with livesky. In *MM*, 2009.
[74] H. Yu, C. Shi, M. Kaminsky, P. B. Gibbons, and F. Xiao. Dsybil: Optimal sybil-resistance for recommendation systems. In *IEEE S&P*, 2009.
[75] J. Zhang, H. Duan, W. Liu, and J. Wu. Anonymity analysis of p2p anonymous communication systems. *Computer Communications*, 2011.
[76] L. Zhang, F. Zhou, A. Mislove, and R. Sundaram. Maygh: Building a cdn from client web browsers. In *EuroSys*, 2013.
[77] M. Zhao, P. Aditya, A. Chen, Y. Lin, A. Haeberlen, P. Druschel, B. Maggs, B. Wishon, and M. Ponec. Peer-assisted content distribution in akamai netsession. In *IMC*, 2013.

A Degree of Initiator/Responder Anonymity in APAC

Initiator Anonymity. In our analysis, we consider the worst case for the system that the adversary can learn the distance between any two controlled peers by passively logging and timing attacks. We assume that peers are compromised independently. Let O_I be the observation that the adversary can determine the initiator for a given circuit with probability $P(O_I)$. For a given circuit in APAC, the probability of identifying the initiator of the circuit equals to the probability that the adversary determines the correct circuit's length as well as controls the first relay and the responder. Due to the distribution factors α_{init} and α_{res}, the first relay may be in $\mathbb{S}_{init}$, $\mathbb{S}_{im}$ or $\mathbb{S}_{res}$. Based on Algorithm 1, for a circuit in APAC, the responder is randomly chosen in $\mathbb{S}_R$. Following the notations in Table 1, for a given circuit with l intermediate nodes, the probability of controlling the first relay and the responder can be computed as:

$$P(Q_I|L=l) = \begin{cases} f_{init}f_R & \text{if } l_1 \geq 1 \\ f_{im}f_R & \text{if } l_1 = 0, l_2 \geq 1 \\ f_{res}f_R & \text{if } l_1 = l_2 = 0, l_3 \geq 1 \end{cases} \quad (9)$$

where $l_1 = \lfloor \alpha_{init} l \rfloor$, $l_2 = l - \lfloor \alpha_{init} l \rfloor - \lfloor \alpha_{res} l \rfloor$, $l_3 = \lfloor \alpha_{res} l \rfloor$, and $f_{im} = \frac{fN - f_{init}N_{init} - f_{res}N_{res}}{N - N_{init} - N_{res}}$.

In APAC, the length of the circuit is variable, and number of intermediate nodes is randomly chosen from 1 to L_{max}. Thus the probability that the number of intermediate nodes is l can be computed as $P(L = l) = \frac{1}{L_{max}}$. For a specific circuit with l relays, when the adversary controls the first relay and the responder, he still cannot precisely determine the predecessor of the first relay as the initiator, as he does not know the exact length of the circuit. Therefore, l, $l+1$,..., L_{max} can be considered as the number of intermediate nodes, and the probability that the adversary guesses the correct length is $P(L' = l|(Q_I|L = l)) = \frac{1}{L_{max}-l+1}$. The probability that O_I occurs when $L = l$ can be represented as $P(O_I|L = L) = P(L' \cap Q_I|L = l) = P(Q_I|L = l)P(L' = l|(Q_I|L = l)) = P(Q_I|L = l)\frac{1}{L_{max}-l+1}$. Generally, for a specific circuit, the probability that the adversary can identify the initiator is:

$$\begin{aligned} P(O_I) &= \sum_{l=1}^{L_{max}} P(L=l)P(O_I|L=l) \\ &= \frac{1}{L_{max}} \sum_{l=1}^{L_{max}} \frac{P(Q_I|L=l)}{L_{max}-l+1} \end{aligned} \quad (10)$$

Under this condition, the adversary can precisely to determine which peer is the initiator. Thus the entropy for identifying the initiator is $H(I|O_I) = 0$.

When the adversary does not have the observation O_I as above, it is impossible for the adversary to directly identify the circuit's initiator. From the adversary's perspective, the initiator anonymity set is $(1-f)N$. Therefore the entropy under this situation is: $H(I|\neg O_I) = -\sum_{i=1}^{(1-f)N} \frac{1}{(1-f)N} \log_2 \frac{1}{(1-f)N} = \log_2((1-f)N)$. From the adversary's observation O (i.e., O_I and $\neg O_I$), the overall entropy for the system can be represented as $H(I|O) =$

$(1 - P(O_I)) \log_2 ((1-f)\,N)$. For the ideal case, every peer has the equal probability of $\frac{1}{N}$ to be identified as the initiator. The maximum entropy $H_M = \log_2 N$. Therefore, the degree of initiator anonymity for the system can be computed as:

$$D(I|O) = \frac{H(I|O)}{H_M} = (1 - P(O_I)) \frac{\log_2 ((1-f)N)}{\log_2 N} \quad (11)$$

Responder Anonymity. Analogous to derivation for degree of initiator anonymity, we can derive Equation 7 for degree of responder anonymity. We briefly illustrate the derivation for degree of responder anonymity. Let O_R be the observation that the adversary can determine the responder for a given circuit with probability $P(O_R)$ to occur. For a given circuit in APAC, the probability of identifying the responder of the circuit equals to the probability that the adversary determines the correct circuit's length and controls the last relay and the initiator. Following the notations in Table 1, we show the difference between $P(Q_R|L = l)$ and $P(Q_I|L = l)$ below. For a given circuit with l intermediate nodes, the probability of controlling the last relay and the initiator can be computed as:

$$P(Q_R|L = l) = \begin{cases} f_{res} f & \text{if } l_3 \geq 1 \\ f_{im} f & \text{if } l_3 = 0, l_2 \geq 1 \\ f_{init} f & \text{if } l_3 = l_2 = 0, l_1 \geq 1 \end{cases} \quad (12)$$

where L_1, l_2, l_3 and f_{im} are defined same as Equation 9, and we assume that every peer is equally to be the initiator for a request.

In APAC, the length of the circuit is variable, and number of intermediate nodes is randomly chosen from 1 to L_{max}. Analogous to Equation 10, for a specific circuit, the probability that the adversary can identify the responder can be represented as $P(O_R) = \frac{1}{L_{max}} \sum_{l=1}^{L_{max}} \frac{1}{L_{max}-l+1} P(Q_R|L = l)$. From the adversary's observation O (i.e., O_R and $\neg O_R$), the overall entropy for the system can be computed as $H(R|O) = (1 - P(O_R)) \log_2 ((1-f)\,N)$. Therefore, the degree of responder anonymity for the system can be computed as:

$$D(R|O) = \frac{H(R|O)}{H_M} = (1 - P(O_R)) \frac{\log_2 ((1-f)N)}{\log_2 N} \quad (13)$$

The adversary's advantage on placement of peers. In the normal analysis, for a 8-node circuit ($l = 6$) as shown in Figure 5, the initiator, intermediate peers and responder are v_{init}, v_1, v_2, v_3, v_4, v_5, v_6, and v_{res} respectively. By controlling v_1 (nearby the initiator), v_3 (globally), v_5 (nearby the responder) and v_{res}, the adversary can definitely determine that v_{init} is the initiator. Referring to the notations in Table 1, we assume that the adversary controls a fraction f of N peers in APAC, a fraction f_{init}/f_{res} of N_{init}/N_{res} peers nearby the initiator/responder are adversarial, and f_R is the fraction of adversarial peers having the requested resource. If the adversary randomly places adversarial peers in APAC, then $f = f_{init} = f_{res}$, and the adversary has the probability $P_{rand} = f^3 f_R$ to infer the initiator. If the adversary places more peers around the initiator/responder to increase f_{init}/f_{res}, she has the probability $P_{near} = f_{init} f_{res} \frac{fN - f_{init}N_{init} - f_{res}N_{res}}{N - N_{init} - N_{res}} f_R$ to determine the initiator. Let $f = f_R = 0.35$, $N = 1000$, $N_{init} = N_{res} = 300$, we find that the adversary's advantage of using the second strategy over the first one is quite limited: $\Delta_{adv} = max\{P_{near}\} - P_{rand} = 0.0154 - 0.0150 = 0.004$, where $P_{near} = max\{P_{near}\}$ when $f_{init} = f_{res} = 0.39$.

B Degree of Anonymity in Current Peer-assisted CDNs

For a given request in peer-assisted CDNs, the initiator can be identified when the responder is adversarial. We assume that O_I is the observation when the responder of a particular request is under the adversary's control with the probability $P(O_I)$ that O_I occurs. Analogous to Section A, we can derive the same Equation 11, but the meaning of $P(O_I)$ is different. The entropy for the system with different observations (O_I and $\neg O_I$) can be computed as:

$$\begin{aligned} H(I|O) &= P(O_I) H(I|O_I) + (1 - P(O_I))\, H(I|\neg O_I) \\ &= (1 - P(O_I)) \log_2 ((1-f)\,N) \end{aligned} \quad (14)$$

where N is the number of all peers. When O_I occurs, the adversary can definitely identify the initiator, thus $H(I|O_I) = 0$; otherwise, from the adversary's perspective, the initiator anonymity set is $(1-f)N$. The probability that a benign peer in the system is the initiator is $\frac{1}{(1-f)N}$. Thus the entropy for the system when $\neg O_I$ occurs is $H(I|\neg O_I) = -\sum_{i=1}^{(1-f)N} \frac{1}{(1-f)N} \log_2 \frac{1}{(1-f)N} = \log_2 ((1-f)N)$. For the ideal case, every peer has the equal probability of $\frac{1}{N}$ to be identified as the initiator. The maximum entropy H_M can be calculated as $H_M = -\sum_{i=1}^{N} \frac{1}{N} \log_2 \frac{1}{N} = \log_2 N$. Therefore, the degree of responder anonymity can be computed as:

$$D(I|O) = \frac{H(I|O)}{H_M} = (1 - P(O_I)) \frac{\log_2 ((1-f)\,N)}{\log_2 N} \quad (15)$$

When the adversarial peers are uniformly scattered in the system, then an adversarial peer having the resource has the probability f to be chosen as the responder, i.e., the probability that the responder is adversarial $P(O_I) = f_R = f$. As the first plot ($L_{max} = 0$) in Figure 7 shows, the degree of initiator anonymity is less than 0.8 when $f \geq 0.2$. Analogous to the initiator anonymity, the current peer-assisted CDNs cannot preserve an adequate degree of responder anonymity when over 20% peers are adversarial.

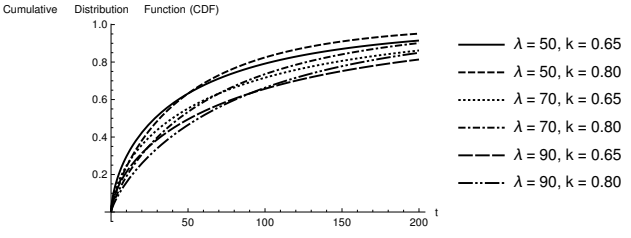

Fig. 20. λ becomes larger and k turns smaller, when more users stay longer on the page.

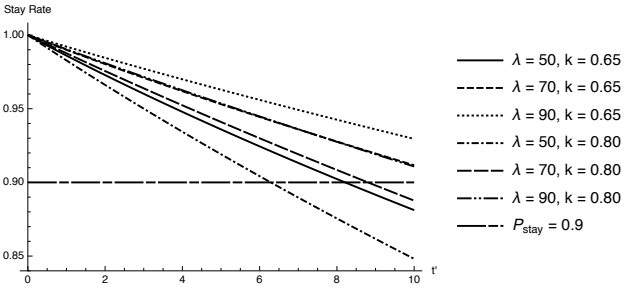

Fig. 21. The stay rate decreases when the duration of a circuit and k increase, as well as λ decreases.

C Analysis of Churn in APAC

A user joins APAC when visiting the deployed site from her web browser, and leaves APAC when closing the tab. As an inherent property of peer-to-peer systems, the dynamics of peer participation, or *churn* [68], especially the stay time of peers, affect the success rates of data transmission for circuits. To provide a mathematical understanding of users' page-leaving behaviors, Liu *et al.* analyzed page-visit stay time for 205,873 different pages for which they had captured upwards of 10,000 visits, and showed that the time users spend on a web page follows Weibull distribution [55]. Based on their finding, we can assume that the stay time[13] of users on our deployed site also follows Weibull distribution. We further calculate the stay rate of users in the duration of a circuit, and then compute the success rate of the circuit, i.e., all intermediate nodes and the responder stay in the network till data transmission via the circuit is completed.

The cumulative distribution function for the Weibull distribution is

$$\omega(t|k,\lambda) = 1 - \exp\left(-\left(\frac{t}{\lambda}\right)^k\right) t \geq 0 \qquad (16)$$

with t, λ and k are the stay time (dwell time), the scale and shape parameters, respectively. As shown in Figure 20, the

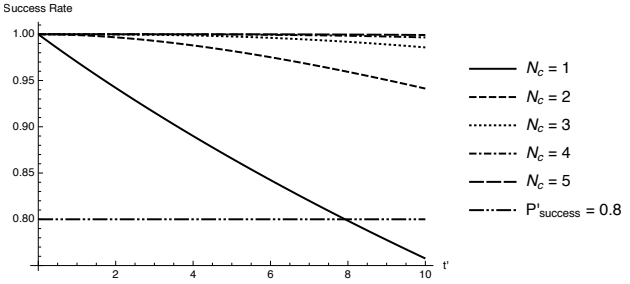

Fig. 22. The success rate increases when the number of created circuits increases.

longer of the average stay time is, the larger λ is and the smaller k is. For a $(l+2)$-node circuit created at time T, we assume that the duration of the circuit is t' and the average number of page visits is N_v. It takes t' seconds for the circuit to complete data transmission, and thus the minimal duration requirement for all nodes in this circuit is t'. For users who visit the site t seconds ahead the creation of the circuit, $(1-\omega(t+t'|k,\lambda))\,N_v$ of them are remaining after the teardown of the circuit, and $(1-\omega(t|k,\lambda))\,N_v$ users stay on the page at least till T. Since the peer server selects nodes for the circuit at T, we can compute the stay rate for users existing at T and lasting for t' below:

$$P_{stay}(t'|k,\lambda) = \frac{\int_0^{+\infty}(1-\omega(t+t'|k,\lambda))\,N_v\,dt}{\int_0^{+\infty}(1-\omega(t|k,\lambda))\,N_v\,dt} \qquad (17)$$

Furthermore, the probability that all intermediate nodes and the responder stay for t' or the success rate of the circuit is[14]:

$$P_{success}(t', l|k,\lambda) = P_{stay}(t'|k,\lambda)^{l+1} \qquad (18)$$

Based on the study of Liu *et al.* [55], λ for 80% of stay-time distributions is no more than 70, and the median value of k vary from 0.65 to 0.80. Figure 21 shows that when the duration t' and k are larger, or λ is smaller, then the stay rate is lower, more users existing at T may leave before the teardown of the circuit. For $\lambda = 70$ and $k = 0.65, 0.80$ (representing that the median stay time is 39.8 s or 44.3 s[15]), the stay rate is always over 90% when the duration of the circuit is 9.42 s (i.e., the fetching time of a 2 MB resource from the remote server). Figure 18 presents that with $\lambda = 70$ and $k = 0.65$ (the median stay time is 39.8 s), the longer length of the circuit may

13 In this paper, we treat Dwell time studied in Liu *et al.*'s work as the stay time of users on a page.

14 We assume that the number of peers for the creation of a circuit is large enough to ignore the probability that the same peer is selected as the intermediate node for multiple circuits.

15 The median stay time for Weibull distribution is:

$$T_{median}(k,\lambda) = \lambda\,(\ln 2)^{\frac{1}{k}} \qquad (19)$$

cause the success rate of data transmission lower. For a 4-node circuit, the success rate is always over 75% even if $t' = 10$. If the duration t' is short enough (e.g., 2 s), the success rate is always over 90%, which means over 90% circuits can be successfully completed. For the retrofitted site of APAC, the site can incentivize users to stay longer on the page, then the stay-time distribution can have larger λ and smaller k, further the success rate becomes higher, e.g., over 95% when $\lambda = 120$ and $k = 0.35$ (the median stay time is 42.1 s) in Figure 19.

Once the site operator observes that the distribution of stay time does not provide a proper success rate for a circuit (e.g., 80%), the site operator can suspend the circuit-based data transmission and reuse the client-server mode. Alternatively, as a relaxation of anonymity, the site operator can create several backup circuits for one request, and the complete data transmission through any circuit is counted as the success for the request. In this case, the success rate can be represented as:

$$P'_{success}(N_c, t', l|k, \lambda) = 1 - (1 - P_{success}(t', l|k, \lambda))^{N_c} \tag{20}$$

with the number N_c of circuits for one request. Figure 22 shows that with $\lambda = 70$, $k = 0.65$ and $l = 2$, 1 backup circuit[16](in total 2 circuits) for one request can drastically increase the success rate (e.g., over 90%). For over 2 backup circuits, the success rate of one request is always over 99% even the duration is 10 s. In the worst case, as a relaxation of compatibility, the site can employ APAC in an extension and ask users to install it. The extension checks whether the client is in any circuit whenever the client is attempting to leave APAC. If so, the extension will take over and complete data transmission for the pending circuits.

16 In Figure 22, we use the same-length circuits as backup circuits for one request. In fact, the peer server can set up backup circuits of any length (not larger than L_{max}).