ASAP: A Low-Latency Transport Layer

Qingxi Li, Wenxuan Zhou, Matthew Caesar, Brighten Godfrey
Dept. of Computer Science, University of Illinois at Urbana-Champaign
Urbana, Illinois, USA
qli10@illinois.edu, wzhou10@illinois.edu, caesar@cs.illinois.edu, pbg@illinois.edu

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols

General Terms

Design, Security, Performance

Keywords

Latency, TCP, DNS

1. MOTIVATION

Achieving low latency is a key challenge for interactive web applications and other online services, as even relatively small delays cause user frustration, loss of usability of web services, and loss of customers and revenue. A recent study by Google [2] found that delays as small as 100 milliseconds measurably reduced users' frequency of conducting searches; the effect increased over time and persisted for weeks after the artificial delay was eliminated. As most web requests are small, and downloading one web page usually involves connecting to several web servers, application latency is fundamentally limited by the connection establishment delay in the underlying network protocols.

This poster studies how transport protocols can be designed to perform transactions with delay as close as possible to a single round-trip time between the endpoints and proposes a new transport protocol, Accelerated Secure Association Protocol (ASAP), to achieve that goal. ASAP merges functionality of DNS and TCP's connection establishment functions by piggybacking the connection establishment procedure atop the DNS lookup process and when the server receives the request, instead of doing three-way handshaking (3WH), the server directly sends data back to the client. From our evaluation, ASAP can reduce the response time of small web request by up to two thirds.

However, doing this in a naive fashion suffers from potential problems. First, eliminating TCP's 3WH creates the possibility of denial of service (DoS) attacks. Most significantly, the attacker can perform *reflection and amplification attacks* [6]: it sends requests with the source address set to a *victim*'s address. The attacker thus instructs servers to conduct DoS attacks on the attacker's behalf.[1] To solve this, we design a *Provenance Verifier* (PV) which verifies a client's location, providing a certificate that the client can give to a server when opening a transport connection. In addition, embedding connection establishment information into a DNS request requires care to interact well with DNS caching.

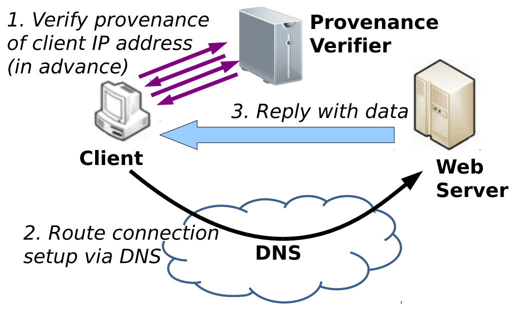
1. Verify provenance of client IP address (in advance)
Provenance Verifier
3. Reply with data
Client
Web Server
2. Route connection setup via DNS
DNS

Figure 1: ***Overview of ASAP.***

2. DESIGN OF ASAP

ASAP comprises a new protocol that combines the functions of DNS and TCP. We next describe ASAP's transport-layer design (§2.1) and name resolution (§2.2).

2.1 Fast transport connection establishment

In this subsection, we present ASAP's core transport connection establishment protocol.

Verifying provenance without a handshake: Typically, the 3WH lets the server test the provenance of an incoming request from some source IP S by sending a pseudorandom initial sequence number (ISN) back to S. If the client is able to reply acknowledging this ISN, then it is *effectively* located at S. Without the 3WH, some DoS vulnerabilities would be exposed (§1). However, we demonstrate a practical means for a server to verify source provenance similar to the 3WH's guarantee, yet without introducing an RTT delay.

First, the client handshakes with a **provenance verifier (PV)** to obtain a **provenance certificate (PC)**. Before this process, the client and PV have each generated a public/private key pair (K^c_{pub}/K^c_{priv} and $K^{pv}_{pub}/K^{pv}_{priv}$ respectively) using a cryptosystem such as RSA. The client

[1]Even with a 3WH, an attacker can reflect packets off a server; but only a single packet will be reflected for every packet the attacker sends, limiting the damage per unit work by the attacker.

Copyright is held by the author/owner(s).
SIGCOMM'11, August 15–19, 2011, Toronto, Ontario, Canada.
ACM 978-1-4503-0797-0/11/08.

then sends a request to the PV of the form

$$\{K_{pub}^c, d_c\}$$

where d_c is the duration for which the client requests that the PC be valid. The PV replies with the PC:

$$PC = \{K_{pub}^c, a_c, t, d\}_{K_{priv}^{pv}}.$$

Here a_c is the address of the client, t is the time when the PC becomes valid (current time), and d is the duration for which the PC is valid. The PV sets d to the minimum of d_c and the PV's internal maximum time, for example, 1 day (that is, the client only needs to contact the PV once per day). Once the client has a current PC for its present location, it can contact a server using the ASAP protocol and include the PC in its request to bypass the 3WH. To do this, the client begins by constructing a **request certificate (RC)** encrypted with its private key:

$$RC = \{hash(m_{net}, m_{trans}, data), t_{req}\}_{K_{priv}^c}.$$

Here $hash$ is a secure hash function, m_{net} is the network-layer metadata (source and destination IP address, protocol number), m_{trans} is the transport-layer metadata for the connection (source and destination port, initial sequence number), t_{req} is the time the client sent the request, and $data$ is the application-level data (such as an HTTP request). The RC makes it more difficult for adversaries (such as malicious web servers) to replay a connection request or use the PC for requests not initiated by the client.

The client then opens a transport connection to the server with a message of the form:

$$m_{net}, m_{trans}, PC, RC, data.$$

Upon receipt, the server verifies validity of the request. The server must already know the public key of each PV that it trusts. It determines whether the PC is valid by checking that it decrypts correctly, the current time lies within $[t, t+d]$, and a_c matches the source address. If so, it uses the client's public key K_{pub}^c from PC to check that RC decrypts correctly, the hash value in RC matches $hash(m_{net}, m_{trans}, data)$, and the time t_{req} is recent, e.g., within the last 5 minutes. (This timeout only needs to be long enough to cover most clock inaccuracy, and packet transit time.) If all these tests pass, then the request is accepted and the connection proceeds as in TCP after the 3WH: $data$ is passed to the application, and an application-level response (e.g., a web page) may be sent immediately to the client. Thus, the client can receive results within a single RTT.

Security properties: (i) Even if the attacker is able to eavesdrop on the RC and PC, it can only replay existing client requests for an amount of time limited by the timeout. (ii) If the attacker is able to take over the IP address owned by a client (e.g., by hijacking a prefix or compromising a router along the path), it can contact the PV and generate the RC itself. To mitigate this, we can run multiple PVs, and require clients to authenticate with all of them.

2.2 Fast name resolution

To reduce delay, we encode the connection information (e.g., initial sequence number, the source IP address and port, the PC and RC and data request) into the hostname field of the DNS request. For example, if the client is looking up `www.xyz.com`, it would generate a DNS request for CI`.www.xyz.com` where CI is the connection information. Since CI is unique, the local DNS will route the DNS request to the authoritative nameserver which supports ASAP. The ADNS will then strip off the CI and forward it to the server. It will also return an A record mapping CI`.www.xyz.com` to the web server's IP address. However, since CI is unique, even though the local nameserver caches this A record, it will never be used by the other ASAP queries even if they are looking up the same host. To address this, thereby reducing the workload of the ADNS, in addition to the ASAP query, we let the client send a normal DNS query (Fig. 1) which will cause the name to be cached at the local nameserver. Using the IP address result, the client then sends a transport connection request directly to the server.

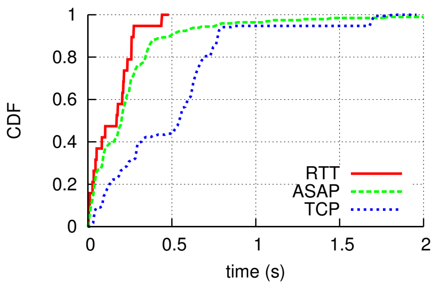

Figure 2: ***CDF of total delay to download a 1K-byte file.***

3. EVALUATION

We evaluate our implementation of ASAP through a deployment on PlanetLab (Fig. 2). Overall, ASAP reduces transmission time by between one and two round trip times (depending on DNS caching and location), significantly reducing latency of short web traffic.

4. RELATED WORK

T/TCP [1], an extension of TCP, bypasses the 3WHS and truncates TIME-WAIT State to reduce latency, but is vulnerable to malicious attacks [4]. Recently, Google proposed the TCP Fast Open (TCPFO) [3] extension that allows hosts to exchange data during the 3WHS. However, TCPFO only accelerates connections to a single server after the client has connected to that server once, while ASAP clients only need to contact PV once to get a valid certificate to accelarate connections to all the servers that trust the PV. DEW [5] explores ways by which a variety of Web requests and responses could be piggybacked on DNS messages, but requires modifications on both LDNSs and ADNSs.

5. REFERENCES

[1] R. Braden. T/TCP - TCP extensions for transactions, functional specification. 1994.

[2] J. Brutlag. Speed matters for Google web search, June 2009. `http://code.google.com/speed/files/delayexp.pdf`.

[3] Y. Cheng. TCP Fast Open. Unpublished work in progress (IETF Informational), March 2011.

[4] C. Hannum. Security problems associated with T/TCP. Unpublished work in progress (IETF Informational), September 1996.

[5] B. Krishnamurthy, R. Liston, and M. Rabinovich. DEW: DNS-enhanced web for faster content delivery. May 2003.

[6] V. Paxson. An analysis of using reflectors for distributed denial-of-service attacks. *ACM SIGCOMM Computer Communications Review*, July 2001.