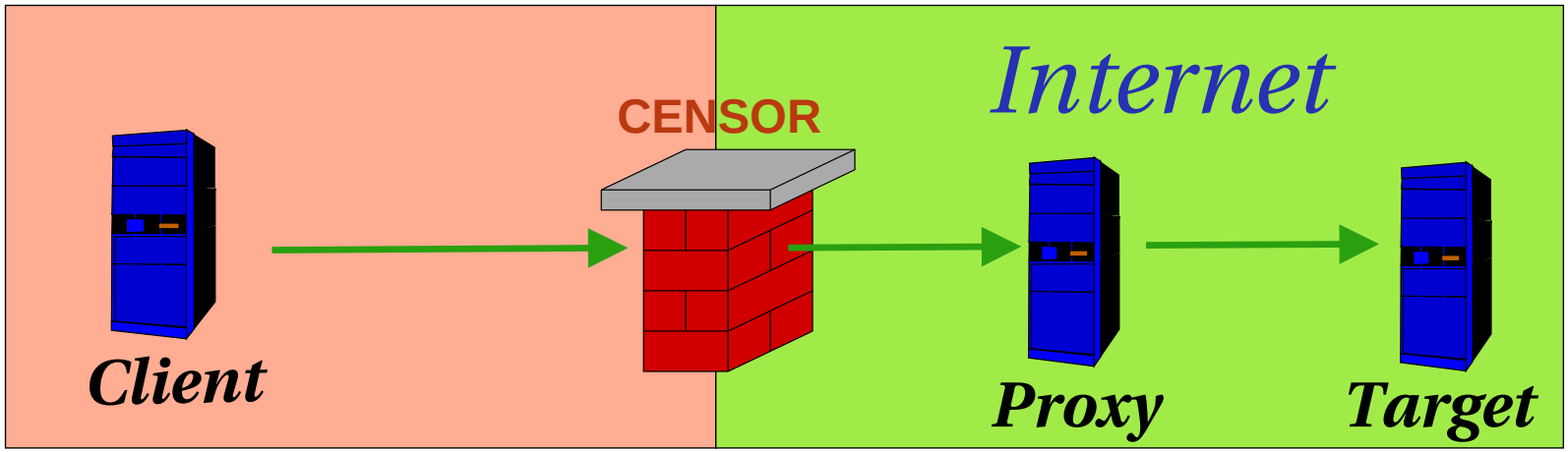
Thwarting Web Censorship with Untrusted Messenger Discovery

Nick Feamster, Magdalena Balazinska,
Winston Wang, Hari Balakrishnan, David Karger
M.I.T. Laboratory for Computer Science

http://nms.lcs.mit.edu/infranet/

Proxy-Based Censorship Systems



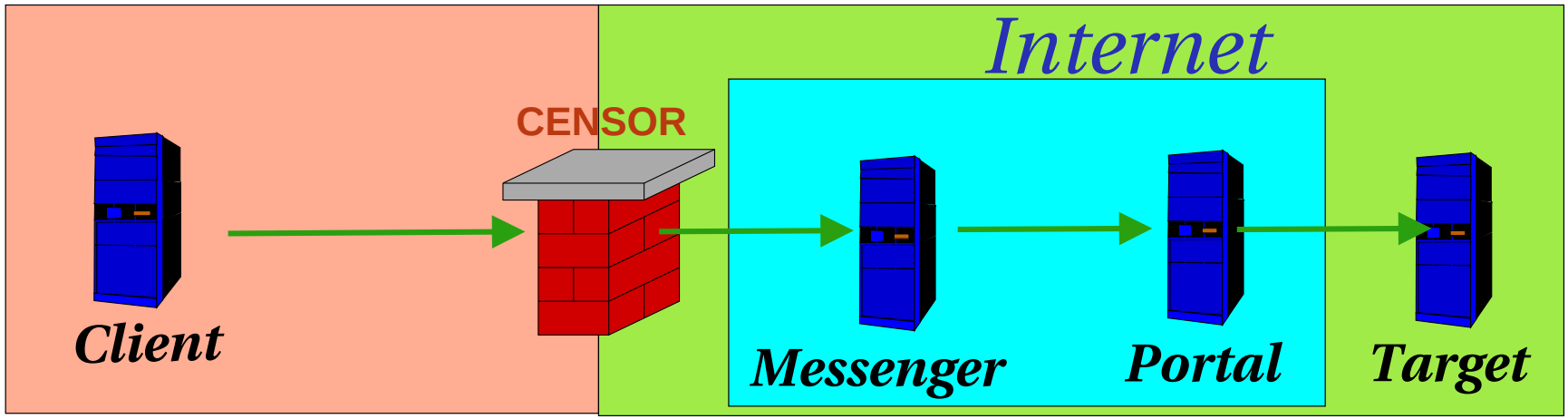
- Infranet, Peekabooty, Triangle Boy
- What's the problem?
  - Trusted proxies can be blocked.
  - Arbitrary proxies can't be trusted.

Tradeoff: Trust vs. Resistance to Blocking

- A small set of well-known proxies
  - *Good:* Clients can trust them
  - *Bad:* Easy to discover and block
- A large set of proxies
  - *Good:* Difficult to discover and block all of them
  - *Bad:* Clients may have to use a proxy they don't trust
- A proxy that clients know and trust can also be discovered by censors.

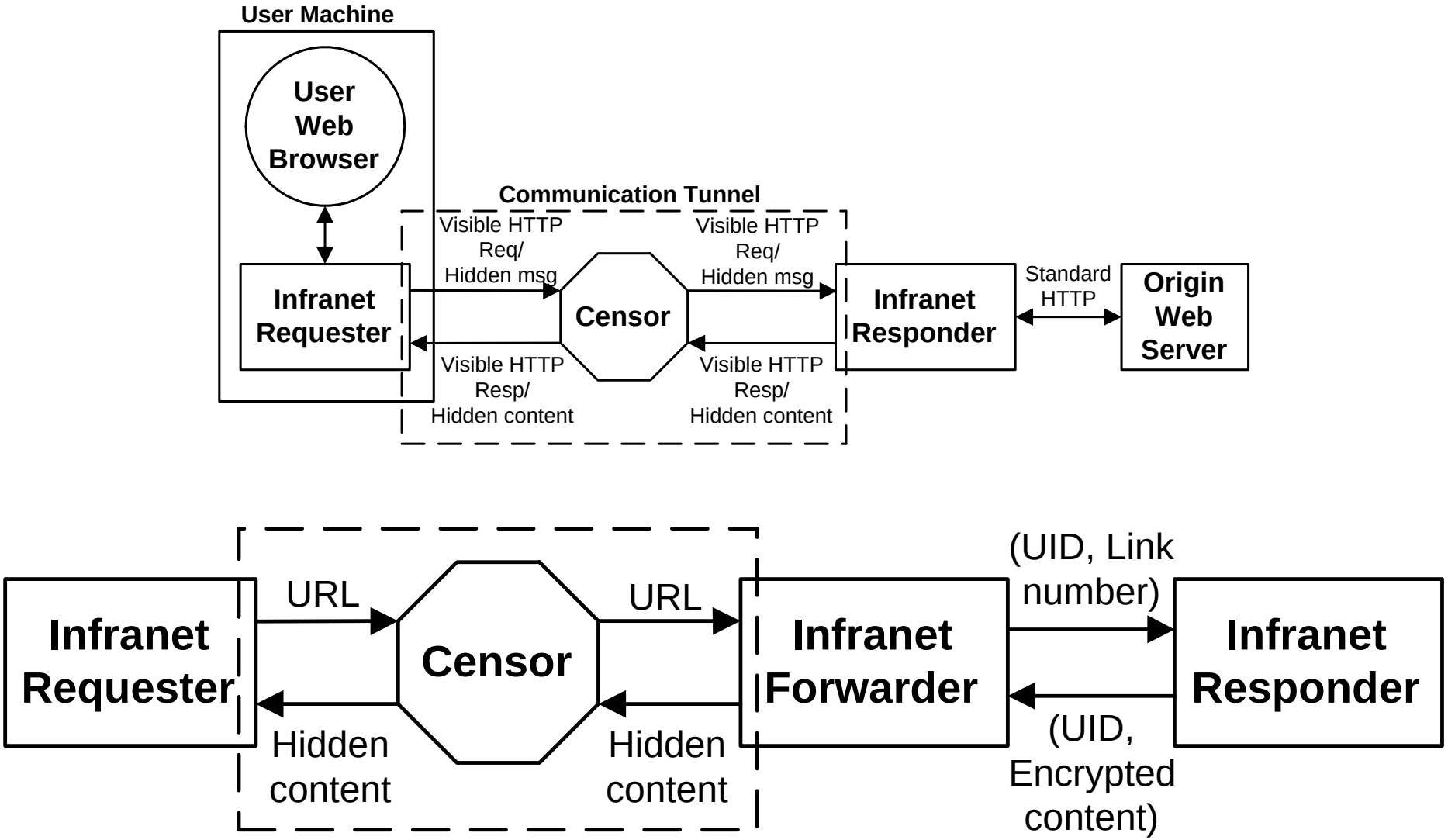
Can we get the best of both worlds?

The Proxy Serves Two Purposes

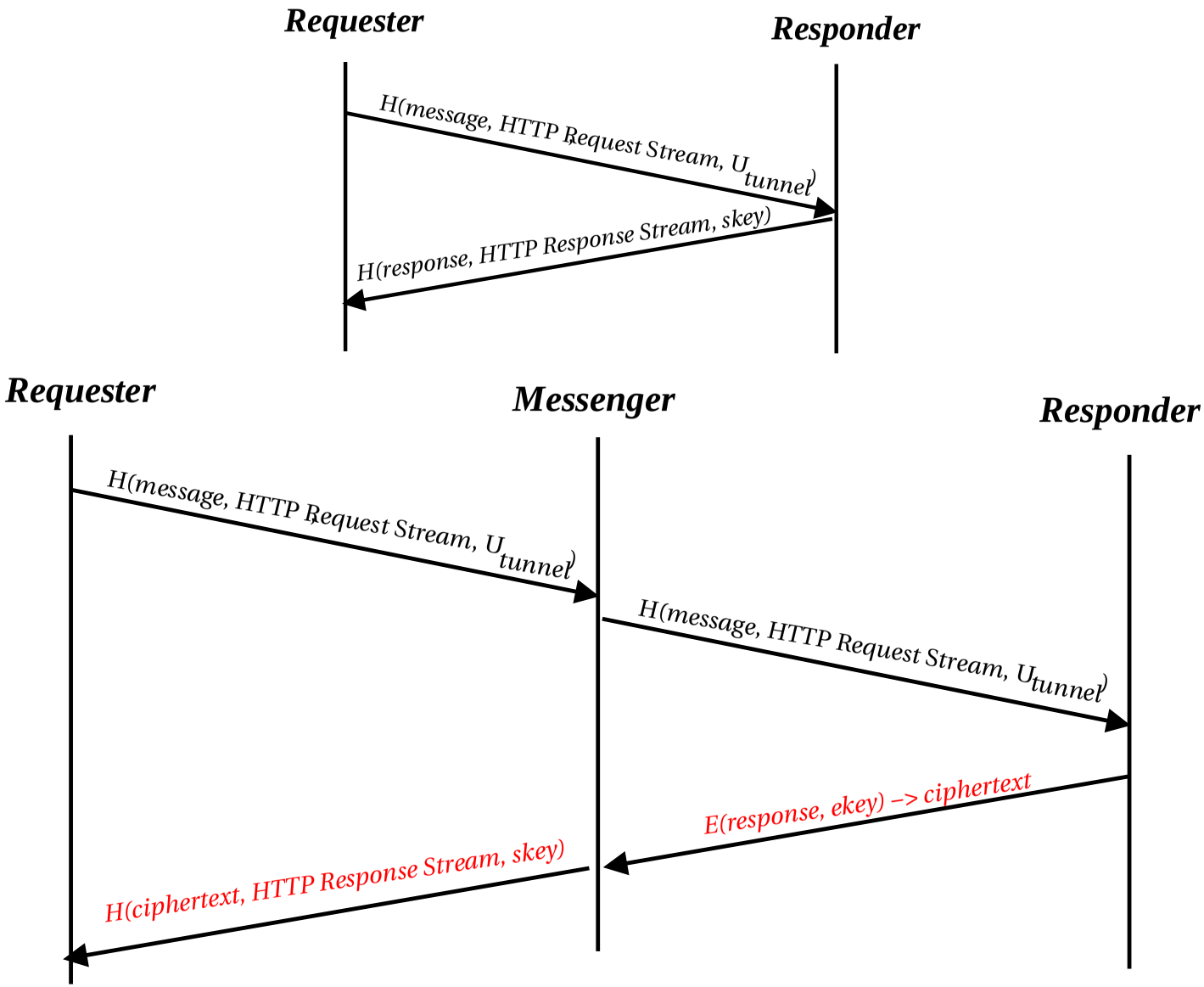


- Untrusted Messenger
  - Client hops between messengers, as before
  - Messengers cannot distinguish Infranet requests from innocuous requests (different from Triangle Boy)
- Trusted Portal
  - Sets codebook between URLs and message fragments
  - Deciphers URLs into message fragments, fetches content

Example: Infranet Protocol



Example: Infranet Protocol



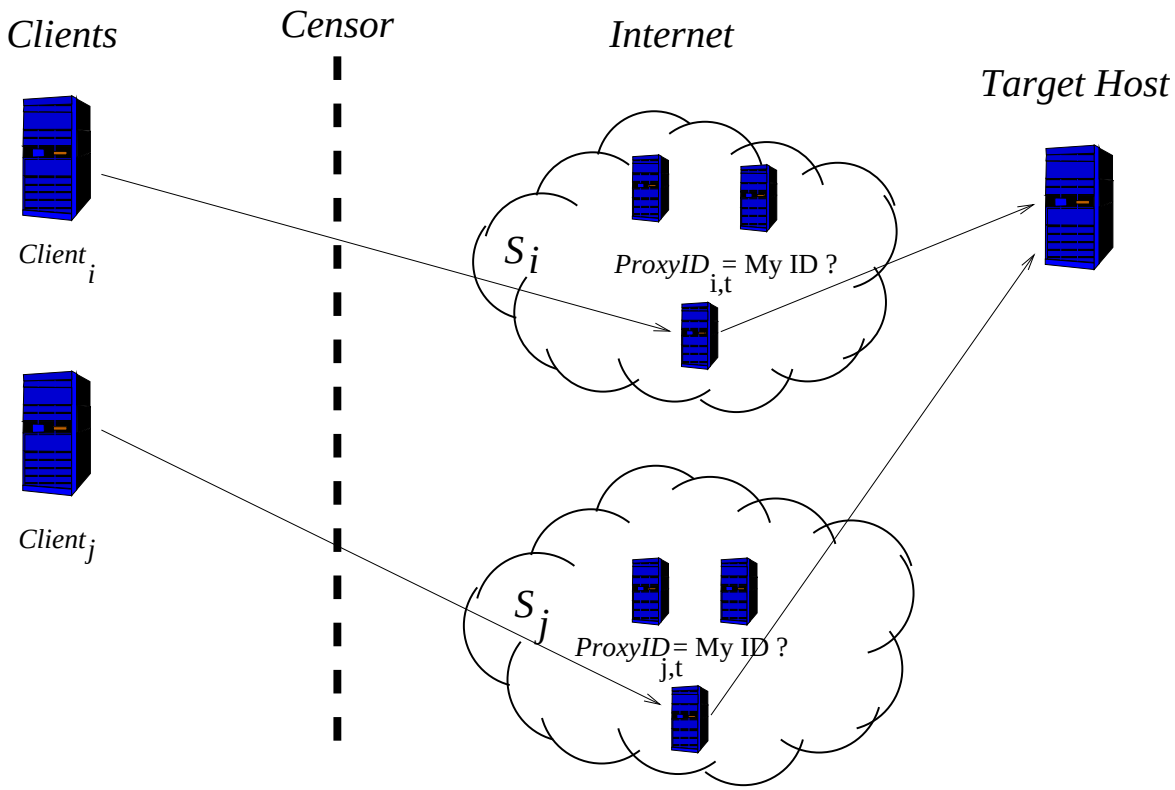
So...What Has Changed?

- Portals can be widely advertised and trusted.
  - Solves the problem faced by Triangle Boy, Infranet, et al.
- Messengers serve only as access points.
  - In the case of Infranet, same threat model as before.
- *Problem:* As clients, censors can discover messengers.
  - Messenger discovery must be designed carefully
  - Don’t let a client discover too many messengers!

Threat Model: Two Types of Discovery

- In-Band: Censor becomes a client
  - *Principle:* No single client should ever know a large portion of the proxies.
  - *Mechanism:* Assign each client a unique set of proxies, require initia investment of resources to discover proxies.
- Out-of-Band: Scan for proxies, traffic patterns, etc.
  - *Principle:* To an arbitrary Web client, a proxy should not appear to b a proxy.
  - *Mechanism:* Clients "hop" between proxies that serve them.

Keyspace Hopping



- Client transmits, proxy receives, each listening on a specific "frequency" for any timeslot.

Nice Features

- One client can’t discover all proxies in-band.
  - Resources and impersonation required for more widespread discovery (more later)
- Clients don’t choose their own proxies.
  - Eliminate odd-looking jumps in traffic at various locations
  - There is room for more intelligent transitions between proxies.
- Proxies don’t look like proxies to everyone.
  - Makes scanning more difficult (dragnet could still expose proxies, though)

Keyspace Hopping Challenges

- *Coordination:* How to figure out which proxies are serving that client?
- *Bootstrapping:* Client must know at least one proxy to get started!
  - How will the set of proxies know about this new client?
- *Lookup:* How to find a proxy with a particular ID?
  - Can’t let a censor perform arbirary lookups!

Coordination

$$
\begin{aligned}
PreProxyID_{t,i} &\leftarrow \mathcal{G}(\text{IP address}, hkey) \\
B_i &\leftarrow \mathcal{H}(\text{IP address}, hkey) \\
ProxyID_{t,i} &\leftarrow (B_i + (PreProxyID_{t,i} \mod |S_i|)) \mod C
\end{aligned}
$$

- Pseudorandom dance through portion of keyspace
- Client-specific info (IP address, hkey) determines
  - which portion of keyspace
  - how the client hops through that space
- Tradeoff of proxy set size
  - Larger means that client has more proxies to try.
  - ...but a single censor can then discover more proxies in one shot!

Bootstrapping and Lookup

- Bootstrapping: Client learns parameters
  - Can’t have one bootstrapping proxy (back to original problem)
  - Start with out-of-band discovery of one forwarder.
  - Forwarder services first-time visitors with some probability
- Lookup: Mapping Proxy IDs to IP addresses
  - Can’t use a typical lookup service
  - Send mapping after a client has solved a puzzle

How hard should the puzzle be?

How Difficult Should the Puzzles Be?

- The censor can impersonate any relevant IP address with ease.
- Each puzzle yields a chunk of known proxies.
- Time is on our side
  - 100,000 forwarders (reasonable)
  - Bernoulli trials: 300,000 puzzles for about 50% of proxies
  - Each puzzle takes a week to solve: 1,400 compute-years to block 50% of proxies

Rendezvous

- Rendezvous can expose clients
  - Upstream: does a rendezvous imply the client's guilt?
  - Downstream: telegraphing content for a client will expose Infranet clients to messengers
- *Upstream:* Increase the set of clients that a messenger services per timeslot
- *Downstream:* serve content per timeslot, not per client

Conclusion and Open Problems

- Proxy-based systems co-mingle trust and discovery.
- Separation into untrusted messengers and a few trusted portals provides the best of both worlds.
- Discovering forwarders: a few is easy; all is hard.
- Open Problems
  - Better ways to slow the censor?
  - Hopping should more closely mimic the habits of a normal user.
  - Impersonating multiple clients should be more difficult.
  - Timing, etc.