Maygh: Building a CDN from client web browsers

Liang Zhang[†§] Fangfei Zhou[†§] Alan Mislove[†] Ravi Sundaram[†]

[†]*Northeastern University* [§]*Student*

Motivation

Web has enabled information exchange at massive scale
E.g., News (NYT), OSNs (Facebook), Content sharing (Flickr)

Result: Popular sites must serve significant amounts of content

Options for serving popular web sites:
1. Serve on your own (purchase machines, etc)
2. Pay CDNs (Akamai, etc)
3. Pay cloud computing services (S3/EC2, etc)

All options result in significant monetary costs for operator

How do popular sites afford these costs?
1. User subscriptions (small user base)
2. Advertising (third parties, privacy concerns)

Limited choice of business models limits sites that can exist
What about sites that do not fit into either business model?

Goal: *Alternate way for popular web sites to distribute content*
Recruit web clients visiting site to help out

Related Work

Others have explored client-assisted web content distribution

Browser plug-ins
FireCoral, Swarm plug-in

Client-side software
Akamai's NetSession, PPLive

But, both require user to install and run/activate
Plug-ins can only serve other plug-in users
Existing approaches have somewhat unclear incentives

Example: AdBlock Plus installed on only 4.2% of FireFox users
With much more clear incentives for users to install

Maygh

Build a drop-in content distribution system for web content
Serves as a dynamically built CDN; content always available from origin

Want to make it work with today's sites, browsers
Do not require users to do anything different

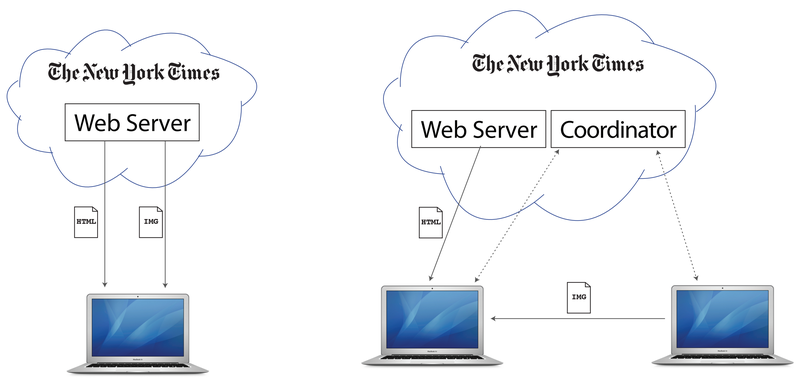
Key challenge: Browsers not designed to communicate directly
RTMFP (Flash) or WebRTC (W3C) for browser-to-browser communication

Design

Add coordinator: Middlebox run by website operators
1. Serves as a directory for content
   Keeps track of content in user's browsers

2. Allows browsers to establish direct connections
   Supports NAT traversal using STUN with RTMFP/WebRTC

With Maygh, browsers connect to coordinator, download content from others
Can scale multiple coordinators to support 1000s requests/second



Client-side Maygh library implemented in JavaScript (ActionScript for RTMFP)
Use LocalStorage to persistently store content

All content is static, identified by content-hash
Prevents forgery of content by malicious users

Using Maygh

1. Include Maygh JavaScript

```
<script src="maygh.js">
```

2. Change mechanism for loading content

```
<img id="pic-id" src="http://www.foo.com/..."/>
```

with

```
<img id="pic-id"/>
<script>
  maygh.load("pic-hash", "pic-id");
</script>
```

Evaluation

Implemented Maygh using RTMFP
Also have proof-of-concept WebRTC implementation

How much additional latency is there?

Accessed from	Served from Maygh: LAN (Boston)	Cable (Boston)	DSL (New Orl.)
LAN (Boston)	229 / 87 ms 72 / 16 ms	618 / 307 ms 364 / 120 ms	1314 / 707 ms 544 / 354 ms
Cable (Boston)	771 /283 ms 284 / 57 ms	702 / 314 ms 577 / 107 ms	1600 / 837 ms 765 / 379 ms

Fetch 50 KB objects from other peer
First/Subsequent loading time with RTMFP and WebRTC
RMTFP has protocol overhead; WebRTC is sufficiently fast

How much bandwidth can Maygh save?

Obtain one-week of Akamai image access logs from etsy.com
205 M requests, 5.7 M IP addresses
Simulate Maygh deployment; 75% 95th-percentile reduction