**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

**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

**Evaluation**

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

**How much additional latency is there?**

<table>
<thead>
<tr>
<th>Accessed from</th>
<th>Serviced from</th>
<th>LAN (Boston)</th>
<th>Cable (Boston)</th>
<th>DSL (New Orl.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAN (Boston)</td>
<td>Maygh</td>
<td>229 / 87 ms</td>
<td>618 / 307 ms</td>
<td>1314 / 707 ms</td>
</tr>
<tr>
<td></td>
<td></td>
<td>72 / 16 ms</td>
<td>364 / 120 ms</td>
<td>544 / 354 ms</td>
</tr>
<tr>
<td>Cable (Boston)</td>
<td></td>
<td>771 / 283 ms</td>
<td>702 / 314 ms</td>
<td>1600 / 837 ms</td>
</tr>
<tr>
<td></td>
<td></td>
<td>284 / 57 ms</td>
<td>577 / 107 ms</td>
<td>765 / 379 ms</td>
</tr>
</tbody>
</table>

Fetch 50 KB objects from other peer  
First/Subsequent loading time with RTMFP and WebRTC  
RTMFP 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