Motivation
Motivation

- Want server-to-server, interoperable messaging
  - Decentralized social networks
  - Federated messages (email++)
  - Instant syndication
  - Data for mashups
Motivation

- Requirements
  - Easy to deploy
  - Topic-oriented
  - High-throughput
  - Enables new use-cases

- Not trying to solve client/server
Many protocols out there
- Improved polling -- changes.xml, weblogs.com, Ping-o-Matic, rssCloud, SUP
- Publish/Subscribe -- Jabber/XMPP pubsub, AMQP, RestMS, WS-Notification, ...

Good ideas, but
- Not the right tool for topic-based messaging... or
- More complex than most people need... or
- Hard or unfamiliar to deploy... or
- Not scalable in the long term
Motivation

- Goal compared to Wave
  - Wave is a reinvention of collaboration on the web
  - PubSubHubbub bootstraps the web for real-time

- Meet in the middle
Polling

Publisher

Subscriber

Feed
Please give me your feed.
Please give me your feed.

Publisher

Subscriber

Feed
Polling

Publisher

Subscriber

Are we there yet!?

No!!!
PubSubHubbub

Publisher → Hub → Subscriber

Feed

Please give me your feed.

I use this hub for real-time delivery
Are you sure?

Hub

Subscribe me to the feed at this URL

Publisher

Hub

Subscriber

PubSubHubbub

Yes!
My feed at this URL has updated.

Please give me your feed.
Find what's new

Here's new content!

Publisher

Feed

Subscriber

Okay!
Here's new content, everyone!
Demo
PubSubHubbub benefits

- Bootstraps existing feeds to be real-time
- Simple for publishers and subscribers
- Hub provides reliability, scalability
- Easy security model
Scale
Scaling work with feeds

- How much does a single feed update cost?
- Very important as update frequency increases!

\[
\text{Total Cost} = \text{Frequency} \times \text{Marginal Cost}
\]
Scenario: Managing a library

- Collection of books for borrowing
- Need to tell the central office which books were borrowed
Doing work late
<table>
<thead>
<tr>
<th>Date</th>
<th>Title</th>
<th>Borrower</th>
</tr>
</thead>
<tbody>
<tr>
<td>11/9</td>
<td>Tale of two...</td>
<td>Bob</td>
</tr>
<tr>
<td>11/10</td>
<td>Don Quixote</td>
<td>Marsha</td>
</tr>
<tr>
<td>11/11</td>
<td>1984</td>
<td>Sally</td>
</tr>
</tbody>
</table>
Doing work late
Doing work late
Doing work late

<table>
<thead>
<tr>
<th>Date</th>
<th>Title</th>
<th>Borrower</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Date</td>
<td>Title</td>
<td>Borrower</td>
</tr>
<tr>
<td>--------</td>
<td>------------------</td>
<td>----------</td>
</tr>
<tr>
<td>11/9</td>
<td>Walden</td>
<td>Sally</td>
</tr>
<tr>
<td>11/10</td>
<td>Cat in the Hat</td>
<td>Bob</td>
</tr>
<tr>
<td>11/12</td>
<td>Huck Finn</td>
<td>Coco</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Problem: Generating a list of borrowers is slow
  – Need to do full inventory each time the list asked for
  – Incremental cost is high

Duh! Who would ever run a library this way?
Today with polling and ping/polling
- Must completely regenerate the feed on each update
- Similar to our library

Problems
- Database load producing the feed
- Bandwidth of the full feed document (to all subscribers)
- Invalidation of cached feeds
Doing work late with feeds

Total Cost = Frequency \times Marginal Cost
<table>
<thead>
<tr>
<th>Date</th>
<th>Title</th>
<th>Borrower</th>
</tr>
</thead>
<tbody>
<tr>
<td>11/9</td>
<td>Walden</td>
<td>Sally</td>
</tr>
<tr>
<td>11/10</td>
<td>Cat in the Hat</td>
<td>Bob</td>
</tr>
<tr>
<td>11/12</td>
<td>Huck Finn</td>
<td>Coco</td>
</tr>
<tr>
<td>Date</td>
<td>Title</td>
<td>Borrower</td>
</tr>
<tr>
<td>-------</td>
<td>-------------------</td>
<td>----------</td>
</tr>
<tr>
<td>11/9</td>
<td>Walden</td>
<td>Sally</td>
</tr>
<tr>
<td>11/10</td>
<td>Cat in the Hat</td>
<td>Bob</td>
</tr>
<tr>
<td>11/12</td>
<td>Huck Finn</td>
<td>Coco</td>
</tr>
<tr>
<td>11/18</td>
<td>War &amp; Peace</td>
<td>John</td>
</tr>
</tbody>
</table>
Doing work early

- Bookkeeping saves time
  - Before: Linearly increasing cost
  - After: Constant marginal cost

- Incremental, immediate picture of book status
Doing work early with feeds

Publisher/Hub combined

Hub
Doing work early with feeds

- Hub pushes content directly
  - Only generate the piece of the feed that has changed
  - Do all work at write time!

- Benefits
  - No extra database load (beyond update cost)
  - Bandwidth of the partial feed document
  - No need for cache invalidation
Total Cost = Frequency \times Marginal Cost
PubSubHubbub provides the best of both
- Delegate to one or more Hubs to be real-time **today**
- Run your own hub for peer-to-peer, high-throughput

*A migration path for new use-cases!*
Progress
Progress

▪ Over 100 million feeds are PubSubHubbub enabled

▪ Companies
  – Google, FriendFeed (Facebook), livedoor, Six Apart, LiveJournal, Posterous, Pheedo, TwitterFeed, Superfeedr, LazyFeed, ...
Progress

- Open-source libraries
  - Perl, PHP, Python, Ruby, Java, Haskell, C#, MovableType, WordPress, Django, Zend, Drupal, ...

- Open-source hubs
  - App Engine, Erlang, Python, Ruby, Perl, ...
Progress

- Will be licensed under Open Web Foundation
  - No Patents, Royalty-free, etc
Getting involved

- Active mailing list, 250+ members
  - Join!

- Review the specification yourself
  - Open process for feedback and peer review

- Write some sample code; contribute to open source

- Write on your blog why we need **push** for the future
Conclusion
Conclusion

- Make your feeds real-time with PubSubHubbub
- Enable new use-cases in the future
- Push it to the limit!
pubsubhubbub.googlecode.com
Questions?