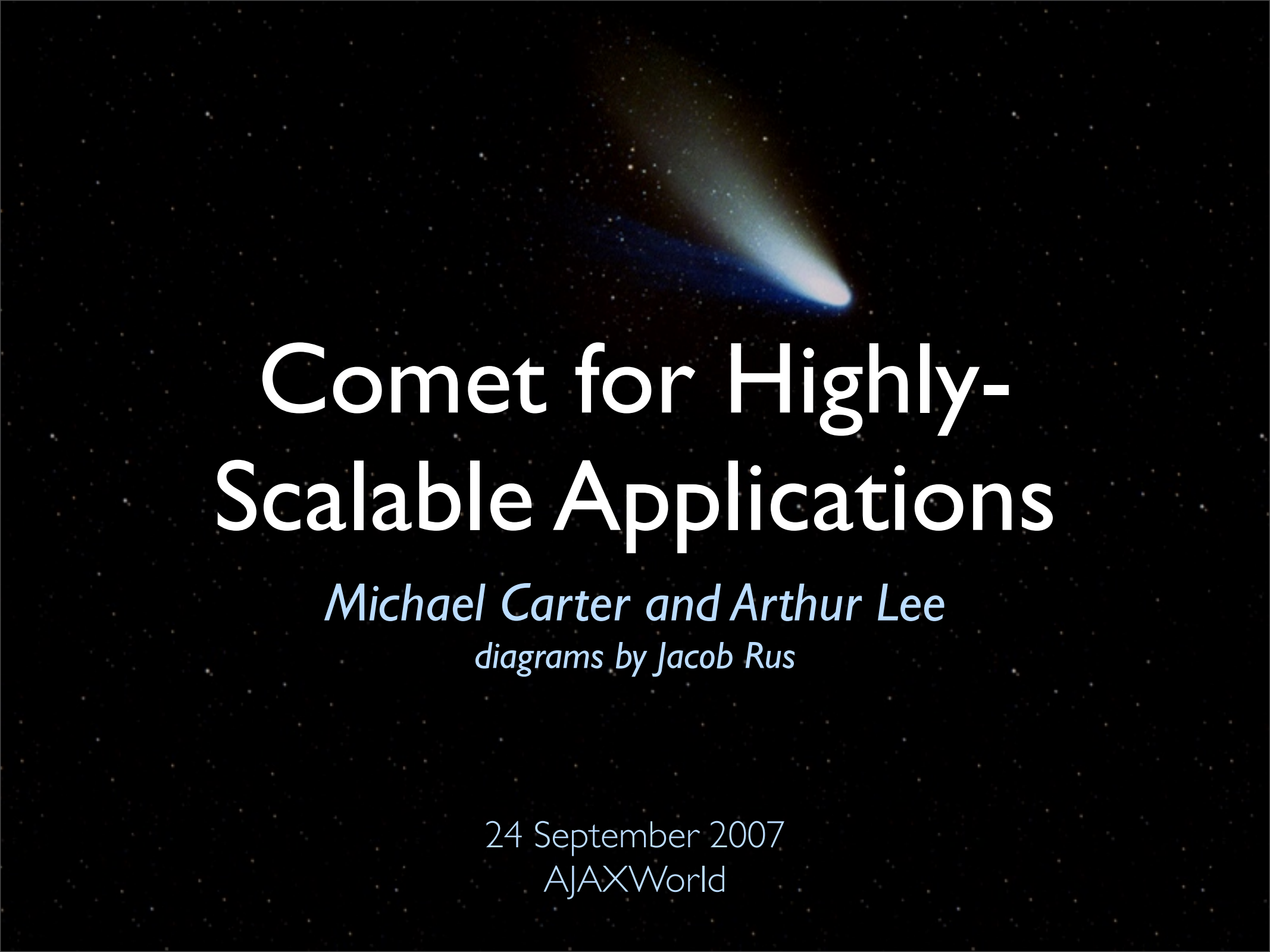


Comet for Highly-Scalable Applications

Michael Carter and Arthur Lee

diagrams by Jacob Rus

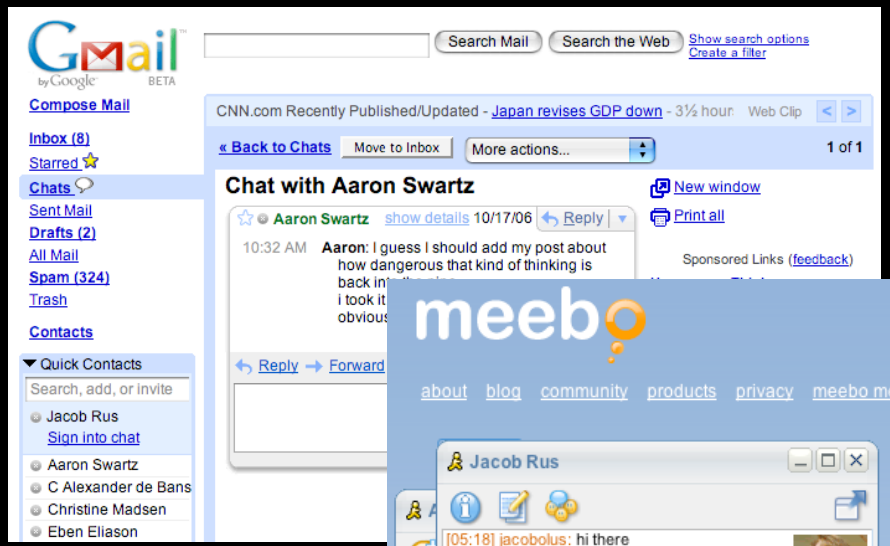
24 September 2007

AJAXWorld

What is Comet?

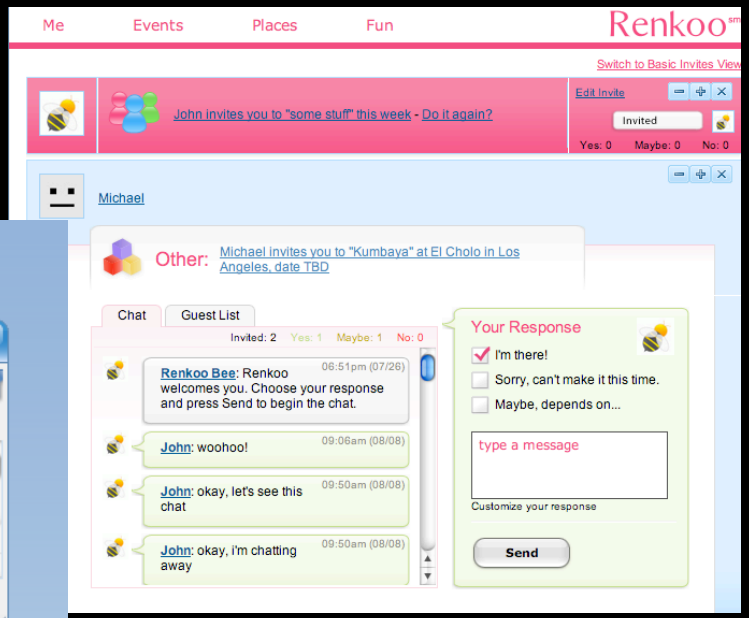
"Comet" describes a model of user interaction

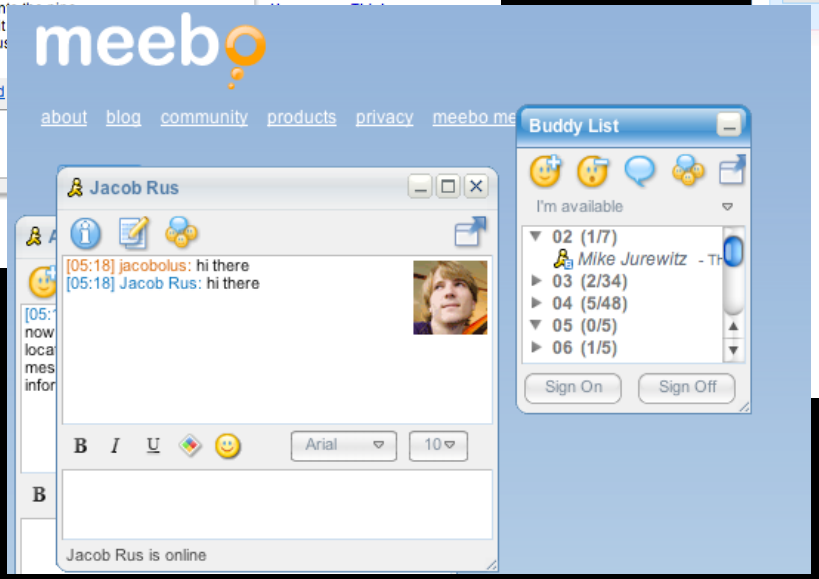
Pushing real-time updates to web browsers



Gmail Chat

Renkoo

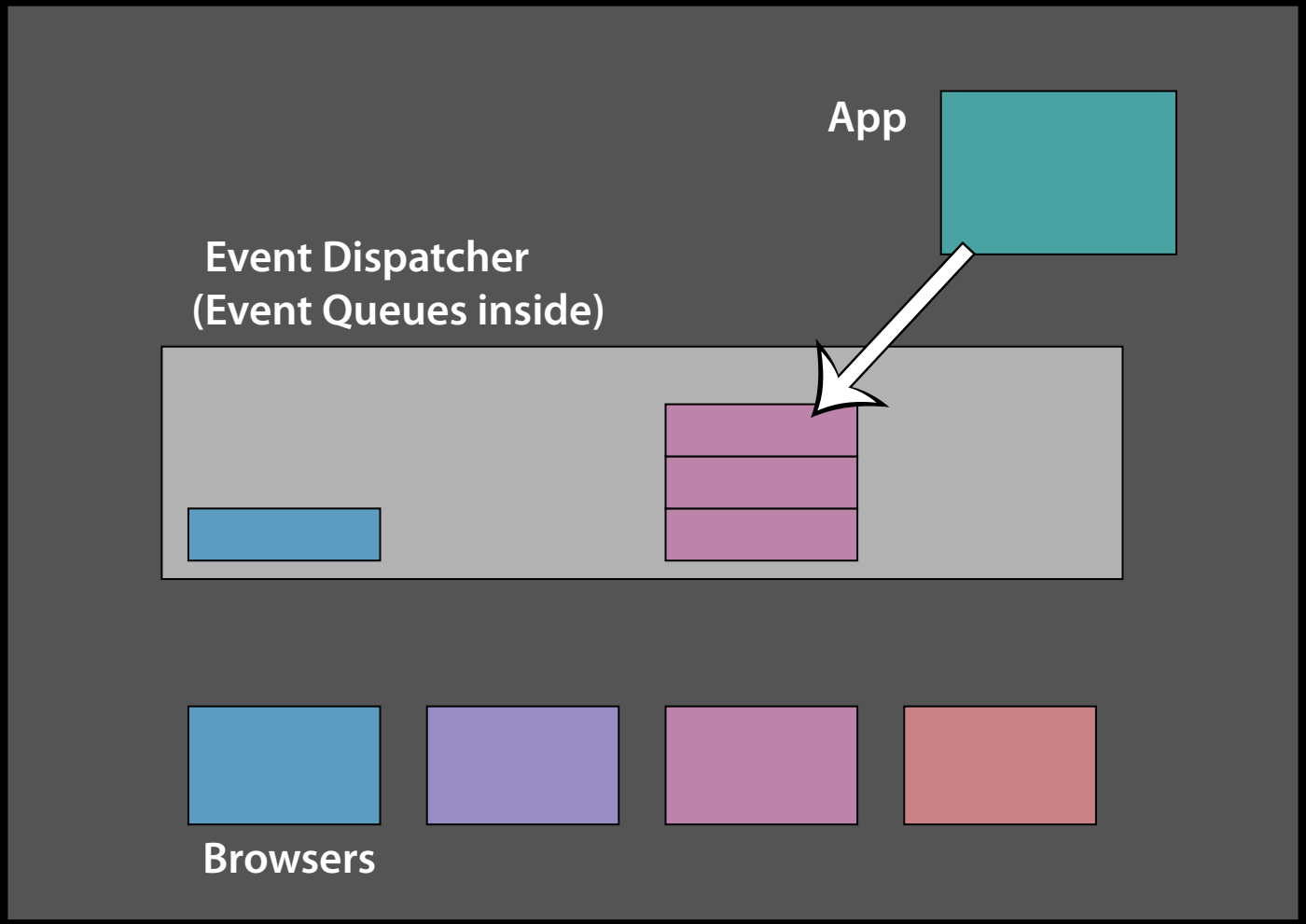




Meebo

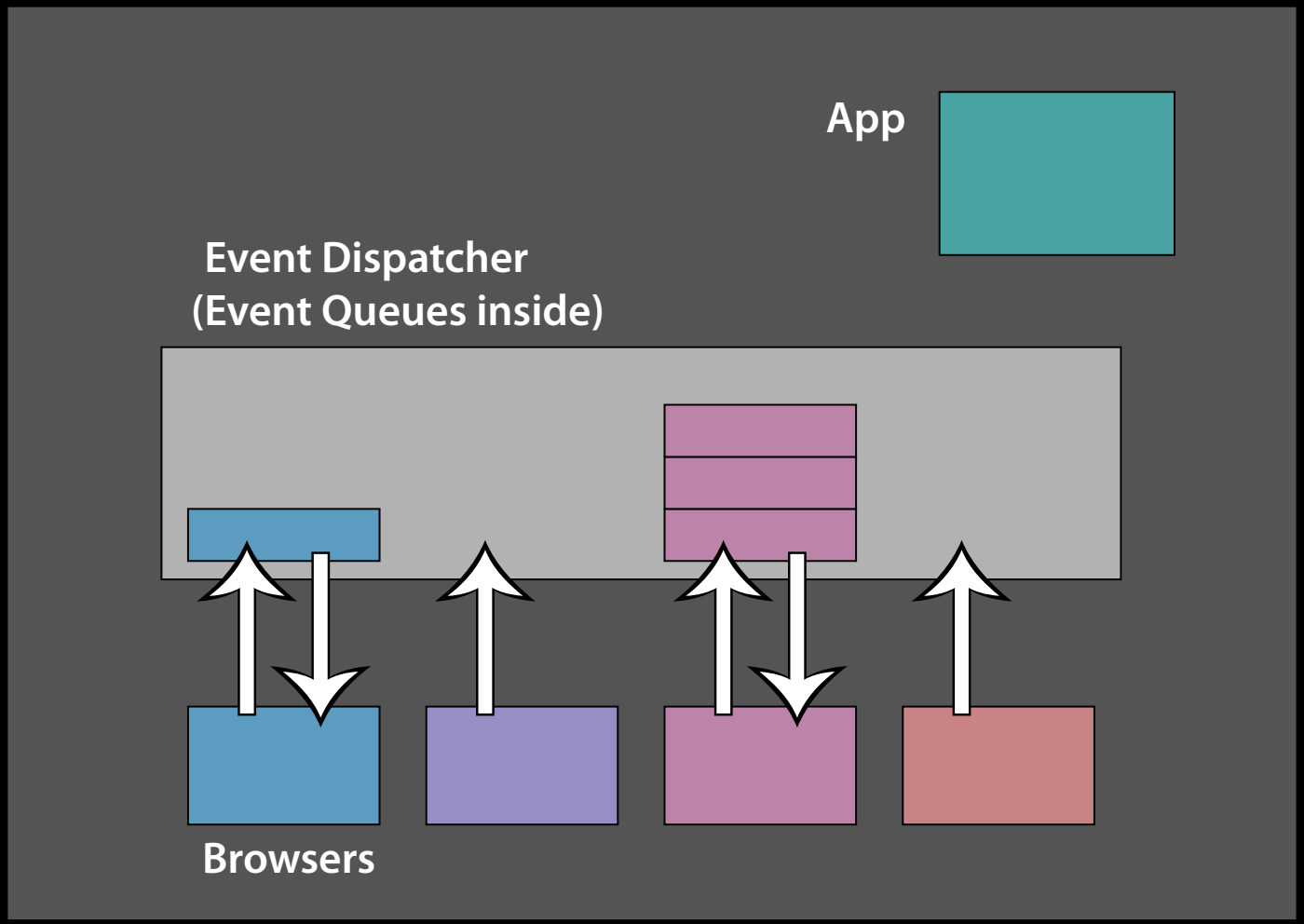
Old Method: Polling

Otherwise known as "hammering the server"



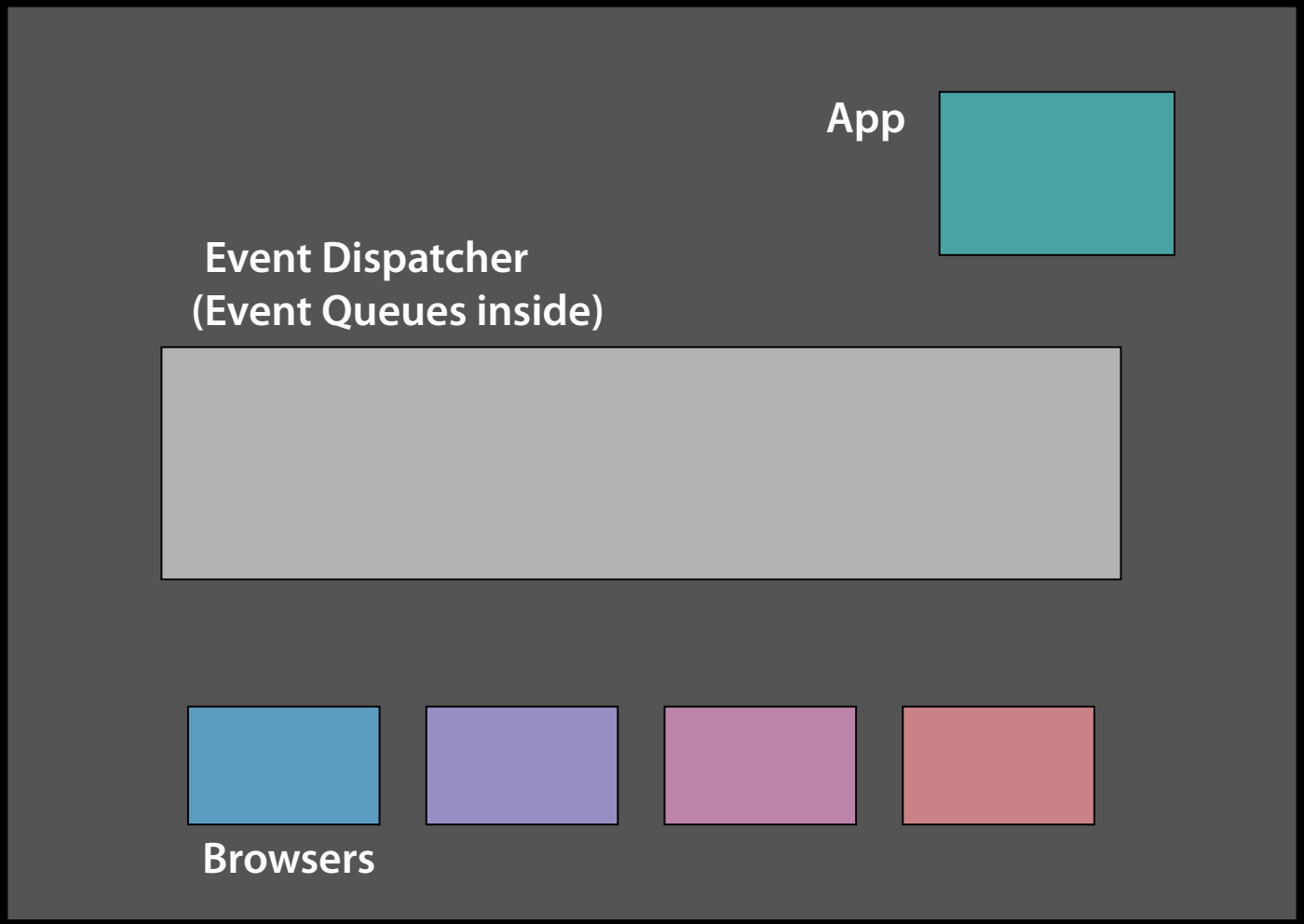
Old Method: Polling

Otherwise known as "hammering the server"



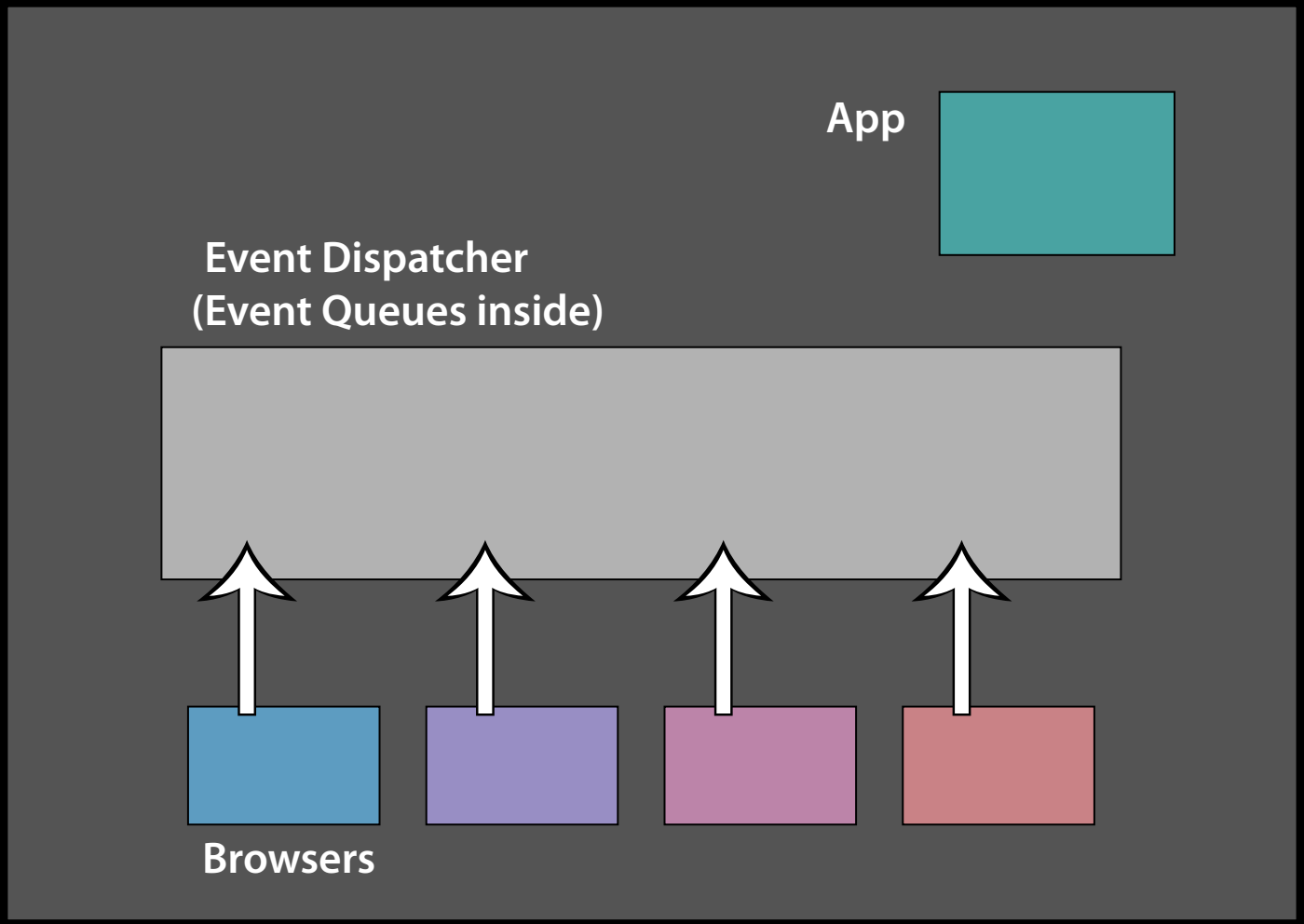
Old Method: Polling

Otherwise known as "hammering the server"



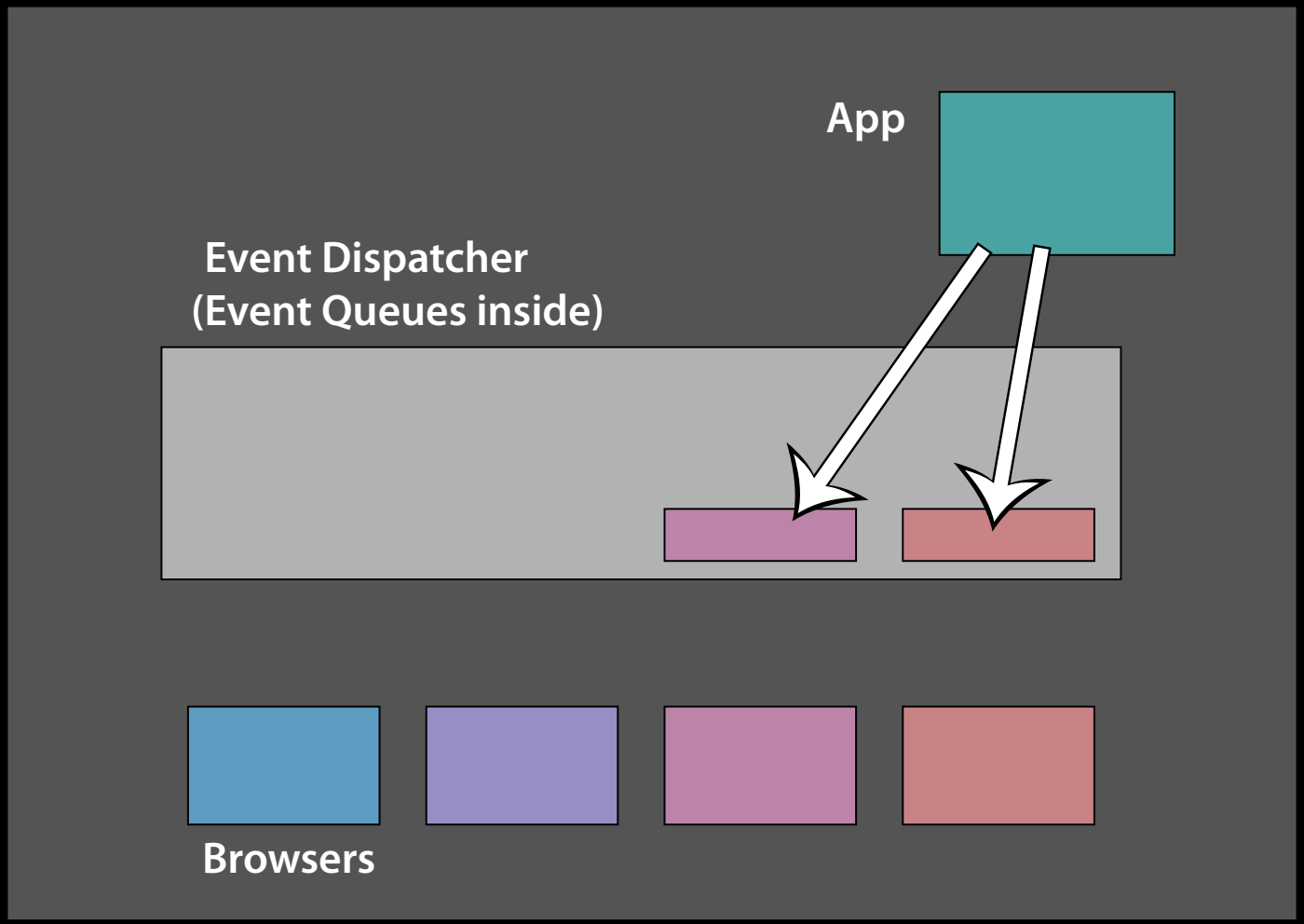
Old Method: Polling

Otherwise known as "hammering the server"



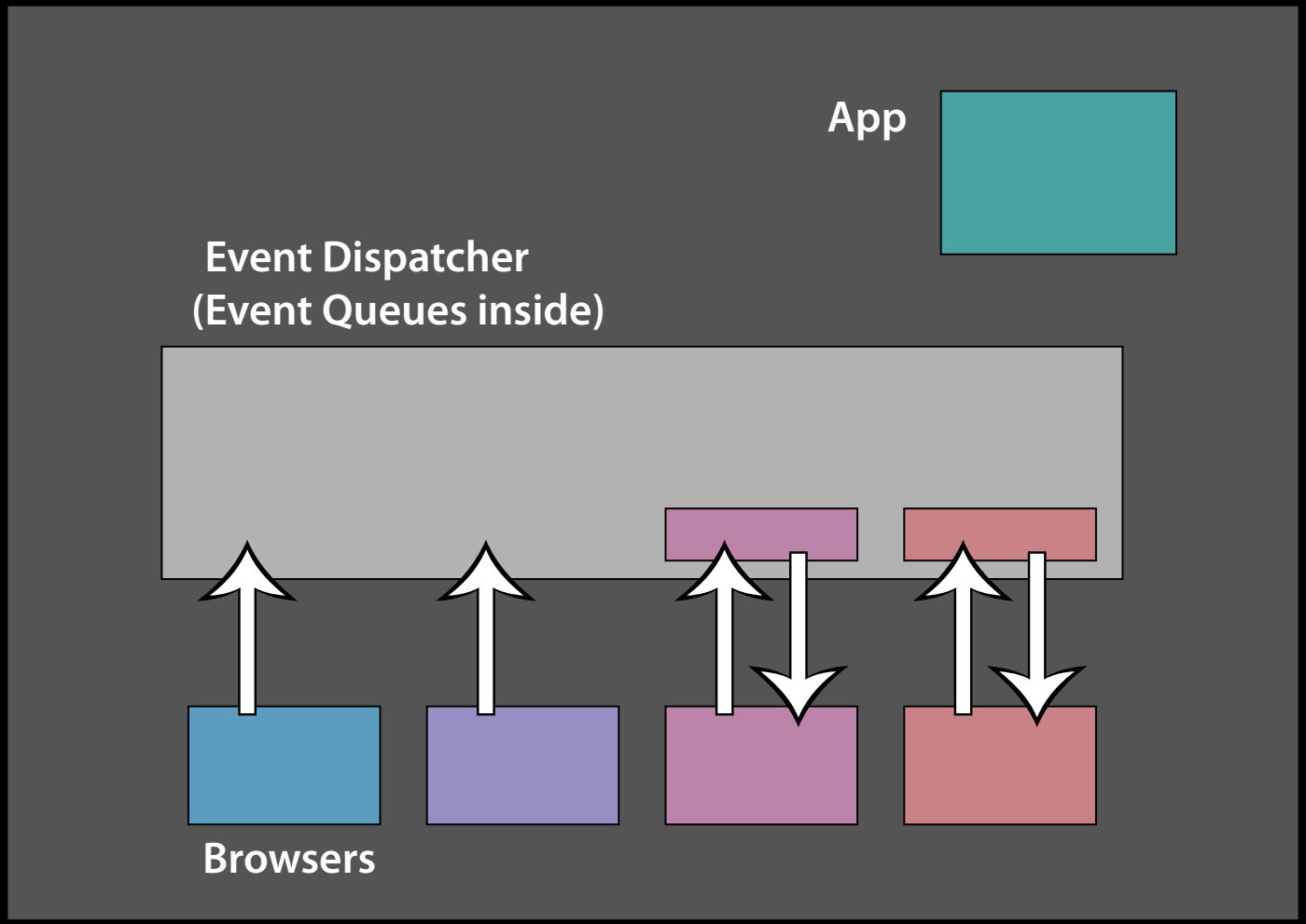
Old Method: Polling

Otherwise known as "hammering the server"



Old Method: Polling

Otherwise known as "hammering the server"



Polling Overview

Old and busted

Pros:

- Easy to implement
  - Client side
  - Server side

Cons:

- Wasted requests
- Additional latency

Enter Long-Polling

The new hotness

Pros:

- Latency—gone!
- Client side still easy

Cons:

- Difficult to scale server-side
- Conceptually more complicated

Vertical Scaling:

How do you scale comet on a single server?

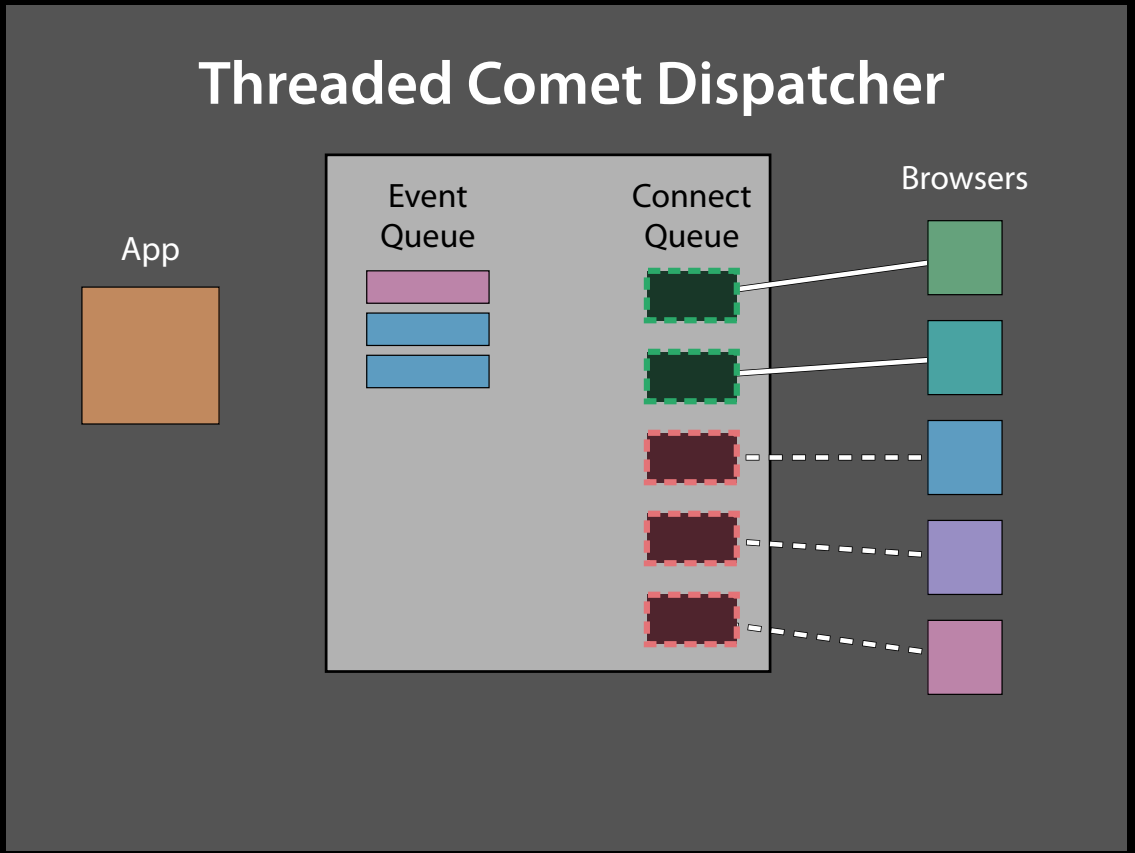
photo by Michael (mx5tx)

Vertical Scaling:

How do you scale comet on a single server?

Wrong way: Use a threaded server like Apache or IIS

- Supports a few dozen users
- CPU usage and latency go through the roof as users are added

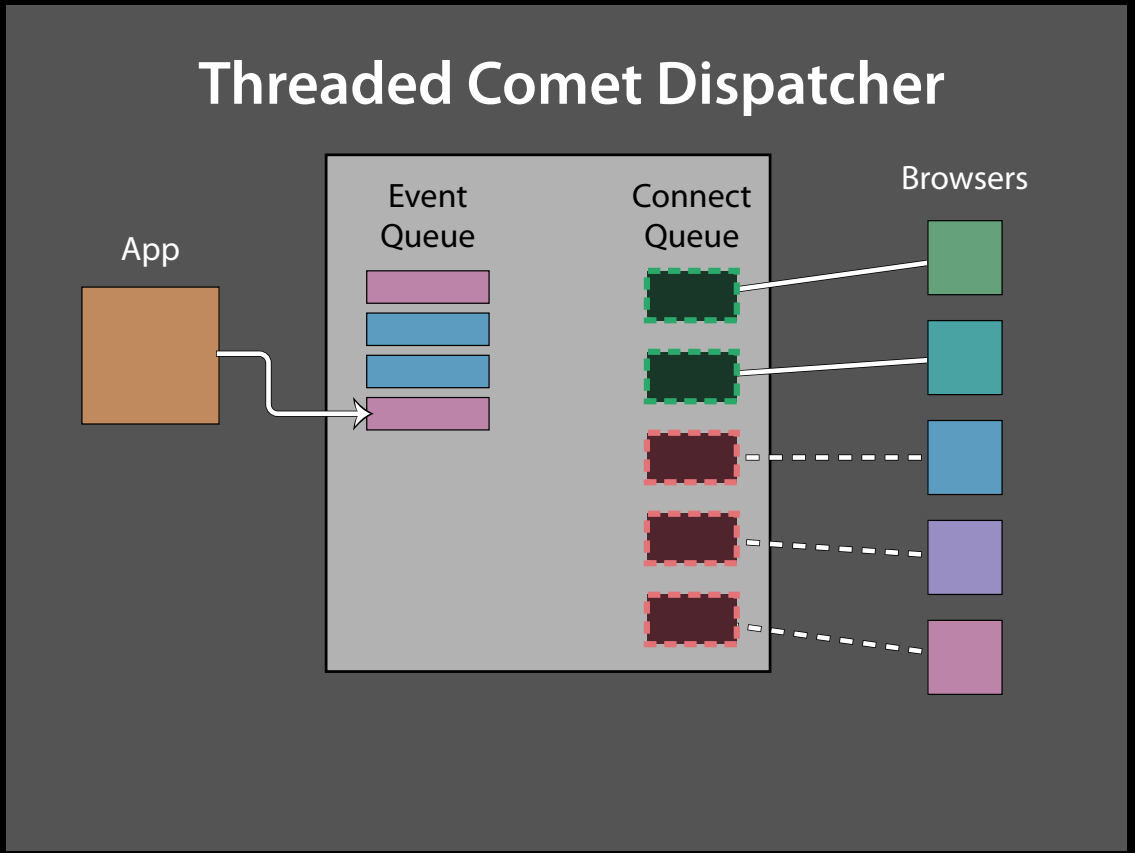


Vertical Scaling:

How do you scale comet on a single server?

Wrong way: Use a threaded server like Apache or IIS

- Supports a few dozen users
- CPU usage and latency go through the roof as users are added

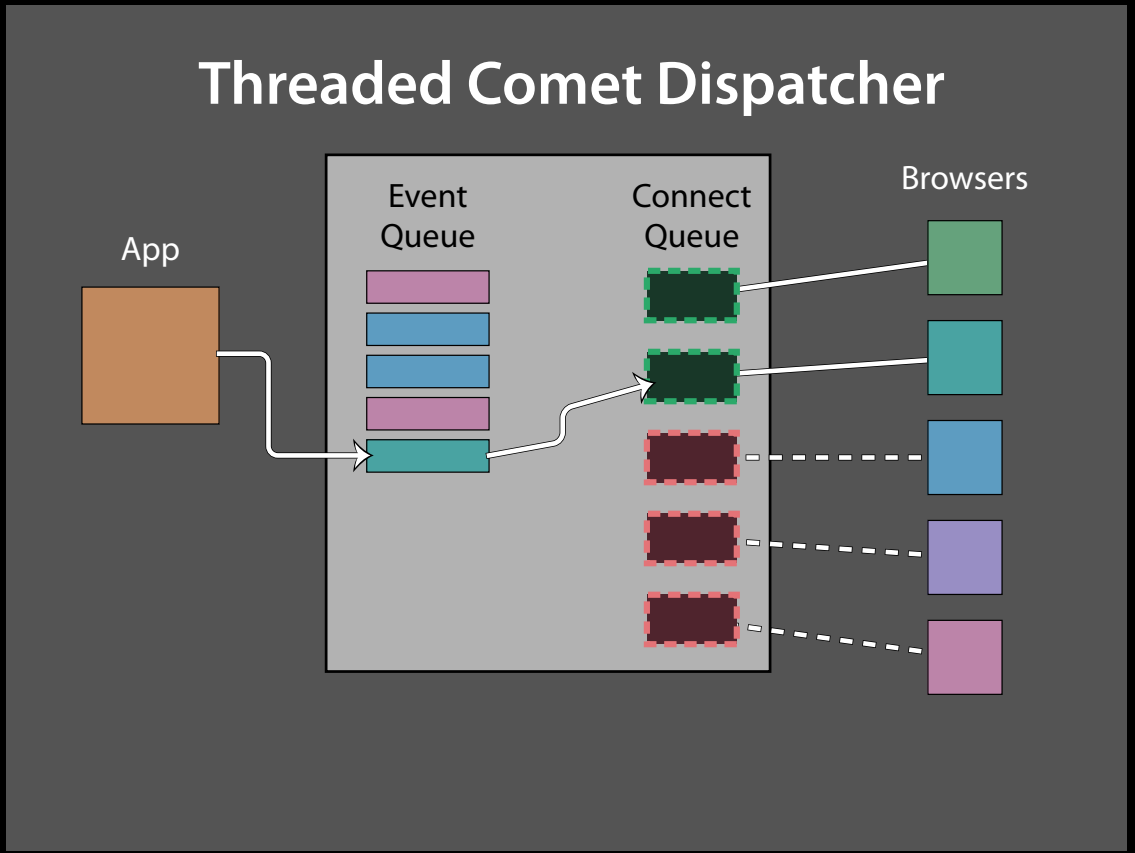


Vertical Scaling:

How do you scale comet on a single server?

Wrong way: Use a threaded server like Apache or IIS

- Supports a few dozen users
- CPU usage and latency go through the roof as users are added

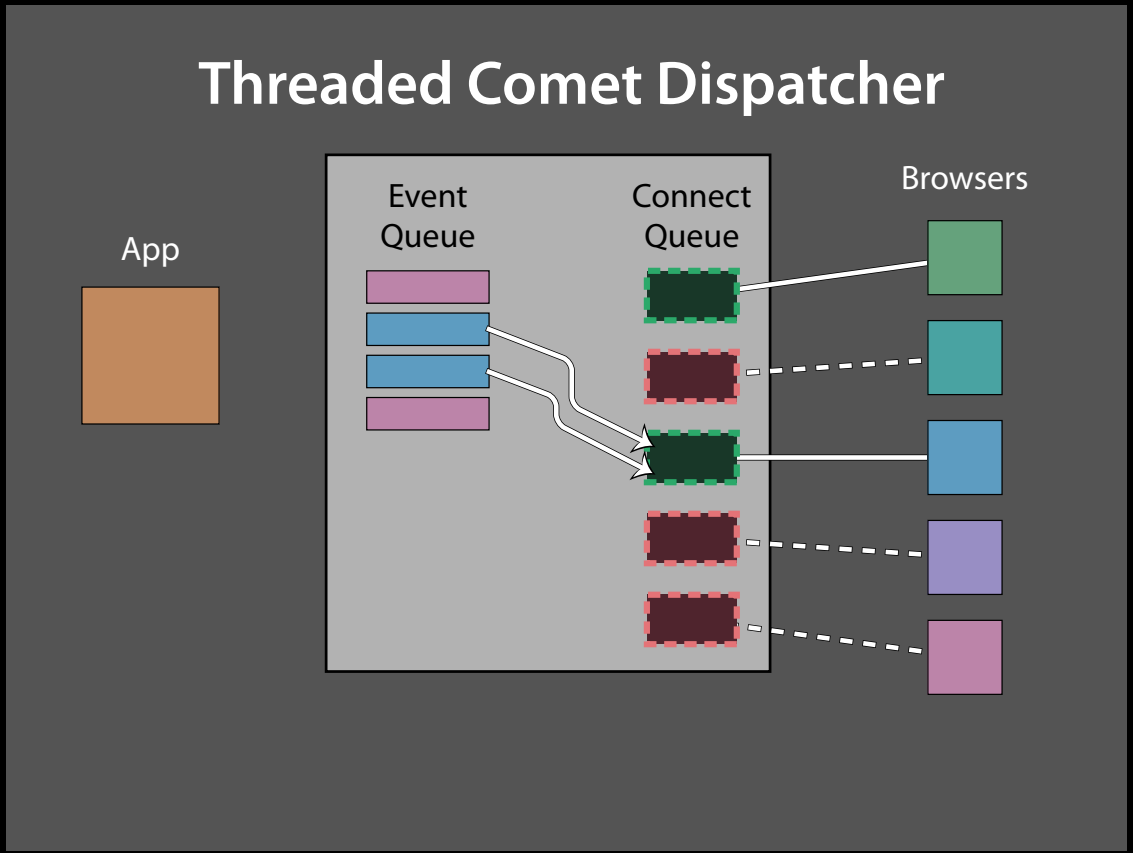


Vertical Scaling:

How do you scale comet on a single server?

Wrong way: Use a threaded server like Apache or IIS

- Supports a few dozen users
- CPU usage and latency go through the roof as users are added

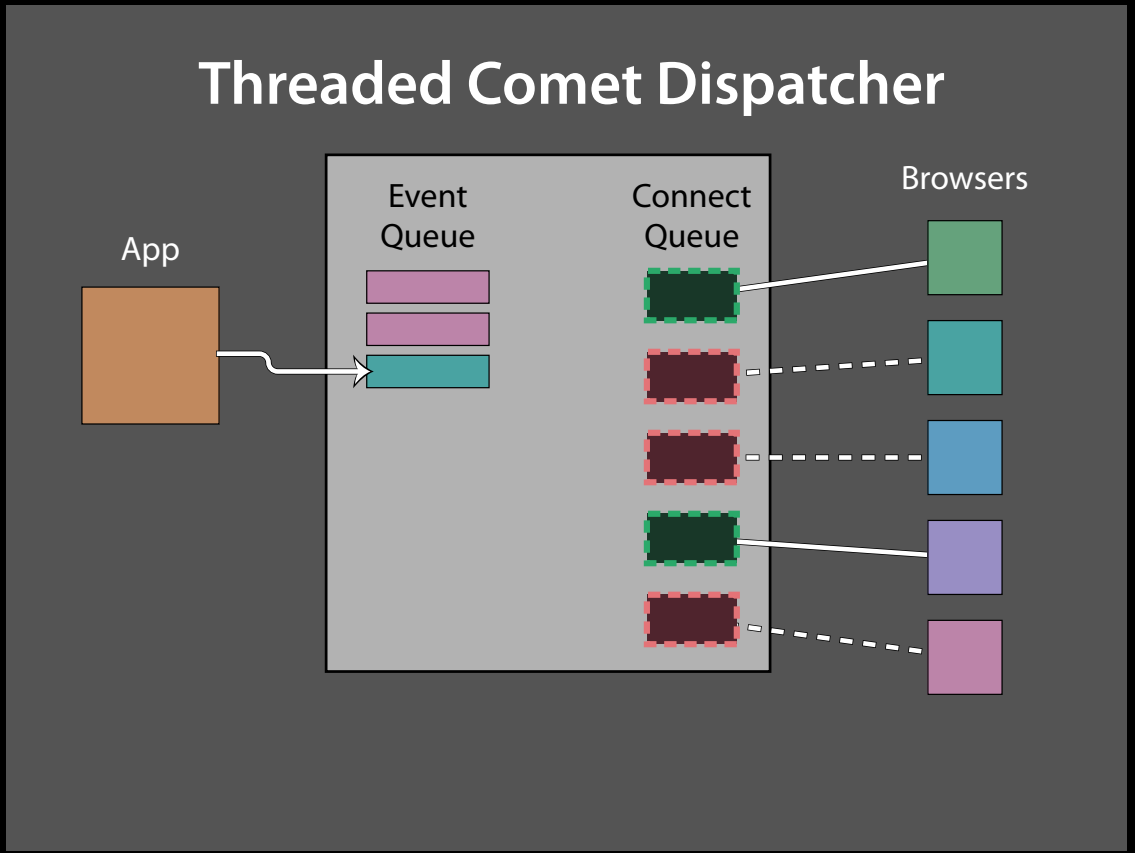


Vertical Scaling:

How do you scale comet on a single server?

Wrong way: Use a threaded server like Apache or IIS

- Supports a few dozen users
- CPU usage and latency go through the roof as users are added



Vertical Scaling:

How do you scale comet on a single server?

Wrong way: Use a threaded server like Apache or IIS

Compared to polling:

⇒ Latency reduced for very light load

⇒ Avg. latency depends on the order of events

⇒ As much as 7× the CPU for moderate load

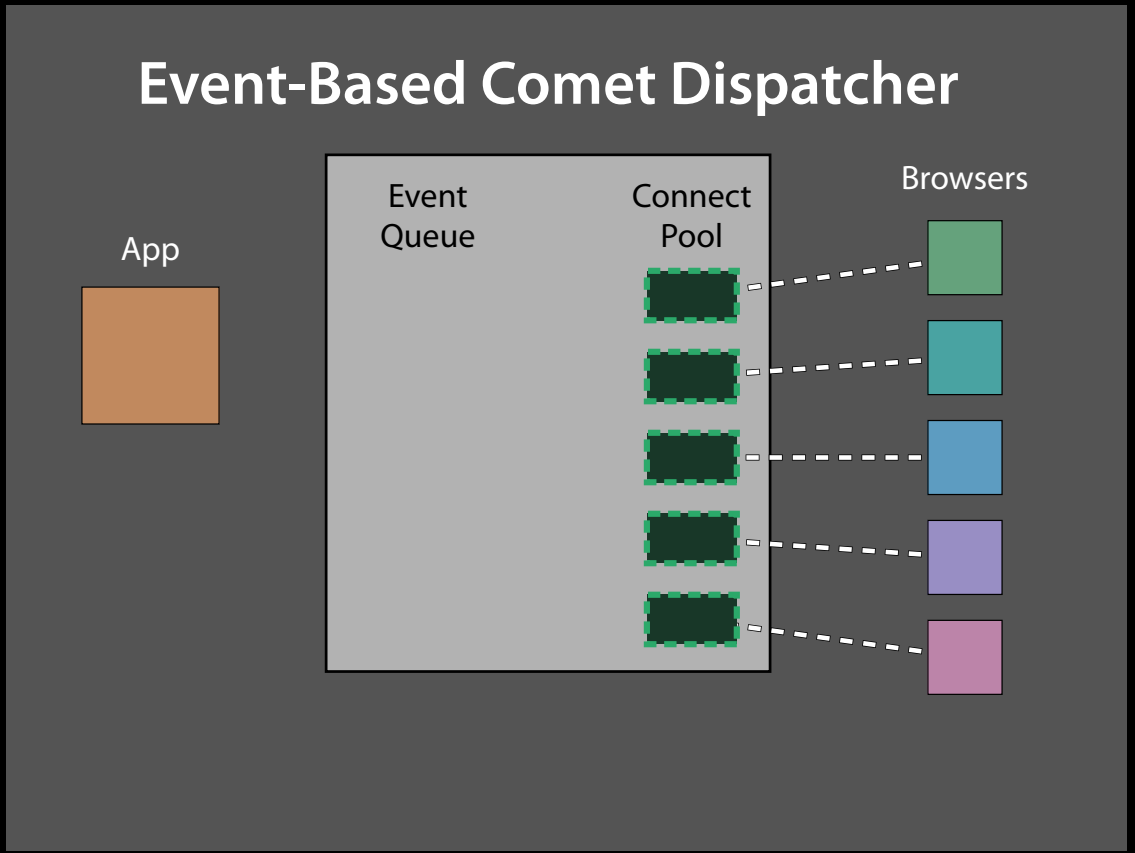
⇒ CPU-bound at moderate load

Vertical Scaling:

How do you scale comet on a single server?

Right way: Use an event-based architecture

- Supports as many users as CPU and I/O can handle
- CPU usage scales linearly and latency remains low

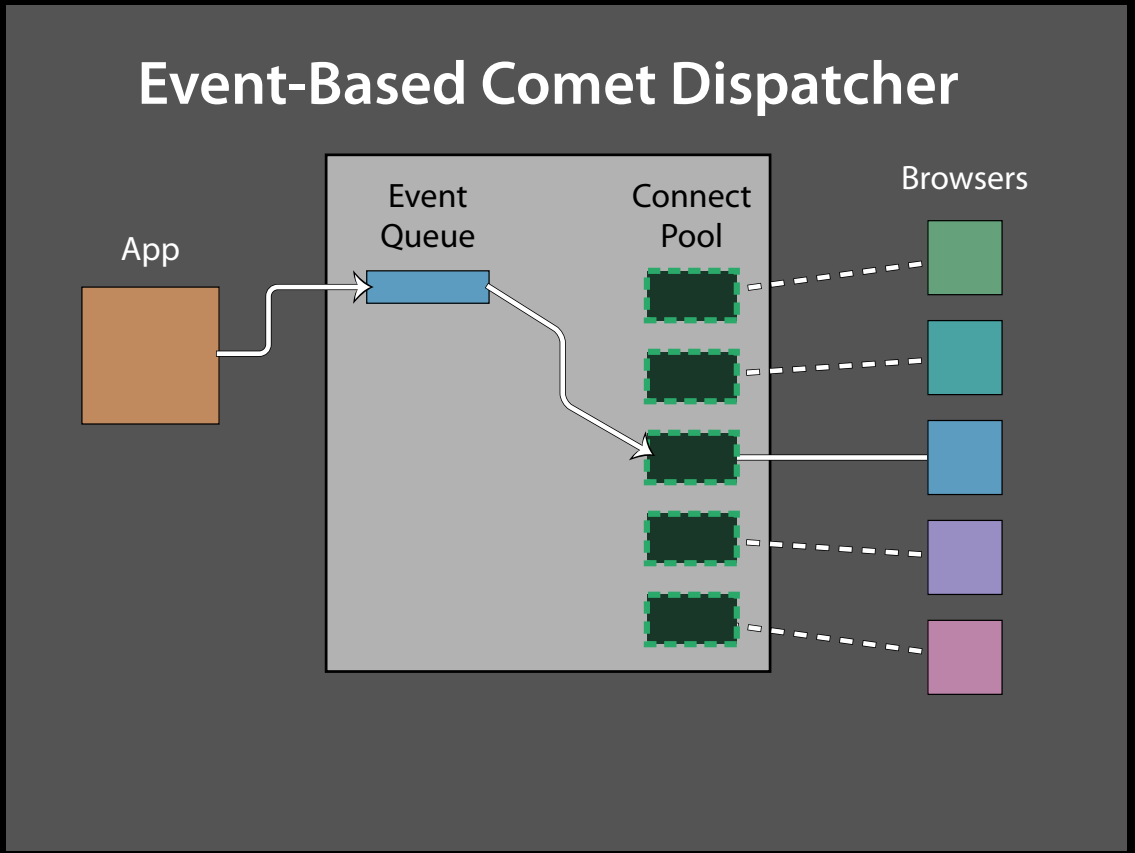


Vertical Scaling:

How do you scale comet on a single server?

Right way: Use an event-based architecture

- Supports as many users as CPU and I/O can handle
- CPU usage scales linearly and latency remains low

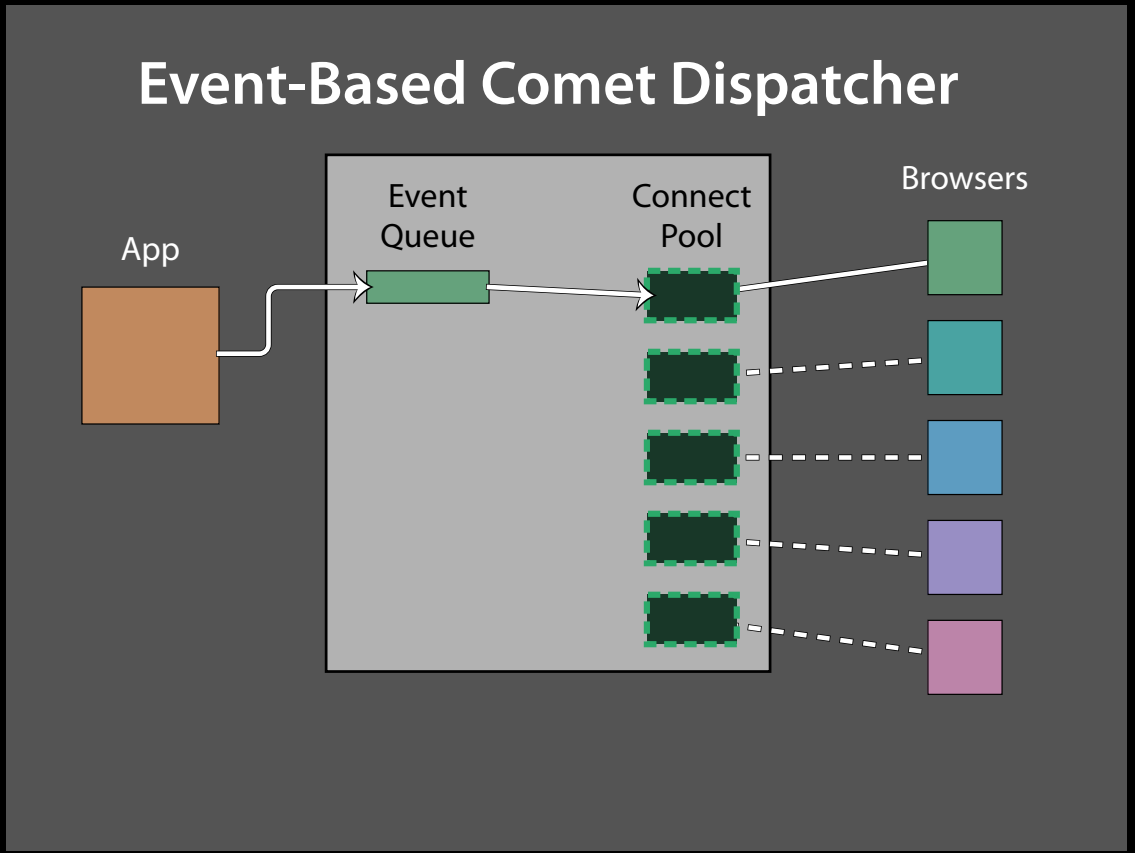


Vertical Scaling:

How do you scale comet on a single server?

Right way: Use an event-based architecture

- Supports as many users as CPU and I/O can handle
- CPU usage scales linearly and latency remains low



Vertical Scaling:

How do you scale comet on a single server?

Right way: Use an event-based architecture

Compared to polling:

⇒ Decreased latency and CPU usage

Compared to threaded Comet:

⇒ Vastly decreased CPU usage

⇒ Order of events irrelevant

⇒ Performance degrades gracefully

Horizontal Scaling:

How do you scale comet across many servers?

photo by ulybug

Horizontal Scaling:

How do you scale comet across many servers?

Wrong way: Build the whole software stack into the comet server

⇒ Embed publish-subscribe architecture into comet server

⇒ Keep application state within comet nodes

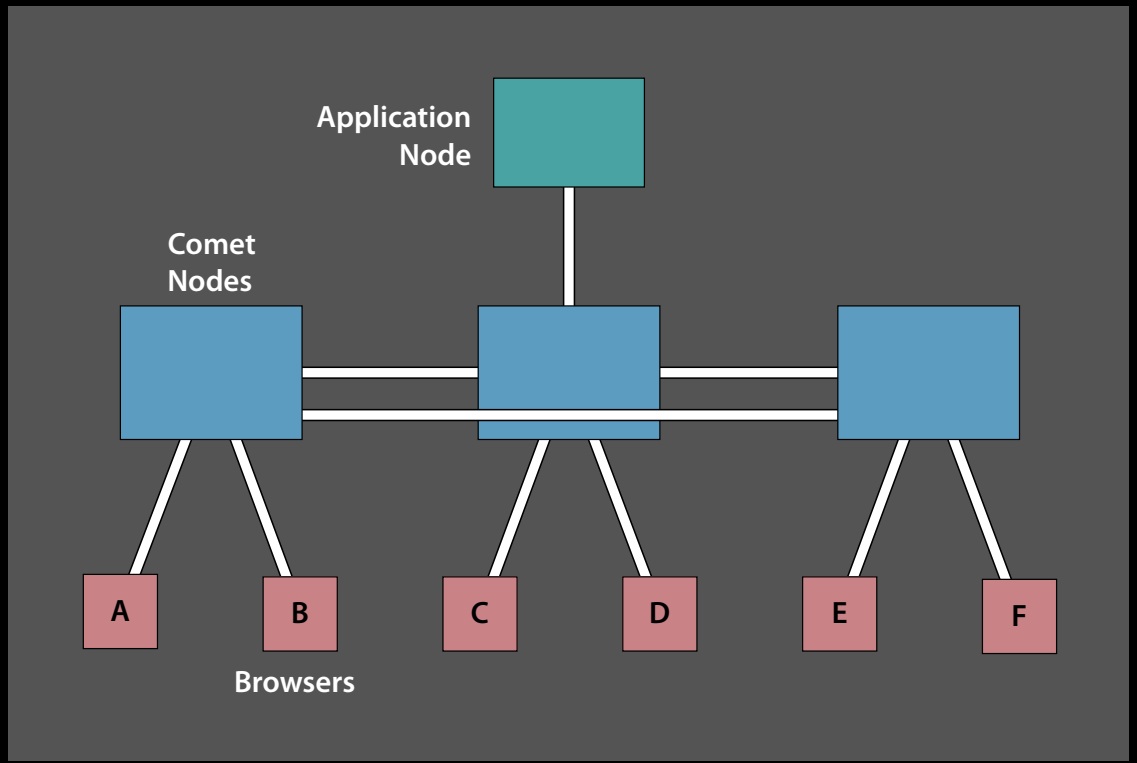
⇒ Allow comet nodes to communicate with each-other

Horizontal Scaling:

How do you scale comet across many servers?

Wrong way: Build the whole software stack into the comet server

- Design is inflexible, can't scale for some uses
- Hard to scale to many nodes in the best case

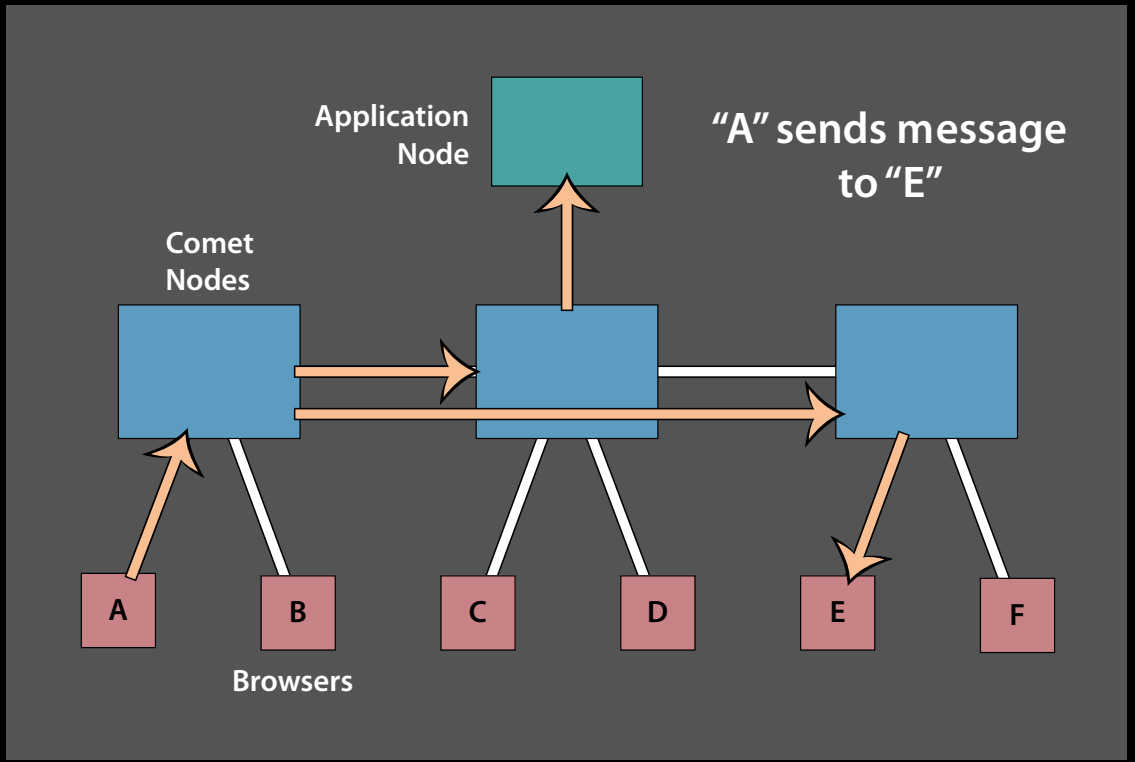


Horizontal Scaling:

How do you scale comet across many servers?

Wrong way: Build the whole software stack into the comet server

- Design is inflexible, can't scale for some uses
- Hard to scale to many nodes in the best case

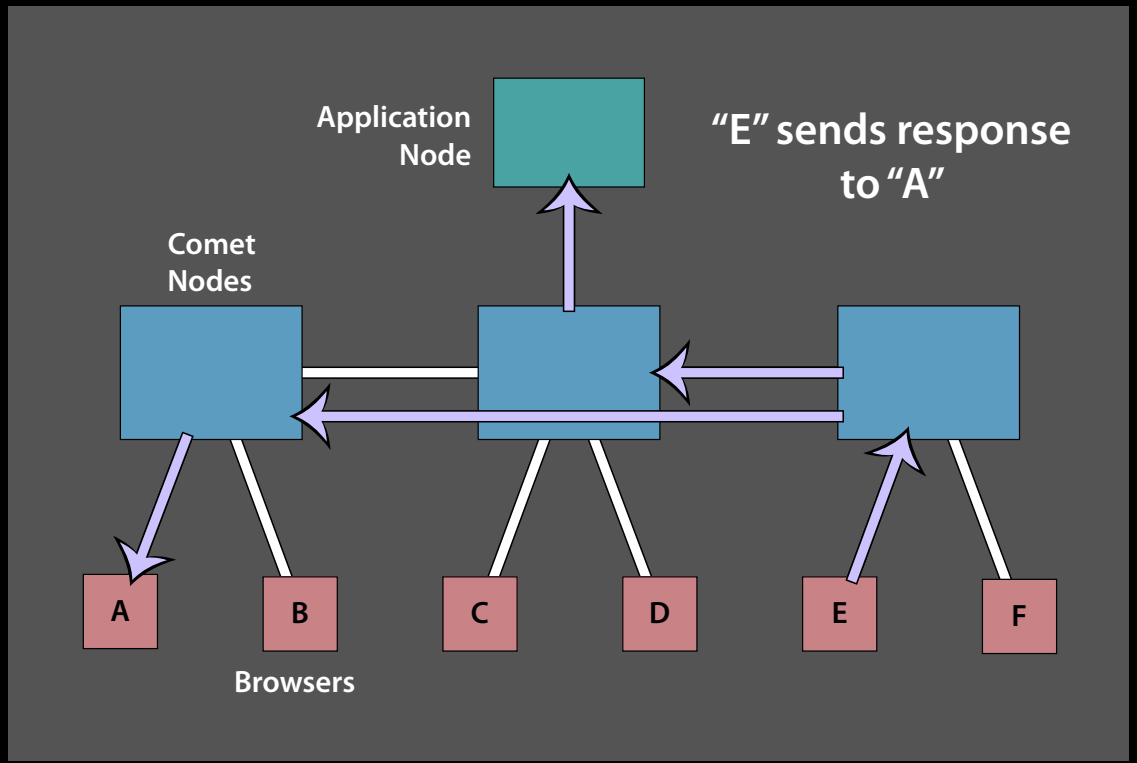


Horizontal Scaling:

How do you scale comet across many servers?

Wrong way: Build the whole software stack into the comet server

- Design is inflexible, can't scale for some uses
- Hard to scale to many nodes in the best case

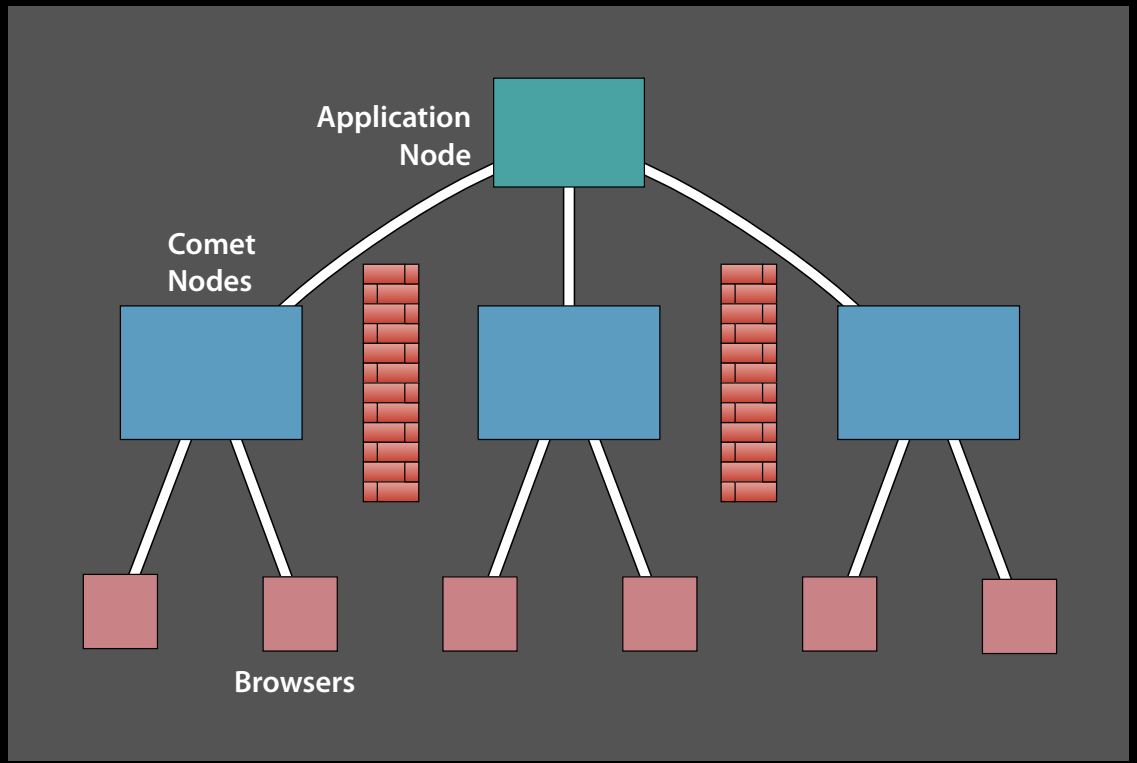


Horizontal Scaling:

How do you scale comet across many servers?

Right way: Share nothing between nodes

- No CPU or I/O overhead of sharing state
- CPU usage scales linearly and latency remains low



Horizontal Scaling:

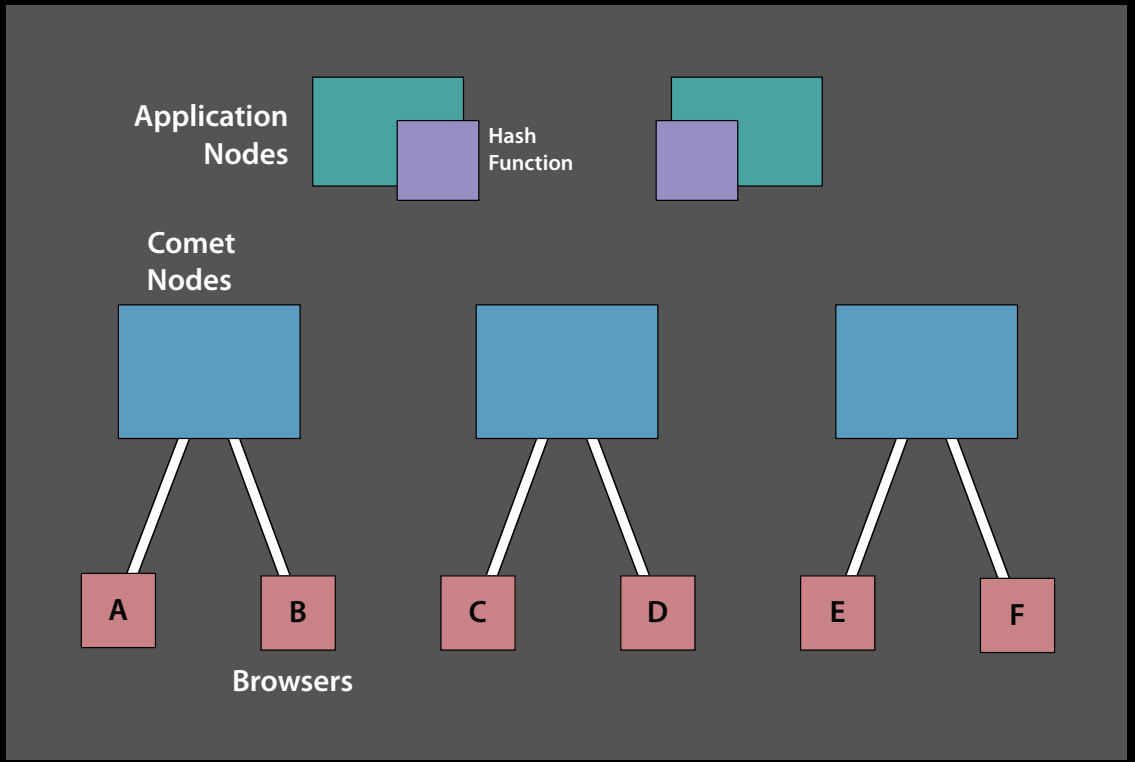
How do you scale comet across many servers?

Right way: Use Orbited *«wink»*

⇒ Distributed hash table

⇒ Pre-defined hash function

⇒ Leave authenti-cation to app layer



Horizontal Scaling:

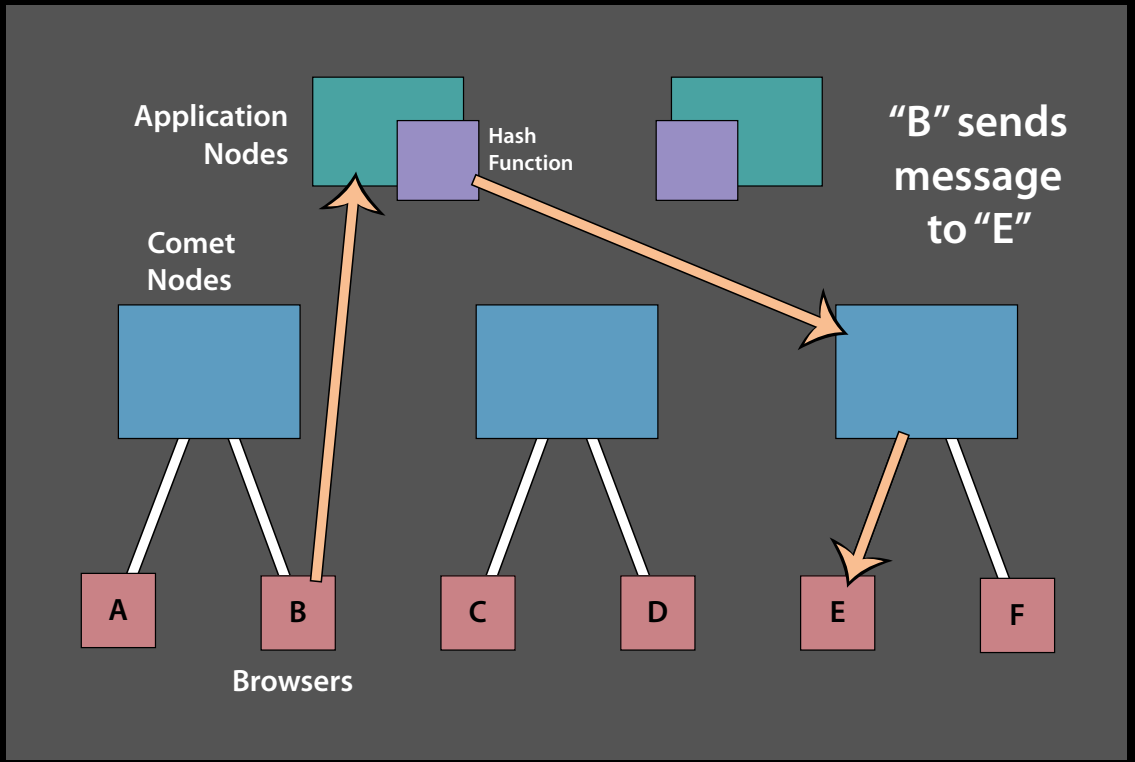
How do you scale comet across many servers?

Right way: Use Orbited *«wink»*

⇒ Distributed hash table

⇒ Pre-defined hash function

⇒ Leave authenti-cation to app layer



Horizontal Scaling:

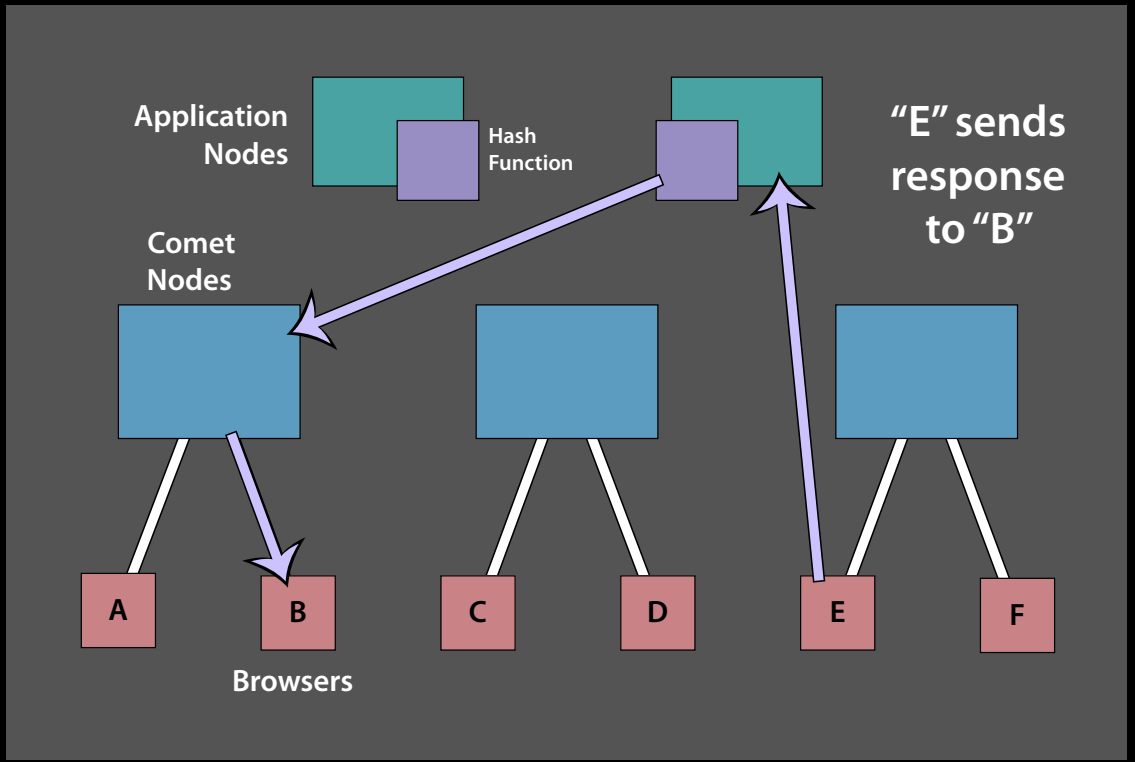
How do you scale comet across many servers?

Right way: Use Orbited *«wink»*

⇒ Distributed hash table

⇒ Pre-defined hash function

⇒ Leave authenti-cation to app layer



Publish / Subscribe

Pushing published events to groups of browsers

What's it good for? Applications for which multiple users need the same live updates

Publish / Subscribe

Pushing published events to groups of browsers

What's it good for? Applications for which multiple users need the same live updates

real-time chat

Publish / Subscribe

Pushing published events to groups of browsers

What's it good for? Applications for which multiple users need the same live updates

real-time chat

stock tickers

Scaling Pub/Sub

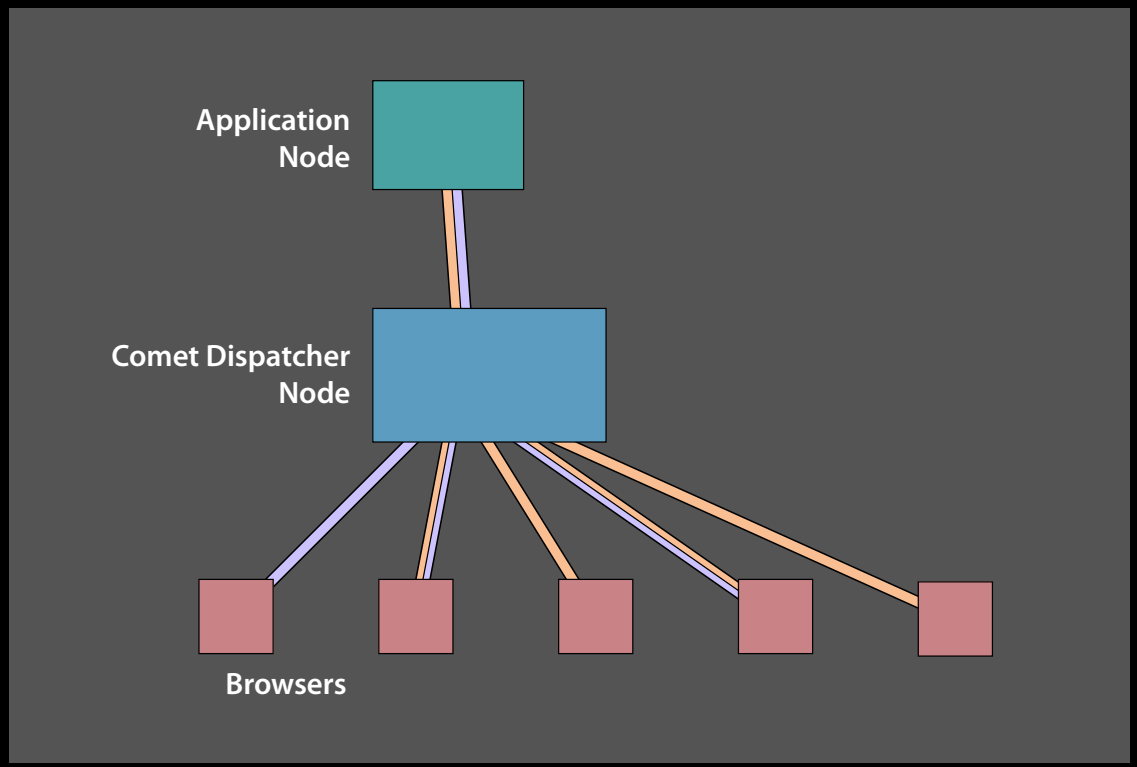
How do you scale publish/subscribe horizontally?

If we start with a *single* comet node, which dispatches events to multiple users in multiple groups then how do we scale up to *multiple* comet nodes?

Scaling Pub/Sub

How do you scale publish/subscribe horizontally?

If we start with a single node, which dispatches events to multiple users in multiple groups…

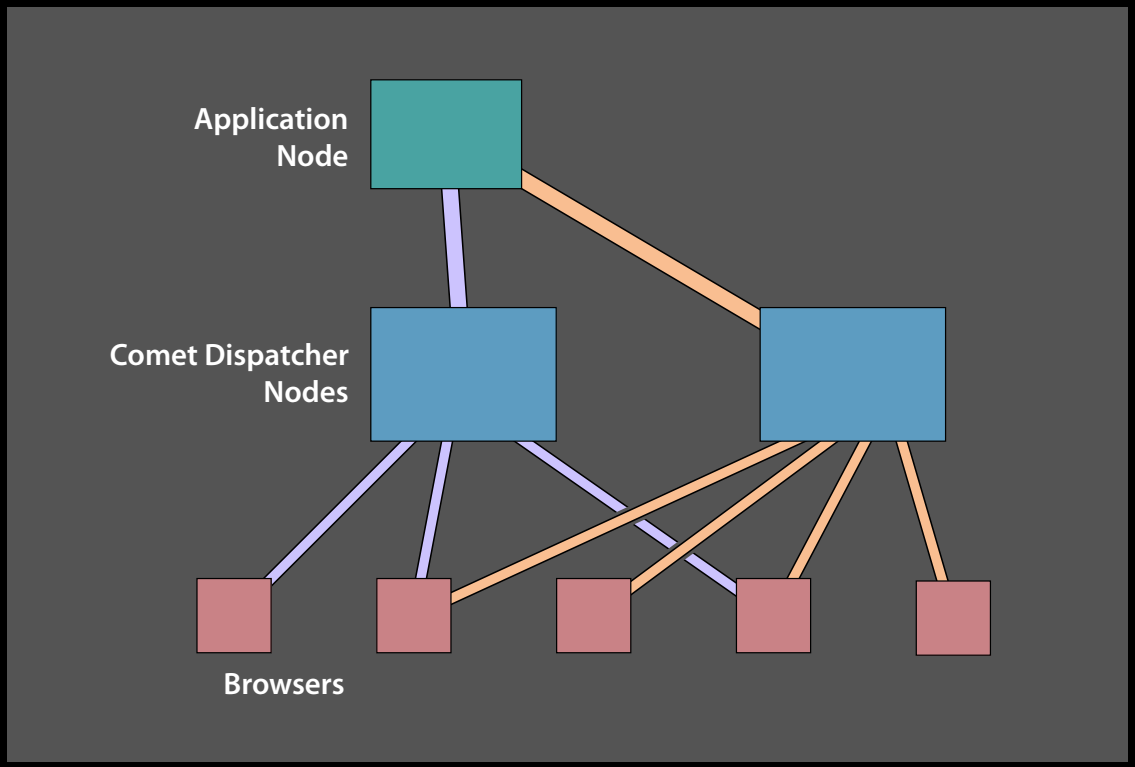


A single comet node serves all users and all groups

Scaling Pub/Sub

How do you scale publish/subscribe horizontally?

…then how do we scale up to a pair of nodes?



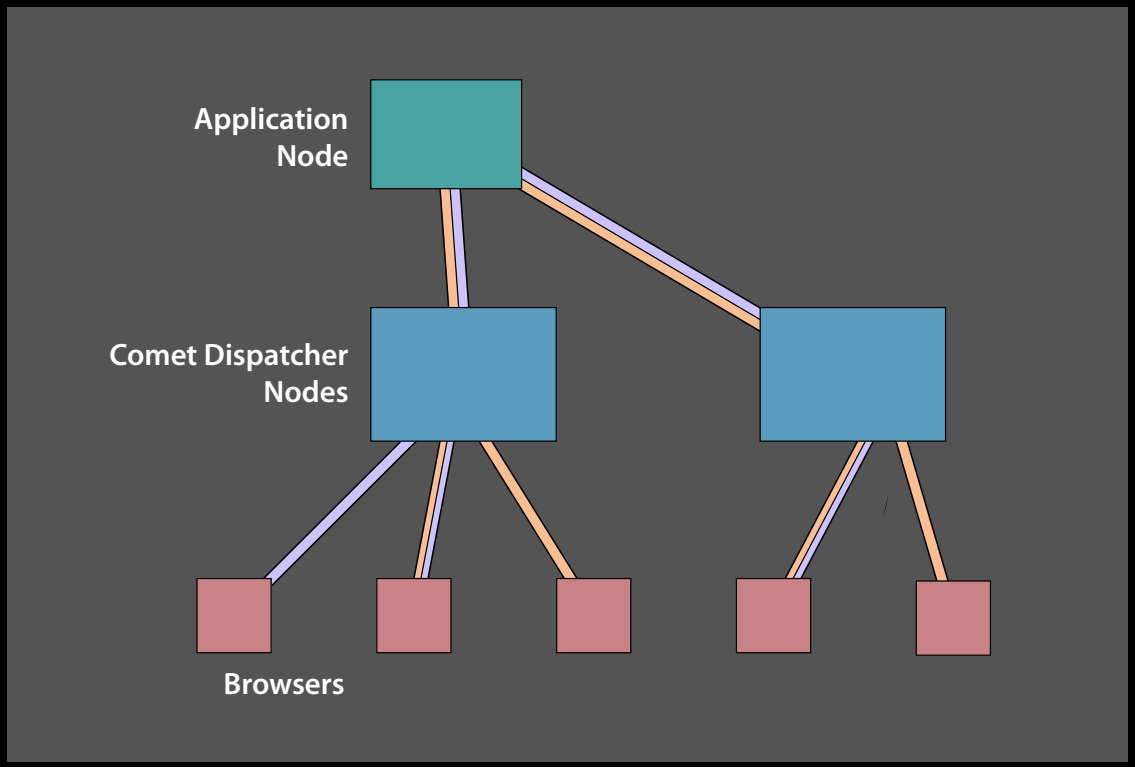
Either we arrange nodes by group:

All of a group's events are sent by one node

Scaling Pub/Sub

How do you scale publish/subscribe horizontally?

…then how do we scale up to a pair of nodes?



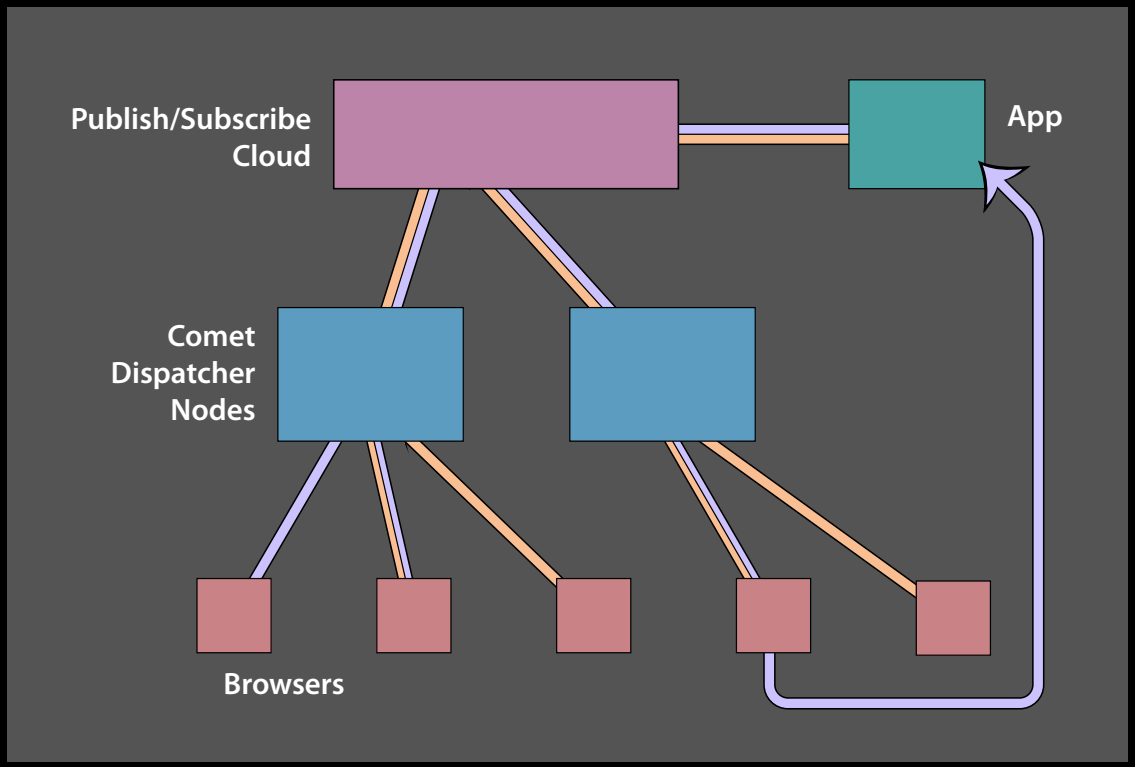
Or we arrange nodes by user:

A user receives all events from the same node

Scaling Pub/Sub

How do you scale publish/subscribe horizontally?

But how, specifically? Split the architecture into layers



⇒ Do not mix pub/sub layer with comet dispatcher

⇒ Treat pub/sub layer as a black box

Pub/Sub Options

Publish/subscribe already has solutions

Don't reinvent the wheel: Many well-understood, scalable publish/subscribe solutions exist

- Internet Relay Chat: Nearly 20 years old. Scales well for large groups of users.
- XMPP (Jabber): Flexible protocol. Does presence well, in addition to messaging.
- Java Message Service: It's enterprise, baby.

Pub/Sub with Orbited

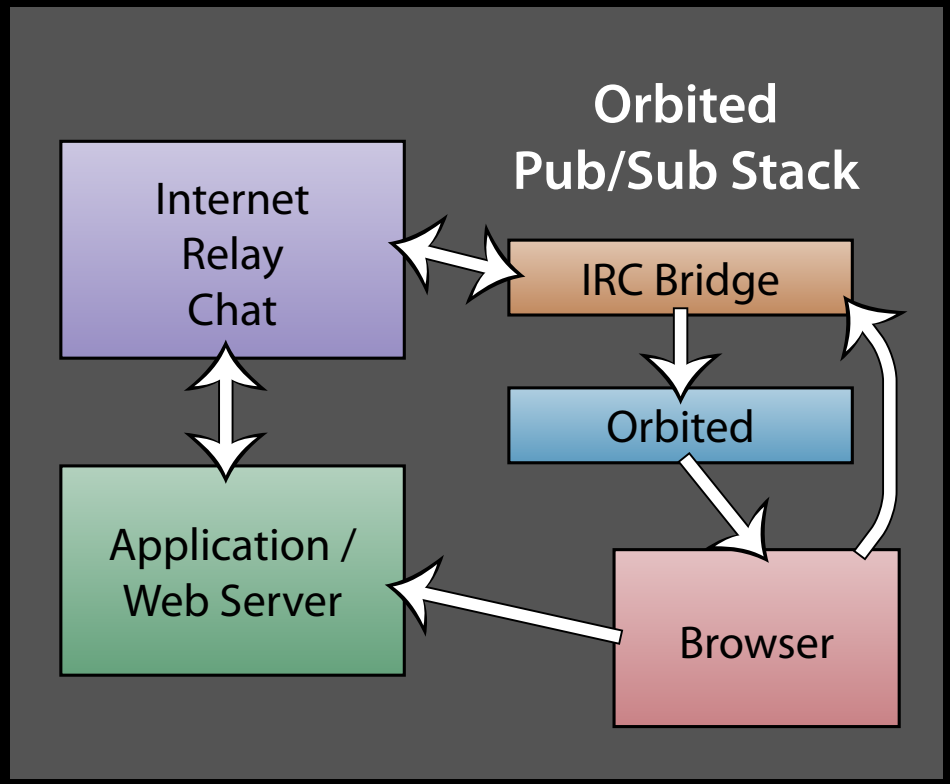
Orbited can be easily added to a pub/sub stack

Scaling through modularity: Its simplicity makes Orbited easy to add to any web app architecture

⇒ Orbited handles comet dispatch

⇒ IRC or Jabber handles publish/ subscribe

⇒ App servers do everything else



Conclusion:

- Polling latency unacceptable
- Comet solves this problem
- Impossible to scale Comet with threaded design
- Event-based architecture is necessary
- Horizontal scaling requires “share nothing” design
- Publish/subscribe is a difficult problem
- Scaling pub/sub is easiest with layered architecture

Michael Carter

CarterMichael@gmail.com
http://www.orbited.org/

24 September 2007
AJAXWorld