Chloe and the RTW

Trotter Cashion
Strange Loop 2011

For those looking only at slides!

MASHION

Co-Founder

@cashion

github.com/
trotter

Go Get It!

http://github.com/mashion/chloe

Why?

Knockout!

simulchart.com

Knockout!

Made me cry :-(

We can Haz?

http://utterinsanity.files.wordpress.com/2008/09/funny-pictures-cat-drives-an-invisible-racecar.jpg%3Fw%3D497%26h%3D318

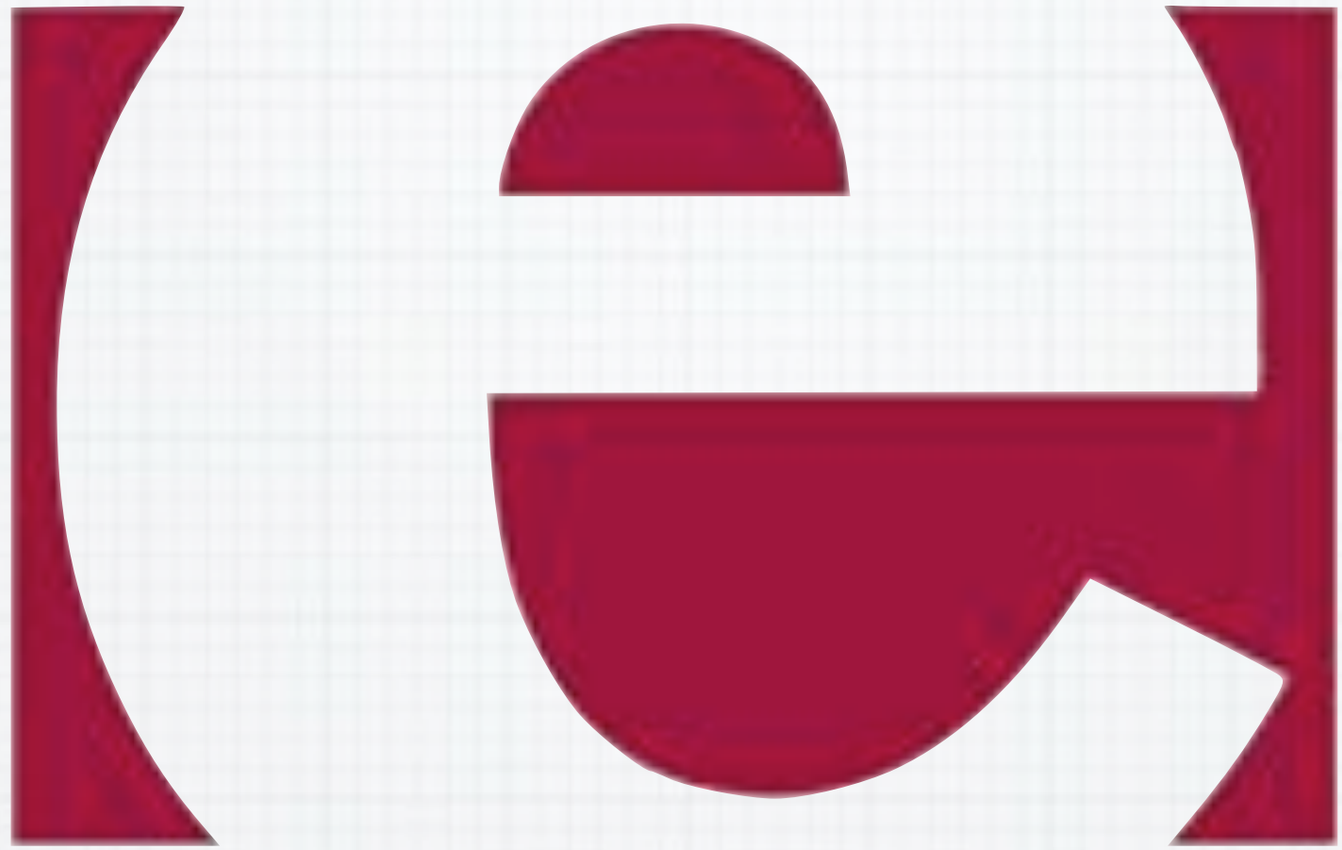
We can Haz?

http://utterinsanity.files.wordpress.com/2008/09/funny-pictures-cat-drives-an-invisible-racecar.jpg%3Fw%3D497%26h%3D318

We can Haz?

http://utterinsanity.files.wordpress.com/2008/09/funny-pictures-cat-drives-an-invisible-racecar.jpg%3Fw%3D497%26h%3D318

Got me thinking...



ERLANG

Chloe

A realtime web-server... that doesn't suck.

Cool History, Right?

http://github.com/mashion/chloe

Usage...

A Simple Chat Service

```
<html>
  <head>
    <script type="text/javascript" src="/jquery-1.6.4.min.js"></script>
    <script type="text/javascript" src="http://localhost:8901/chloe.js">
      </script>
    <script type="text/javascript" src="/chat.js"></script>
    <title>Ajax Chat</title>
  </head>
  <body>
    <form>
      <input name="message" id="message" type="text"/>
      <input name="submit" type="submit" value="Send"/>
    </form>
    <ul id="messages"></ul>
  </body>
</html>
```

```
<html>
  <head>
    <script type="text/javascript" src="/jquery-1.6.4.min.js"></script>
    <script type="text/javascript" src="http://localhost:8901/chloe.js">
      </script>
    <script type="text/javascript" src="/chat.js"></script>
    <title>Ajax Chat</title>
  </head>
  <body>
    <form>
      <input name="message" id="message" type="text"/>
      <input name="submit" type="submit" value="Send"/>
    </form>
    <ul id="messages"></ul>
  </body>
</html>
```

Include Chloe JS

```
<html>
  <head>
    <script type="text/javascript" src="/jquery-1.6.4.min.js"></script>
    <script type="text/javascript" src="http://localhost:8901/chloe.js">
      </script>
    <script type="text/javascript" src="/chat.js"></script>
    <title>Ajax Chat</title>
  </head>
  <body>
    <form>
      <input name="[illegible]" type="text"/>
      <input name="submit" type="submit" value="Send"/>
    </form>
    <ul id="messages"></ul>
  </body>
</html>
```

Include Your JS

```
$(function () {
  var form = $('form'),
      container = $('#messages');
      chloe = new Chloe({host: 'localhost', port: 8901});

  chloe.connect(function () {
    form.submit(function () {
      chloe.send(form.serialize());
      $('#message').val('');
      return false;
    });

    chloe.onmessage(function (message) {
      container.prepend("<li>" + message + "</li>");
    });
  });

});
```

```
$(function () {
  var form = $('form'),
      container = $('#messages');
      chloe = new Chloe({host: 'localhost', port: 8901});

  chloe.connect(function () {
    form.submit(function
      chloe.send(form.serialize());
      $('#message').val('');
      return false;
    });

    chloe.onmessage(function (message) {
      container.prepend("<li>" + message + "</li>");
    });
  });

});
```

Instantiate Chloe

```
$(function () {
  var form = $('form'),
      container = $('#messages');
      chloe = new Chloe({host: 'localhost', port: 8901});

  chloe.connect(function () {
    form.submit(function () {
      chloe.se
      $('#mess
      return false;
    });

    chloe.onmessage(function (message) {
      container.prepend("<li>" + message + "</li>");
    });
  });

});
```

Connect to the Server

```
$(function () {
  var form = $('form'),
      container = $('#messages');
      chloe = new Chloe({host: 'localhost', port: 8901});

  chloe.connect(function () {
    form.submit(function () {
      chloe.send(form.serialize());
      $('#message').val('');
      return false
    });

    chloe.onmessage(function (message) {
      container.prepend("<li>" + message + "</li>");
    });
  });

});
```

Send data to the server

```
$(function () {
  var form = $('form'),
      container = $('#messages');
      chloe = new Chloe({host: 'localhost', port: 8901});

  chloe.connect(function () {
    form.submit(function () {
      chloe.send(form.serialize());
      $('#message').val('');
      return f
    });

    chloe.onmessage(function (message) {
      container.prepend("<li>" + message + "</li>");
    });
  });

});
```

Run this function when a message arrives

```ruby
get '/' do
  erubis :index
end

post '/updates' do
  data = URI.decode_www_form(request.body.read)
  params = Hash[data]
  uri = URI.parse("http://localhost:8901/send")
  Net::HTTP.post_form(uri,
                      {"data" => params["message"]})
  'ok'
end
```

```ruby
get '/' do
  erubis :index
end

post '/up
  data = URI.decode_www_form(request.body.read)
  params = Hash[data]
  uri = URI.parse("http://localhost:8901/send")
  Net::HTTP.post_form(uri,
                      {"data" => params["message"]})
  'ok'
end
```

Get the html from earlier

```ruby
get '/' do
  erubis :index
end

post '/updates' do
  data = URI.decode_www_form(request.body.read)
  params = Hash[data]
  uri = URI.parse("http://localhost:8901/send")
  Net::HTTP.post_form(uri,
                      {"data" => params["message"]})
  'ok'
end
```

```
chloe.send(form.serialize());
```

```ruby
get '/' do
  erubis :index
end

post '/updates' do
  data = URI.decode_www_form(request.body.read)
  params = Hash[data]
  uri = URI.parse("http://localhost:8901/send")
  Net::HTTP.post_form(uri,
                      {"data" => params["message"]})
  'ok'
end
```

Ruby trick!

```ruby
get '/' do
  erubis :index
end

post '/updates' do
  data = URI.decode_www_form(request.body.read)
  params = Hash[data]
  uri = URI.parse("http://localhost:8901/send")
  Net::HTTP.post_form(uri,
                      {"message" => params["message"]})
  'ok'
end
```

Chloe broadcast/multicast url

```
get '/' do
  erubis :index
end

post '/updates' do
  data = URI.                                    est.body.read)
  params = Ha
  uri = URI.parse("http://localhost:8901/send")
  Net::HTTP.post_form(uri,
                      {"data" => params["message"]})
  'ok'
end
```

Send the data to Chloe

How It Works

Chrome

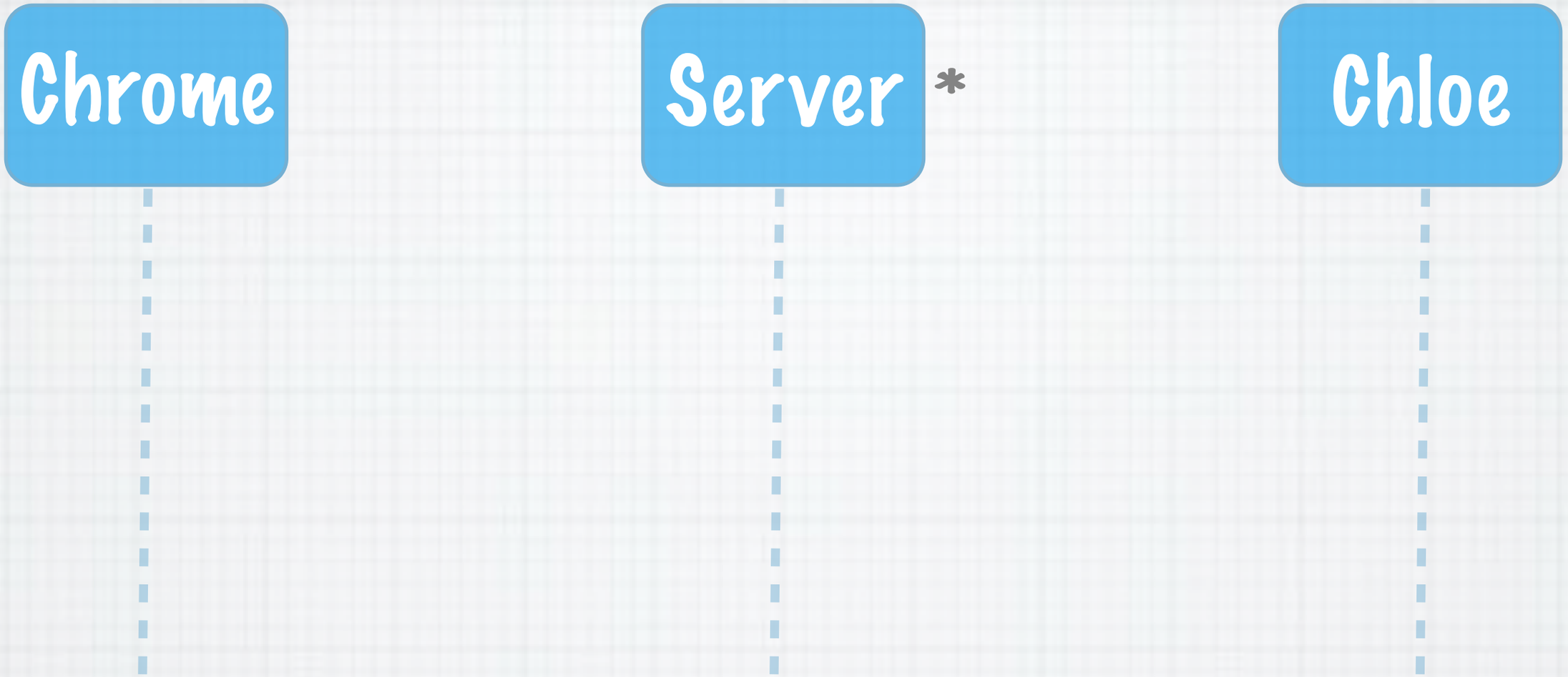
Chrome

Server

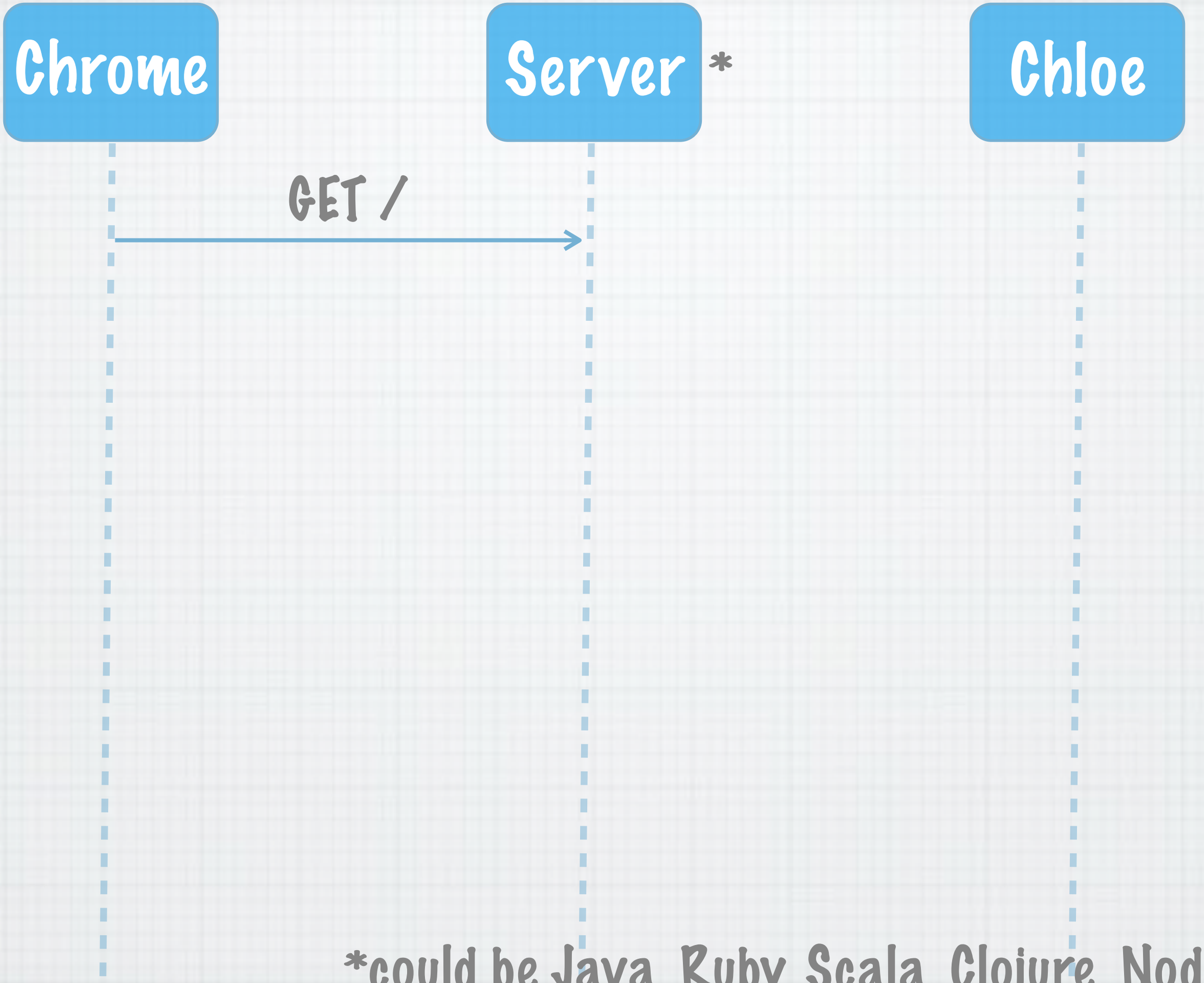
Chrome

Server *

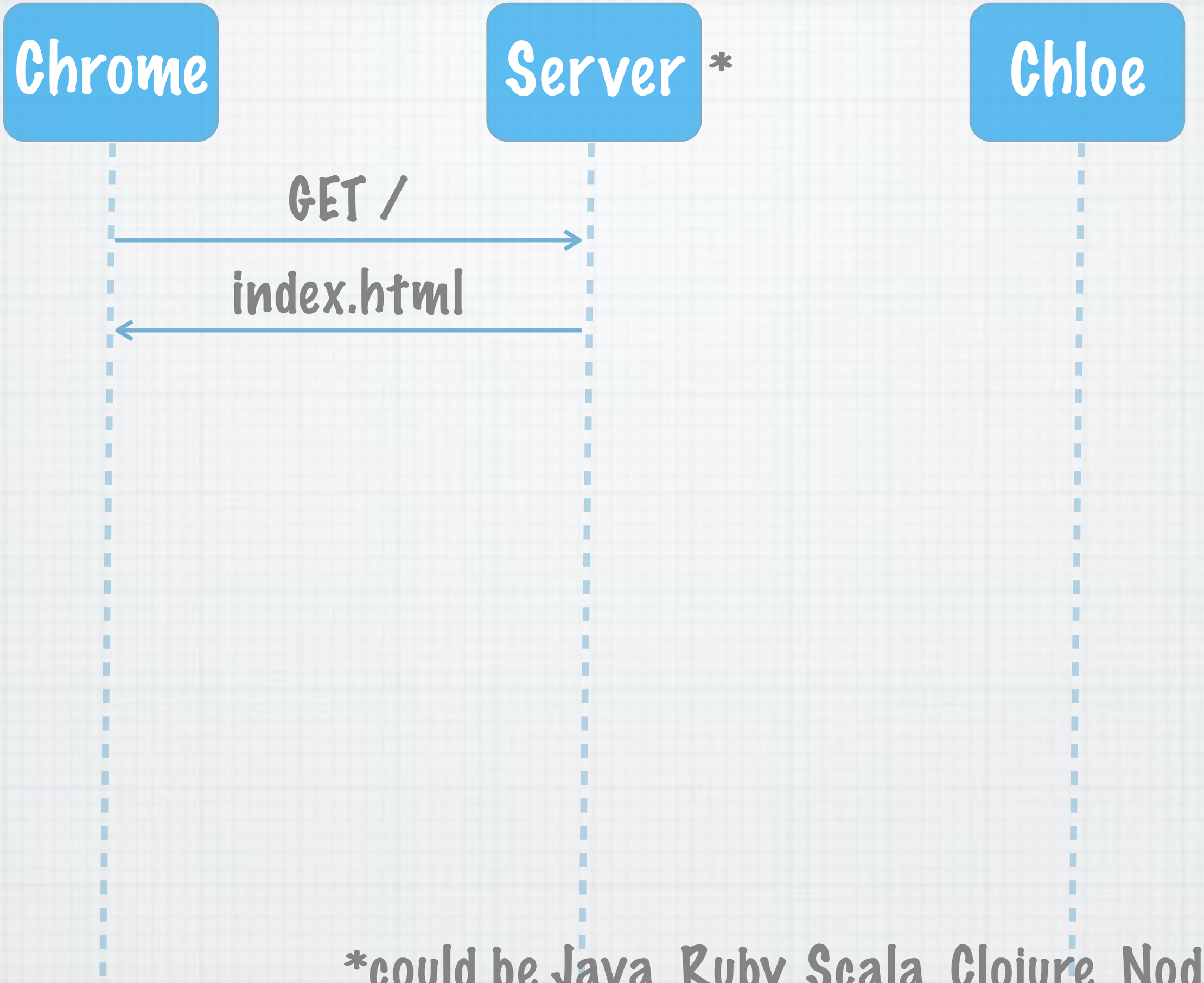
*could be Java, Ruby, Scala, Clojure, Node...



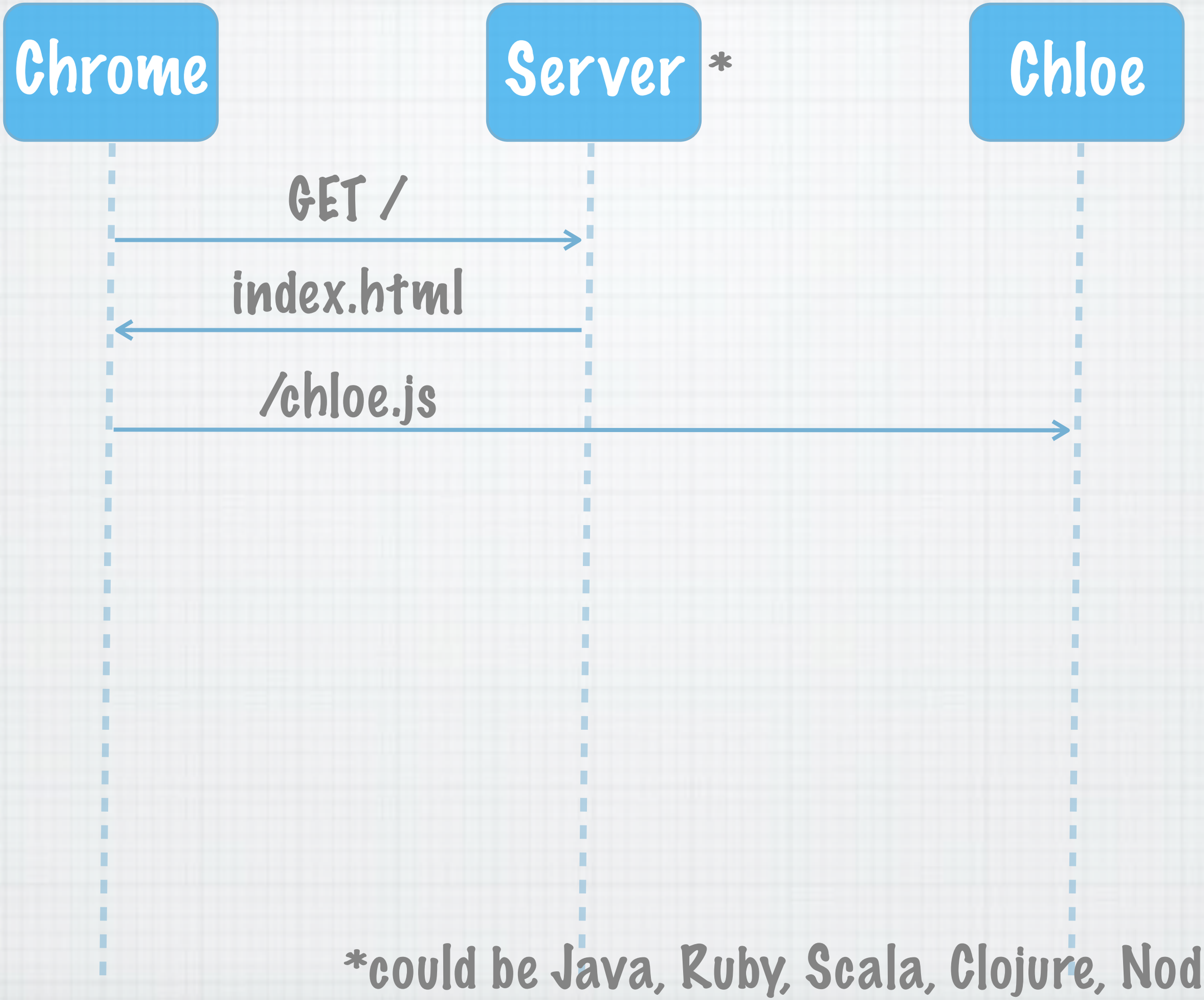
*could be Java, Ruby, Scala, Clojure, Node...



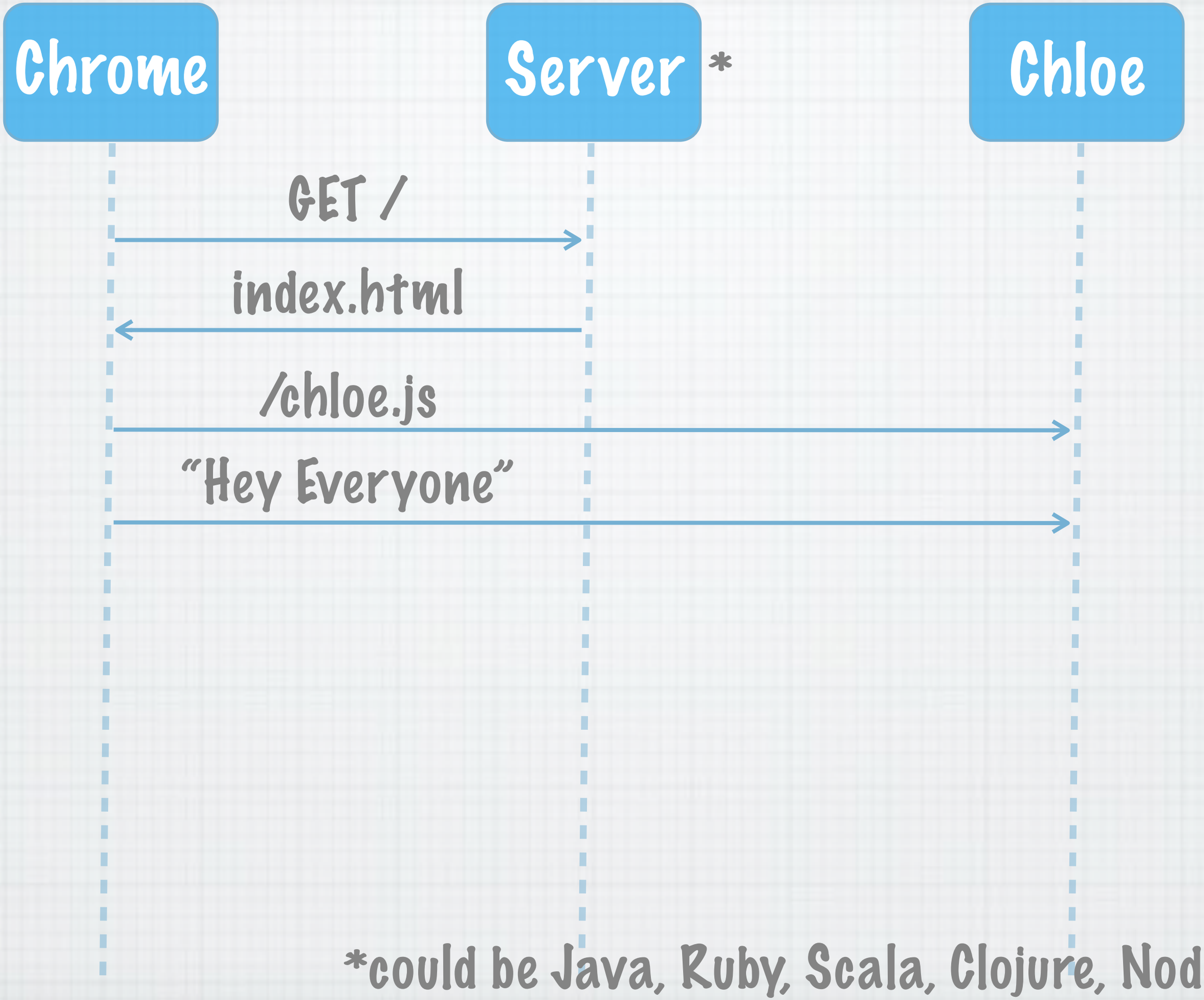
*could be Java, Ruby, Scala, Clojure, Node...

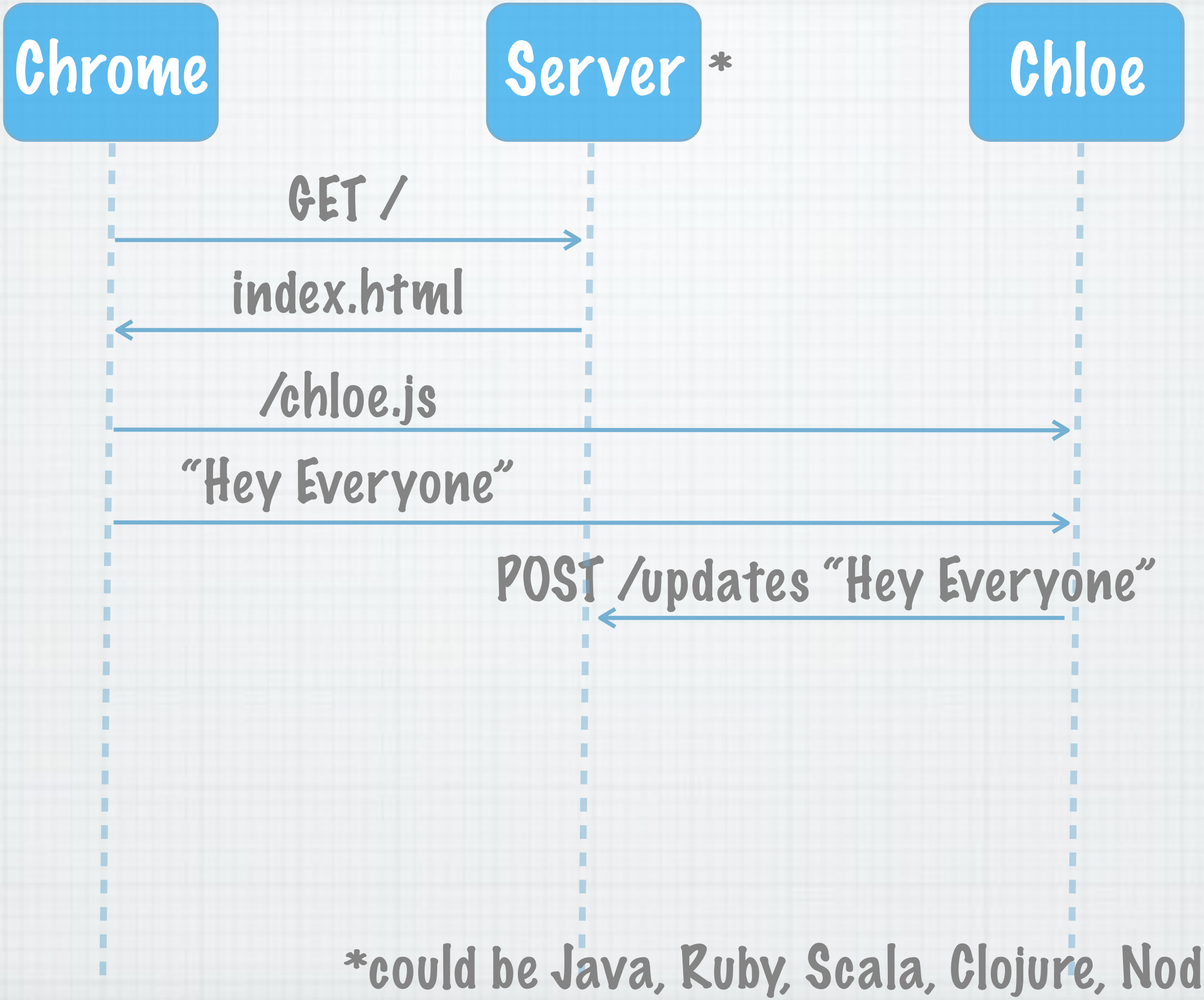


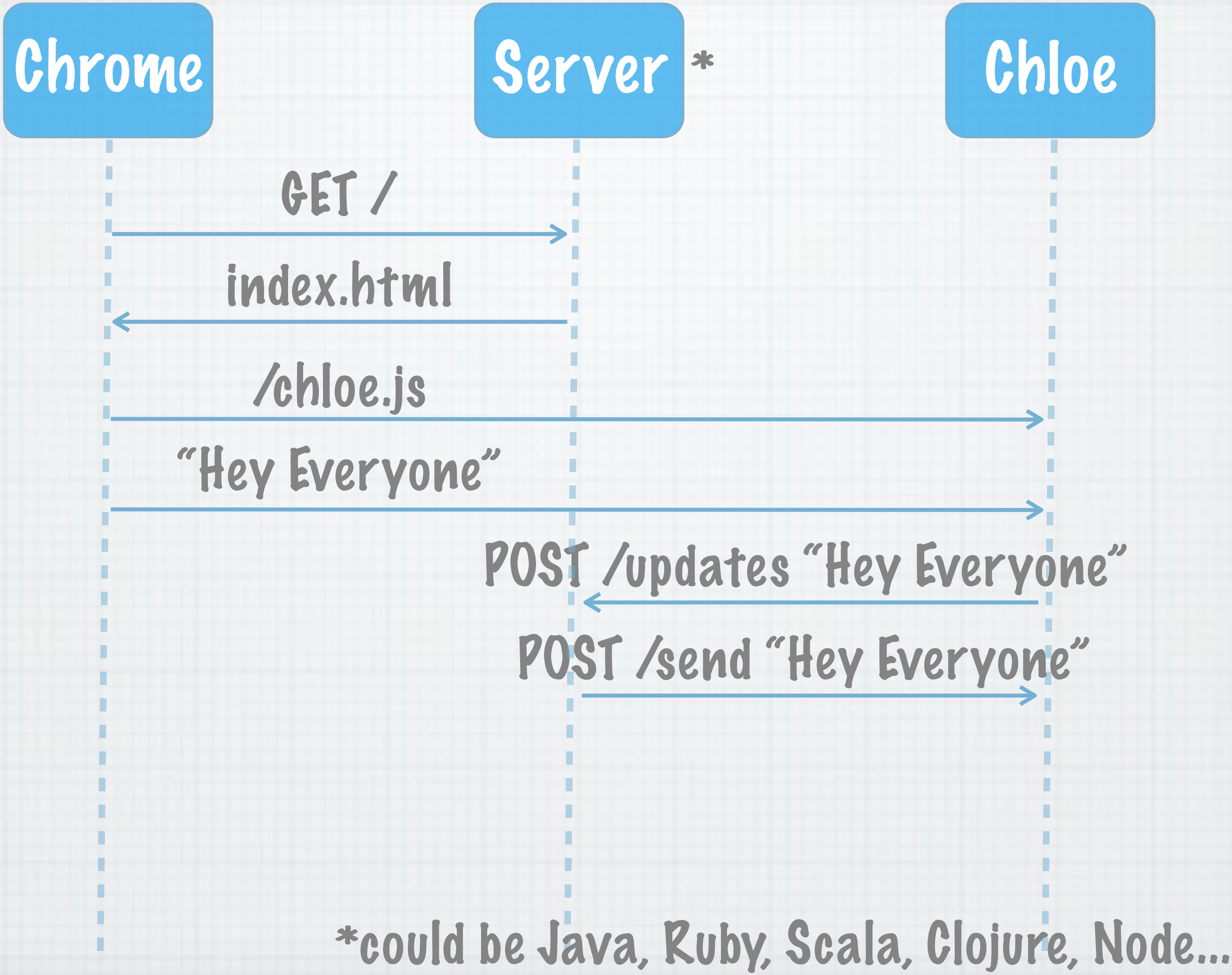
*could be Java, Ruby, Scala, Clojure, Node...

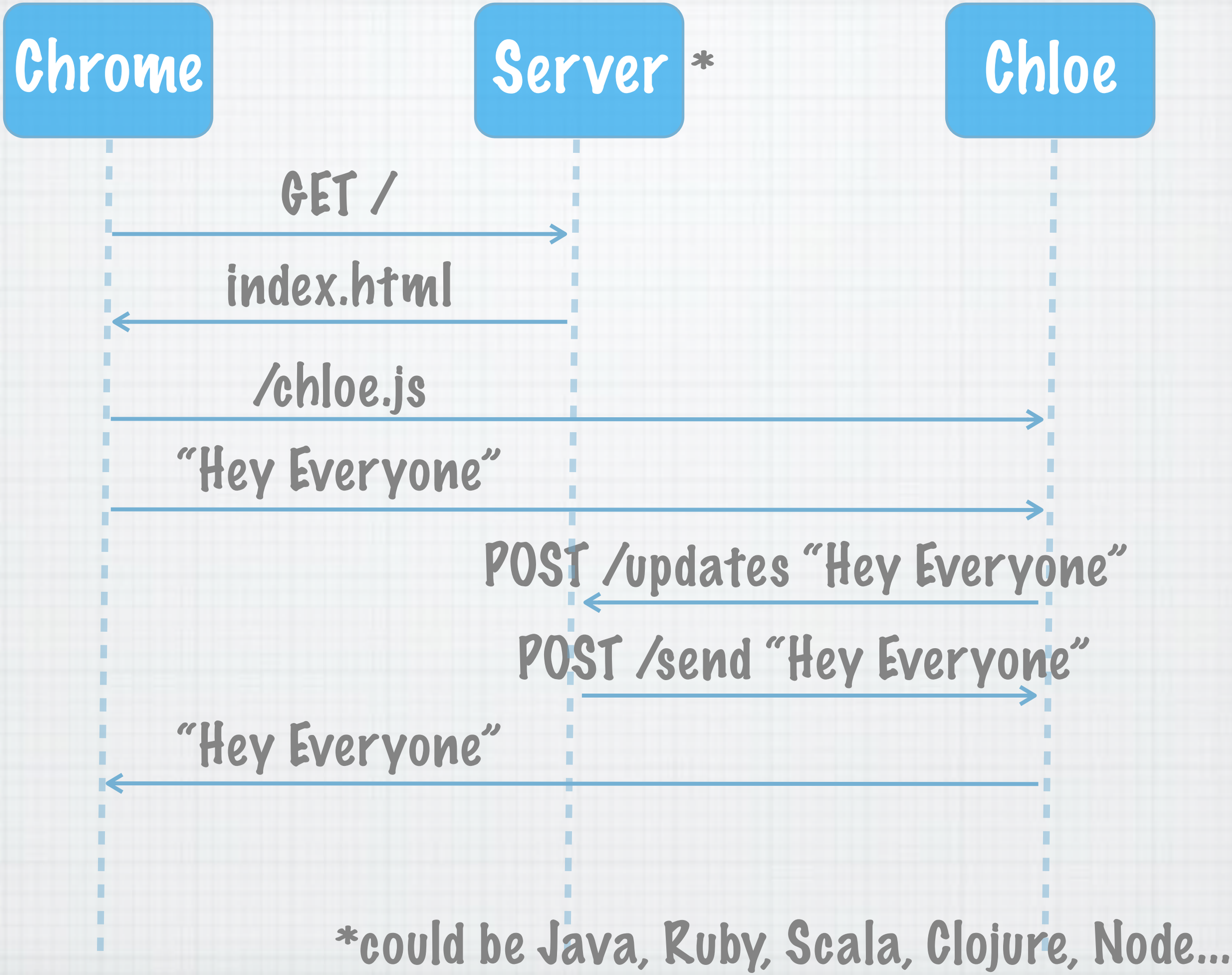


*could be Java, Ruby, Scala, Clojure, Node...


Chrome
Server *
Chloe
GET /
index.html
/chloe.js
"Hey Everyone"
*could be Java, Ruby, Scala, Clojure, Node...


Chrome
Server *
Chloe
GET /
index.html
/chloe.js
"Hey Everyone"
POST /updates "Hey Everyone"
*could be Java, Ruby, Scala, Clojure, Node...


Chrome
Server *
Chloe
GET /
index.html
/chloe.js
"Hey Everyone"
POST /updates "Hey Everyone"
POST /send "Hey Everyone"
*could be Java, Ruby, Scala, Clojure, Node...


Chrome
Server *
Chloe
GET /
index.html
/chloe.js
"Hey Everyone"
POST /updates "Hey Everyone"
POST /send "Hey Everyone"
"Hey Everyone"
*could be Java, Ruby, Scala, Clojure, Node...

Fallbacks

* Websockets
* Long polling (jsonp)
* Cross domain xhr - SOON!
* This seems to be enough for now...

Channels

```
chloe.connect(function () {
  chloe.subscribe('chat-room-5', function (message) {
    $("#messages").prepend("<li>" + message + "</li>")
  });
});
```

```
post '/updates' do
  uri = URI.parse("http://localhost:8901/send")
  Net::HTTP.post_form(uri,
        { "data" => "Hello room!",
          "channel" => "chat-room-5"})
  "ok"
end
```

Channel Name

```
chloe.connect(function () {
  chloe.subscribe('chat-room-5', function (message) {
    $("#messages").prepend("<li>" + message + "</li>")
  });
});
```

```
post '/updates' do
  uri = URI.parse("http://localhost:8901/send")
  Net::HTTP.post_form(uri,
        { "data" => "Hello room!",
          "channel" => "chat-room-5"})
  "ok"
end
```

```
chloe.connect(function () {
  chloe.subscribe('chat-room-5', function (message) {
    $("#messages").prepend("<li>" + message + "</li>")
  });
});
```

Called when message arrives

```
post '/updates' do
  uri = URI.parse("http://localhost:8901/send")
  Net::HTTP.post_form(uri,
        { "data" => "Hello room!",
          "channel" => "chat-room-5"})
  "ok"
end
```

```
chloe.connect(function () {
  chloe.subscribe('chat-room-5', function (message) {
    $("#messages").prepend("<li>" + message + "</li>")
  });
});
```

```
post '/updates' do
  uri = URI.parse("http://localhost:8901/send")
  Net::HTTP.post_form(uri,
        { "data" => "Hello room!",
          "channel" => "chat-room-5"})
  "ok"
end
```

```
chloe.connect(function () {
  chloe.subscribe('chat-room-5', function (message) {
    $("#messages").prepend("<li>" + message + "</li>")
  });
});
```

```
post '/updates' do
  uri = URI.parse("http://localhost:8901/send")
  Net::HTTP.post_form(uri,
        { "data" => "Hello room!",
          "channel" => "chat-room-5"})
  "ok"
end
```

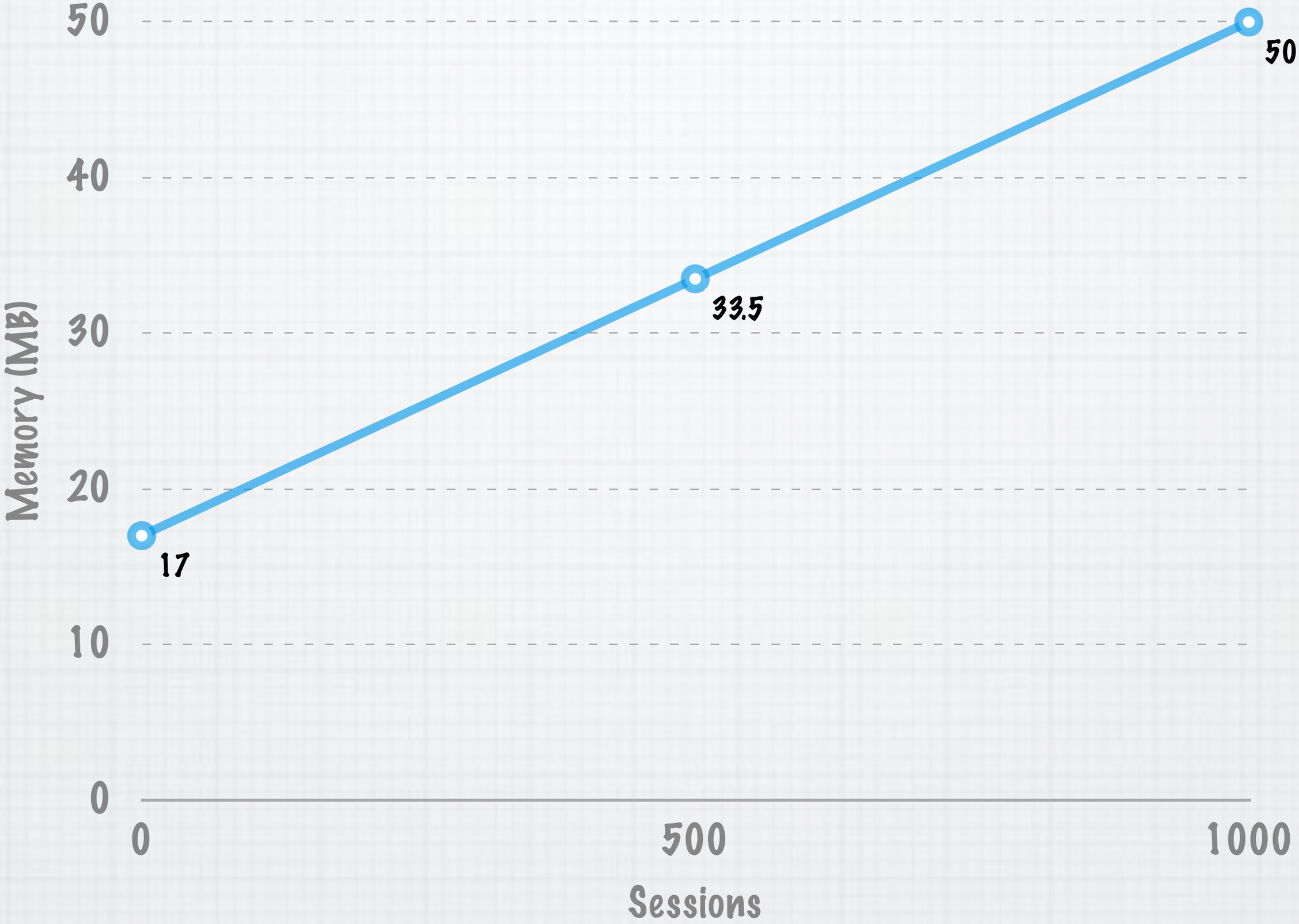
Channel for broadcast

What's left for Channels

* Bi-directionality
* Per Channel callbacks

Performance

Difficult to determine...


50
40
30
20
10
0
Memory (MB)
50
33.5
17
0
500
1000
Sessions

Tune Your TCP Stack

```
# ulimit -n 65536
# ifconfig eth0 txqueuelen 8192
# /sbin/sysctl -w net.core.somaxconn=4096
# /sbin/sysctl -w net.core.netdev_max_backlog=16384
# /sbin/sysctl -w net.core.rmem_max=16777216
# /sbin/sysctl -w net.core.wmem_max=16777216
# /sbin/sysctl -w net.ipv4.tcp_max_syn_backlog=8192
# /sbin/sysctl -w net.ipv4.tcp_syncookies=1
```

From cometd: http://cometd.org/documentation/2.x/howtos/loadtesting

Tune Your TCP Stack

```
# ulimit -n 65536
# ifconfig eth0 txqueuelen 8192
# /sbin/sysctl -w net.core.somaxconn=16384
# /sbin/sysctl -w net.core.rmem_max=16777216
# /sbin/sysctl -w net.core.wmem_max=16777216
# /sbin/sysctl -w net.ipv4.tcp_max_syn_backlog=8192
# /sbin/sysctl -w net.ipv4.tcp_syncookies=1
```

Increase number of file descriptors

From cometd: http://cometd.org/documentation/2.x/howtos/loadtesting

Tune Your TCP Stack

```
# ulimit -n 65536
# ifconfig eth0 txqueuelen 8192
# /sbin/sysctl -w net.core.somaxconn=4096
# /sbin/sysctl -w net.core.netdev_max_backlog=16384
# /sbin/sysctl -w net.core.optmem_max=16384
# /sbin/sysctl -w net.core.wmem_max=16777216
# /sbin/sysctl -w net.ipv4.tcp_max_syn_backlog=8192
# /sbin/sysctl -w net.ipv4.tcp_syncookies=1
```

Increase transaction queue length

From cometd: http://cometd.org/documentation/2.x/howtos/loadtesting

Tune Your TCP Stack

```
# ulimit -n 65536
# ifconfig eth0 txqueuelen 8192
# /sbin/sysctl -w net.core.somaxconn=4096
# /sbin/sysctl -w net.core.netdev_max_backlog=16384
# /sbin/sysctl                                    16
# /sbin/sysctl                                    16
# /sbin/sysctl -w net.ipv4.tcp_max_syn_backlog=8192
# /sbin/sysctl -w net.ipv4.tcp_syncookies=1
```

Increase size of listen queue

From cometd: http://cometd.org/documentation/2.x/howtos/loadtesting

Tune Your TCP Stack

```
# ulimit -n 65536
# ifconfig eth0 txqueuelen 8192
# /sbin/sysctl -w net.core.somaxconn=4096
# /sbin/sysctl -w net.core.netdev_max_backlog=16384
# /sbin/sysctl -w net.core.rmem_max=16777216
# /sbin/sysc                                      6
# /sbin/sysc                                      log=8192
# /sbin/sysctl -w net.ipv4.tcp_syncookies=1
```

Increase incoming packet queue

From cometd: http://cometd.org/documentation/2.x/howtos/loadtesting

Tune Your TCP Stack

```
# ulimit -n 65536
# ifconfig eth0 txqueuelen 8192
# /sbin/sysctl -w net.core.somaxconn=4096
# /sbin/sysctl -w net.core.netdev_max_backlog=16384
# /sbin/sysctl -w net.core.rmem_max=16777216
# /sbin/sysctl -w net.core.wmem_max=16777216
# /sbin/sysctl                                        192
# /sbin/sysctl
```

Increase maximum receive window

From cometd: http://cometd.org/documentation/2.x/howtos/loadtesting

Tune Your TCP Stack

```
# ulimit -n 65536
# ifconfig eth0 txqueuelen 8192
# /sbin/sysctl -w net.core.somaxconn=4096
# /sbin/sysctl -w net.core.netdev_max_backlog=16384
# /sbin/sysctl -w net.core.rmem_max=16777216
# /sbin/sysctl -w net.core.wmem_max=16777216
# /sbin/sysctl -w net.ipv4.tcp_max_syn_backlog=8192
# /sbin/sysctl                                    kies=1
```

Increase send window

From cometd: http://cometd.org/documentation/2.x/howtos/loadtesting

Tune Your TCP Stack

```
# ulimit -n 65536
# ifconfig eth0 txqueuelen 8192
# /sbin/sysctl -w net.core.somaxconn=4096
# /sbin/sysctl -w net.core.netdev_max_backlog=16384
# /sbin/sysctl -w net.core.rmem_max=16777216
# /sbin/sysctl -w net.core.wmem_max=16777216
# /sbin/sysctl -w net.ipv4.tcp_max_syn_backlog=8192
# /sbin/sysctl -w net.ipv4.tcp_syncookies=1
```

How many SYN requests to keep in memory

From cometd: http://cometd.org/documentation/2.x/howtos/loadtesting

Tune Your TCP Stack

```
# ulimit -n 65536
# ifconfig eth0 txqueuelen 8192
# /sbin/sysctl -w net.core.somaxconn=4096
# /sbin/sysctl -w net.core.netdev_max_backlog=16384
# /sbin/sysctl -w net.core.rmem_max=16777216
# /sbin/sysctl -w net.core.wmem_max=16777216
# /sbin/sysctl -w net.ipv4.tcp_max_syn_backlog=8192
# /sbin/sysctl -w net.ipv4.tcp_syncookies=1
```

Makes tcp_max_syn_backlog work

From cometd: http://cometd.org/documentation/2.x/howtos/loadtesting

How much Overhead?

How much Overhead?

?

Up and Running

Binaries Available

* Mac, Ubuntu 32bit, Ubuntu 64bit
* Fully self-contained (Erlang included)
* .deb and .rpm will be coming soon...

It's Easy!

```
# wget http://bit.ly/oqyQfL
# tar xzvf chloe-0.0.3-osx.tgz
# cd chloe-0.0.3
# ./bin/chloe start
# ./bin/chloe stop
```

It's Easy!

```
# wget http://bit.ly/oqyQfL
# tar xzvf chloe-0.0.3-osx.tgz
# cd chloe-0.0.3
# ./bin/chloe start
# ./bin/chloe stop
```

It's Easy!

```
# wget http://bit.ly/oqyOfL
# tar xzvf chloe-0.0.3-osx.tgz
# cd chloe-0.0.3
# ./bin/chloe start
# ./bin/chloe stop
```

It's Easy!

```
# wget http://bit.ly/oqyOfL
# tar xzvf chloe-0.0.3-osx.tgz
# cd chloe-0.0.3
# ./bin/chloe start
# ./bin/chloe stop
```

It's Easy!

```
# wget http://bit.ly/oqyOfL
# tar xzvf chloe-0.0.3-osx.tgz
# cd chloe-0.0.3
# ./bin/chloe start
# ./bin/chloe stop
```

./etc/app.config

```
{chloe,
  [
    {application_server, "http://localhost:4567"},
    {application_server_url, "http://localhost:4567/updates"},
    {port, 8901},
    {doc_root, "./public"},
    {log_dir, "/var/log/chloe"},
    {secret, “SEKRET PASSFRASE”}
  ]}
```

./etc/app.config

For XHR

```
{chloe,
  [
    {application_server, "http://localhost:4567"},
    {application_server_url, "http://localhost:4567/updates"},
    {port, 8901},
    {doc_root, "./public"},
    {log_dir, "/var/log/chloe"},
    {secret, “SEKRET PASSFRASE”}
  ]}
```

./etc/app.config

Callback url on your application

```
{chloe,
  [
    {application_server, "http://localhost:4567",
    {application_server_url, "http://localhost:4567/updates"},
    {port, 8901},
    {doc_root, "./public"},
    {log_dir, "/var/log/chloe"},
    {secret, “SEKRET PASSFRASE”}
  ]}
```

./etc/app.config

```
{chloe,
  [
    {application_s://localhost:4567"},
    {application_server_url, "http://localhost:4567/updates"},
    {port, 8901},
    {doc_root, "./public"},
    {log_dir, "/var/log/chloe"},
    {secret, “SEKRET PASSFRASE”}
  ]}
```

Chloe port

./etc/app.config

```
{chloe,
  [
    {application_server,
    {application_server_u
    {port, 8901},
    {doc_root, "./public"},
    {log_dir, "/var/log/chloe"},
    {secret, “SEKRET PASSFRASE”}
  ]}
```

Where Chloe's JavaScript Lives

./etc/app.config

```
{chloe,
  [
    {application_server, "http://localhost:4567"},
    {application_server_url, "http://localhost:4567/updates"},
    {port, 8901},
    {doc_root, "./public"},
    {log_dir, "/var/log/chloe"},
    {secret, “SEKRET PASSFRASE”}
  ]}
```

Directory for Logs

./etc/app.config

```
{chloe,
  [
    {application_server, "http://localhost:4567"},
    {application_server_url, "http://localhost:4567/updates"},
    {port, 8901},
    {doc_root, "./public"},
    {log_dir, "/var/log/chloe"},
    {secret, “SEKRET PASSFRASE”}
]}
```

For security (not required)...

Signing Requests

```
SECRET = "SEKRET PASSFRASE"

post '/updates' do
  data = URI.decode_www_form(request.body.read)
  message = Hash[data]["message"]
  sig = Digest::MD5.hexdigest(message + SECRET)
  uri = URI.parse("http://localhost:8901/send")
  Net::HTTP.post_form(uri,
                {"data" => message, "sig" => sig})
  'ok'
end
```

Signing Requests

```
SECRET = "SEKRET PASSFRASE"

post '/updates' do
  data = URI.decode_www_form(request.body.read)
  message = Hash[data]["message"]
  sig = Digest::MD5.hexdigest(message + SECRET)
  uri = URI.parse("http://localhost:8901/send")
  Net::HTTP.post_form(uri,
                {"data" => message, "sig" => sig})
  'ok'
end
```

Signing Requests

```
SECRET = "SEKRET PASSFRASE"

post '/update' do
  data = URI.decode_www_form(request.body.read)
  message = Hash[data]["message"]
  sig = Digest::MD5.hexdigest(message + SECRET)
  uri = URI.parse("http://localhost:8901/send")
  Net::HTTP.post_form(uri,
              {"data" => message, "sig" => sig})
  'ok'
end
```

From ./etc/app.config

Signing Requests

```
SECRET = "SEKRET PASSFRASE"

post '/updates' do
  data = URI.decode_www_form(request.body.read)
  message = Hash[data]["message"]
  sig = Digest::MD5.hexdigest(message + SECRET)
  uri = URI.parse("http://localhost:8901/send")
  Net::HTTP.post_form(uri,
                      {"data" => message, "sig" => sig})
  'ok'
end
```

Concat two strings

Signing Requests

```
SECRET = "SEKRET PASSFRASE"

post '/updates' do
  data = URI.decode_www_form(request.body.read)
  message = Hash[data]["message"]
  sig = Digest::MD5.hexdigest(message + SECRET)
  uri = URI.parse("http://localhost:8901/send")
  Net::HTTP.post_form(uri,
              {"data" => message, "sig" => sig})
  'ok'
end
```

Hash them

Signing Requests

```
SECRET = "SEKRET PASSFRASE"

post '/updates' do
  data = URI.decode_www_form(request.body.read)
  message = Hash[data]["message"]
  sig = Digest::MD5.hexdigest(message + SECRET)
  uri = URI.parse("http://localhost:8901/send")
  Net::HTTP.post_form(uri,
                {"data" => message, "sig" => sig})
  'ok'
end
```

Add signature to request

If `secret` is in your config, all requests to `/send` must be signed

Coming Soon

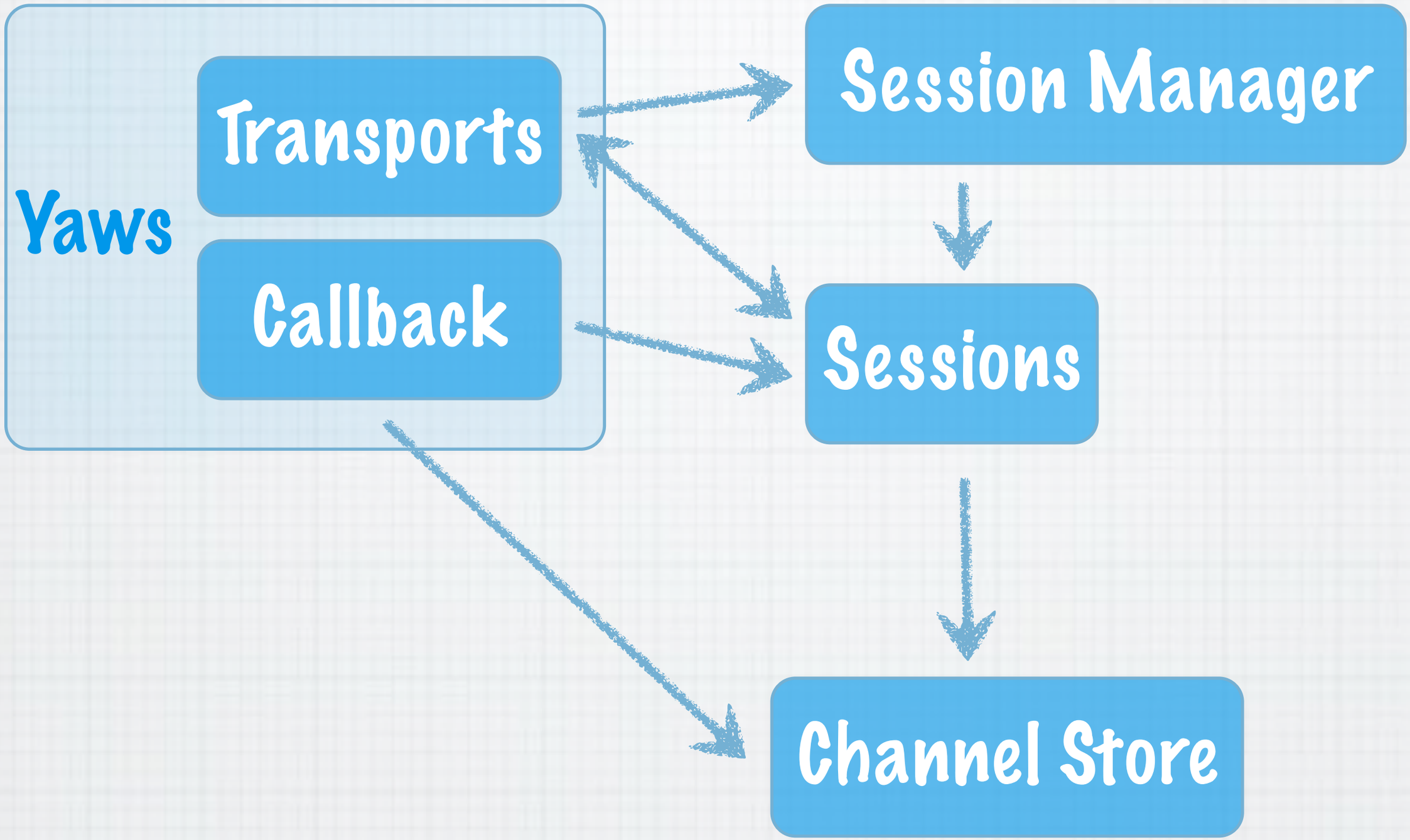
* Improved Performance
* Bi-directional channels
* Per-channel callback urls
* Runtime configuration
* SSL Support

Coming Later

* Improved monitoring
* Explore switch to mochiweb internally
* Client side authentication

Internals

Erlang Nerdery Time!


Yaws
Transports
Callback
Session Manager
Sessions
Channel Store

Yaws

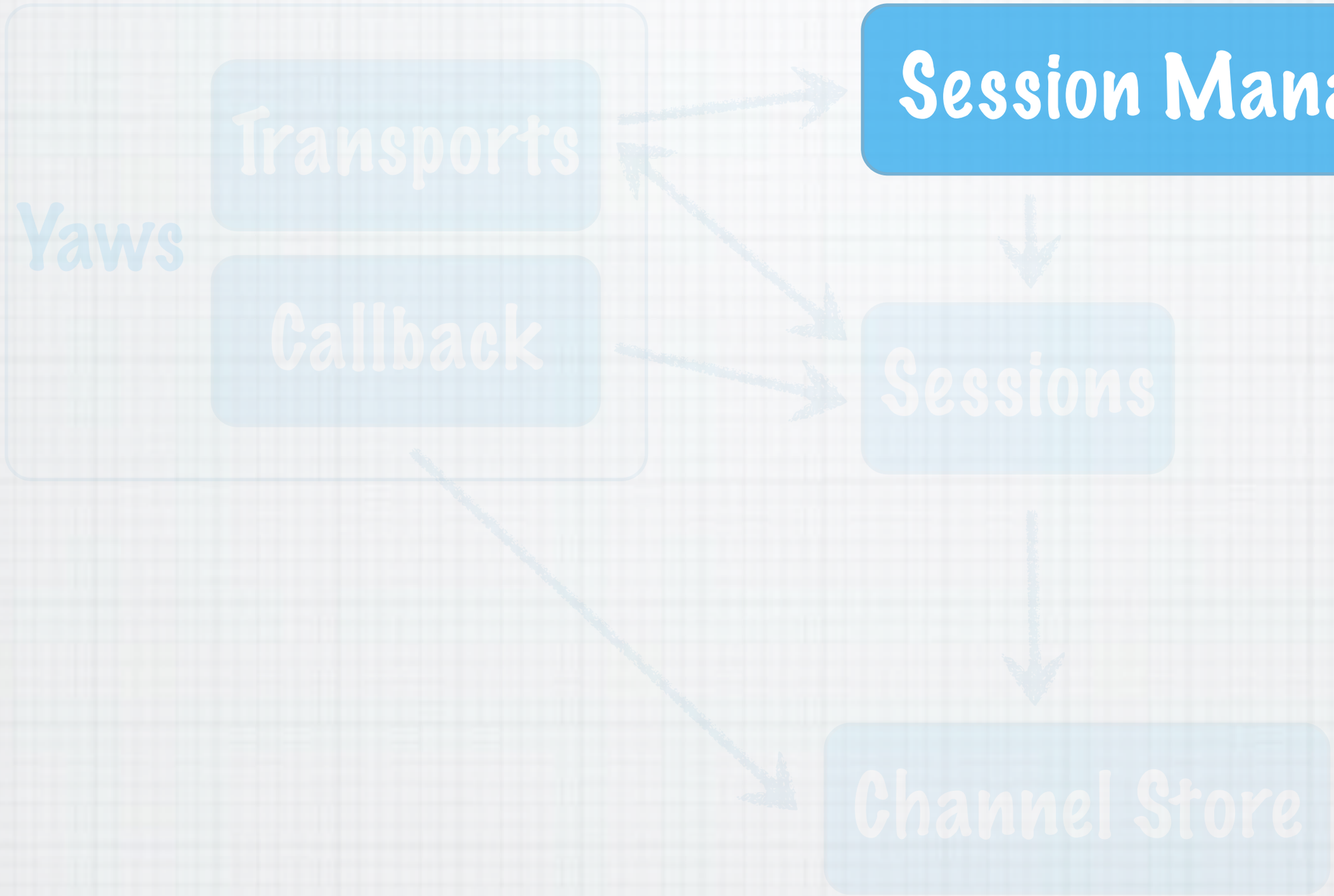
Transports

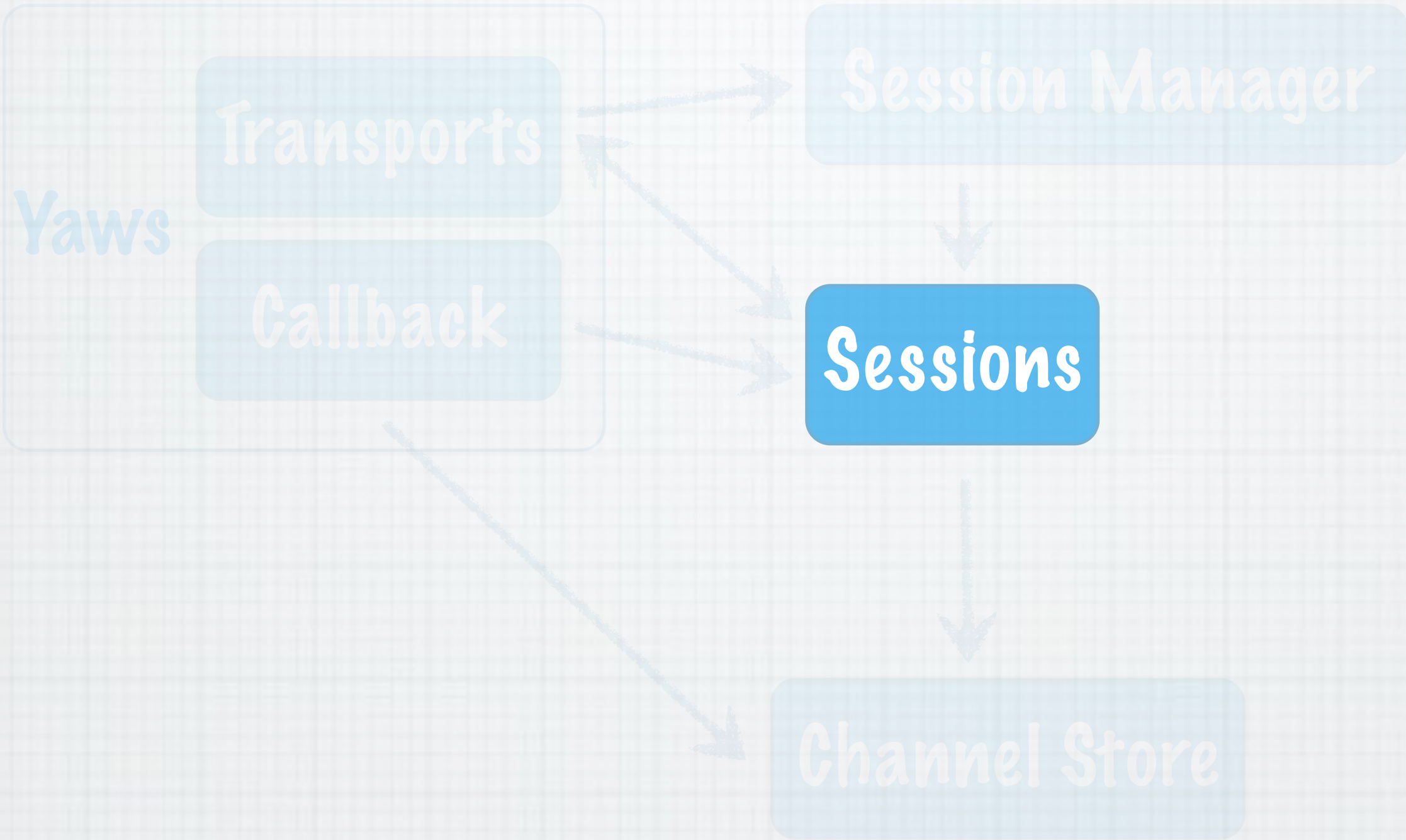
Callback

Session Manager

Sessions

Channel Store


Yaws
Transports
Callback
Session Manager
Sessions
Channel Store


Yaws
Transports
Callback
Session Manager
Sessions
Channel Store

Yaws

Transports

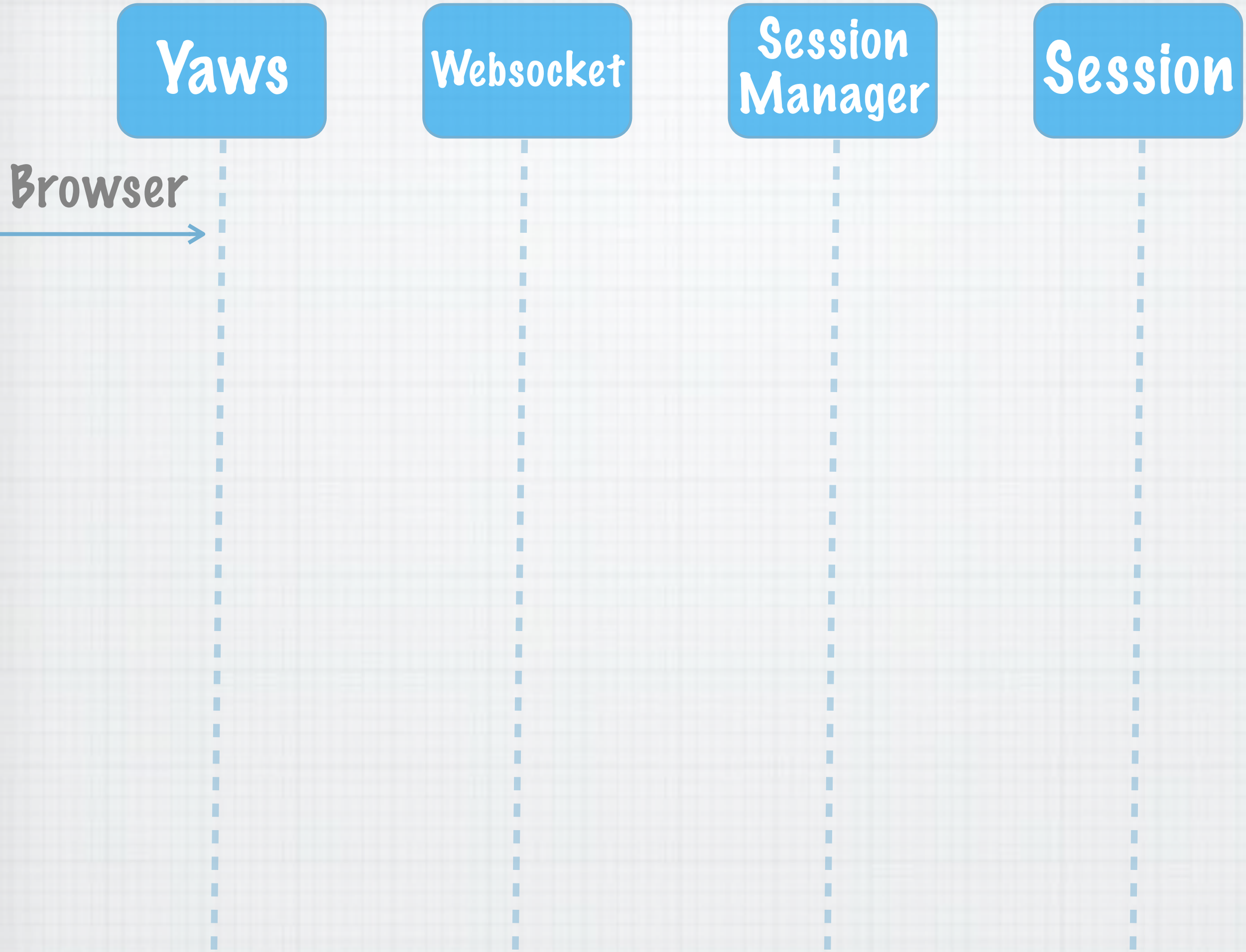
Callback

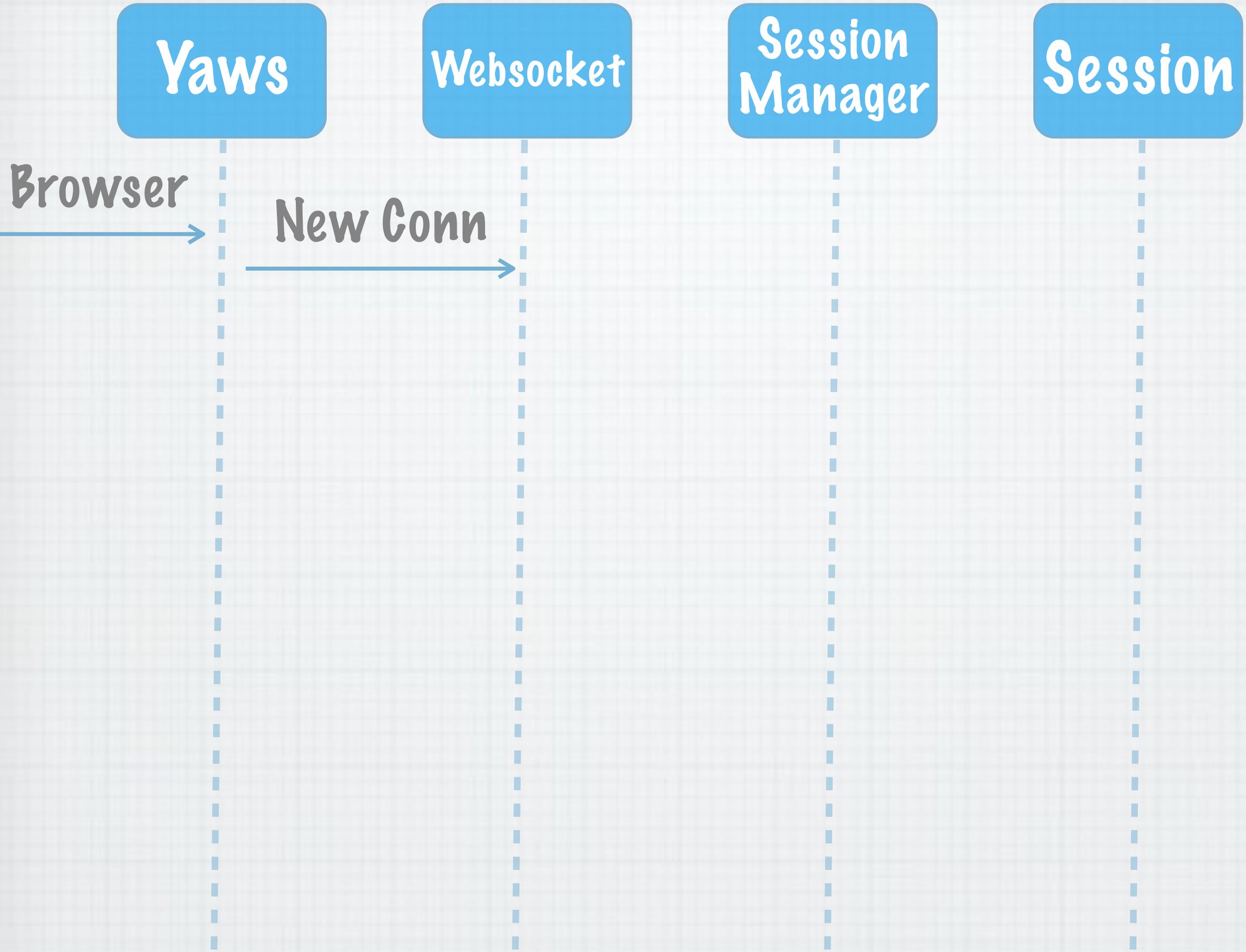
Session Manager

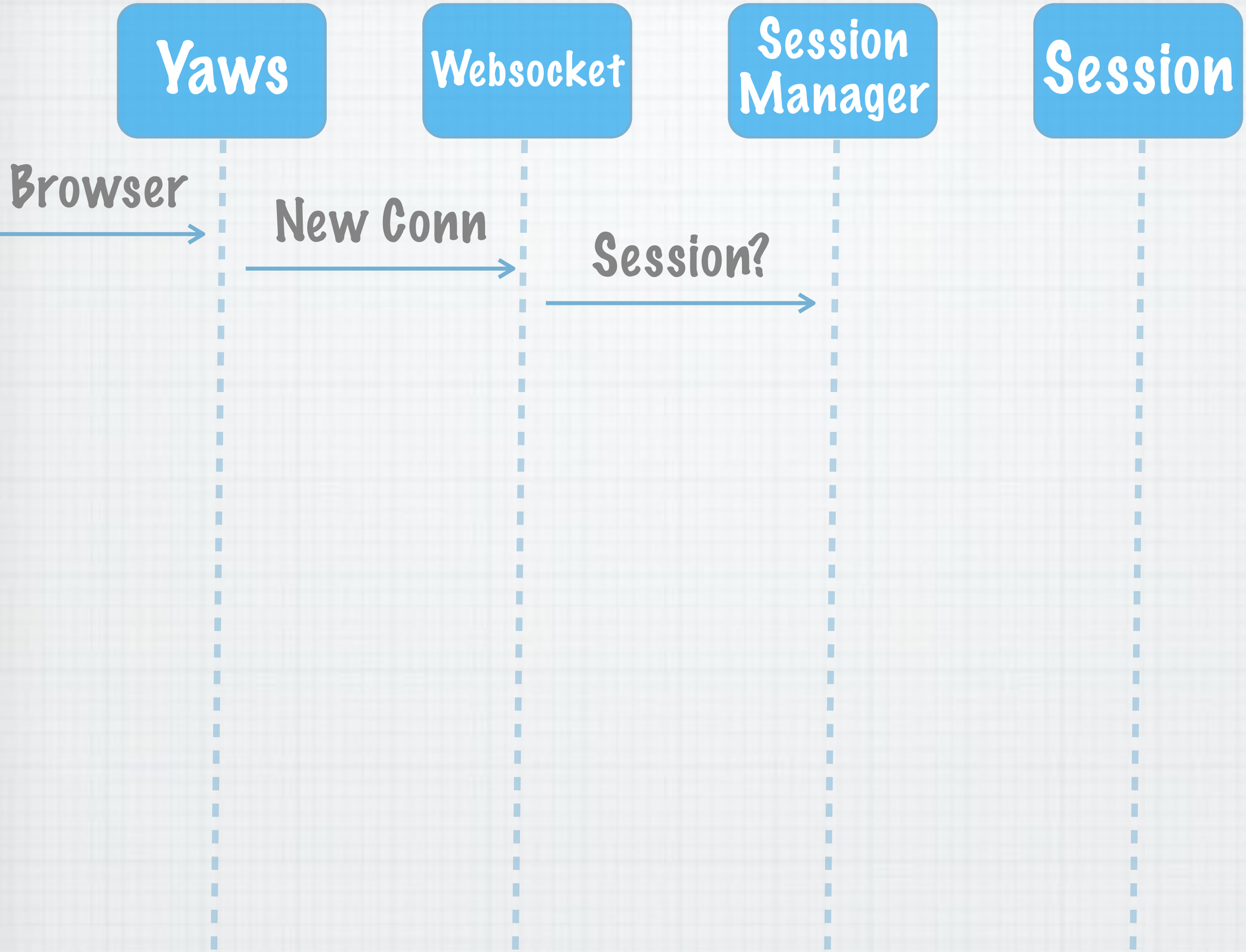
Sessions

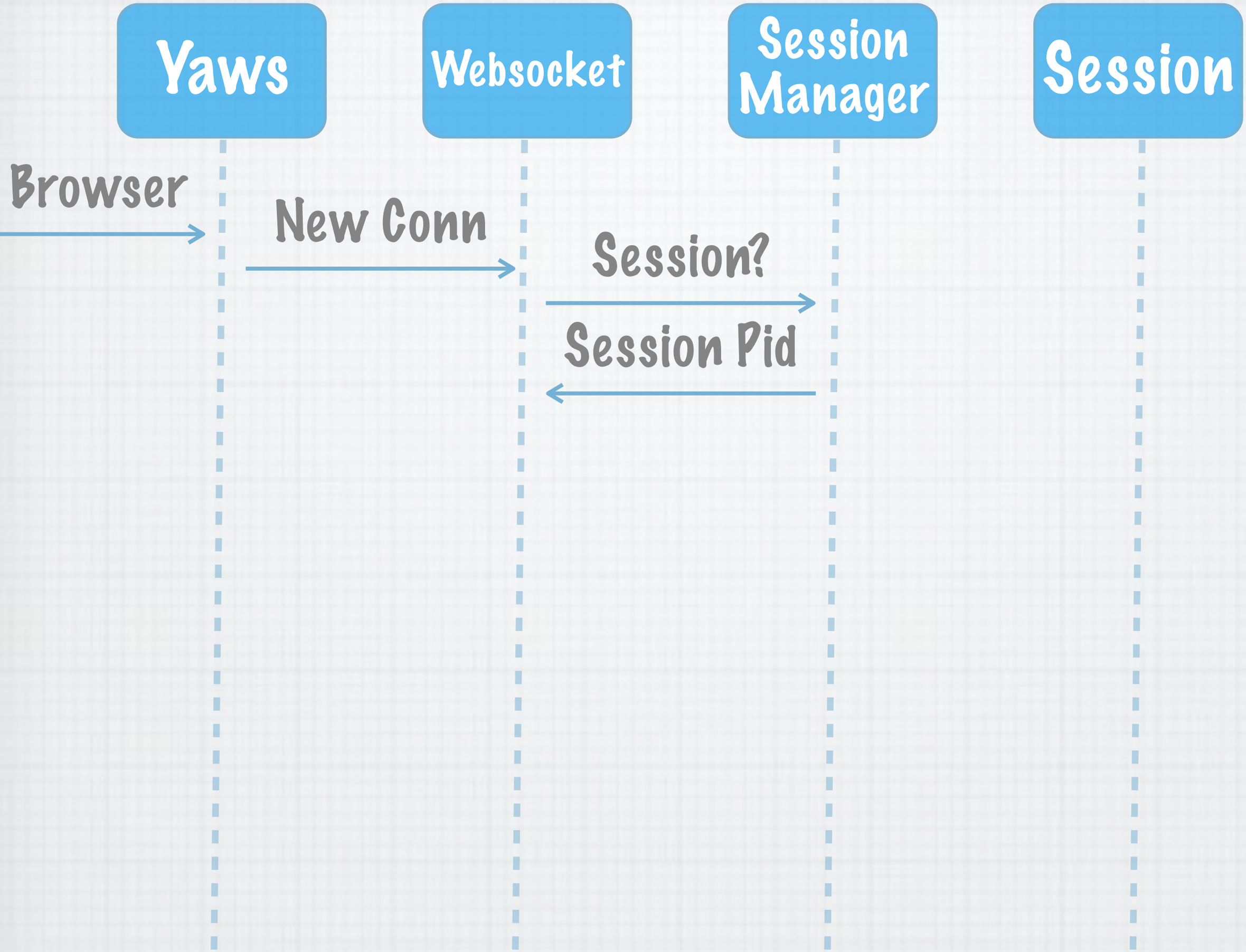
Channel Store

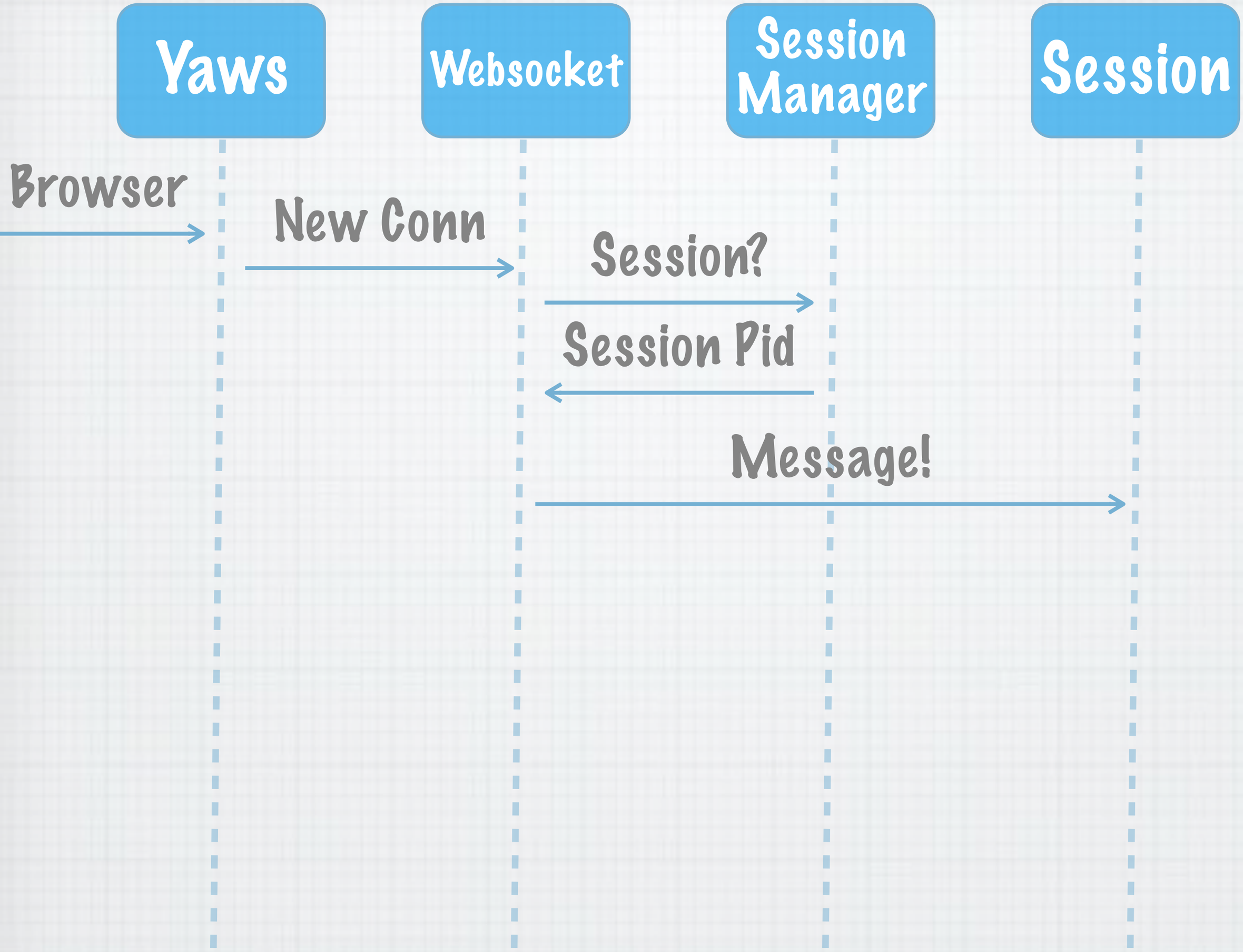

Yaws
Websocket
Session Manager
Session

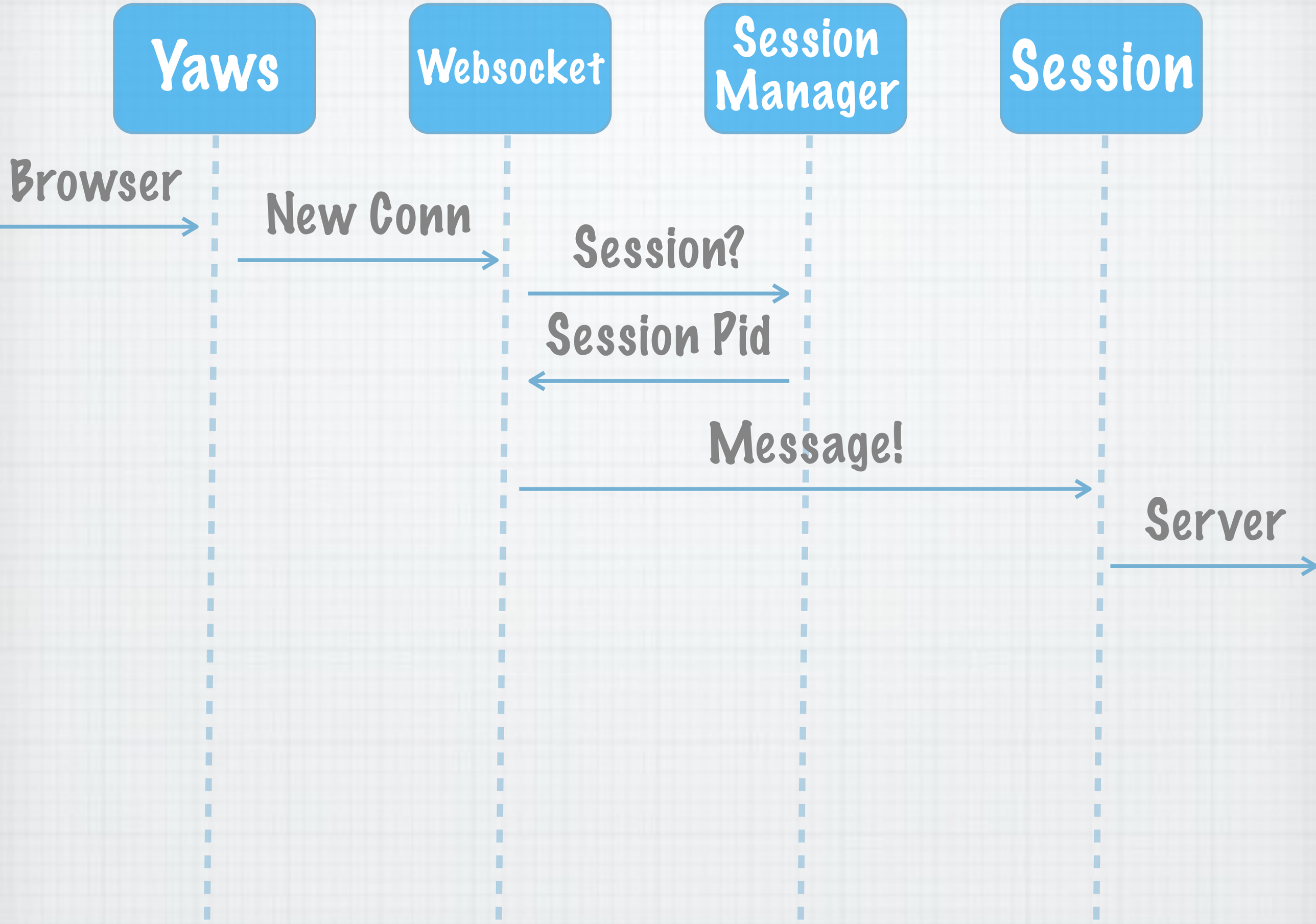

Yaws
Websocket
Session Manager
Session
Browser


Yaws
Websocket
Session Manager
Session
Browser
New Conn


Yaws
Websocket
Session Manager
Session
Browser
New Conn
Session?


Yaws
Websocket
Session Manager
Session
Browser
New Conn
Session?
Session Pid


Yaws
Websocket
Session Manager
Session
Browser
New Conn
Session?
Session Pid
Message!

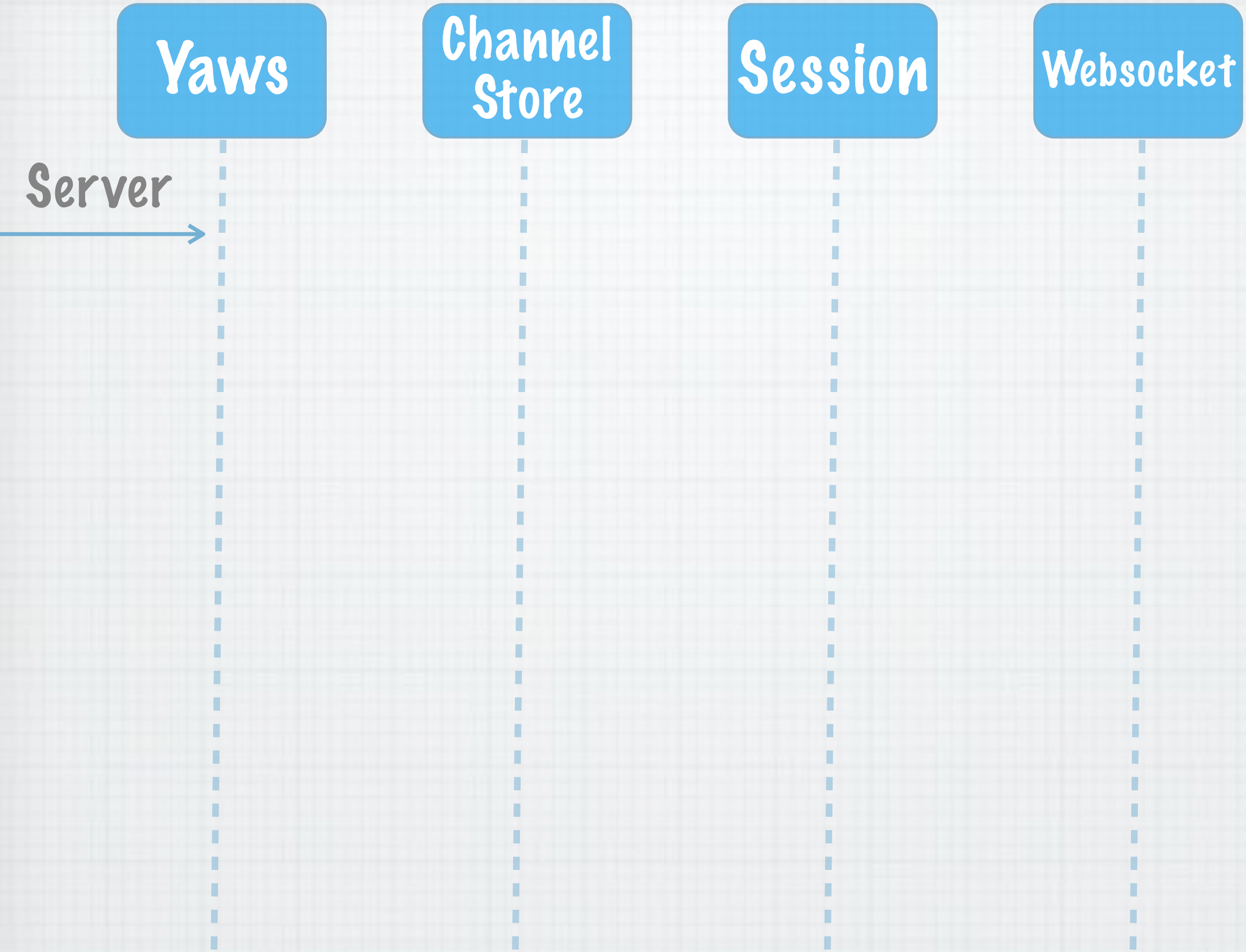

Yaws
Websocket
Session Manager
Session
Browser
New Conn
Session?
Session Pid
Message!
Server

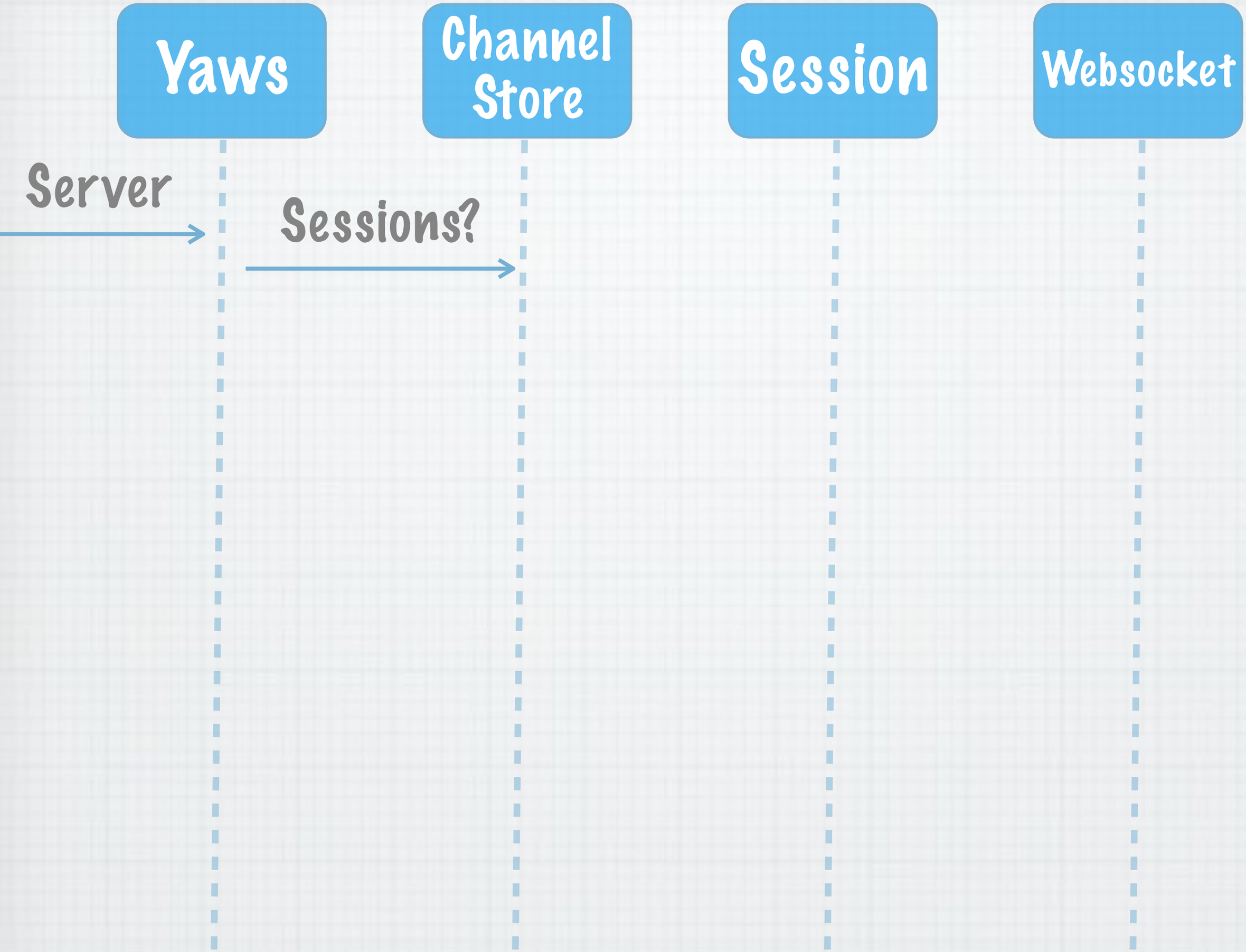
Yaws

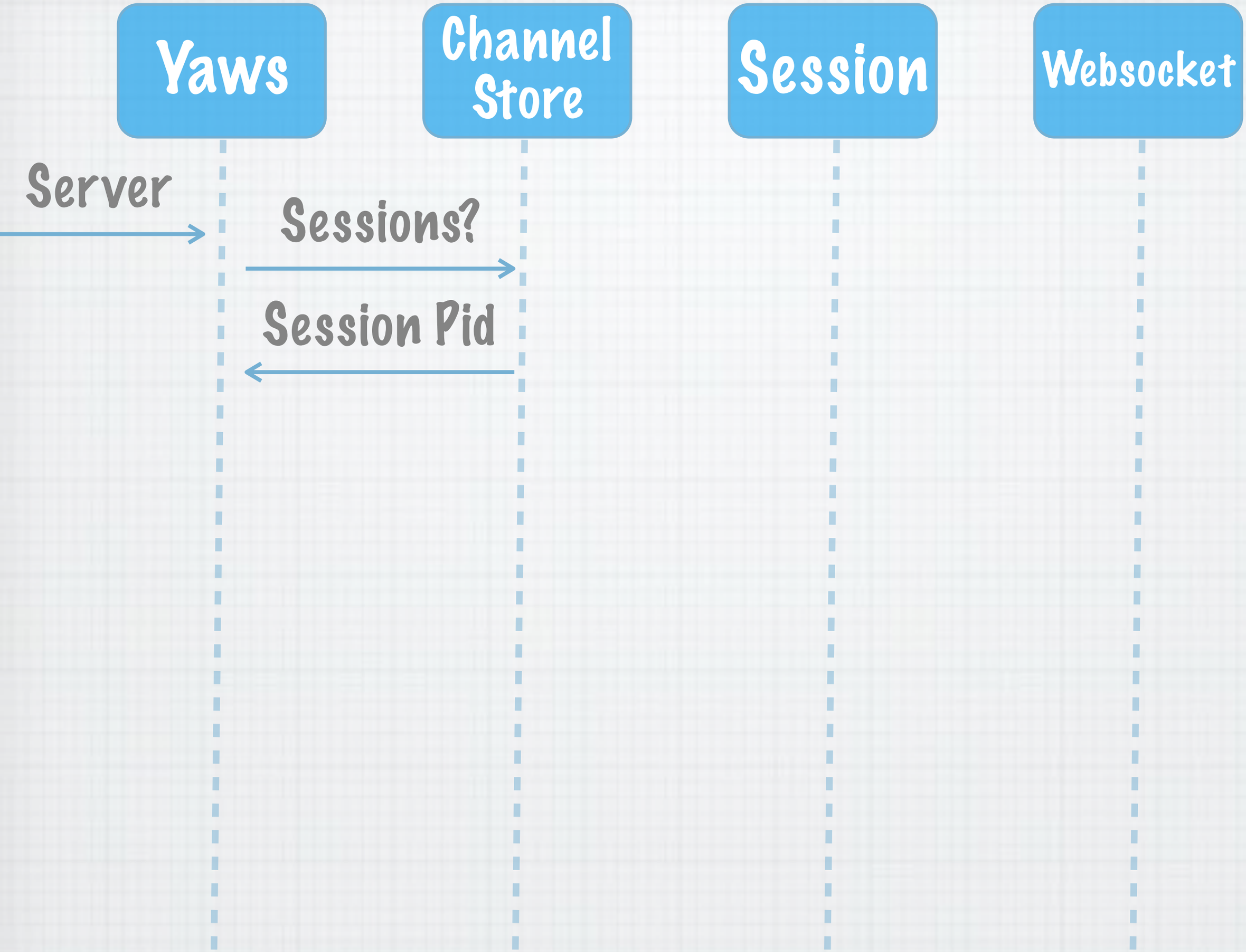
Channel Store

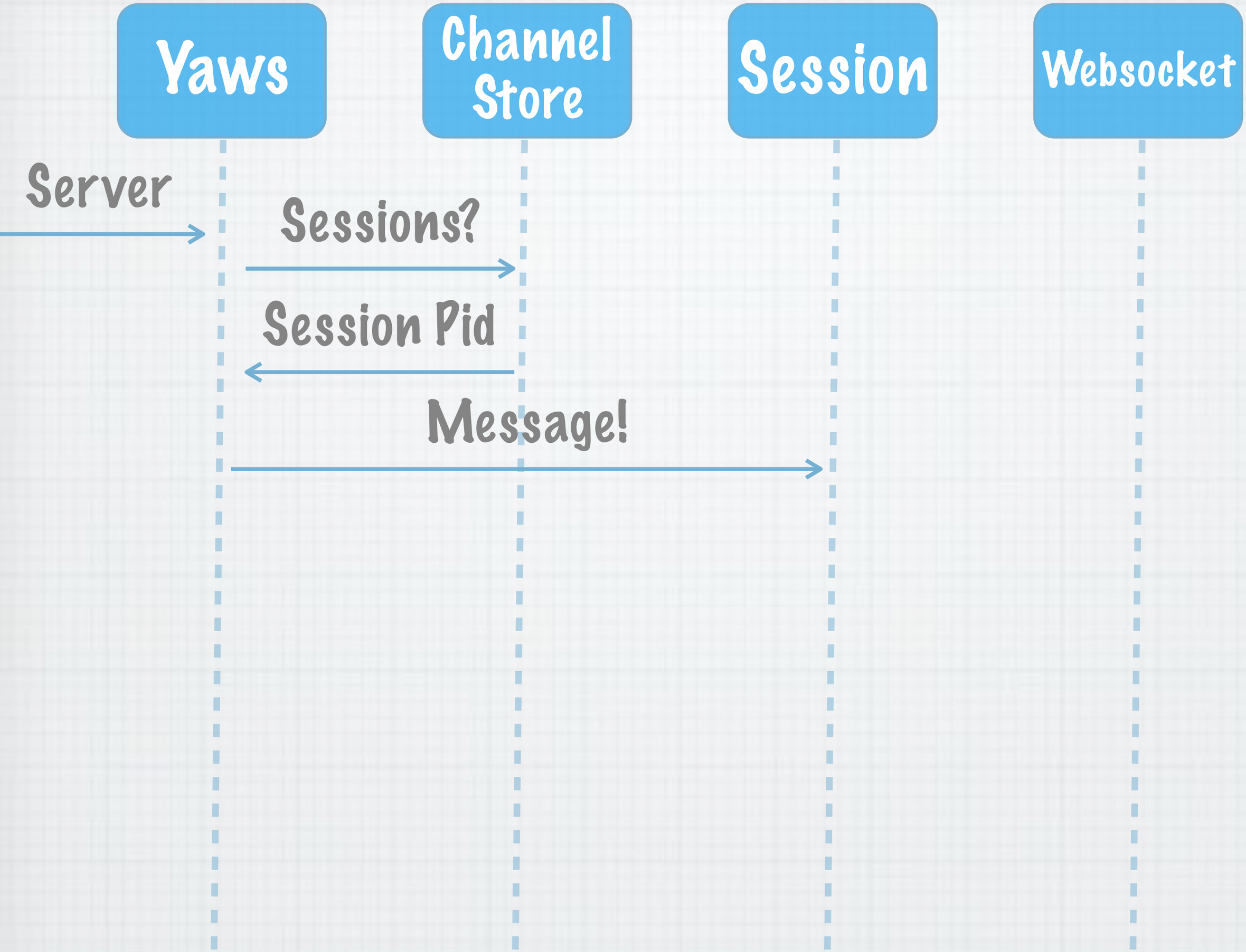
Session

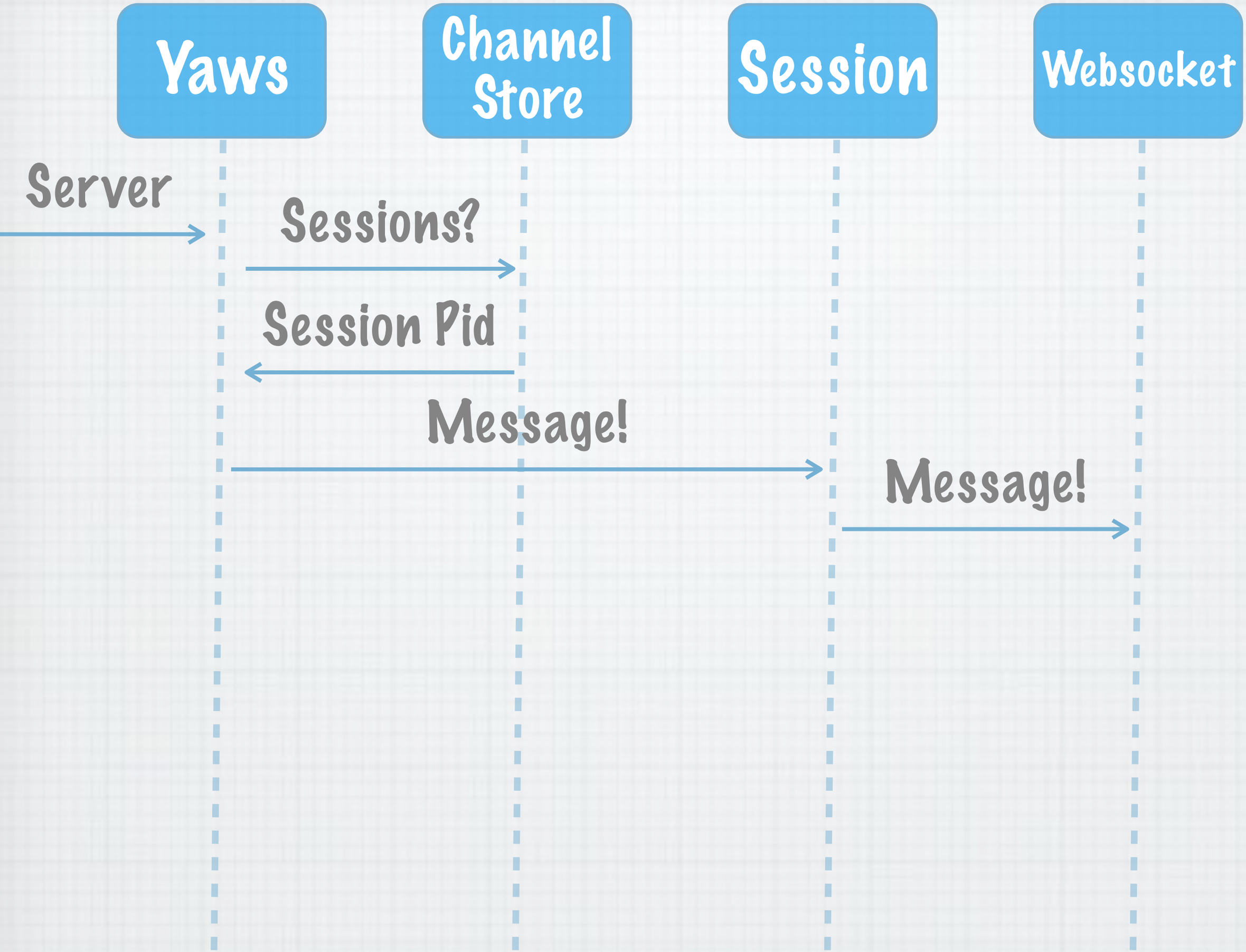
Websocket

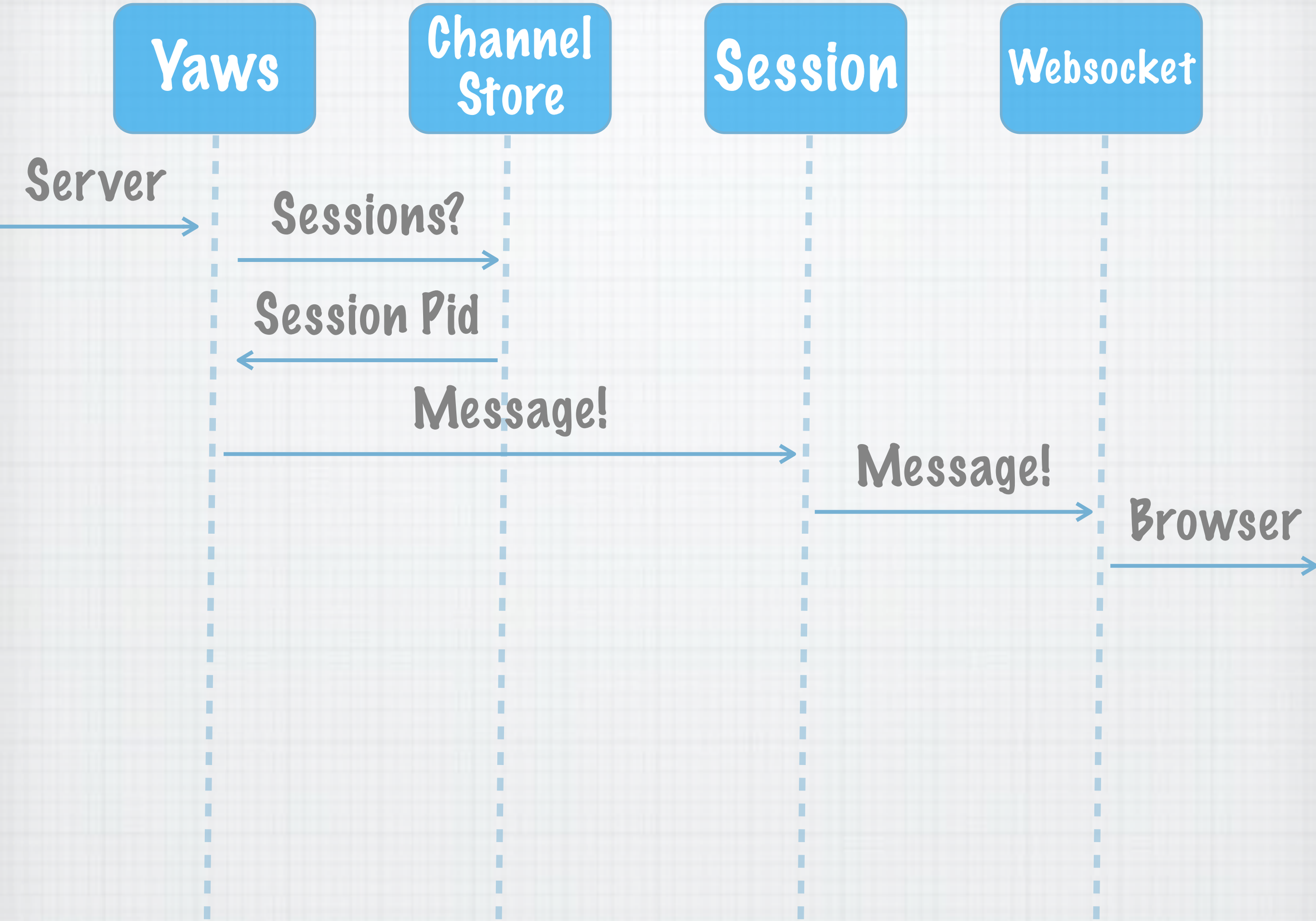

Yaws
Channel Store
Session
Websocket
Server


Yaws
Channel Store
Session
Websocket
Server
Sessions?


Yaws
Channel Store
Session
Websocket
Server
Sessions?
Session Pid


Yaws
Channel Store
Session
Websocket
Server
Sessions?
Session Pid
Message!


Yaws
Channel Store
Session
Websocket
Server
Sessions?
Session Pid
Message!


Yaws
Channel Store
Session
Websocket
Server
Sessions?
Session Pid
Message!
Message!
Browser

Message Structure

```
{ data,
  version,
  type,
  channel,
  id,
  sessionId
}
```

Message Structure

```
{ data,
  version,
  type,
  channel,
  id,
  sessionId
}
```

The message itself

Message Structure

```
{ data,
  version,
  type,
  chann
  id,
  sessionId
}
```

Version of the message protocol

Message Structure

```
{ data,
  version,
  type,
  channel
  id,
  sessionId
}
```

Four types: connect, message, channel-subscribe, poll

Message Structure

```
{ data,
  version,
  type,
  channel,
  id,
  sessi
}
```

Where the message is going

Message Structure

```
{ data,
  version,
  type,
  channel,
  id,
  sessionId
}
```

No Idea...

Message Structure

```
{ data,
  version,
  type,
  channel,
  id,
  sessionId
}
```

Session id for connection

Please Go Play With It

* http://github.com/mashion/chloe
* http://mashion.net
* http://trottercashion.com
* @cashion

Moved to Mountain View

Be my friend?

Thank You!

Good luck w/ Chloe