Message Passing
vs.
Data Synchronization

Anant Narayanan

Pivotal Labs Tech Talk
May 14, 2013

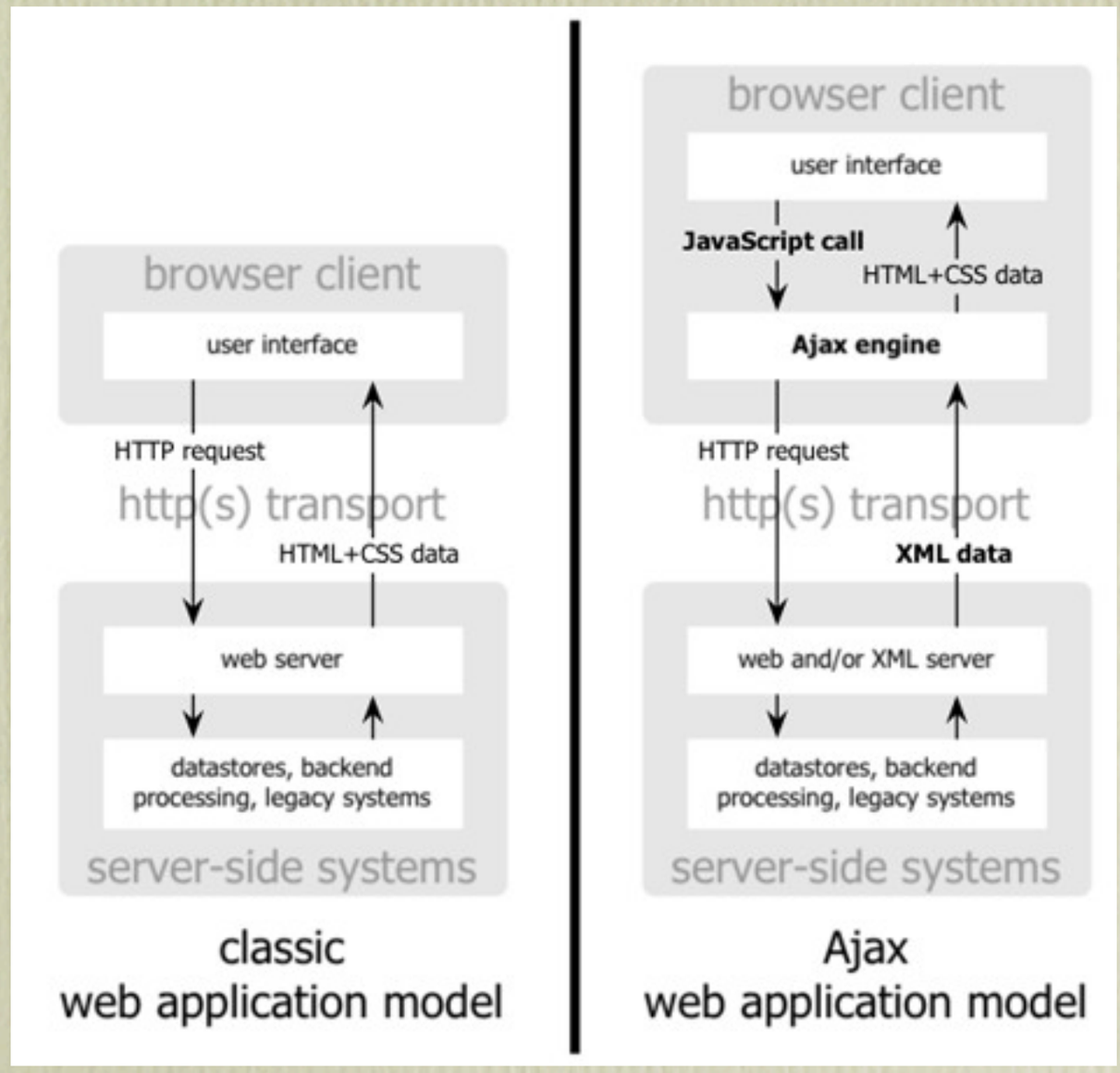
Realtime Synchronization Network

- Build apps fast **without managing servers**
- **Data persistence**, but in **realtime**
- With authentication & security rules, can serve as the **entire backend**

Why?

- Users don't like waiting
- Users want their data wherever they are
  - Between all their devices
  - And sometimes their friends
- 💾 & 🔄 are quickly becoming obsolete

“AJAX”

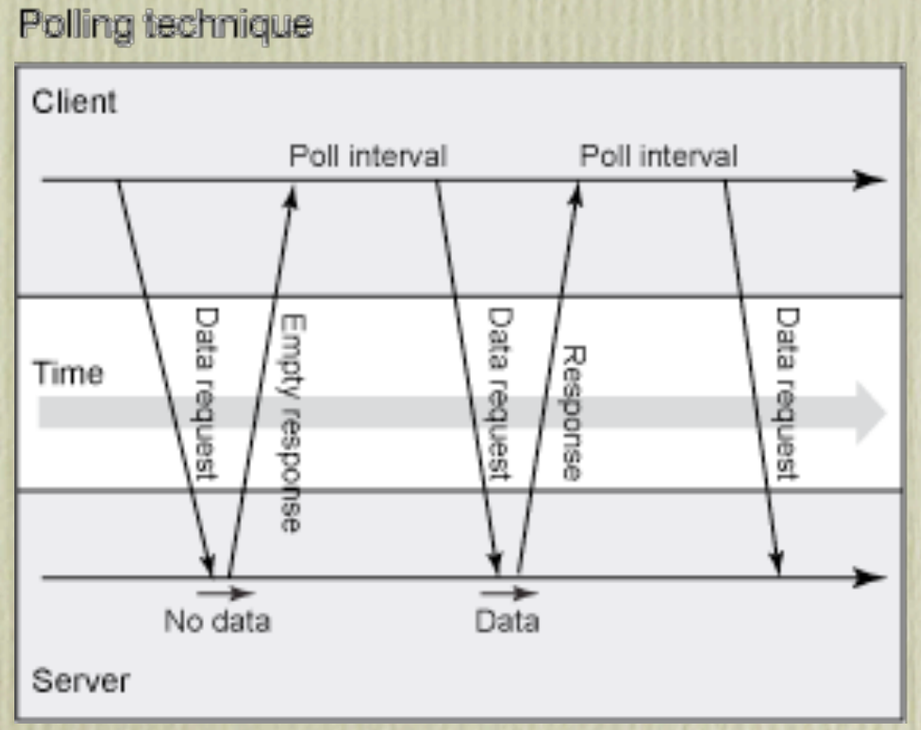

browser client
user interface
HTTP request
http(s) transport
HTML+CSS data
web server
datastores, backend processing, legacy systems
server-side systems
classic web application model
browser client
user interface
JavaScript call
HTML+CSS data
Ajax engine
HTTP request
http(s) transport
XML data
web and/or XML server
datastores, backend processing, legacy systems
server-side systems
Ajax web application model

Not Good Enough

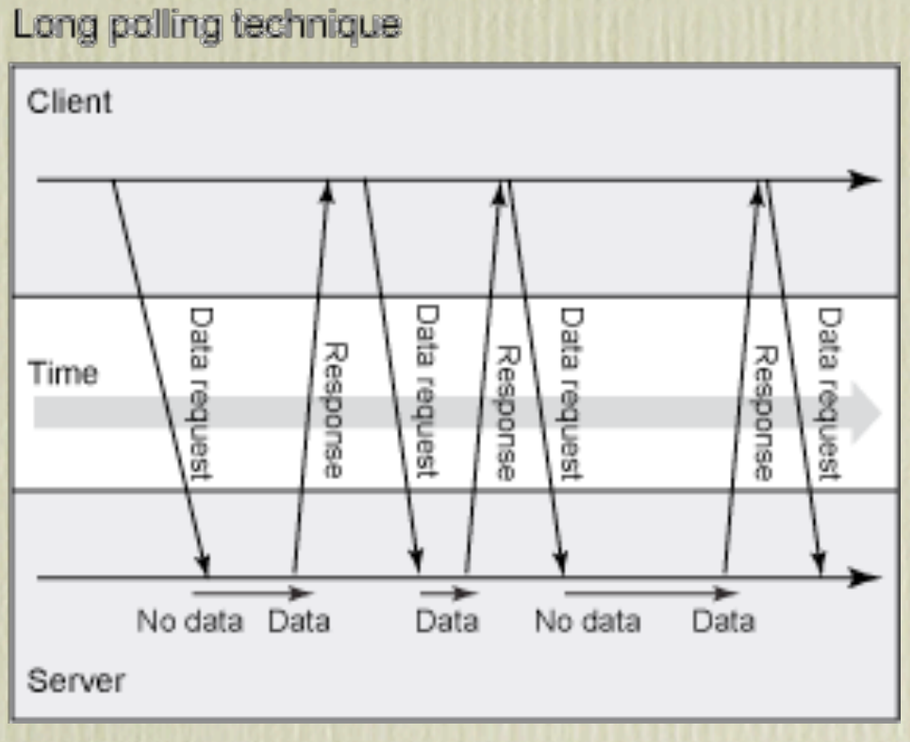
- You still need to know **when** to fetch changes
- Or, you need to **keep asking** about what's new

Long Polling

Polling technique



Long polling technique



- Bidirectional channel between client & server
- Upgrade after initiating a HTTP connection
- Don’t close until you need to

How will you use it?

```
PUBNUB.subscribe({
  channel: "my_channel",
  message: function(m){alert(m)}
});

PUBNUB.publish({
  channel: "my_channel",
  message: "Hello World"
});
```

```
var channel = pusher.subscribe('my-channel');
channel.bind('my-event', function(data) {
  alert('Received my-event with: ' + data.message);
});

pusher.trigger('my-channel',
               'my-event',
               { "message": "hello world" } );
```

Example: Chat

```
var box = PUBNUB.$('box'), input = PUBNUB.$('input');
var channel = 'chat';

PUBNUB.subscribe({
  channel: channel,
  callback: function(text) {
    box.innerHTML = (''+text).replace( /[<>]/g, '' ) +
    '<br>' + box.innerHTML;
  }
});

PUBNUB.bind('keyup', input, function(e) {
  (e.keyCode || e.charCode) === 13 && PUBNUB.publish({
    channel: channel, message: input.value,
    x: (input.value='')
  });
});
```

An Alternative Approach

```
var chatRef =
  new Firebase('https://chat.firebaseio-demo.com/');

$('#messageInput').keypress(function(e) {
  (e.keyCode == 13) && chatRef.push({
    name: $('#nameInput').val(),
    text: $('#messageInput').val()
  }) && $('#messageInput').val('');
});

chatRef.limit(10).on('child_added', function(s) {
  var message = s.val();
  $('<div/>').text(message.text).prepend($('<em/>')
    .text(message.name+': '))
    .appendTo($('#messagesDiv'));
});
```

Why Data Synchronization?

Simpler Client Logic

There's a lot of **complexity** in turning a **stream of messages** into state that is **usable**

Why Data Synchronization?

Greater Efficiency

We have the **flexibility** to **combine** operations

Why Data Synchronization?

Automatic Merge Behavior

With message passing, **conflict resolution** can get tricky, but with data synchronization it can be a **core primitive**

Why Data Synchronization?

Greater Flexibility

Message passing is simply a **subset** of data synchronization

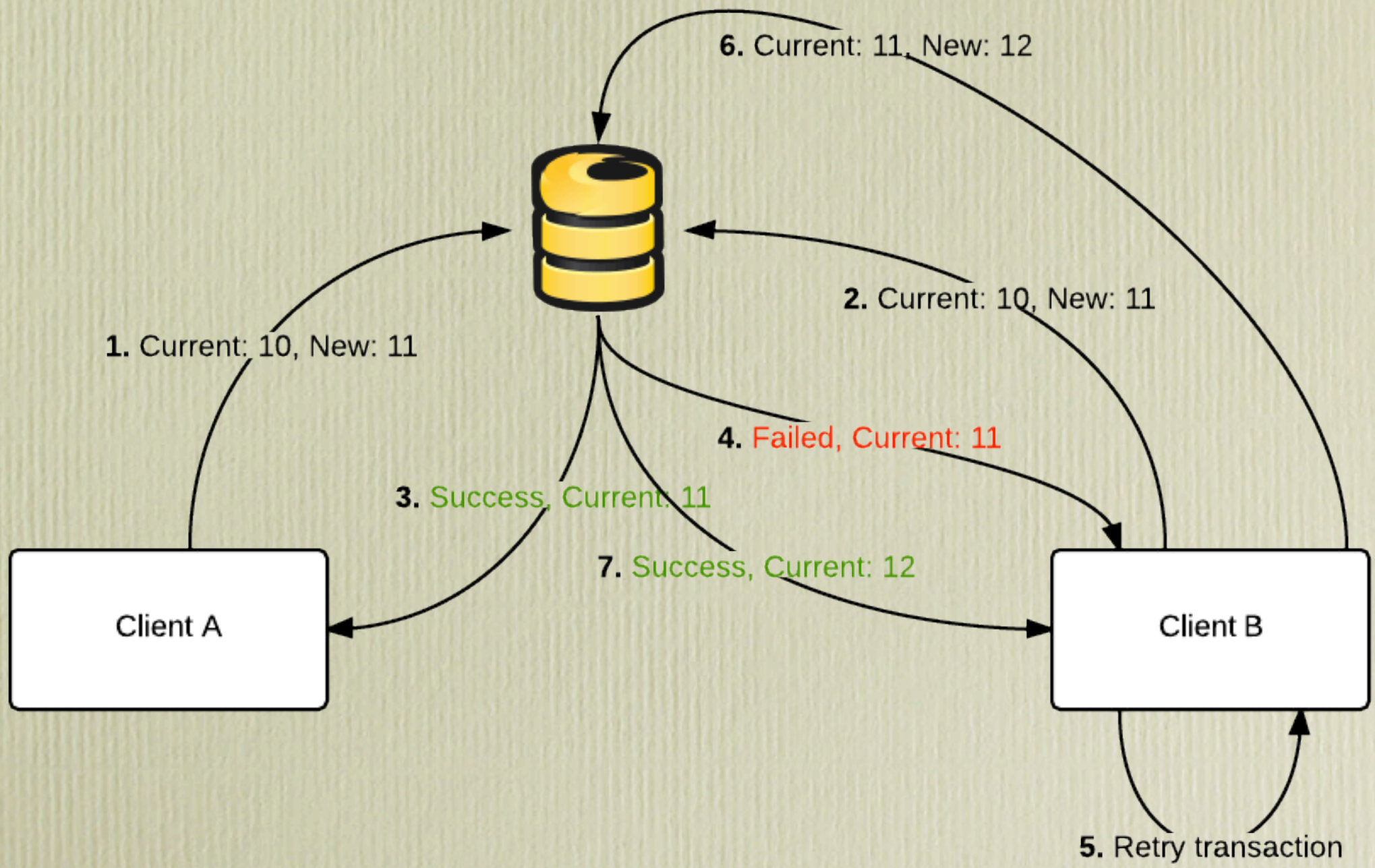
Example: Counter

Transactions can be a built in primitive

```
var countRef = new Firebase(
 'https://example.firebaseio.com/counter');

countRef.transaction(function(current_value) {
  return current_value + 1;
});
```

Example: Counter



Latency Compensation

There may be several "*incorrect" intermediate steps*
but the system as a whole is **eventually consistent**

eg: Events can always be optimistically triggered locally

Thank You

Write your app logic as a function of
current state instead of **deltas**

Store state instead of passing messages

anant@firebase.com

twitter @anantn
www kix.in