CLOUD COMPUTING APPLICATIONS

Cloud Computing Glue: Asynchronous RPC, WebSocket

Prof. Reza Farivar

Asynchronous RPC, aka. Streaming API

- Using old HTTP/0.9 and 1.0, what if the remote server takes a long time?
  - The client can wait, blocked, keeping the HTTP connection open (long polling)
  - It can keep polling the server periodically
  - Ultimately, these are hacks
  - HTTP/1.1 not ready for real-time web
- Web 2.0: Bidirectional client / server communication
  - AJAX
  - Comet
    - Umbrella term: Ajax Push, Reverse Ajax, Two-way-web, HTTP Streaming, and HTTP server push, …
  - Push notifications
  - XMLHttpRequest
    - XMLHttpRequest (XHR) is an API in the form of an object whose methods transfer data between a web browser and a web server.
    - The object is provided by the browser's JavaScript environment.
    - jQuery provides a nice wrapper (it also encapsulates WebSockets and server push)

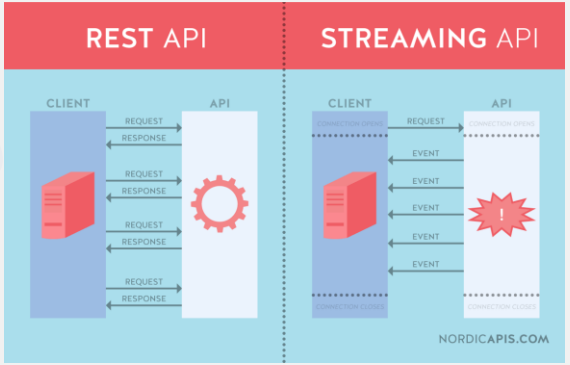


Image from https://nordicapis.com/rest-vs-streaming-apis-how-they-differ/

WebSocket

- RPC libraries can issue asynchronous methods to the server, and the server can inform them of the response later
- Streaming Architecture
  - Minimizing latency
- WebSocket
  - Part of HTML 5 standard
  - Can handle interactive sessions better than RESTful architecture
- Use cases:
  - Chat
  - Stock price update
  - Collaborative Document Editing
  - Location update (I am here now)
  - Multiplayer games

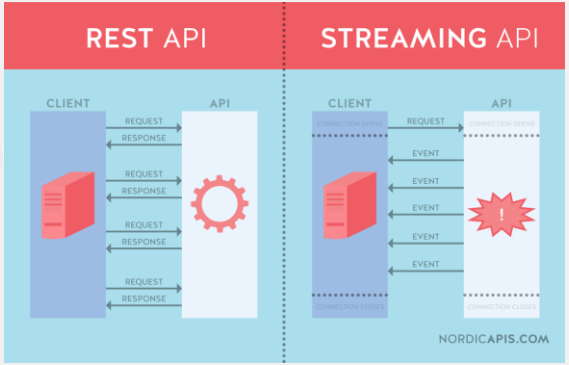


Image from https://nordicapis.com/rest-vs-streaming-apis-how-they-differ/

WebSocket Protocol

- WebSocket is an application protocol, running on top of TCP
  - It uses URIs, but not http://
  - Instead, uses ws://
- It utilizes an initial HTTP session and HTTP port numbers to process handshake phase
- The protocol has 3 phases
  - Opening Handshake
  - Data Transfer
  - Closing Handshake
- Protocol “upgrade”
- Purely event driven
  - Application code listens for events on WebSocket objects to handle incoming data and changes in connection status
- Asynchronous programming
  - Client does not need to do anything (e.g. poll) to receive data

WebSocket 3 phases

1) Opening Handshake
   - HTTP request/response to open WebSocket connection
   - Example client request

```
GET /chat HTTP/1.1
Host: example.com
Upgrade: websocket
Connection: Upgrade
Origin: http://example.com
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Sec-WebSocket-Version: 13
```

   - Example server response

```
HTTP/1.1 101 Switching Protocols
         Upgrade: websocket
         Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
```

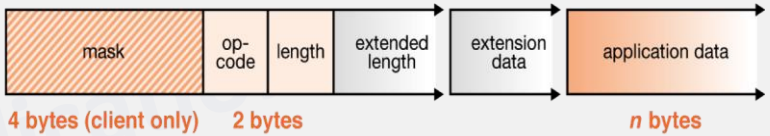
   - HTTP protocol is switched (aka. upgraded) to WebSocket

2) Data Transfer
- Bidirectional communication
- WebSocket frame
- Description of fields:
  - Op-code: Continuation, Text, Binary, Close, Ping, Pong

3) Closing handshake
   - A WebSocket frame with opcode 0x8 is sent

To keep the connection alive

WebSocket Frame



WebSocket API

- Simple Javascript W3C WebSocket API
- four different events:
  - Open - fires to establish a connection
  - Message - contains the data from the server
  - Error - fires in response to an unexpected event (failure)
  - Close - fires when the WebSocket connection is closed

- Primary methods + events
  - `WebSocket(URL, [protocols])` – Create a connection
    - `var ws = new WebSocket("ws://www.websocket.org","SOAP");//"SOAP" is optional`
  - `onOpen()` – WebSocket opened
    - `ws.onOpen = function(e){ console.log(ws.protocol);}`
  - `Send(data)` – Send data (string, Blob or Arraybuffer)
    - `ws.send("Hello WebSocket!");`
  - `onMessage()` – Message received
    - `ws.onMessage = function(e) { log("Message received: " + e.data); ws.close(); }`
  - `onClose()` – Close message received
    - `ws.onOpen = function(e){ console.log ("Disconnected: "+ e.reason);}`
  - `onError()` - Error

WebSocket

- Note that once you allow asynchronous communication in a networked environment, you should handle faults
- There are wrapper packages, encapsulating WebSockets functionality with additional features
  - Node.js supports WebSockets through plugins
  - Example: Socket.io.
  - Consisting of a Node.js server and a Javascript client library, socket.io provides reliability for handling proxies and load balancers as well as personal firewall and antivirus software and even supports binary streaming.
- Java 11 supports both HTTP as well as WebSocket protocols

WebSocket in Cloud Computing

- Many Cloud providers support WebSocket as the API of choice for interactive sessions with the Cloud service and the client
  - Amazon AWS API Gateway
  - Salesforce
  - Many cloud video vendors
    - E.g. easylive.io
  - Basis of Slack and its “Real Time Messaging API”
    - Slack has a simpler Event API based on HTTP/2 Push, but only the RTM guarantees real time delivery of messages
  - Google App Engine Channel API used WebSocket to allow server to client messaging
    - *Now deprecated, replaced by Firebase (a MBaaS solution)*
    - We can always run a Socket.io instance on Computer Engine (or EC2) instance
      - https://cloud.google.com/solutions/real-time-gaming-with-node-js-websocket
    - Or use AWS API Gateway to handle the deployment of WebSocket servers for us