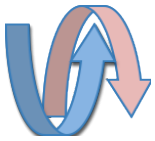
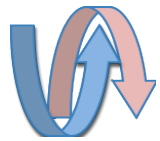
Zodiark --The Atmosphere Framework

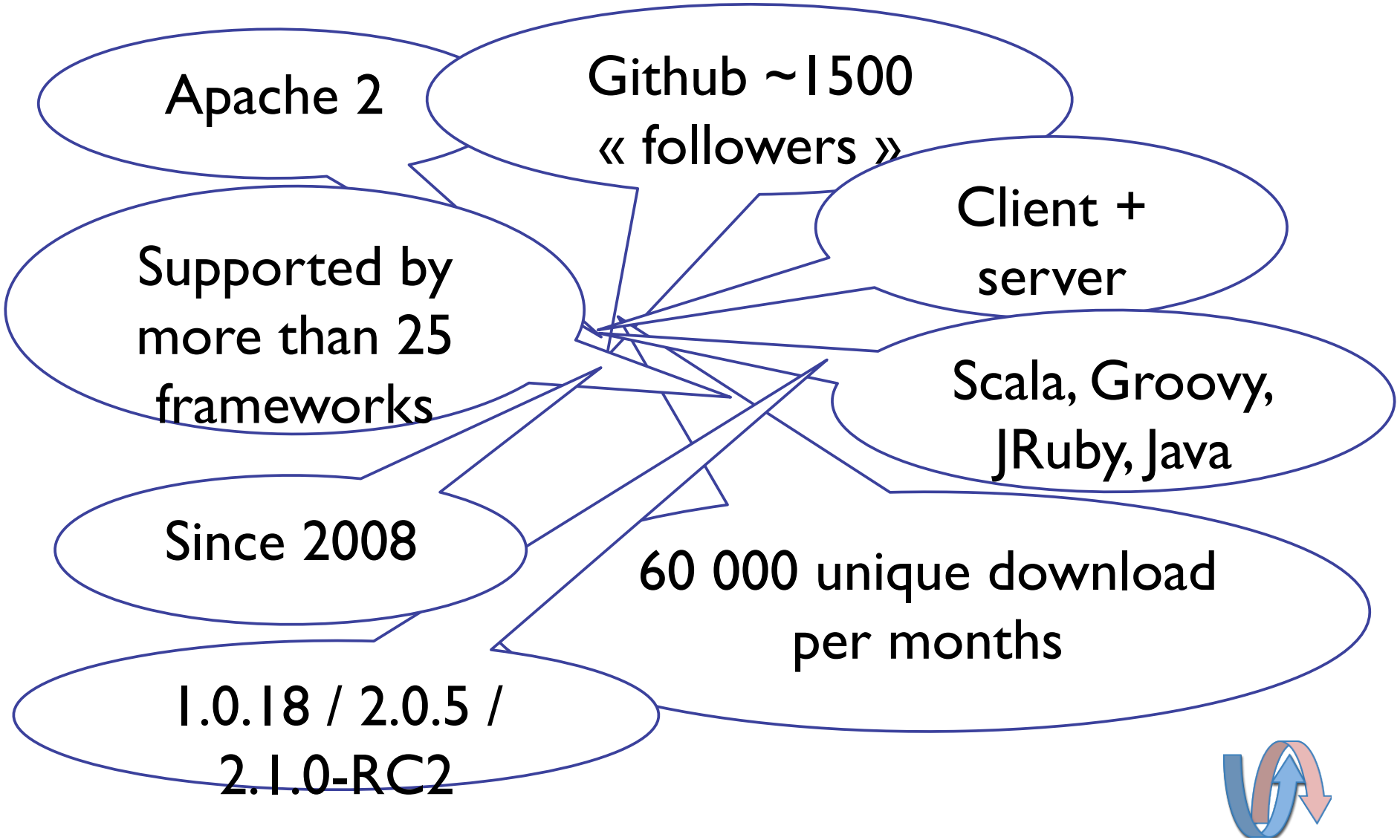


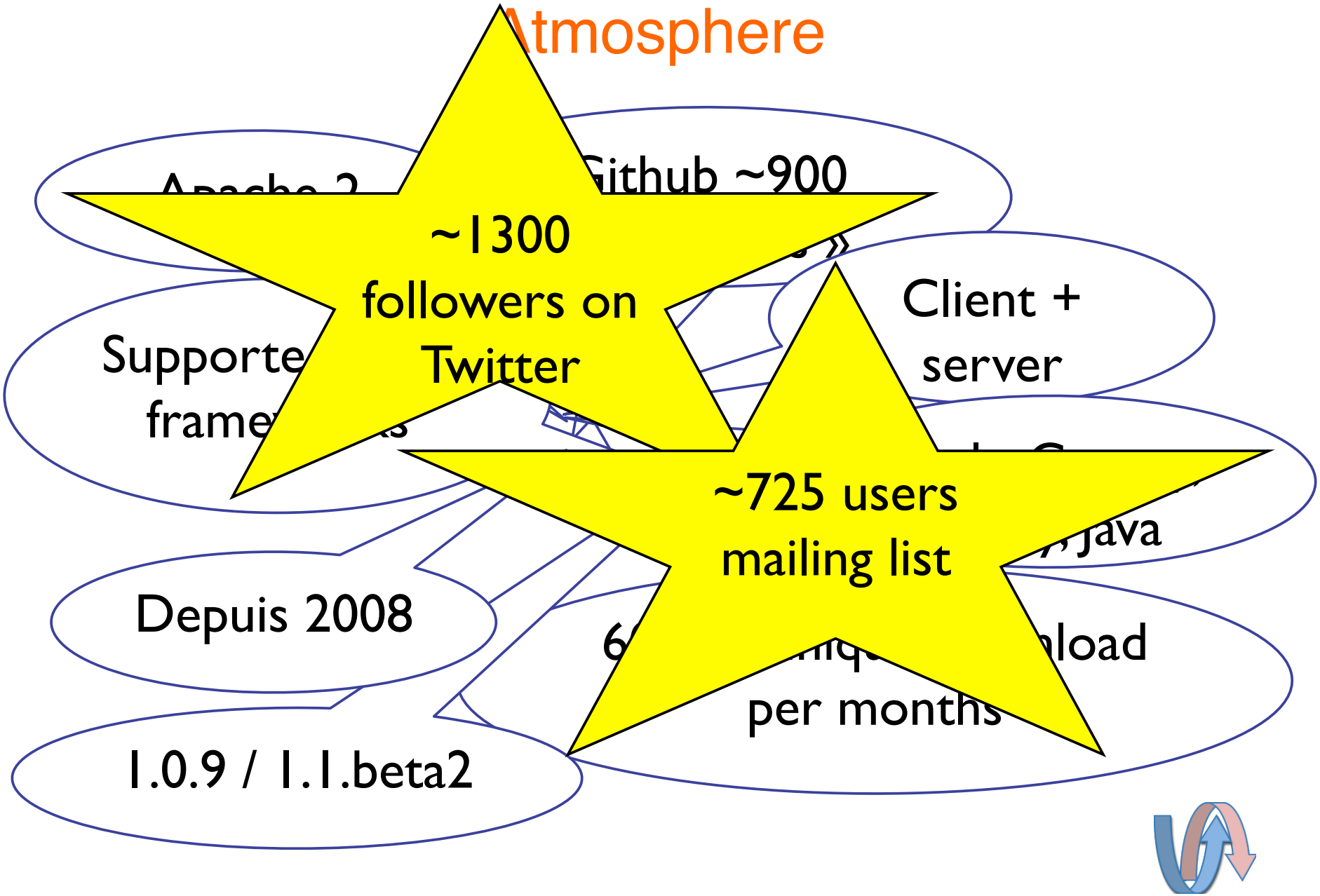
Atmosphere

- Introduction
- Case Study
- WebSockets and Comet: the issues
- Browsers and Server
- Atmosphere Concepts
- Zodiak Concepts



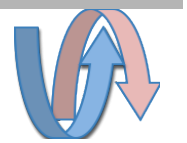
Atmosphere


Apache 2
Github ~1500
« followers »
Client +
server
Supported by
more than 25
frameworks
Scala, Groovy,
JRuby, Java
Since 2008
60 000 unique download
per months
1.0.18 / 2.0.5 /
2.1.0-RC2


Atmosphere
Github ~900
~1300
followers on
Twitter
Client +
server
Supporte
frame
~725 users
mailing list
Depuis 2008
per months
1.0.9 / 1.1.beta2

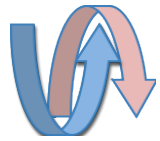
WSJ.com

- **60 millions de Requests par jours**
- Atmosphere 1.0.13/Jetty 8.1.9
- **Websockets => long-polling => JSONP**
- IE 6/7/8/9/10, Chrome XX, Firefox 3/12, Safari XX
- **WebSockets: IE 10, Chrome 14, Firefox 8, Safari 5**



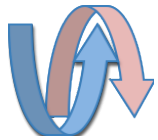
WebSockets

WebSocket: A Socket, that's it!



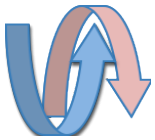
WebSockets

WebSocket is a web technology providing full-duplex communications channels over a single TCP connection. The WebSocket API is being standardized by the W3C, and the WebSocket protocol has been standardized by the IETF as RFC 6455.

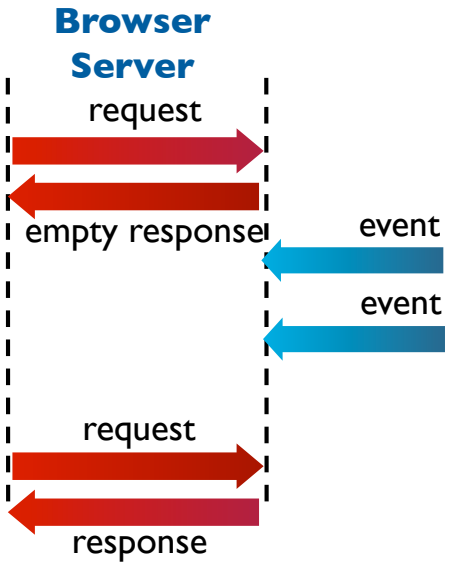


WARNING!!

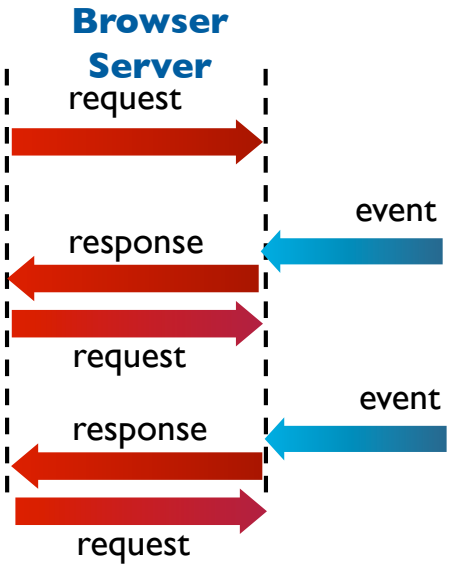
Nobody is/will/was fired because of long-polling!!!



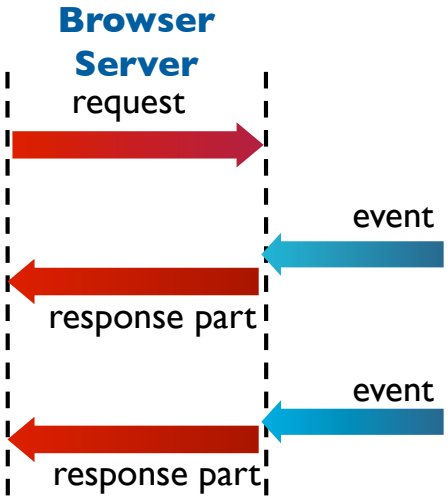
Polling



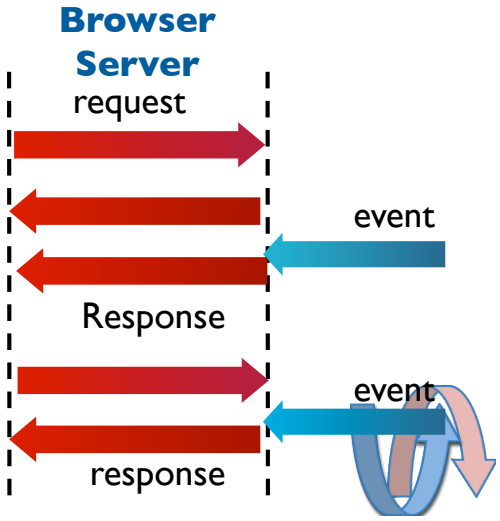
Long Polling



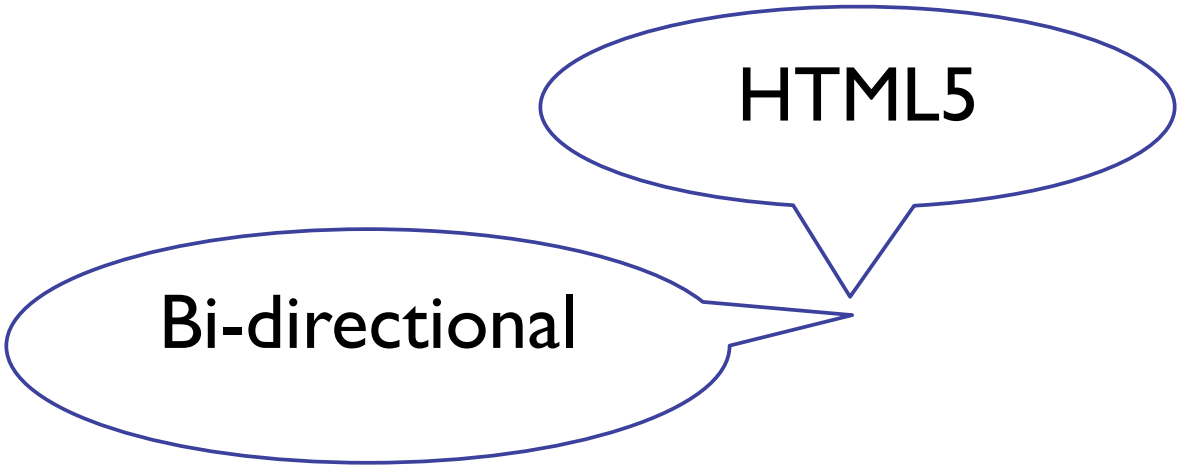
Streaming

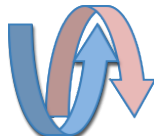


WebSocket =>

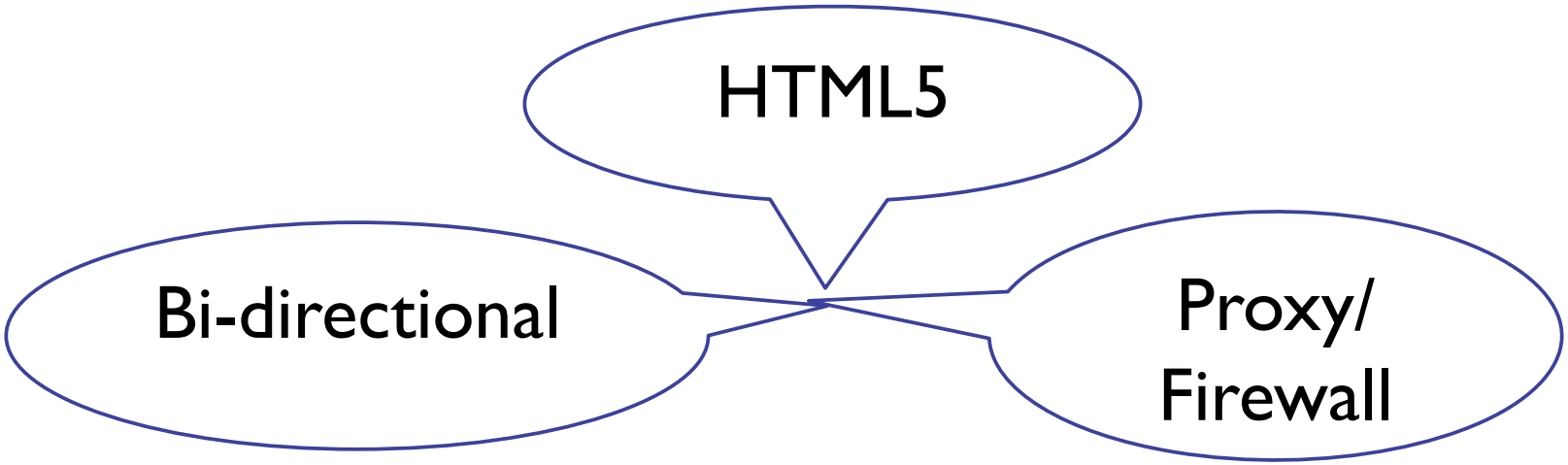


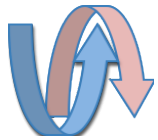
WebSockets



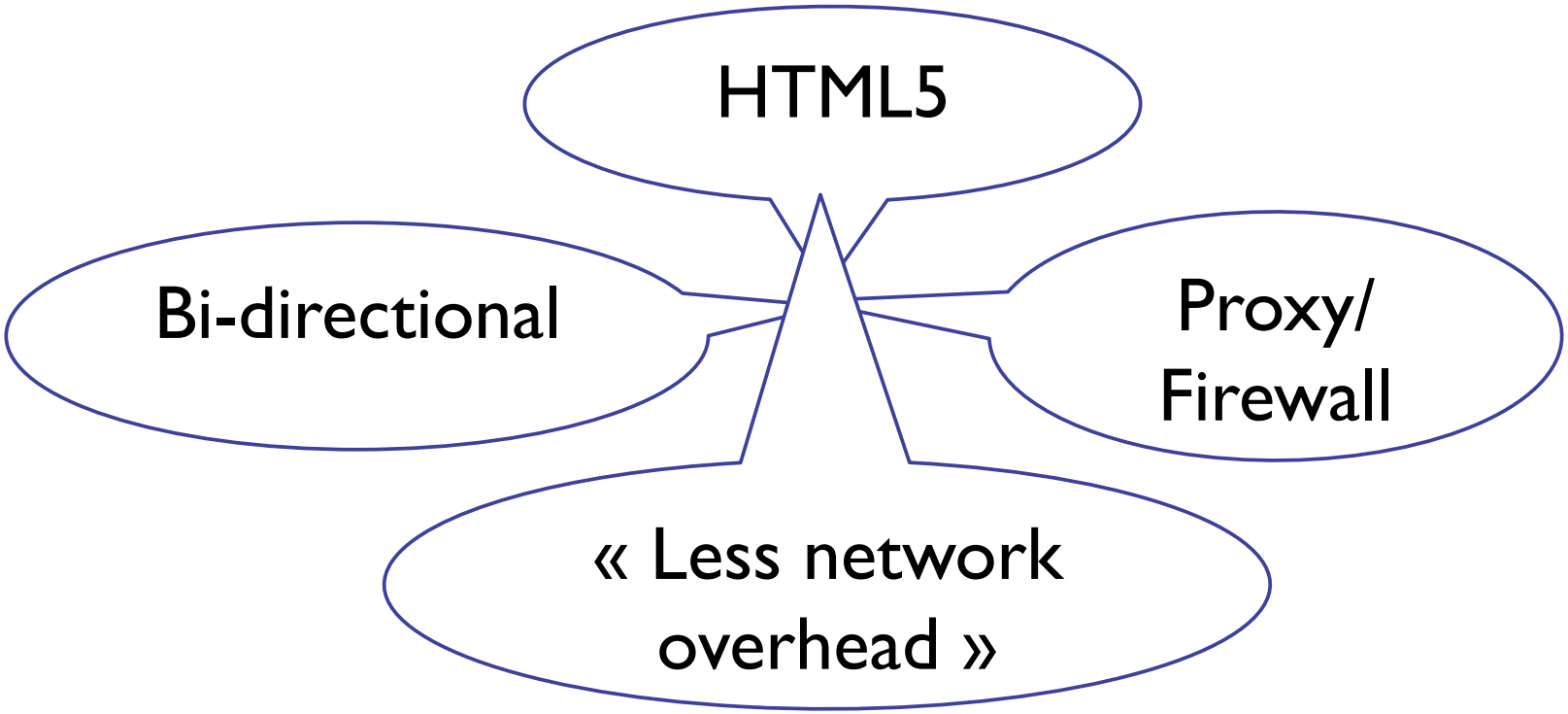


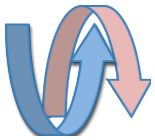
WebSockets



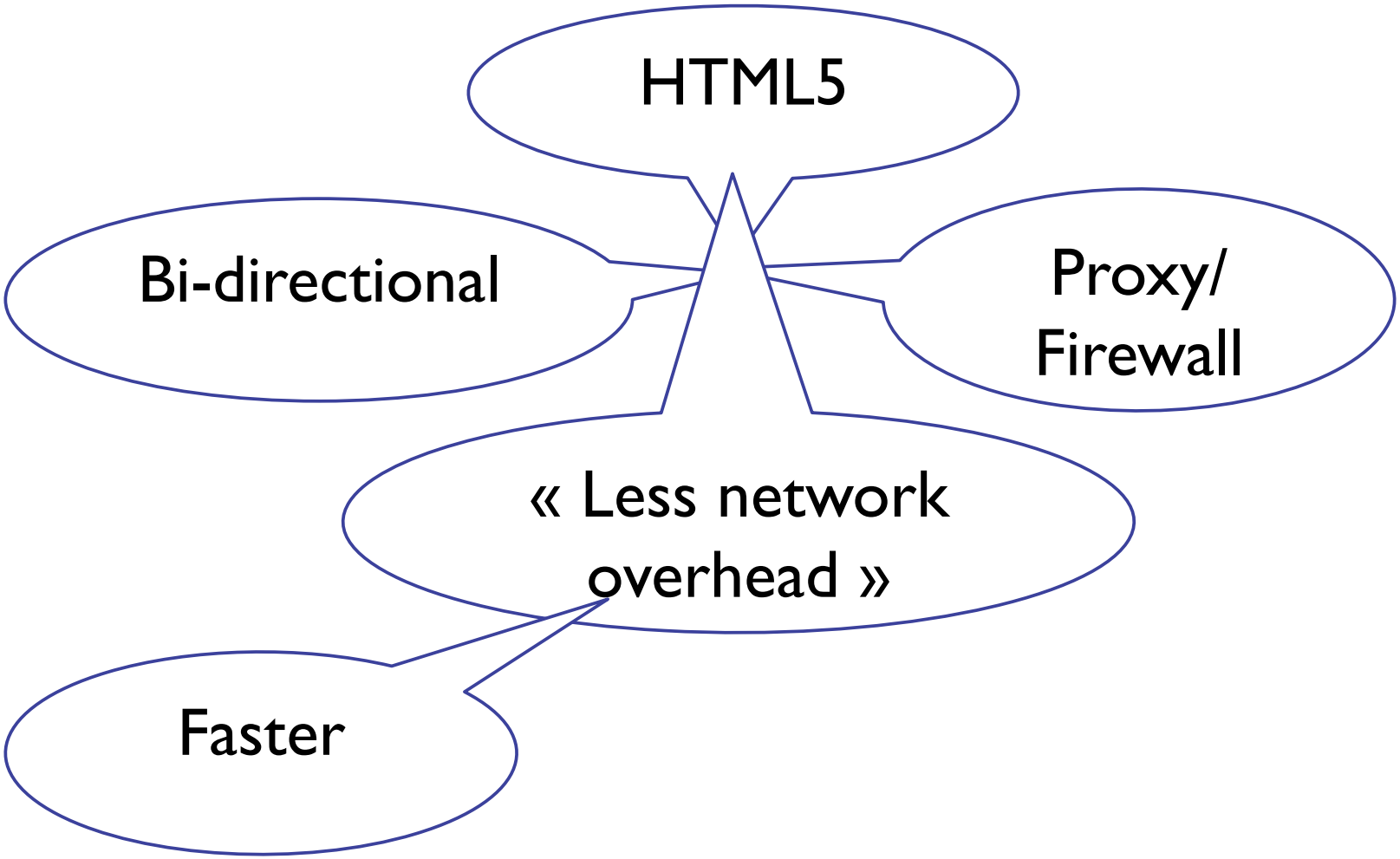


WebSockets

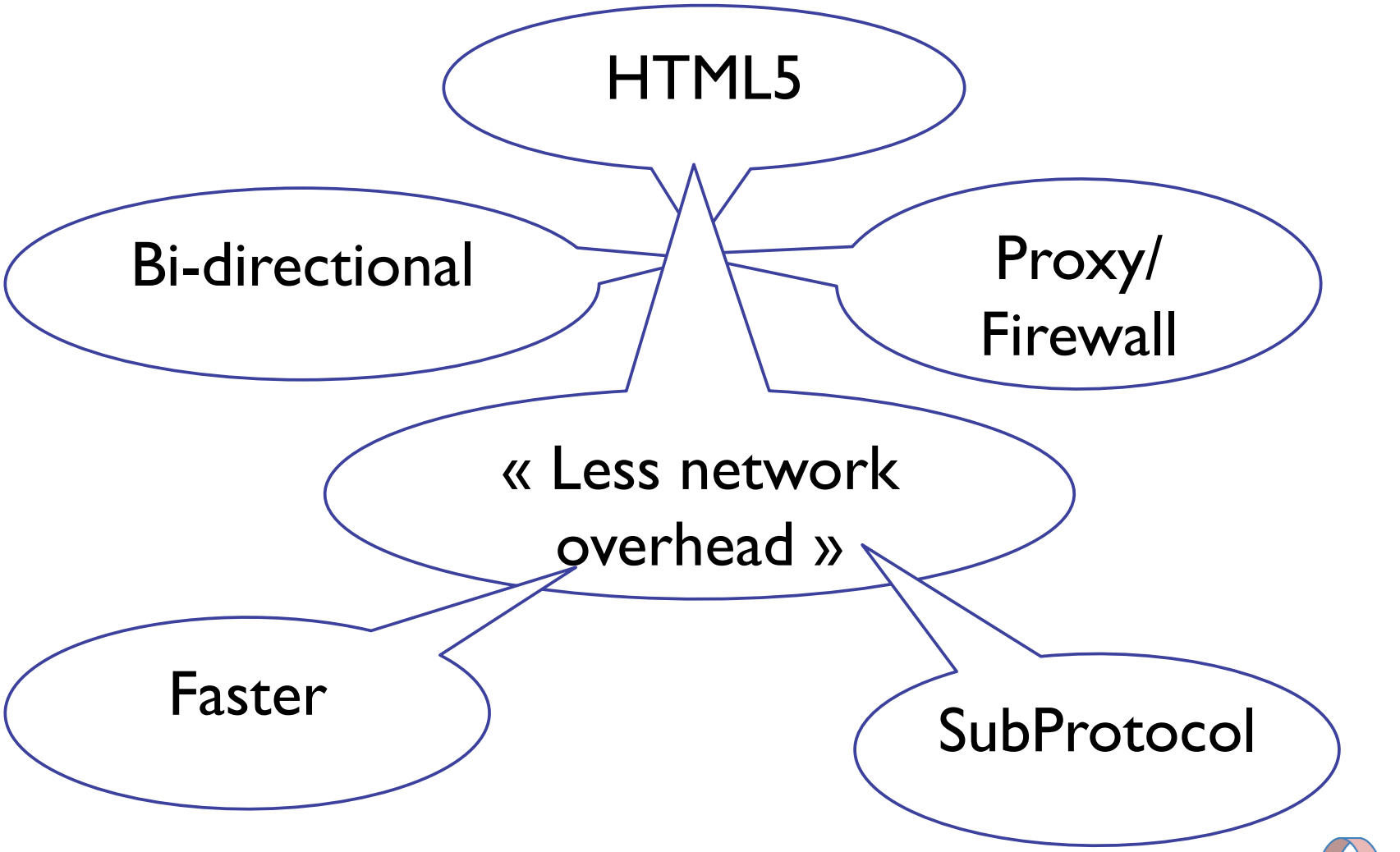

HTML5
Bi-directional
Proxy/
Firewall
« Less network
overhead »



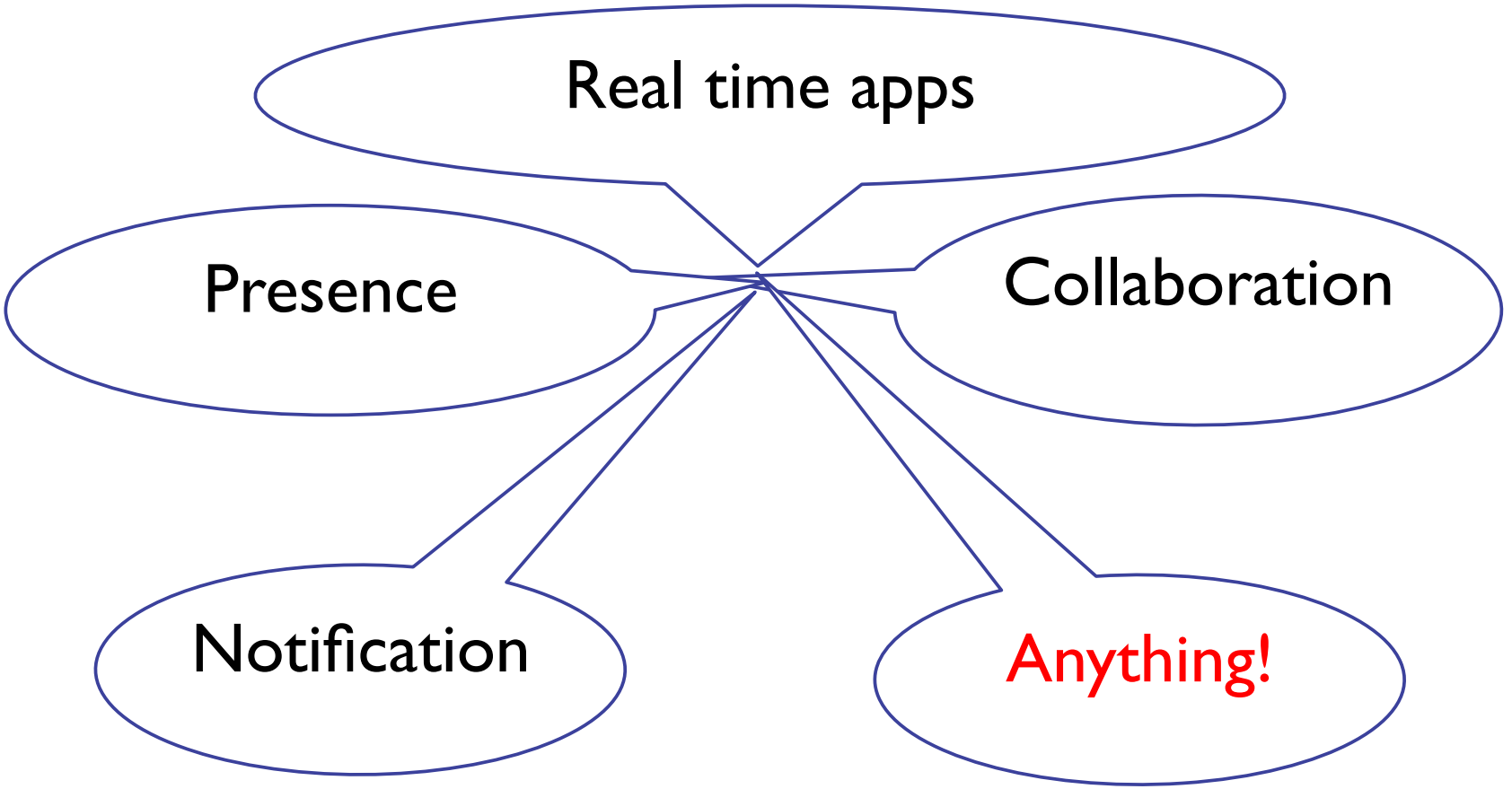
WebSockets

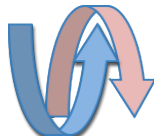

HTML5
Bi-directional
Proxy/
Firewall
« Less network
overhead »
Faster

WebSockets


HTML5
Bi-directional
Proxy/
Firewall
« Less network
overhead »
Faster
SubProtocol

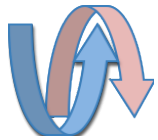
WebSockets


Real time apps
Presence
Collaboration
Notification
Anything!



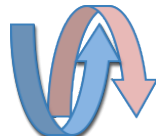
First request

```
T 127.0.0.1:65062 -> 127.0.0.1:8080 [AP]
GET / HTTP/1.1.
Upgrade: websocket.
Connection: Upgrade.
Host: 127.0.0.1:8080.
Origin: http://127.0.0.1:8080.
Sec-WebSocket-Key: Tz9qdt3lmte6Slf+GvpRqQ==.
Sec-WebSocket-Version: 13.
Sec-WebSocket-Extensions: x-webkit-deflate-frame.
```



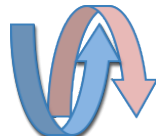
Second Request

```
T 127.0.0.1:8080 -> 127.0.0.1:51292 [AP]
HTTP/1.1 101 Switching Protocols.
Upgrade: WebSocket.
Connection: Upgrade.
Sec-WebSocket-Accept: HVXA7SqH5uYeN6aD9tZ0JQbfTJA=.
```



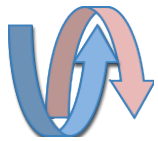
Life if good!

```
T 127.0.0.1:8080 -> 127.0.0.1:51292 [AP]
HTTP/1.1 101 Switching Protocols.
Upgrade: WebSocket.
Connection: Upgrade.
Sec-WebSocket-Accept: HVXA7SqH5uYeN6aD9tZ0JQbfTJA=.
```



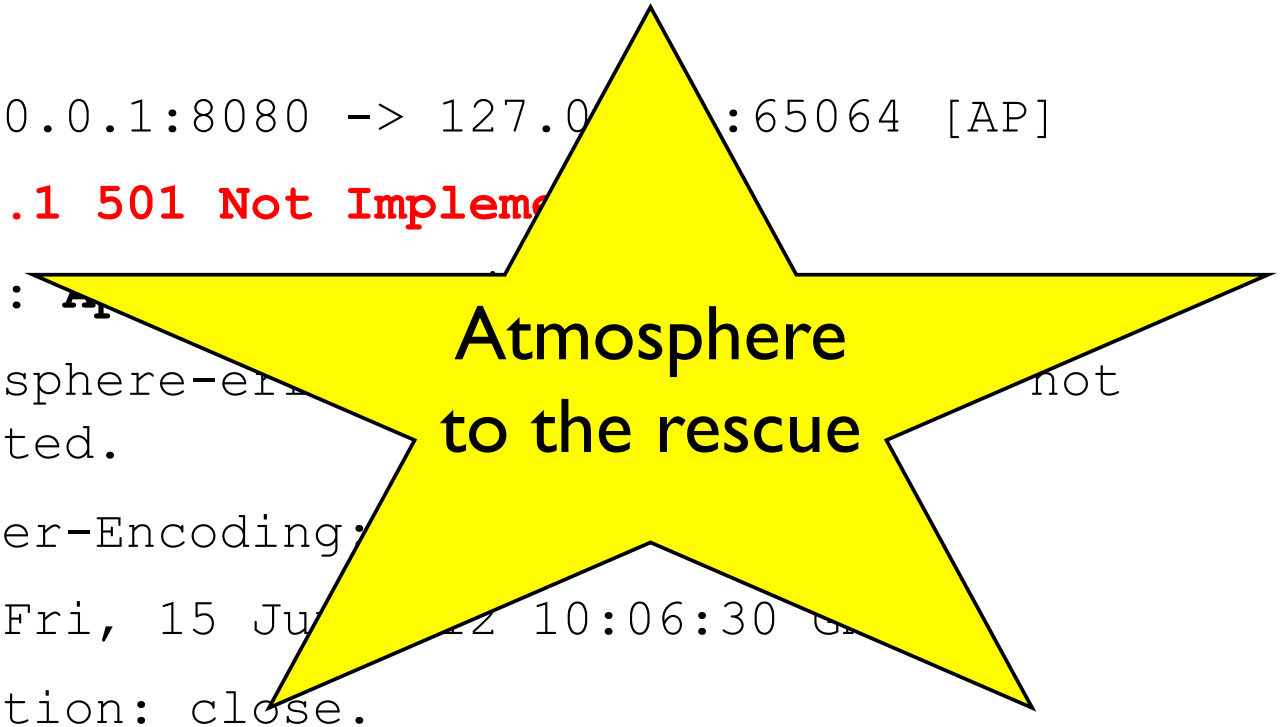
Not supported everywhere

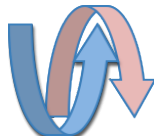
```
T 127.0.0.1:8080 -> 127.0.0.1:65064 [AP]
HTTP/1.1 501 Not Implemented.
Server: Apache-Coyote/1.1.
X-Atmosphere-error: Websocket protocol not supported.
Transfer-Encoding: chunked.
Date: Fri, 15 Jun 2012 10:06:30 GMT.
Connection: close.
.
```



Not supported everywhere

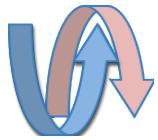
```
T 127.0.0.1:8080 -> 127.0.    :65064 [AP]
HTTP/1.1 501 Not Implem
Server: A
X-Atmosphere-er                                    not
supported.
Transfer-Encoding:
Date: Fri, 15 Ju    12 10:06:30 G
Connection: close.
.
```





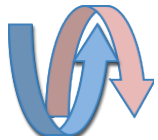
WebSocket API – Standard JavaScript

```
websocket = new WebSocket(wsUri);
websocket.onopen = function(evt) { …};
websocket.onclose = function(evt) { …};
websocket.onmessage = function(evt) { …};
websocket.onerror = function(evt) { …};

webSocket.send(…)
```



Server – No Standard yet

- Node.js
- Pusher
- Jetty
- GlassFish
- Tomcat
- Apache
- NIO Framework like Netty and Grizzly



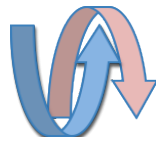
WebSocket API – Java

Jetty 7/8/9, GlassFish 3.1, Netty 3, Tomcat 7.0.27 and up, Resin 4, JBoss 7.1.2 and up

JSR 356: http://jcp.org/en/jsr/detail?id=356

AHC (Client -> De facto)

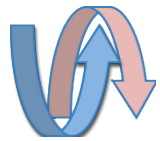
http://github.com/sonatype/async-http-client



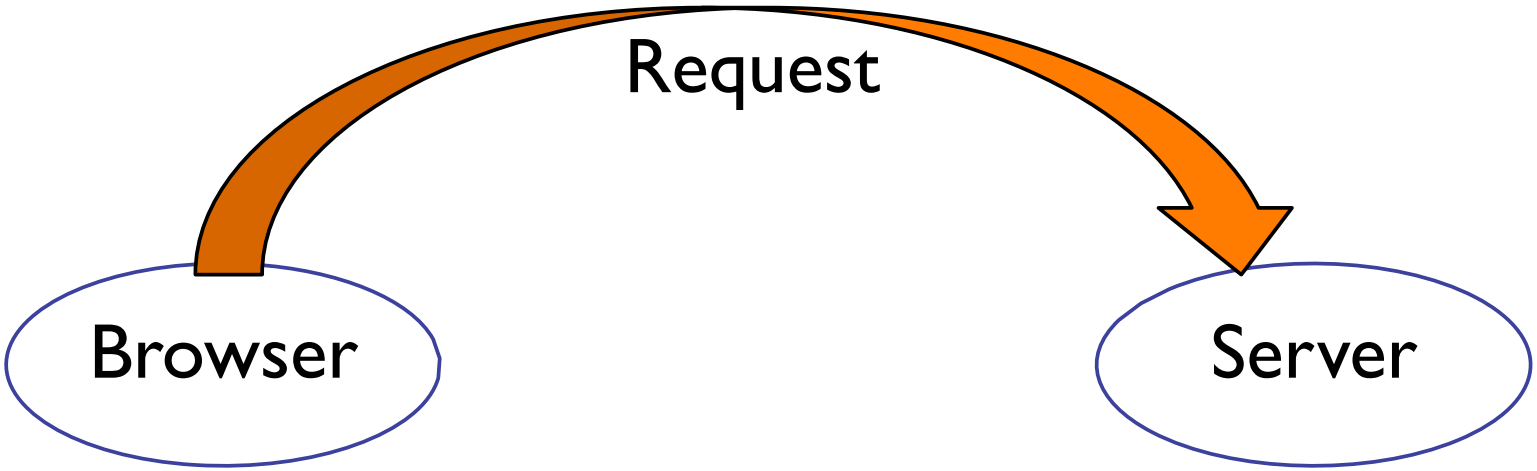
WebSocket API – Java & Scala

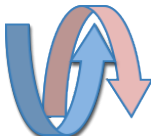
wAsync:

https://github.com/Atmosphere/wasync

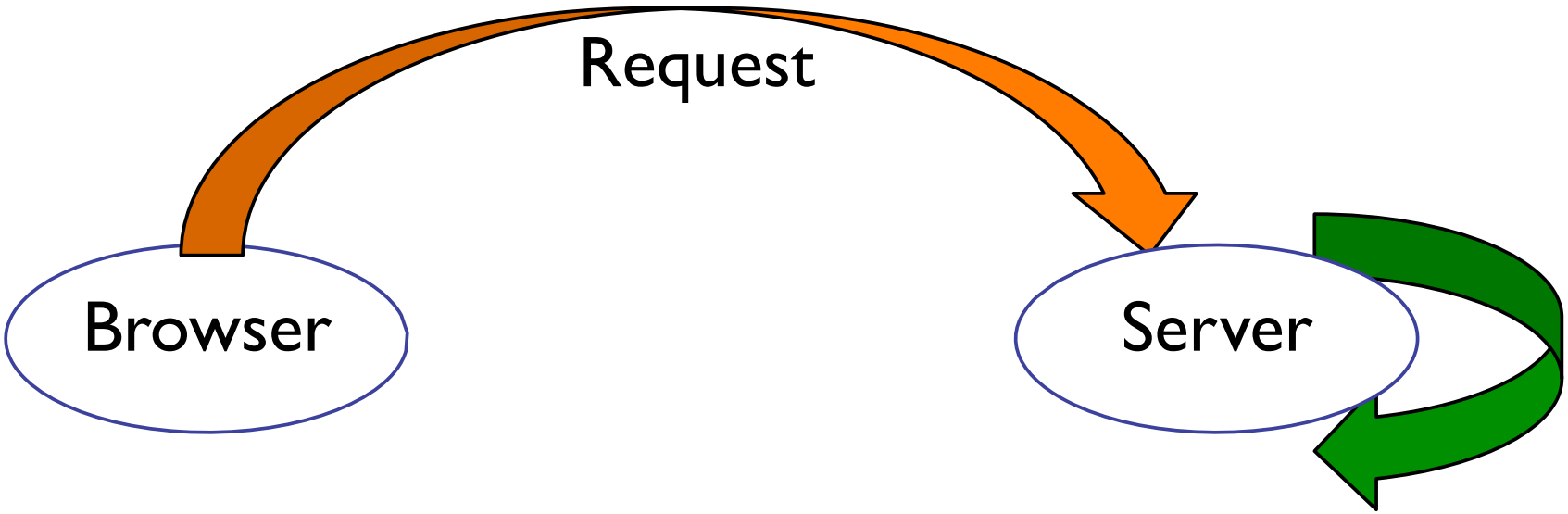


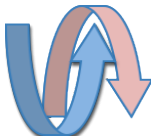
Before (Long-Polling)



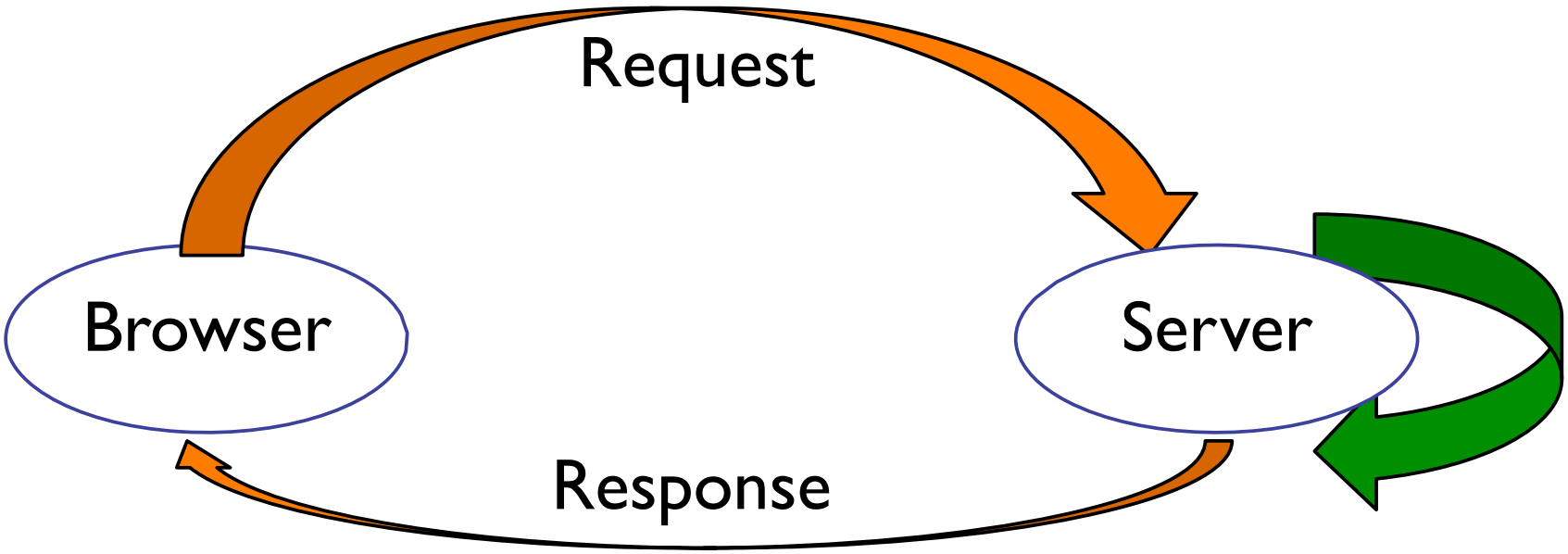


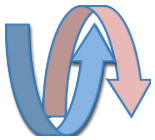
Before (Long-Polling)



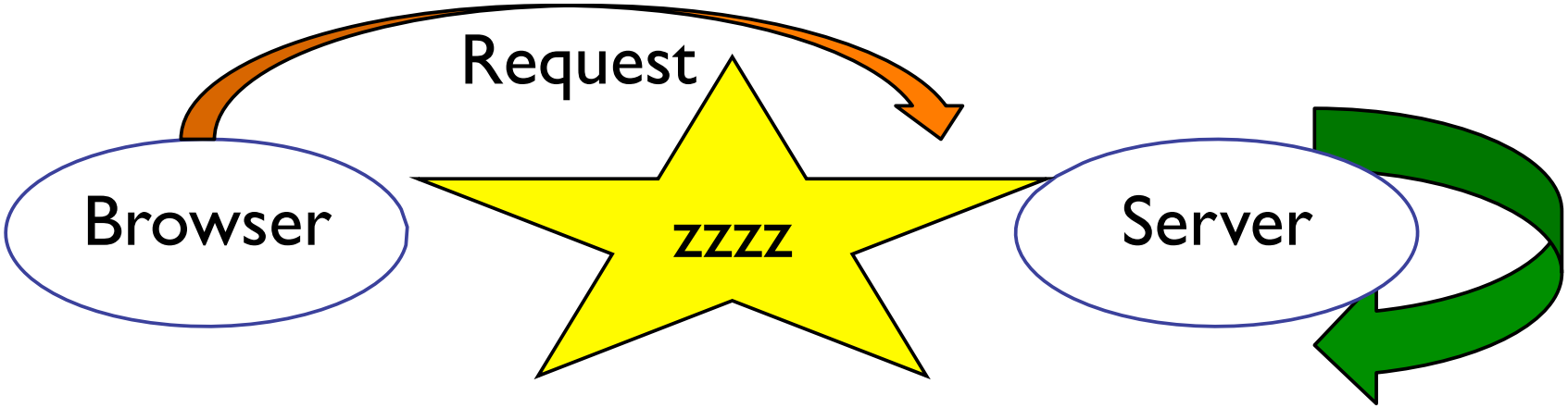


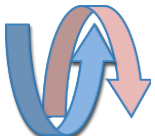
Before (Long-Polling)



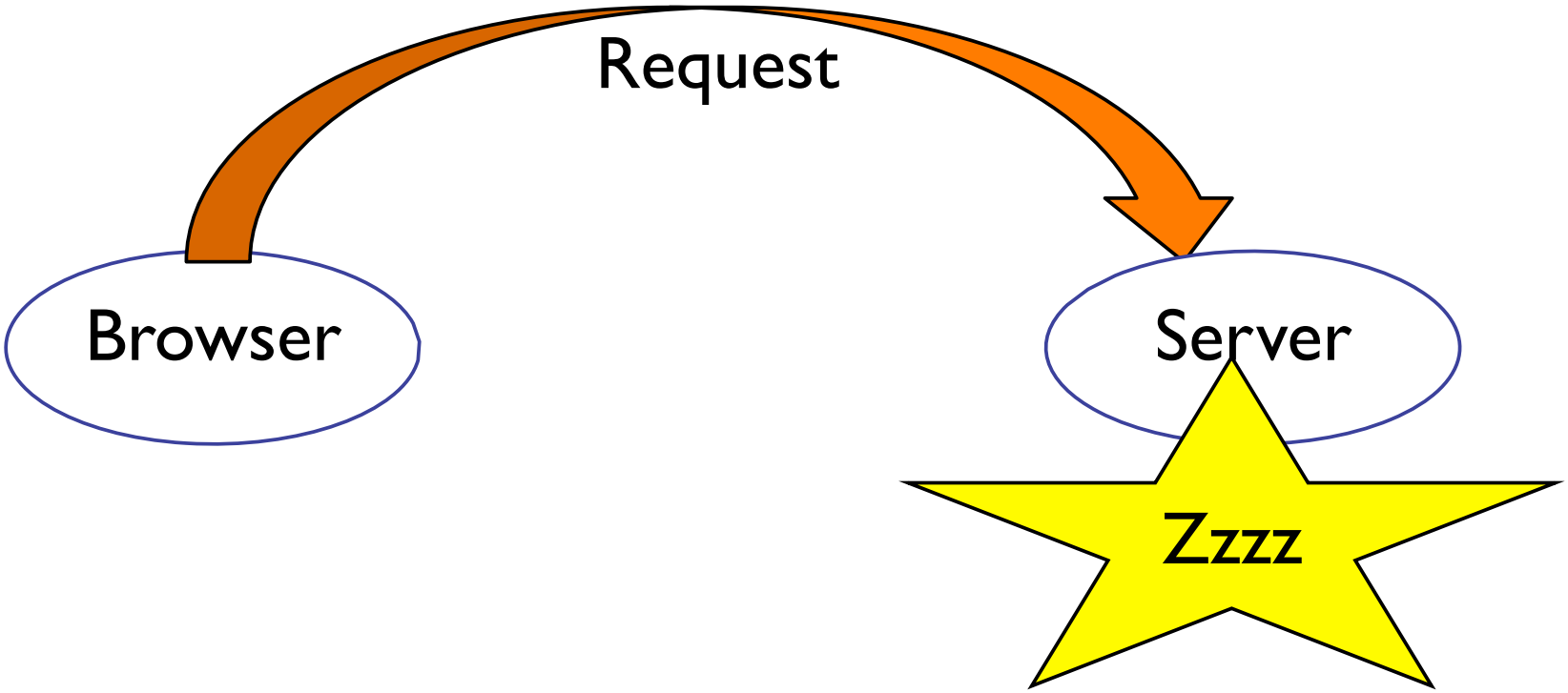


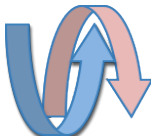
Oups!!



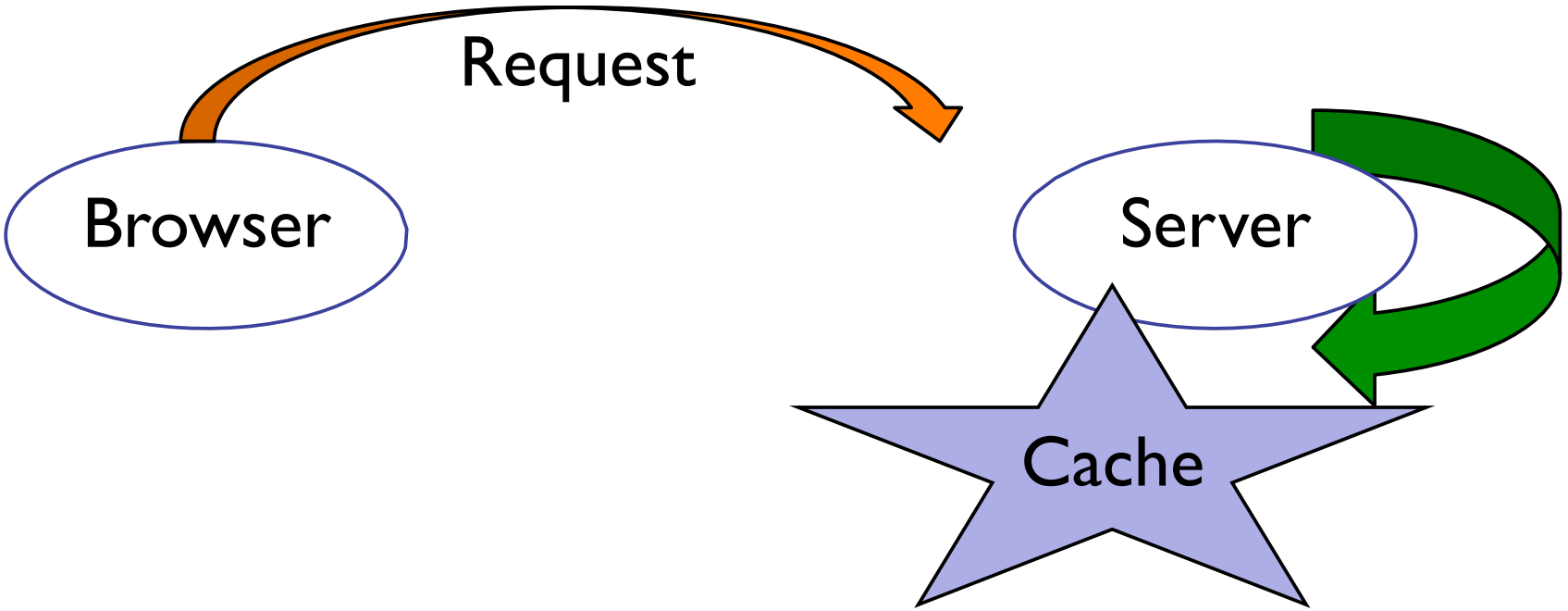


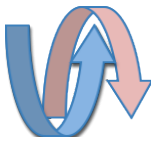
Oups!



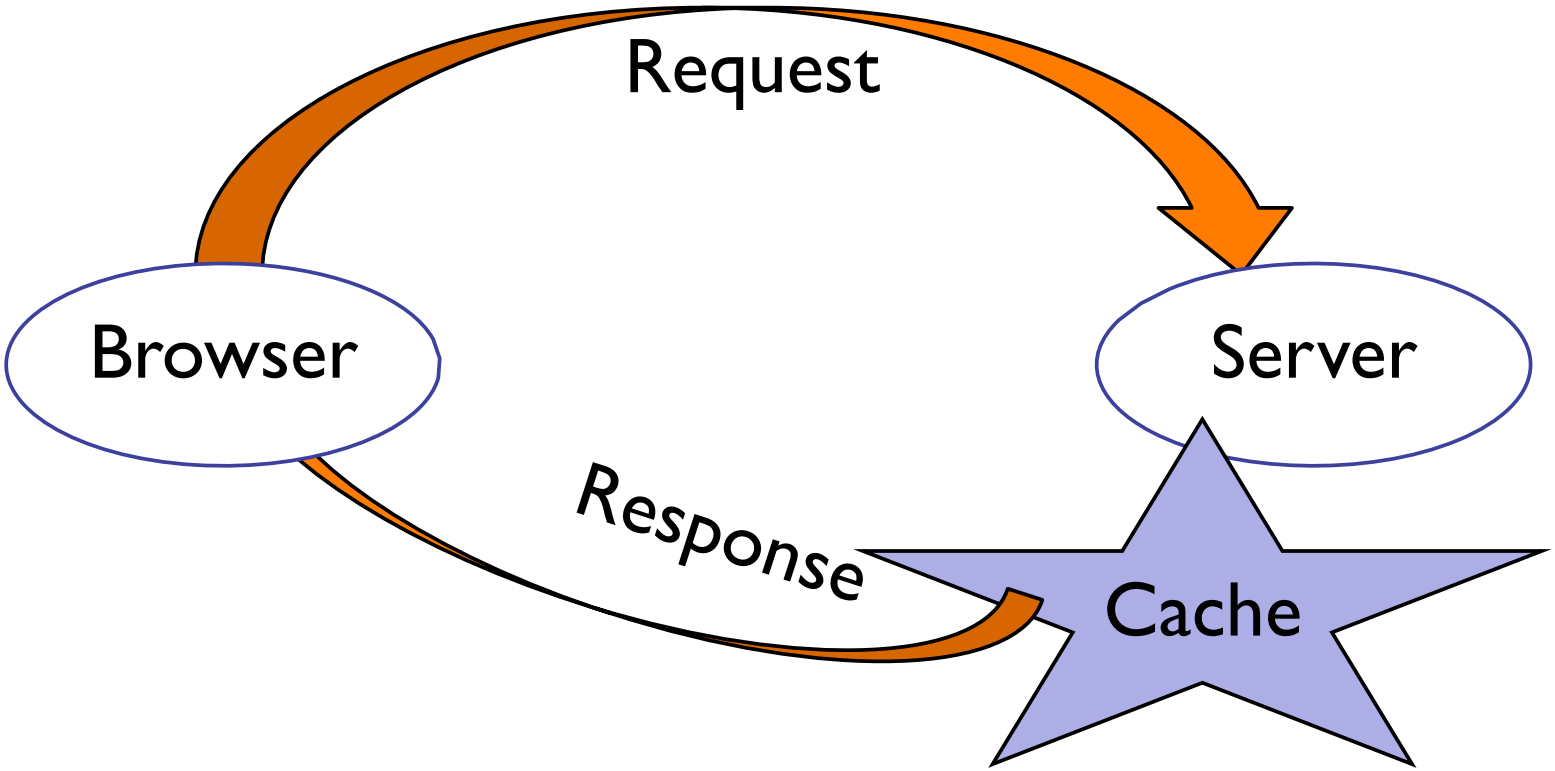


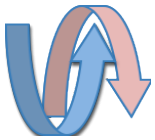
Better!



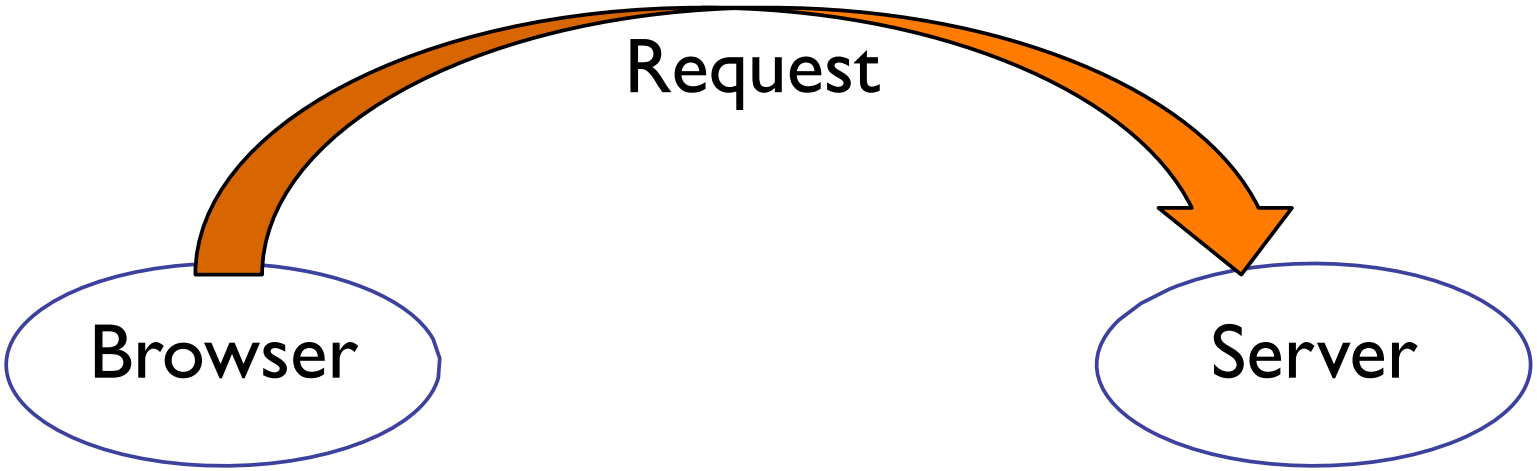


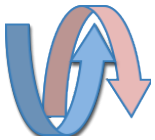
Better!



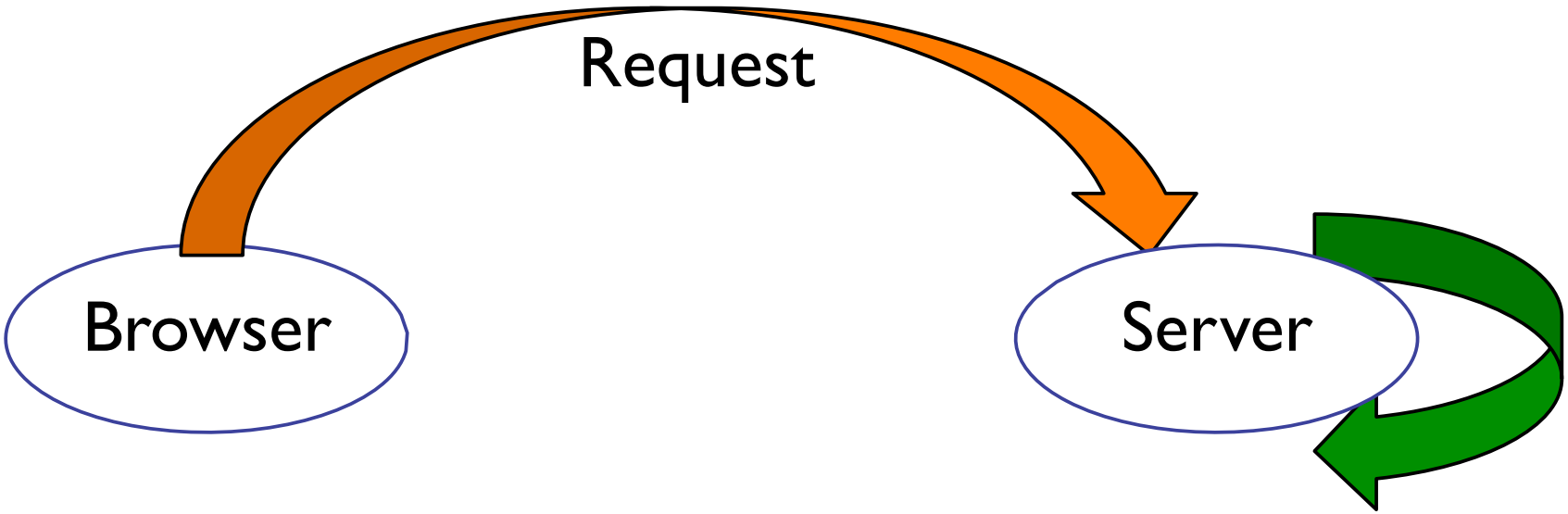


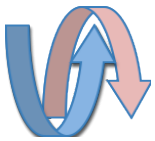
Pushing the limits (HTTP Streaming)



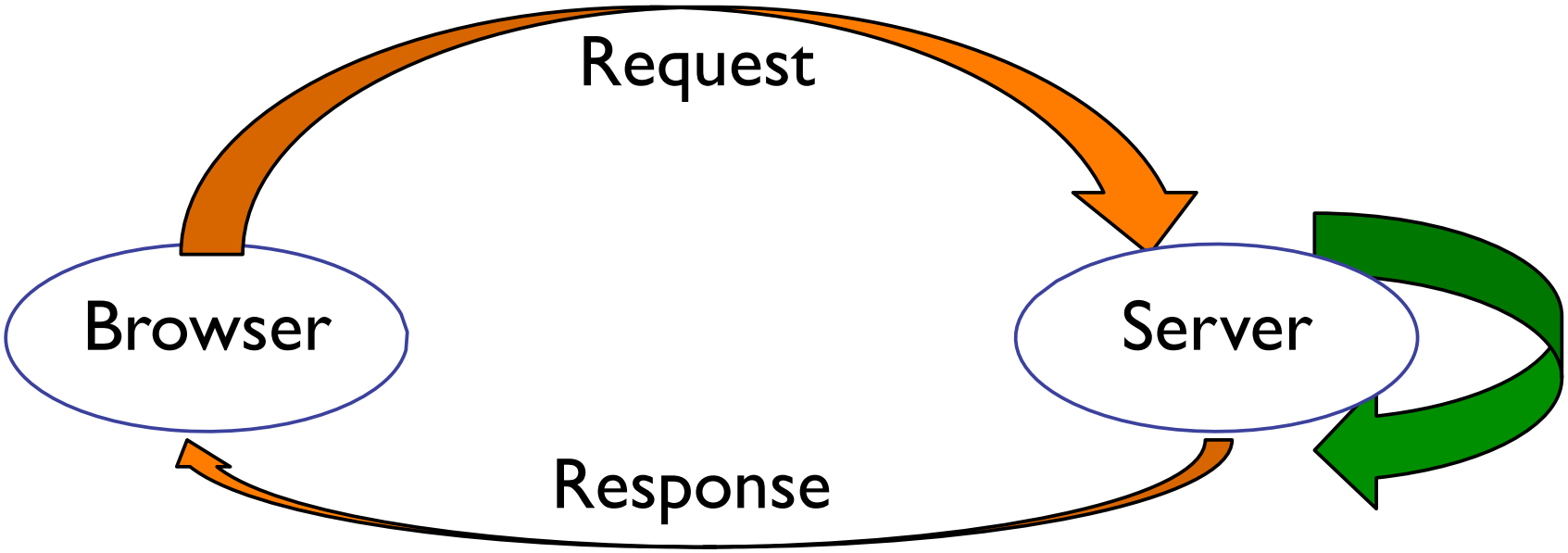


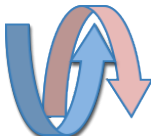
Pushing the limits (HTTP Streaming)



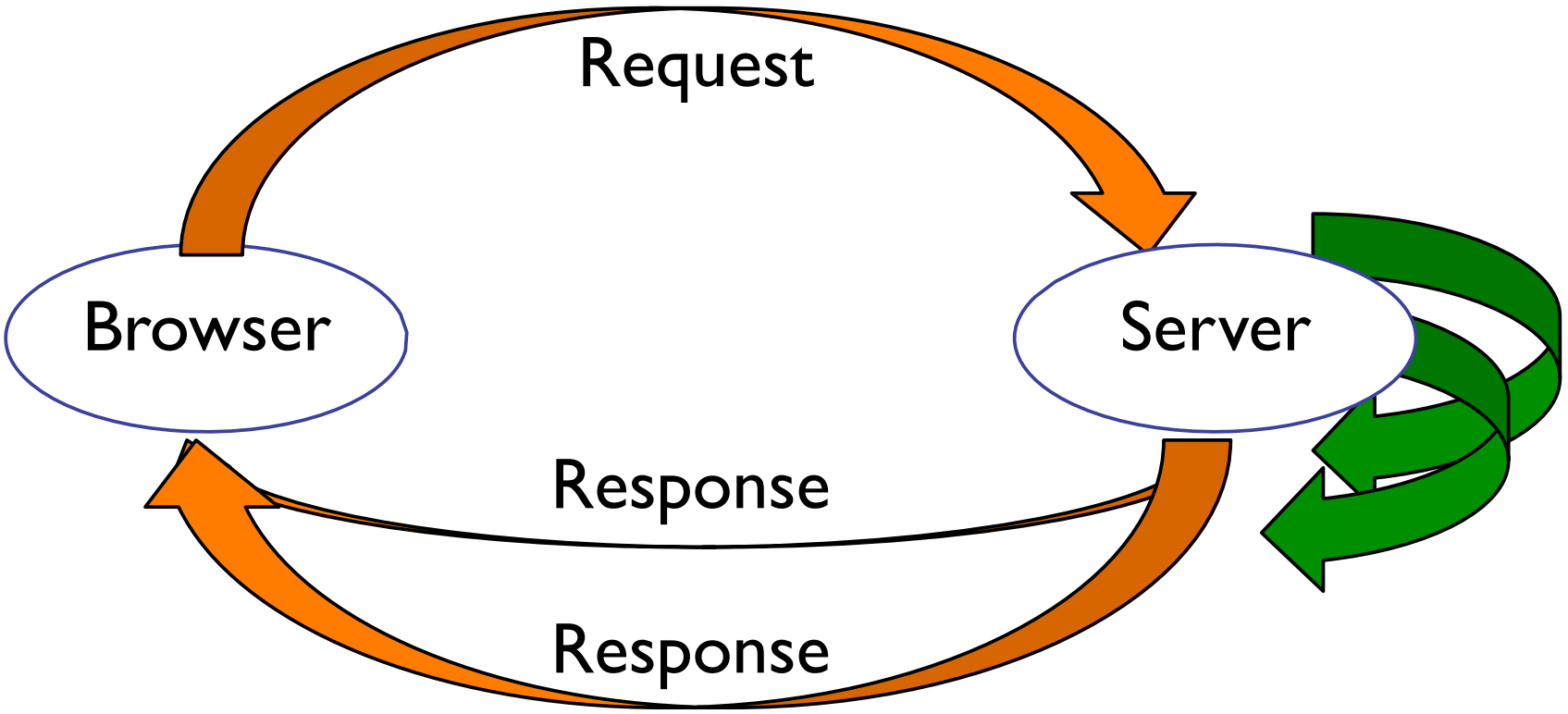


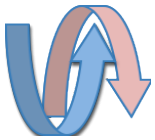
Pushing the limits (HTTP Streaming)



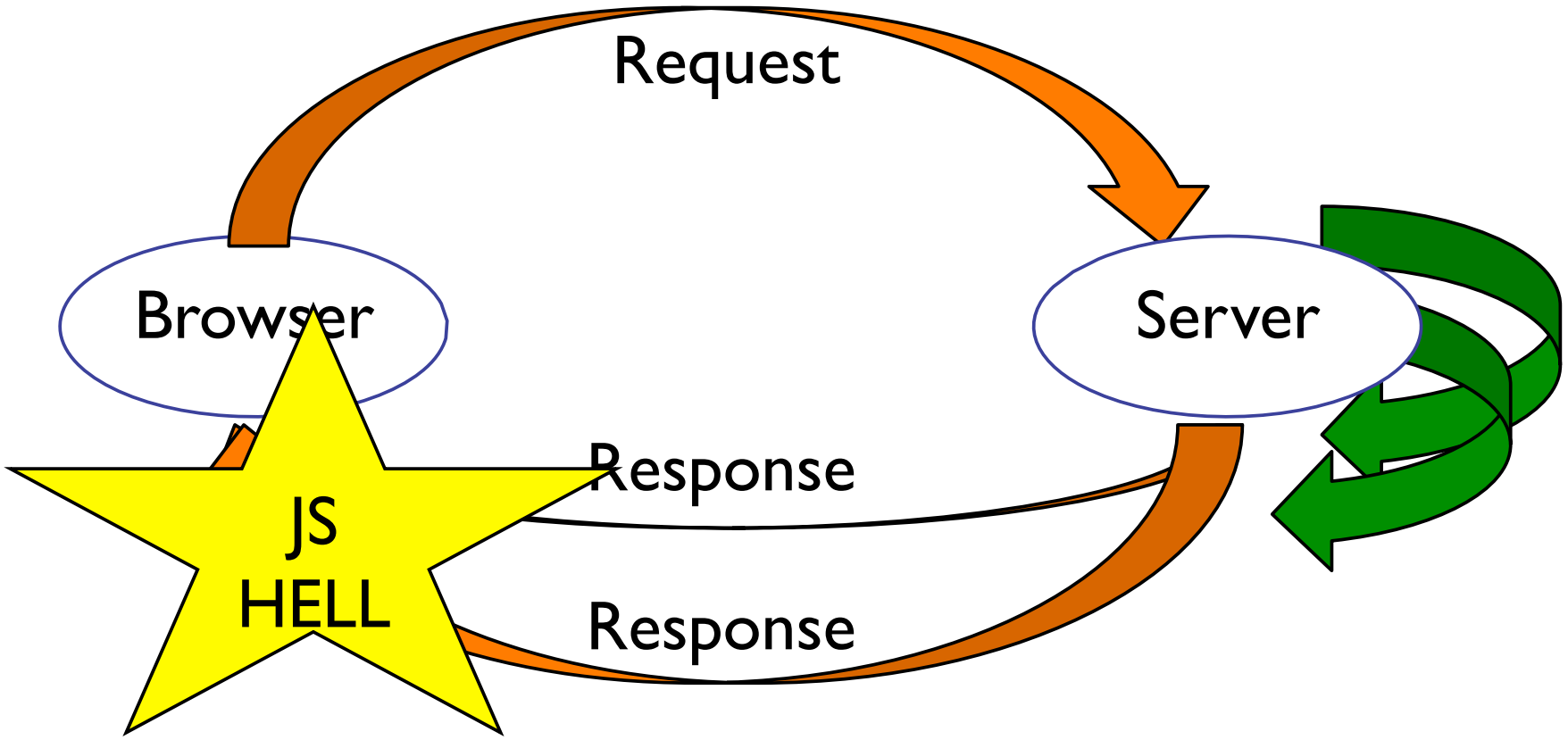


Pushing the limits (HTTP Streaming)

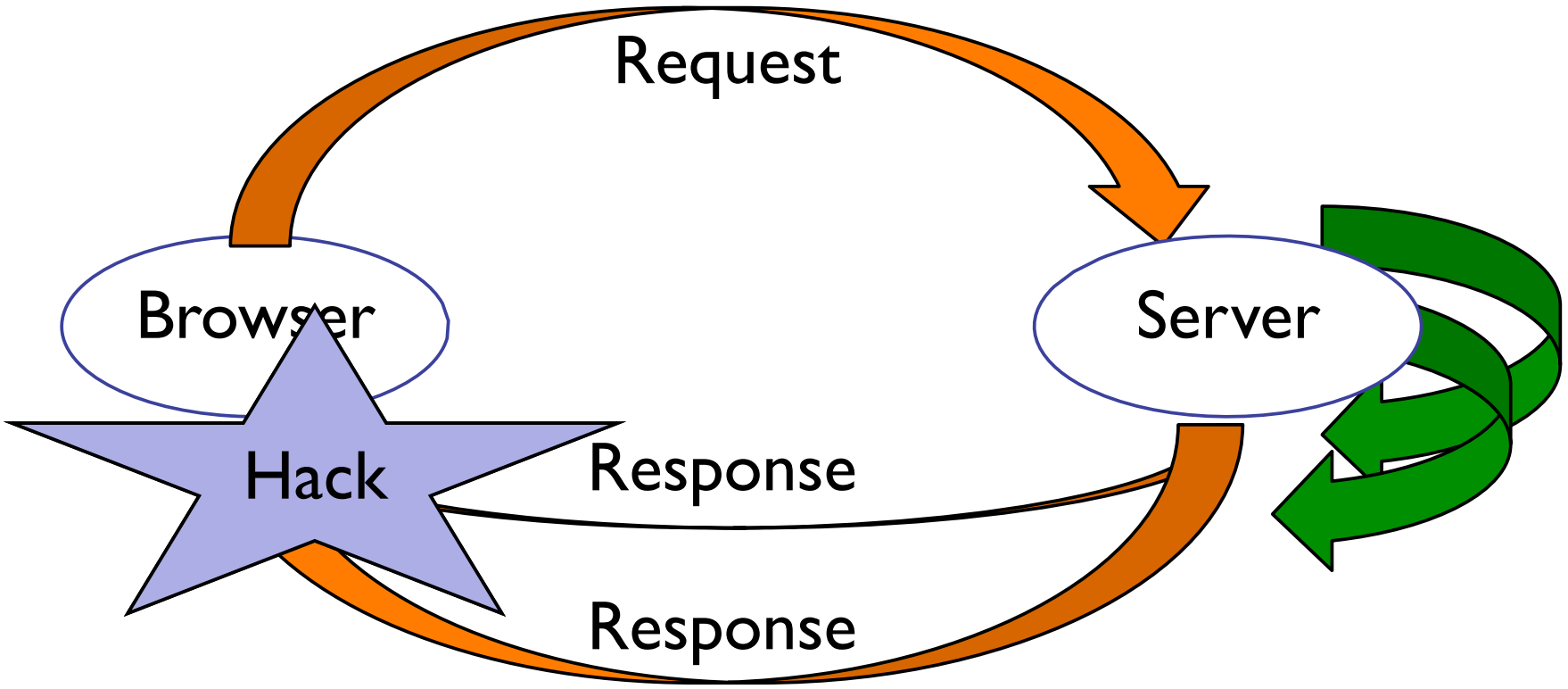


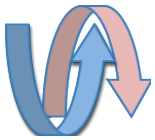


Oups!!

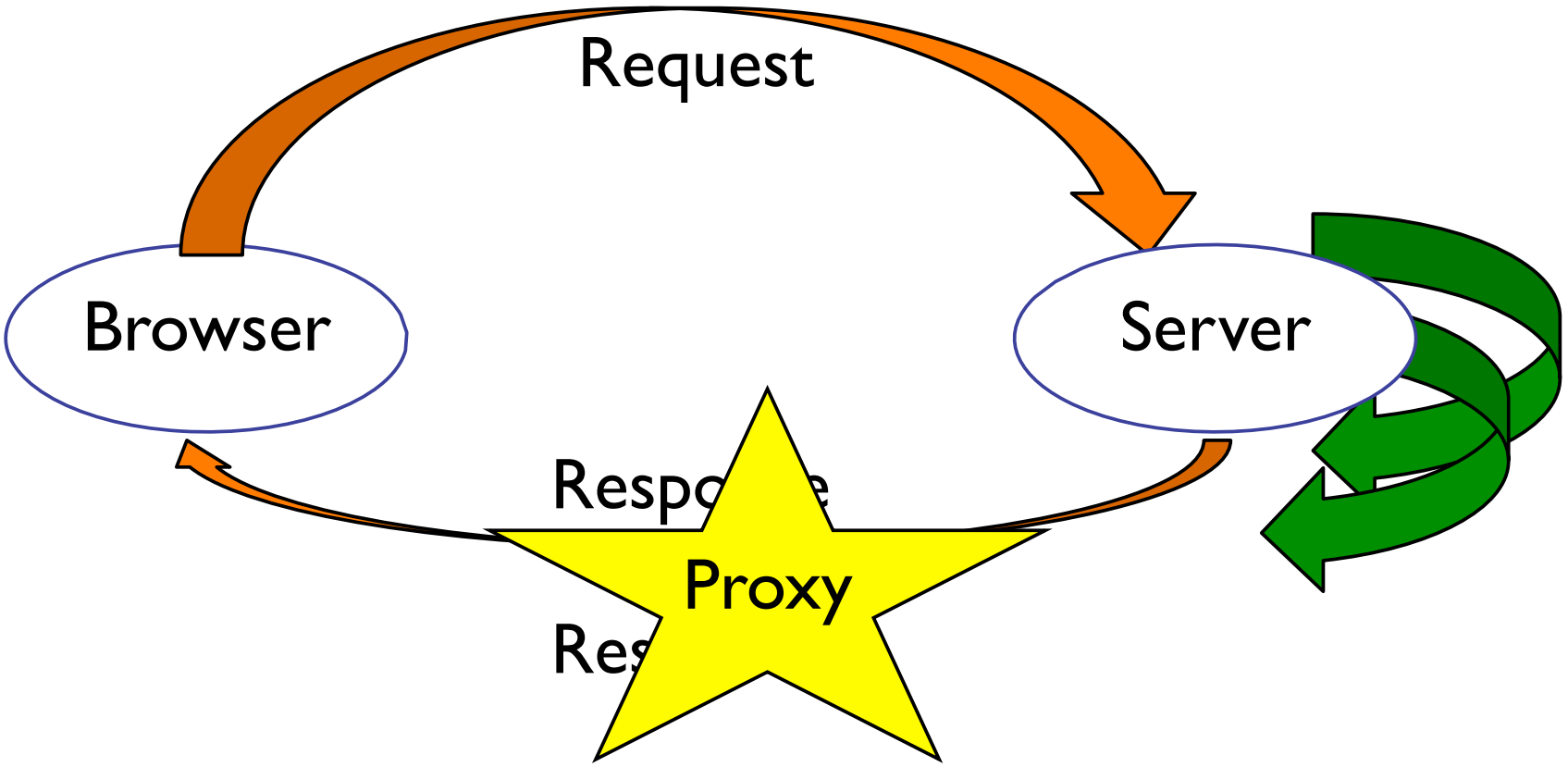


Pushing the limits (HTTP Streaming)

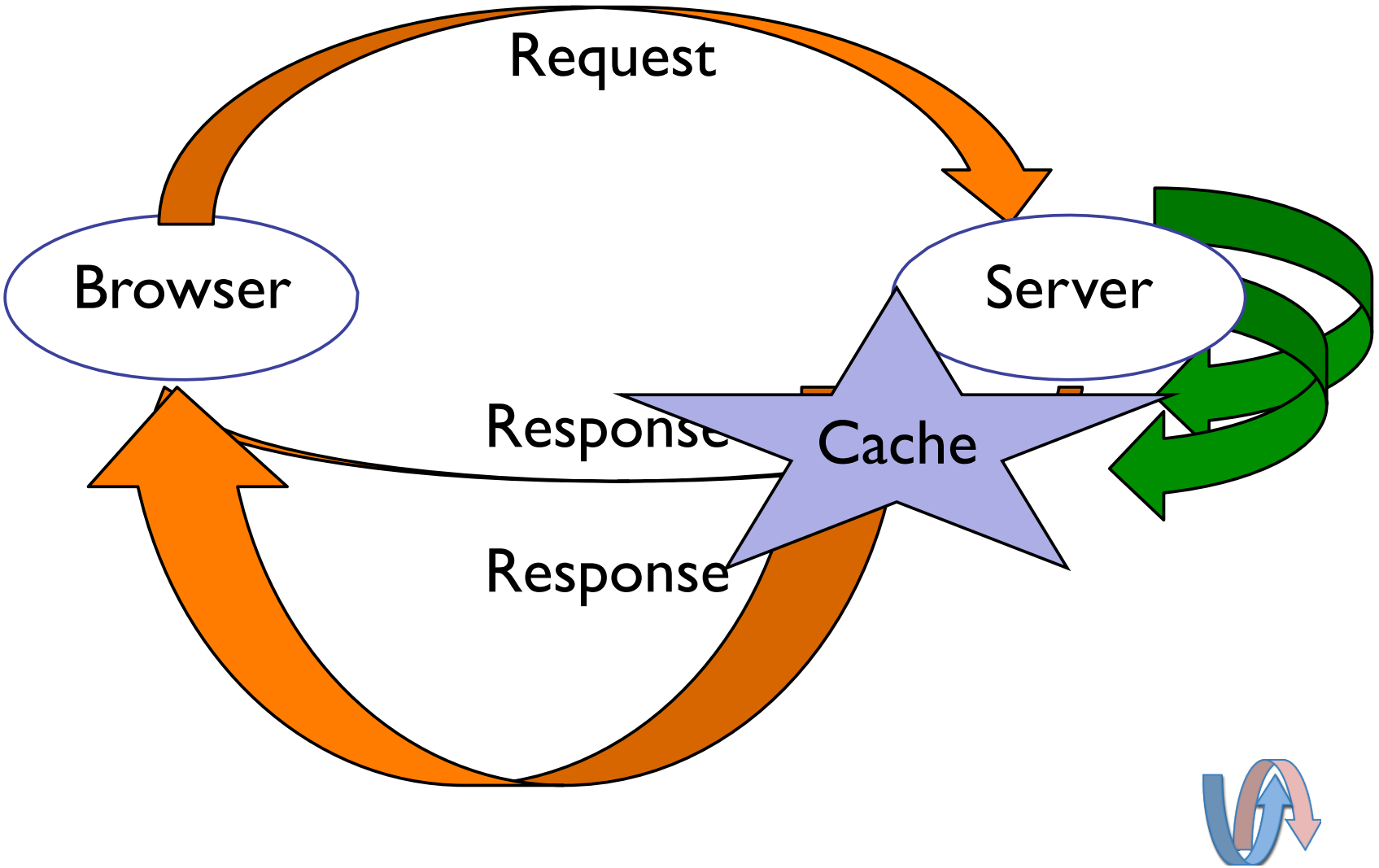




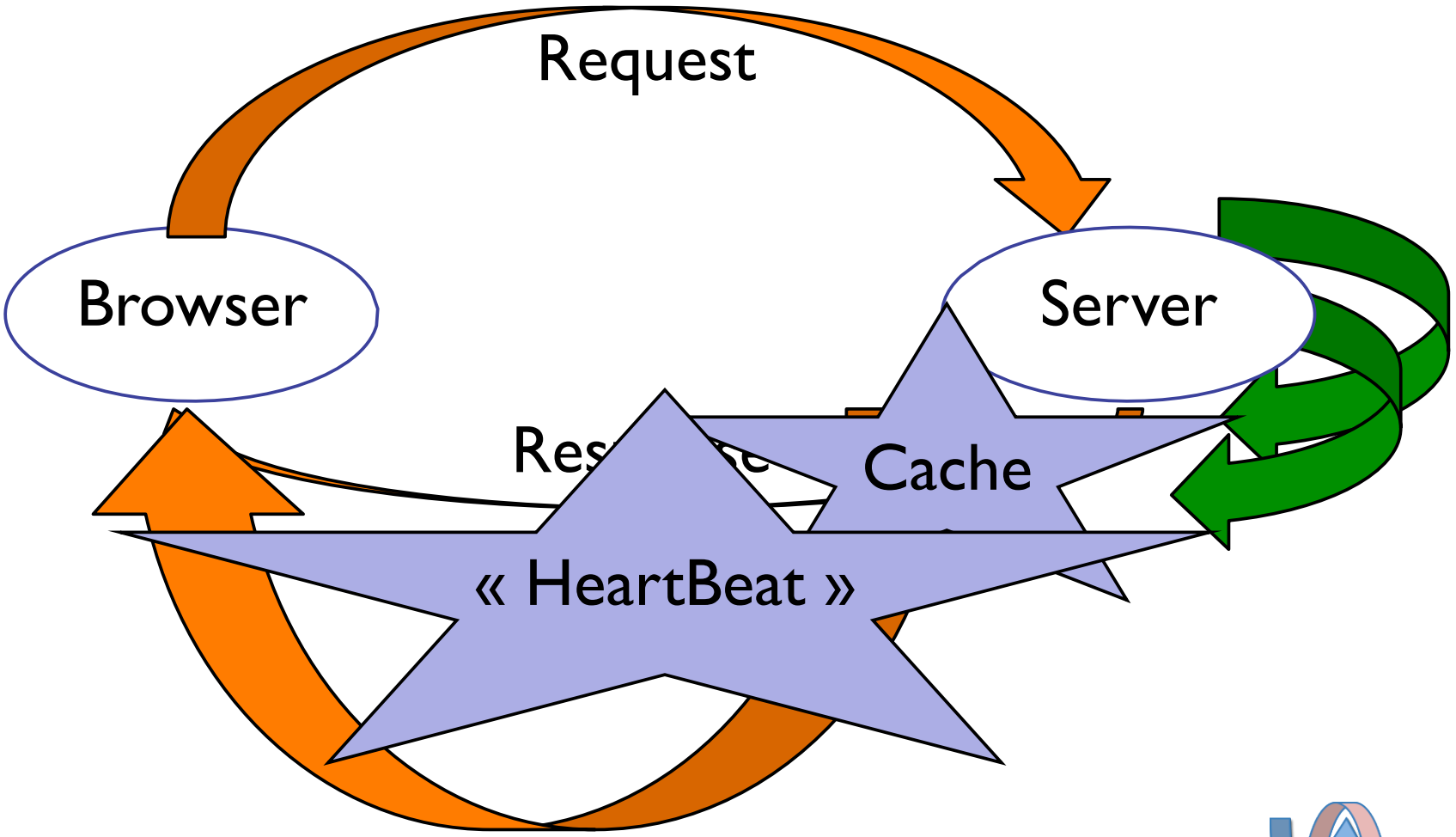
Oups!!!



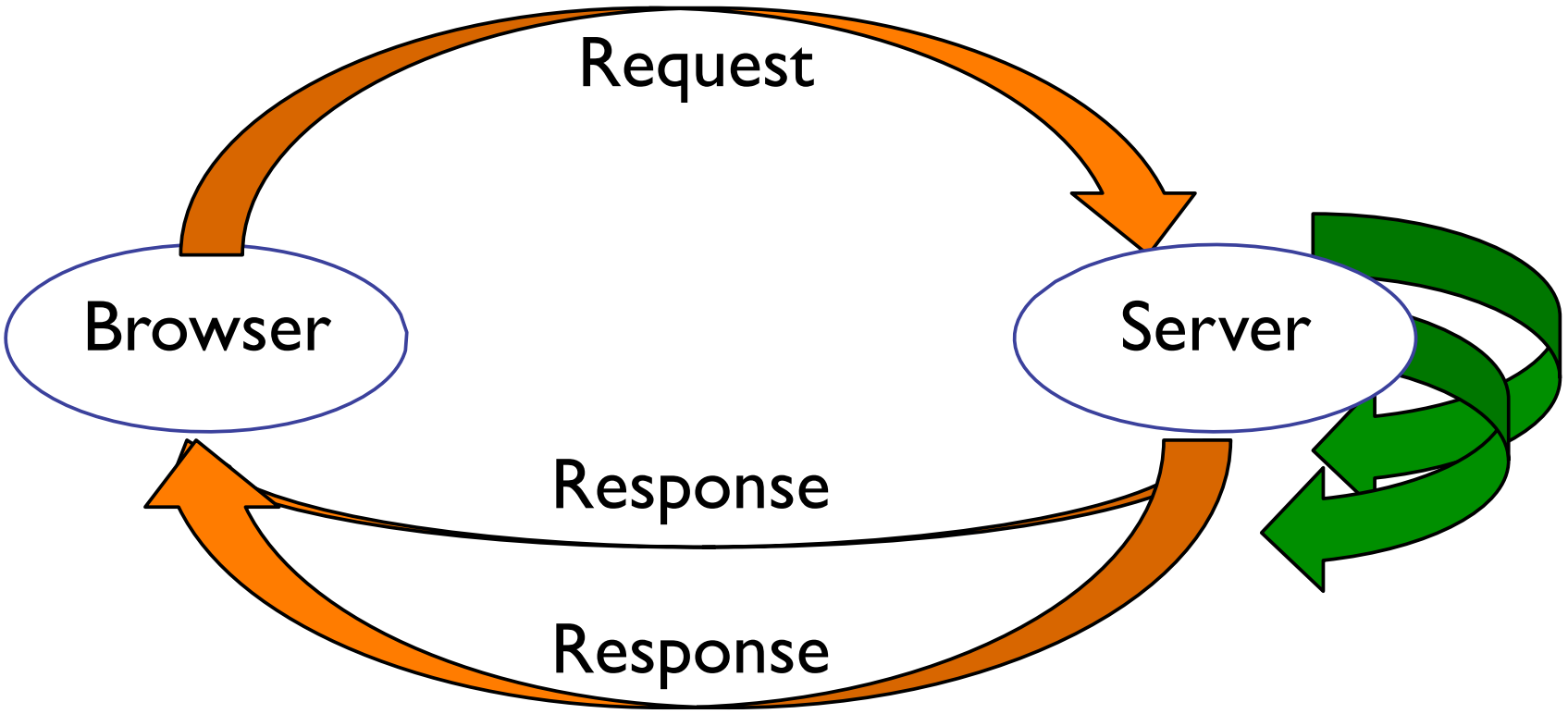
Better

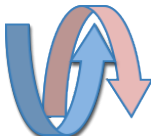


Better

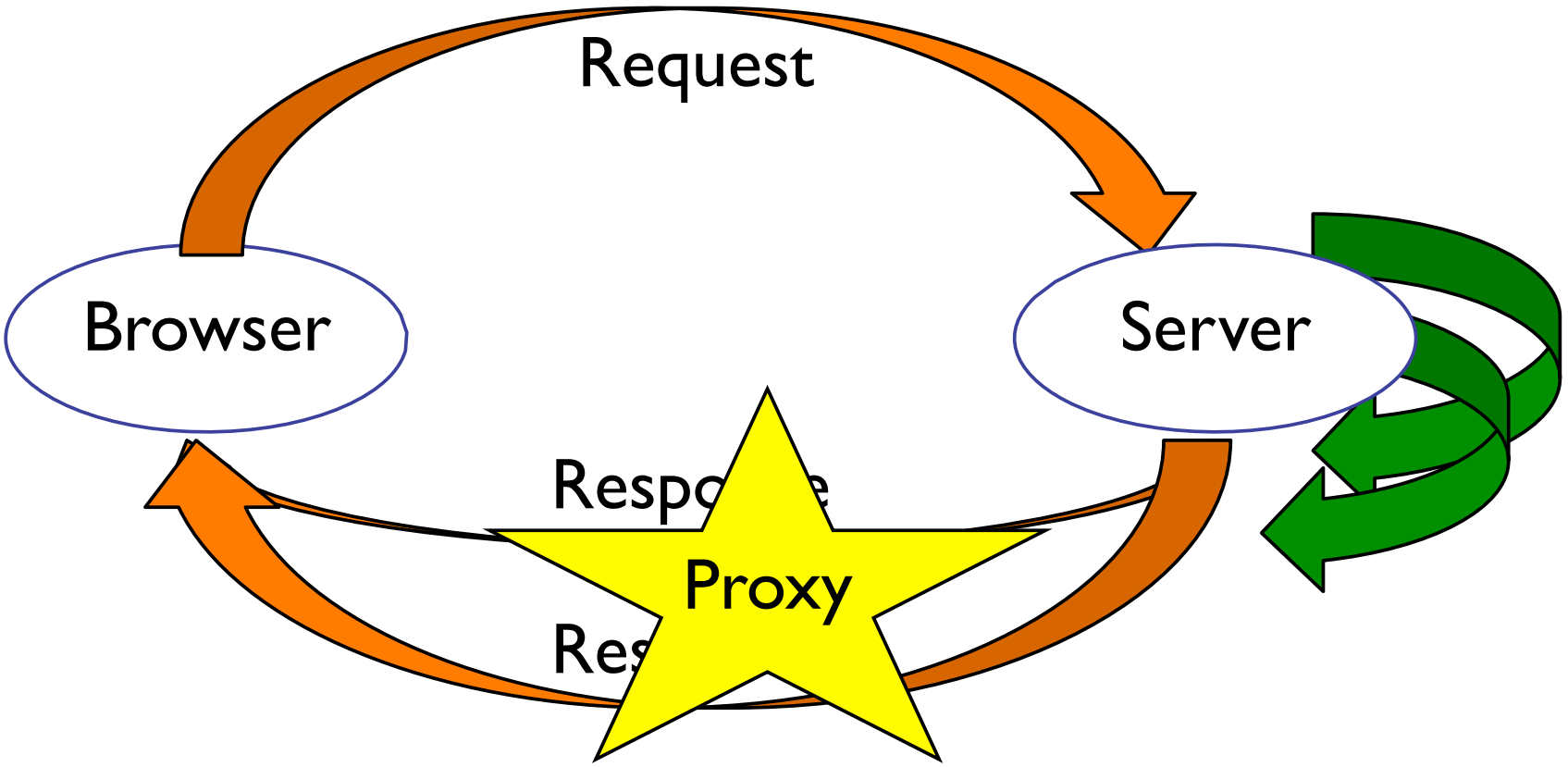


Better: Server Side Events (SSE)

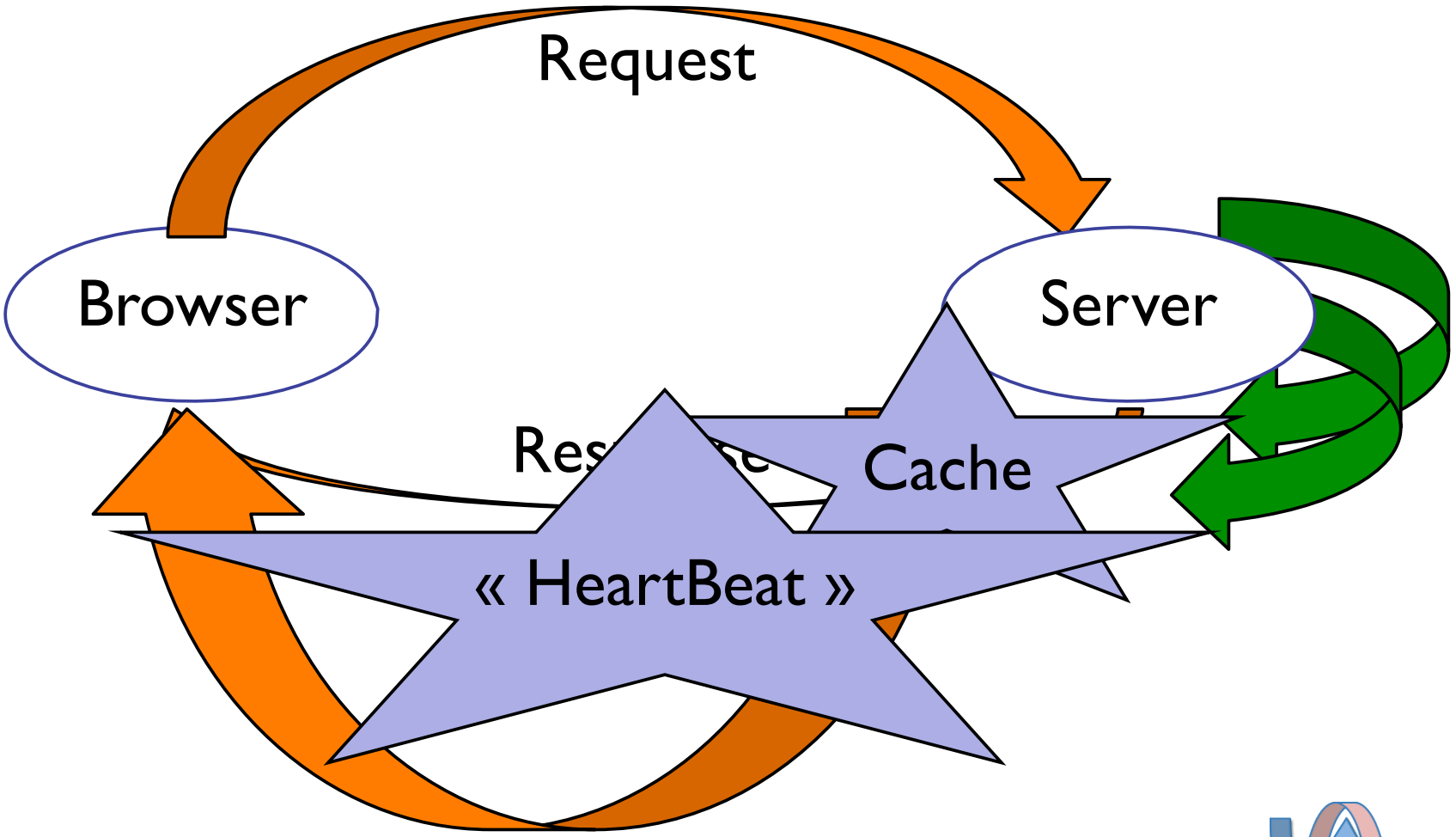




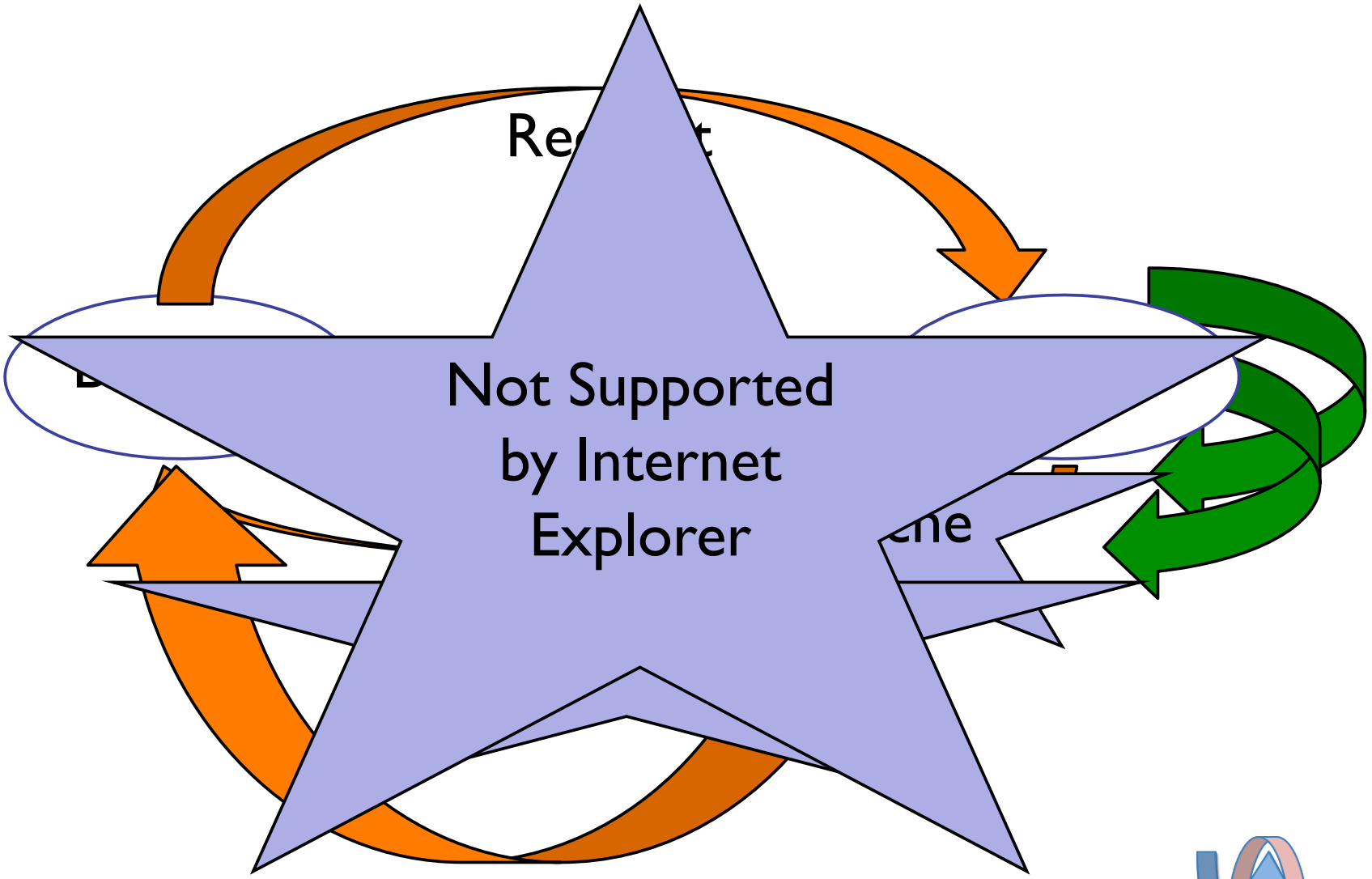
Error!



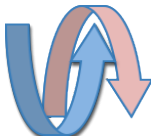
Better



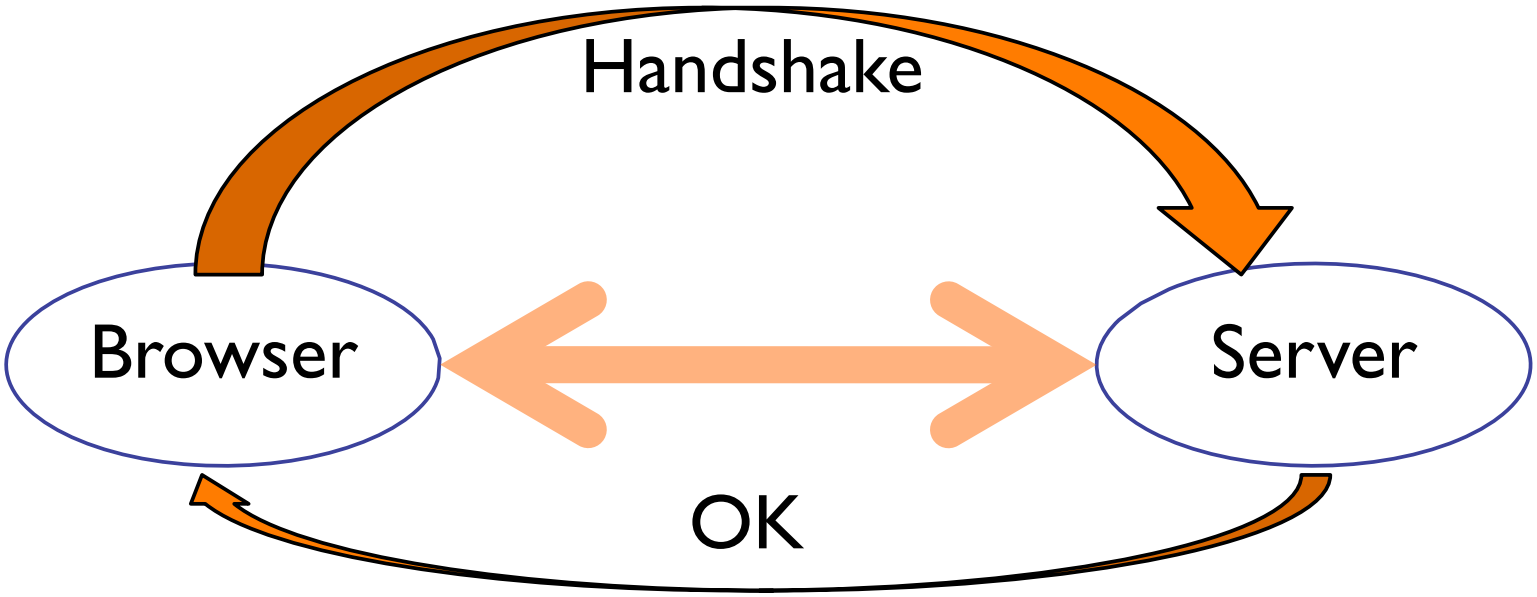
Better

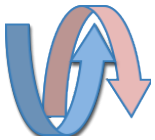


WebSockets

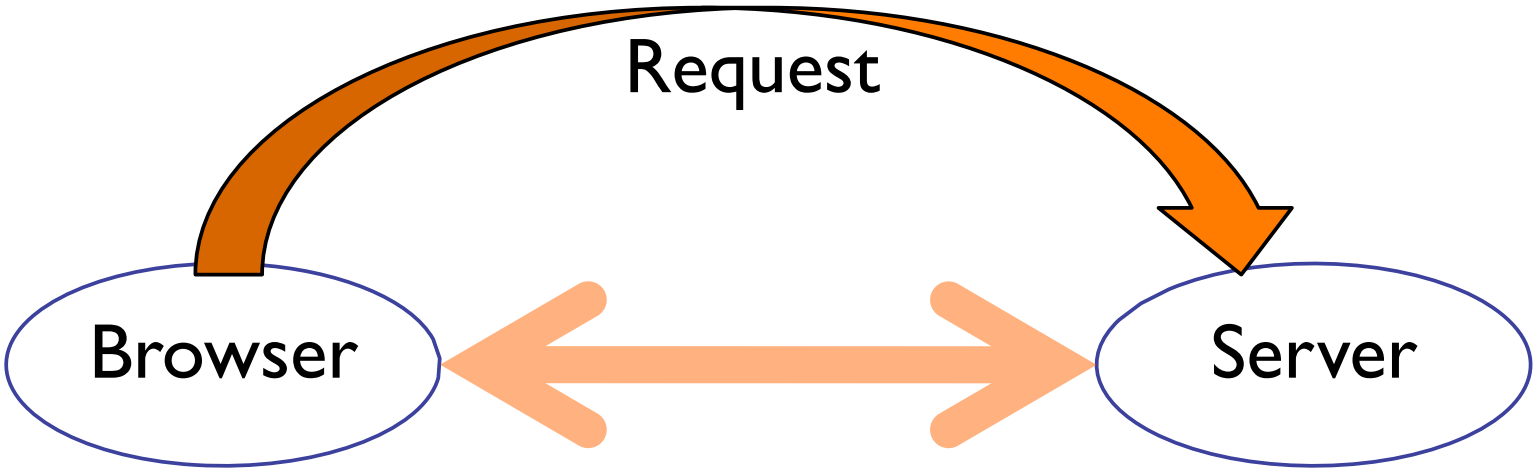


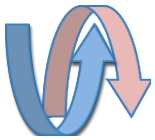
WebSockets



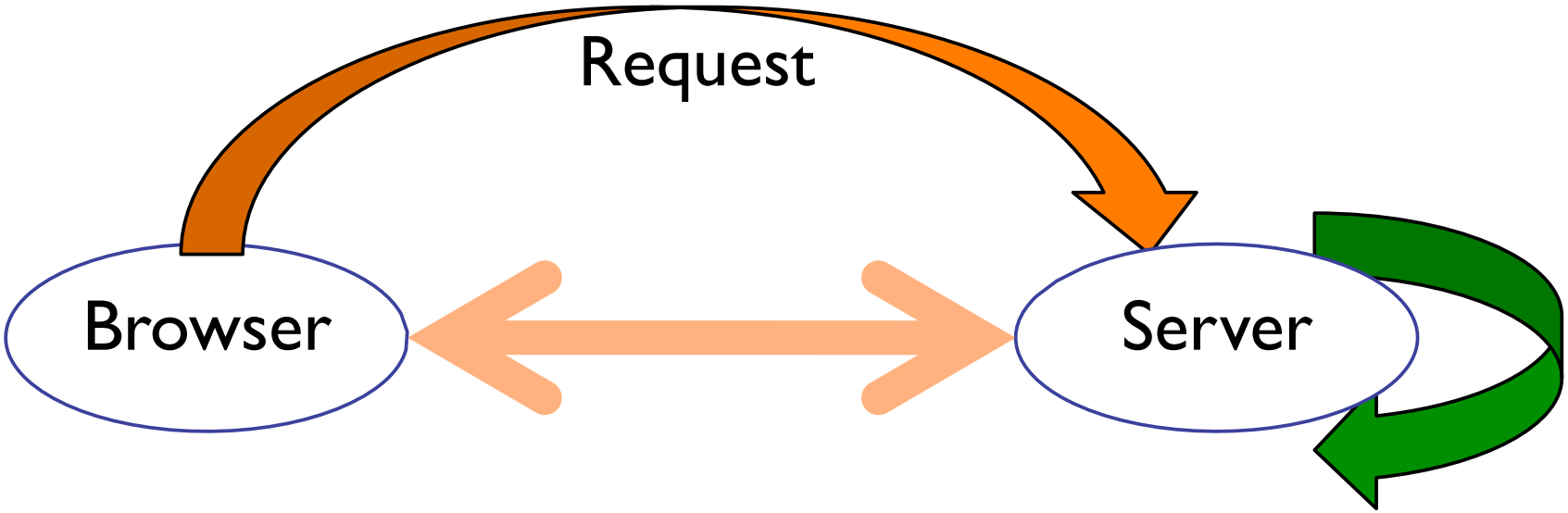


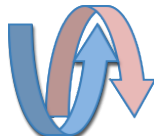
WebSockets



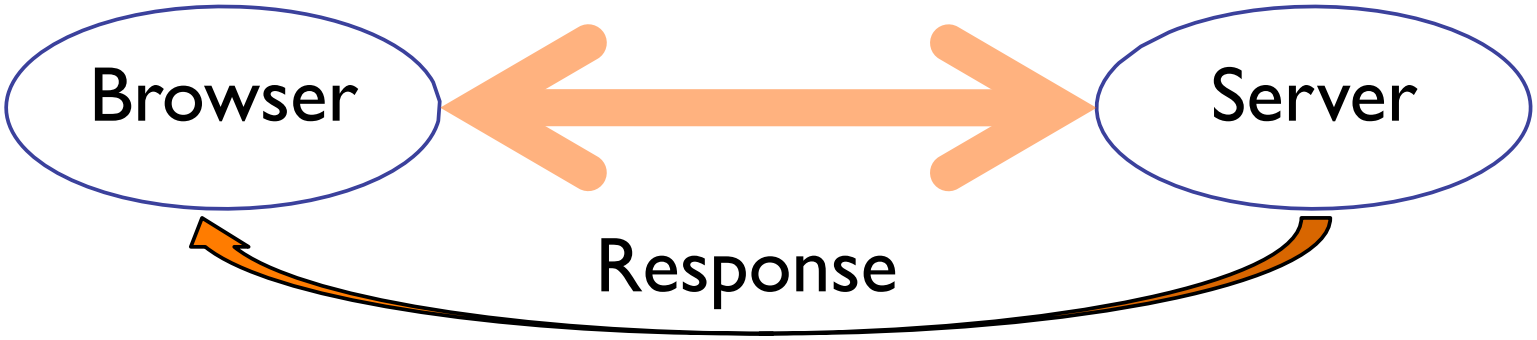


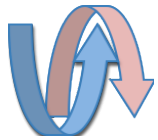
WebSockets



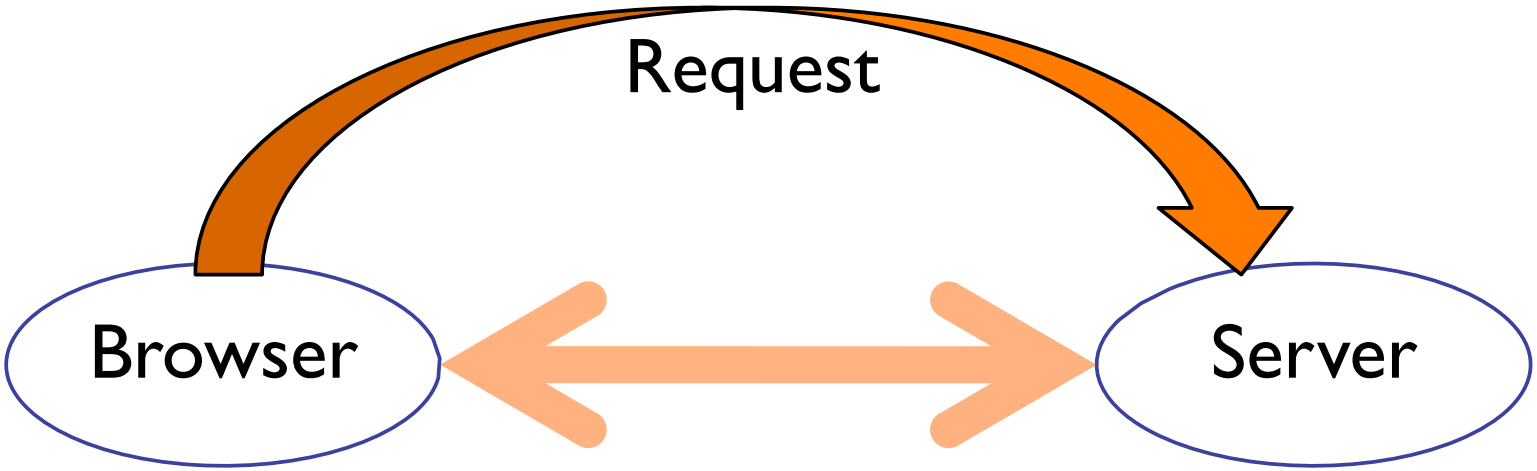


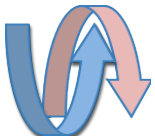
WebSockets



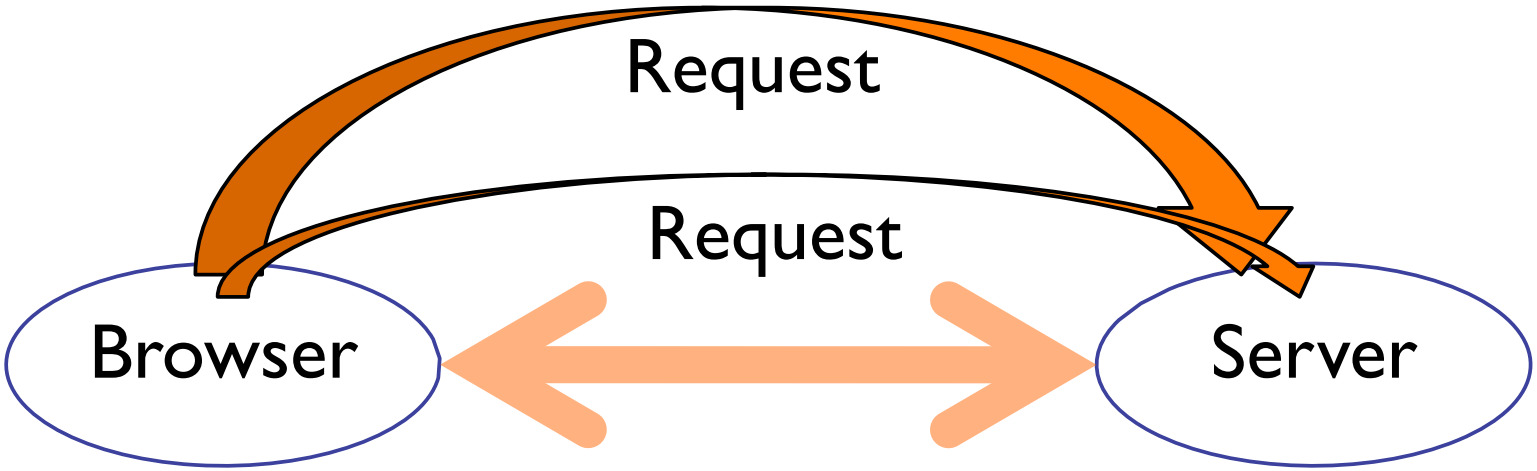


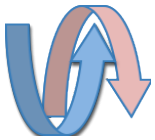
WebSockets



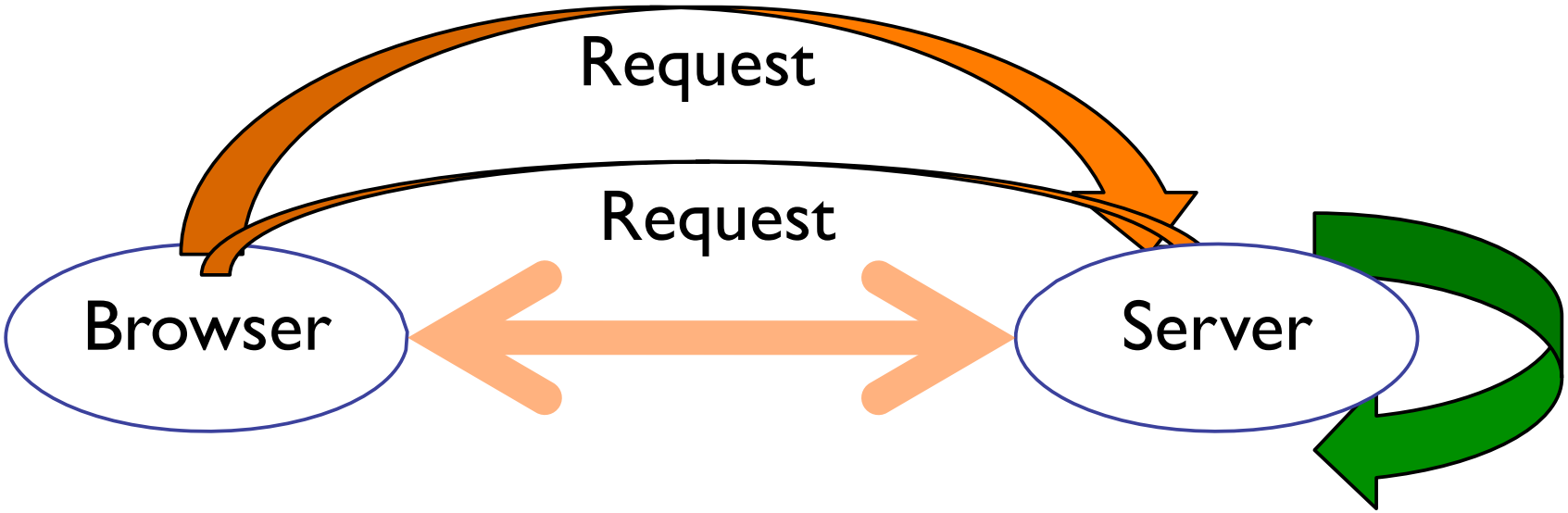


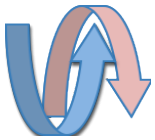
WebSockets



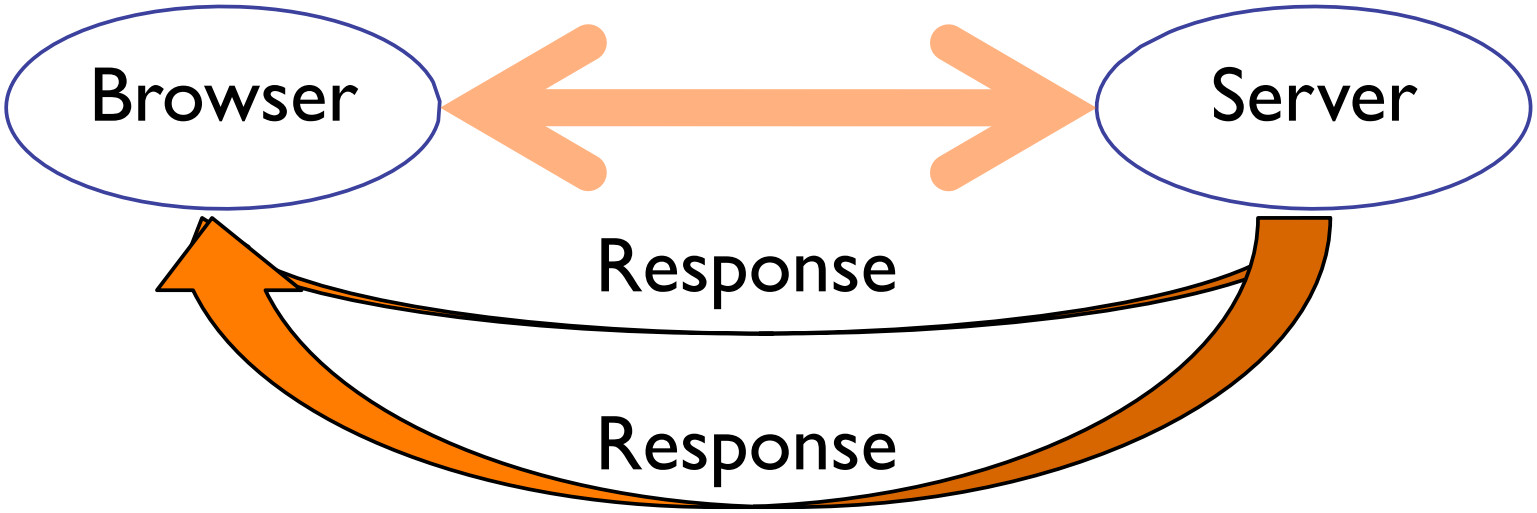


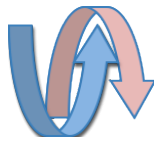
WebSockets



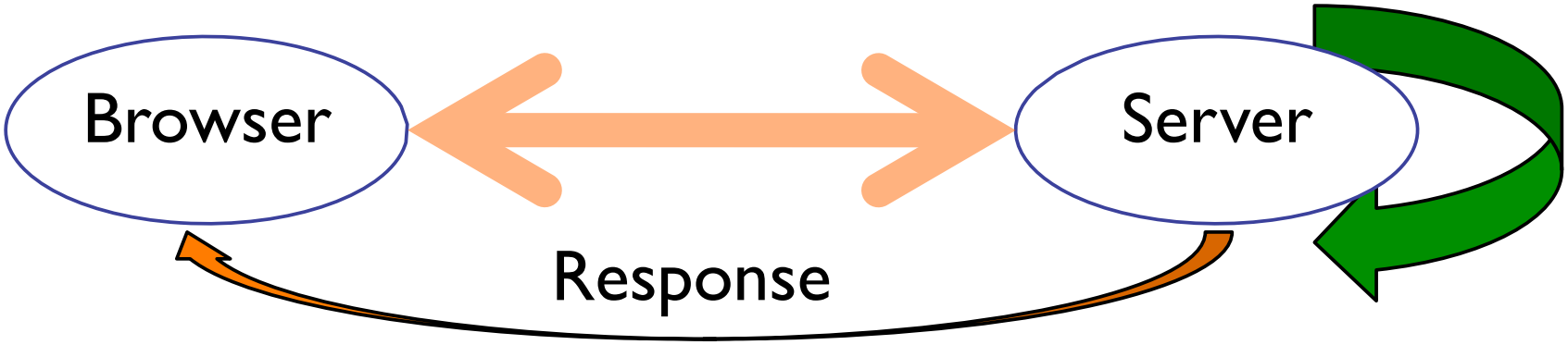


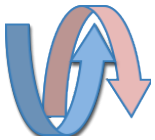
WebSockets



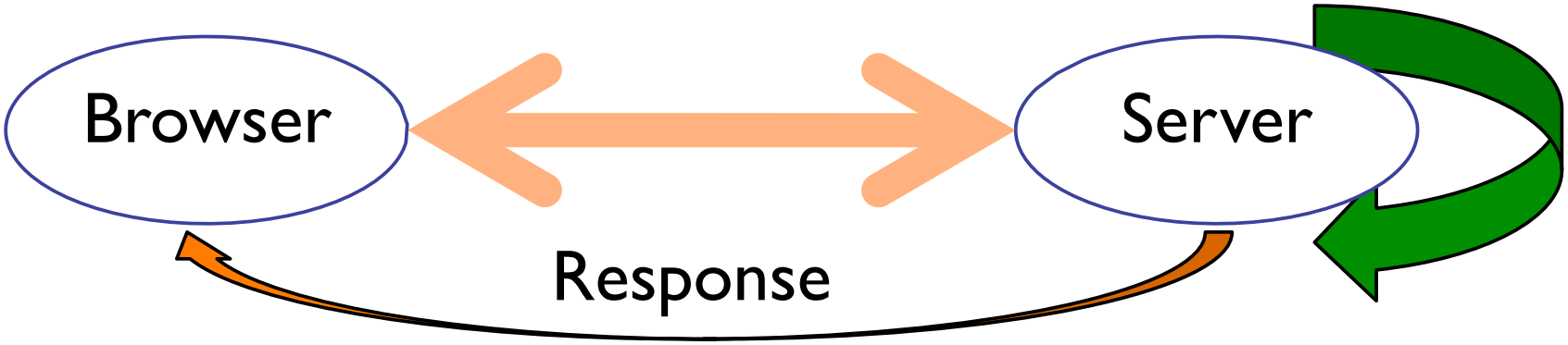


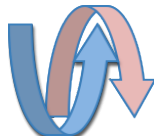
Anytime





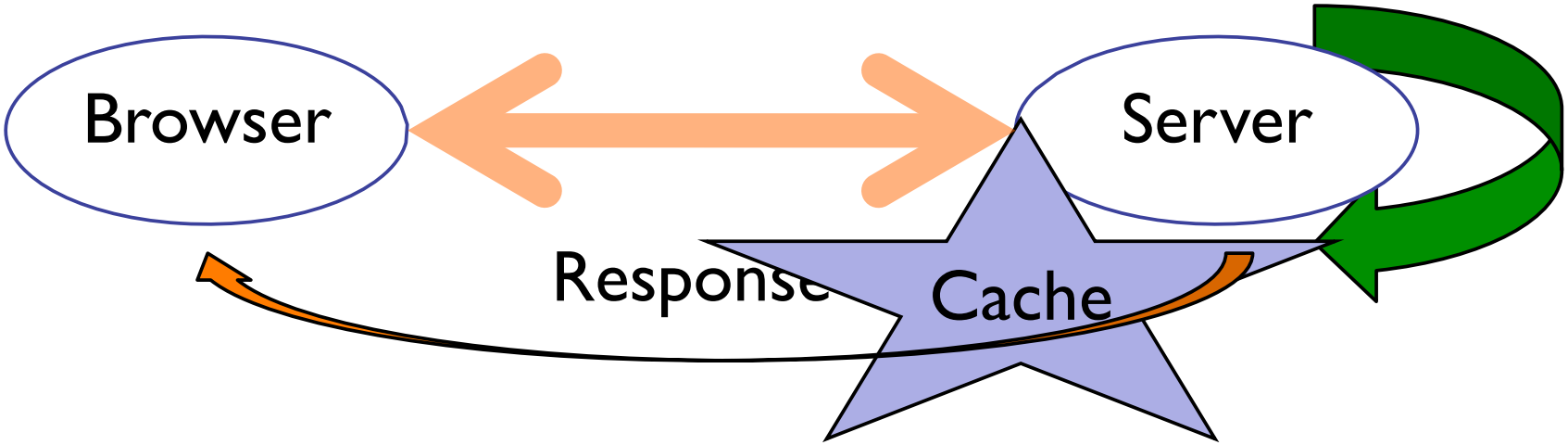
Everything is good

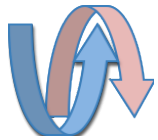




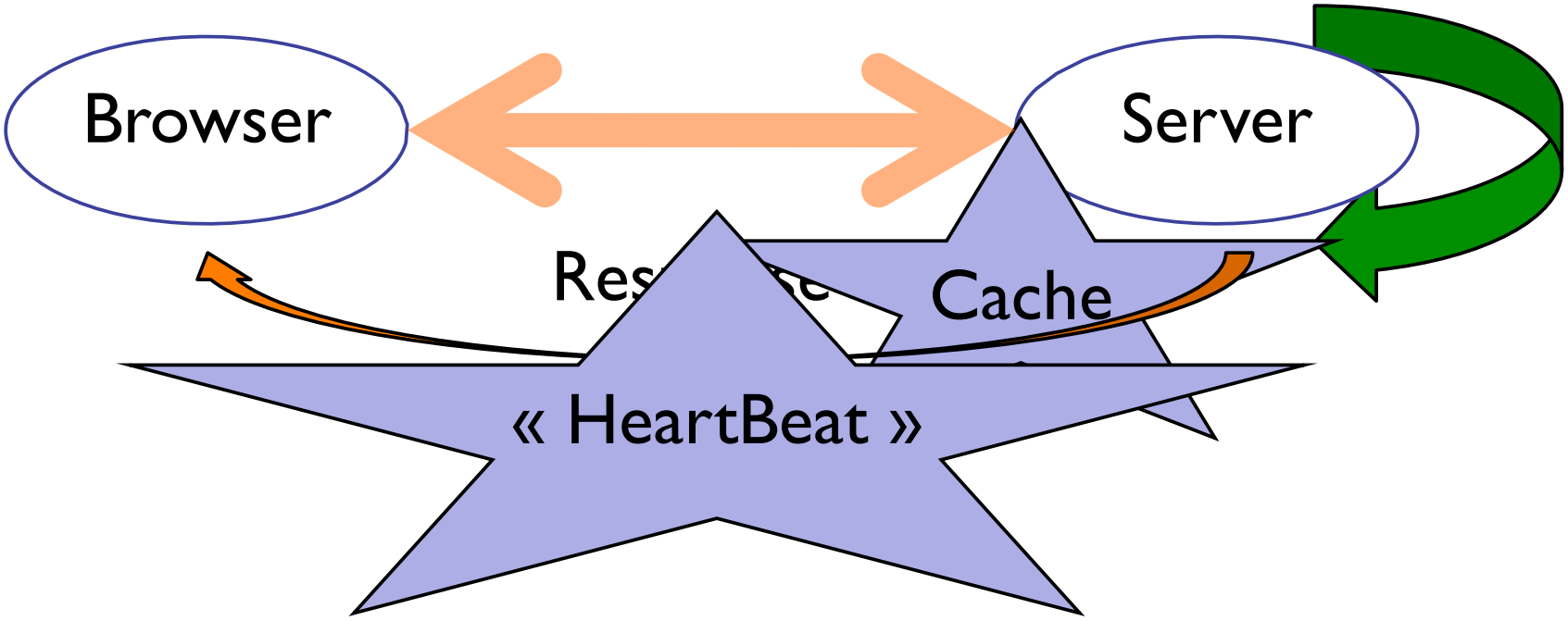
Problems

Better!

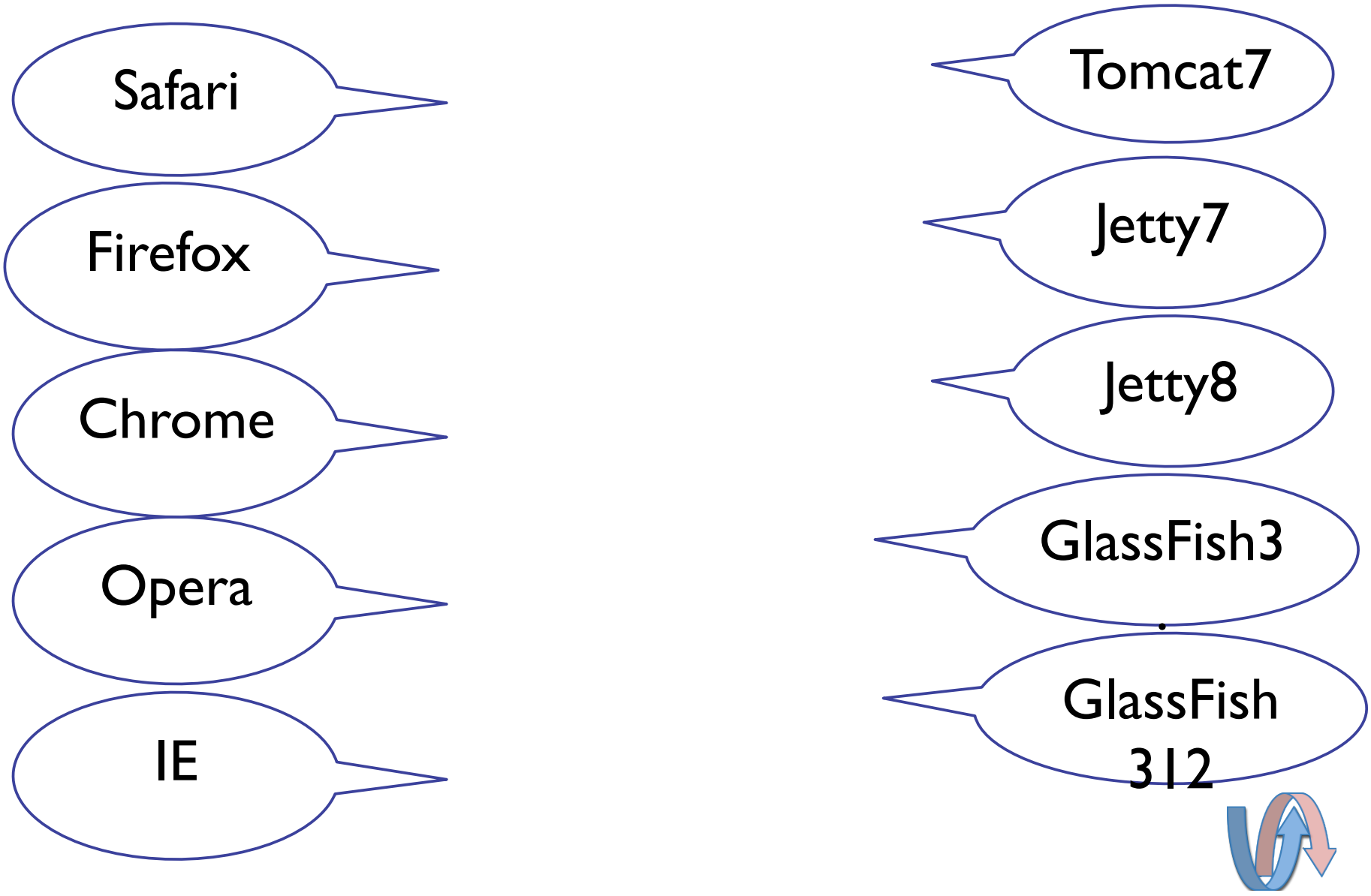




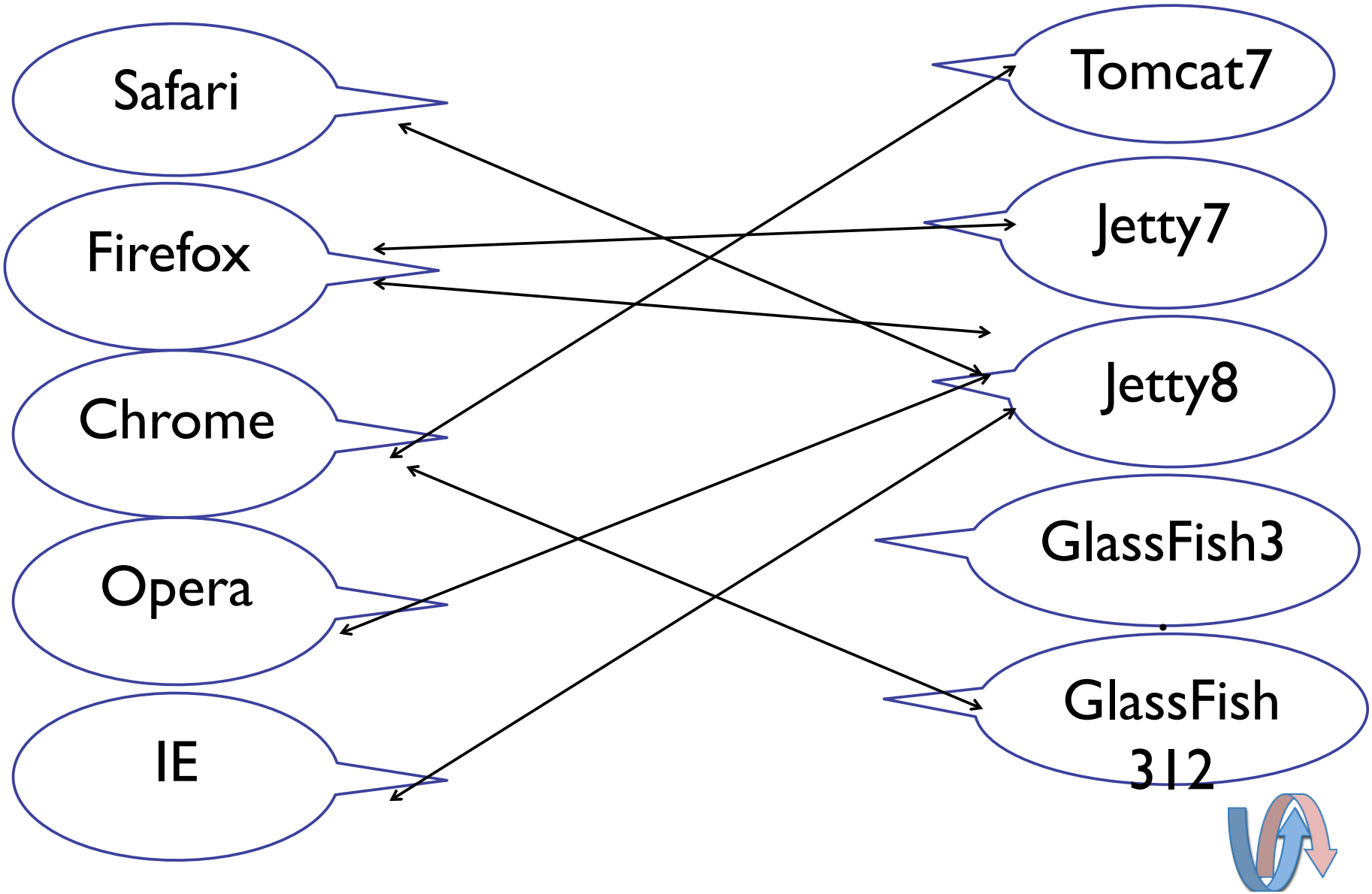
Better!

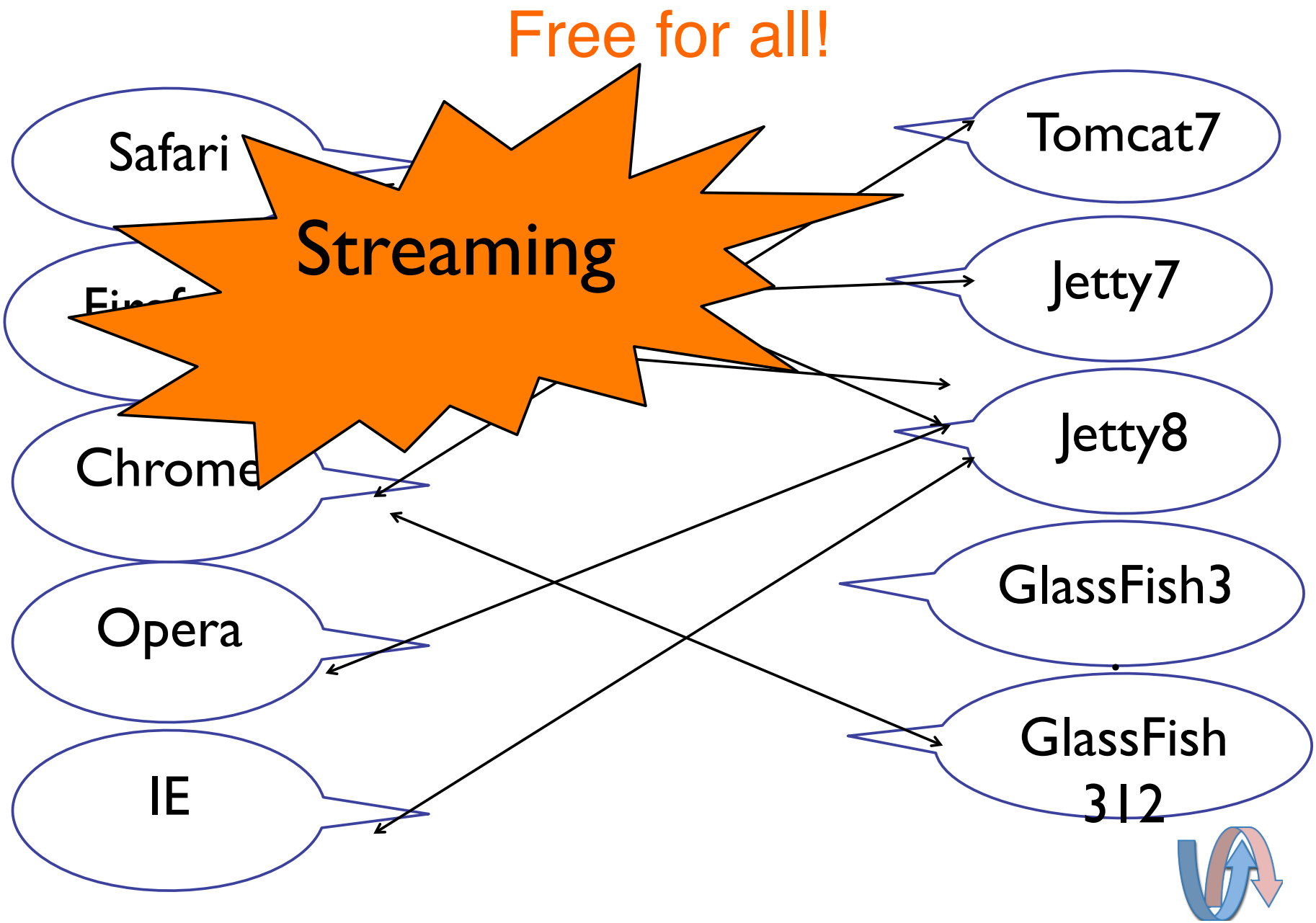


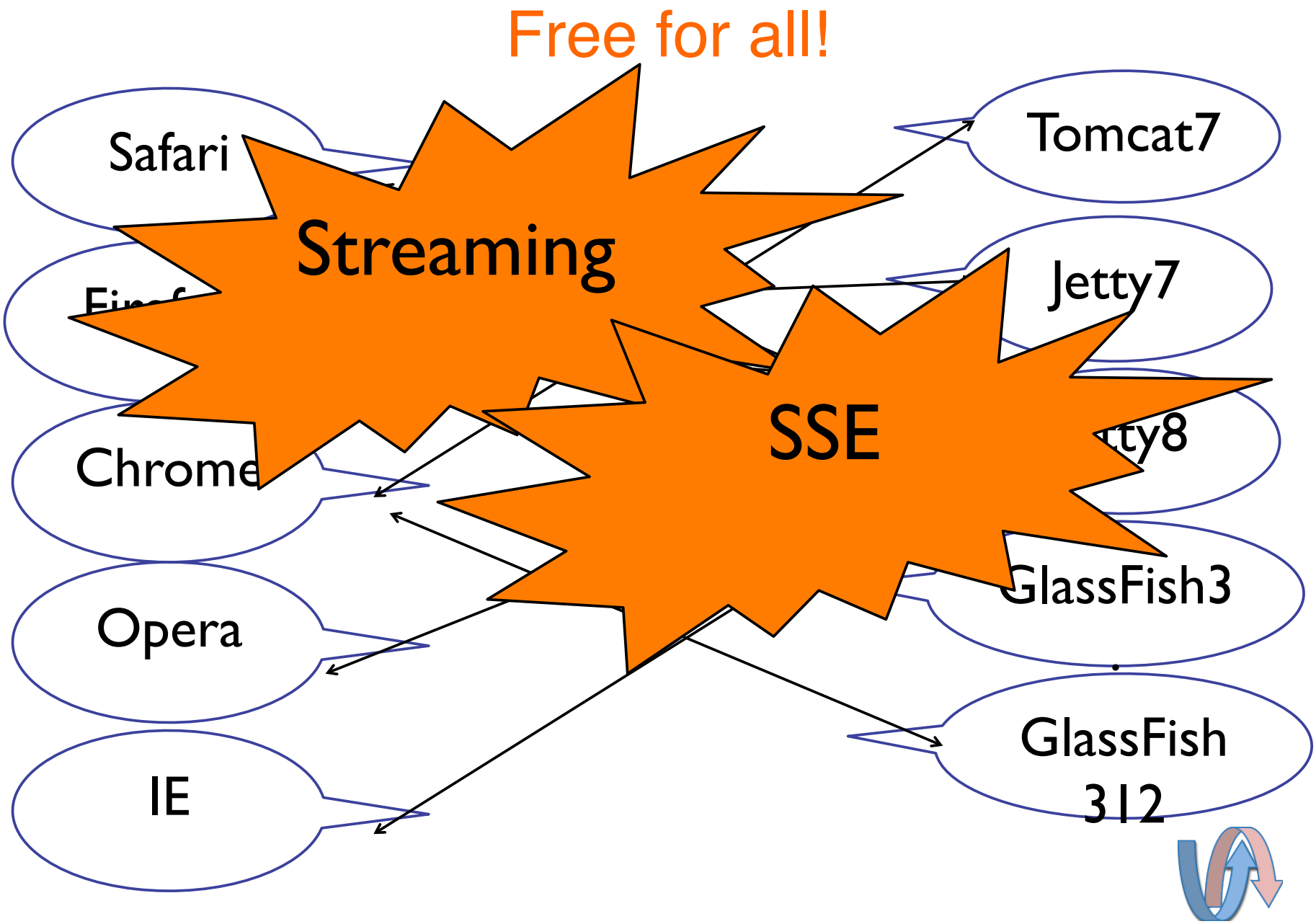
Browser versus Server

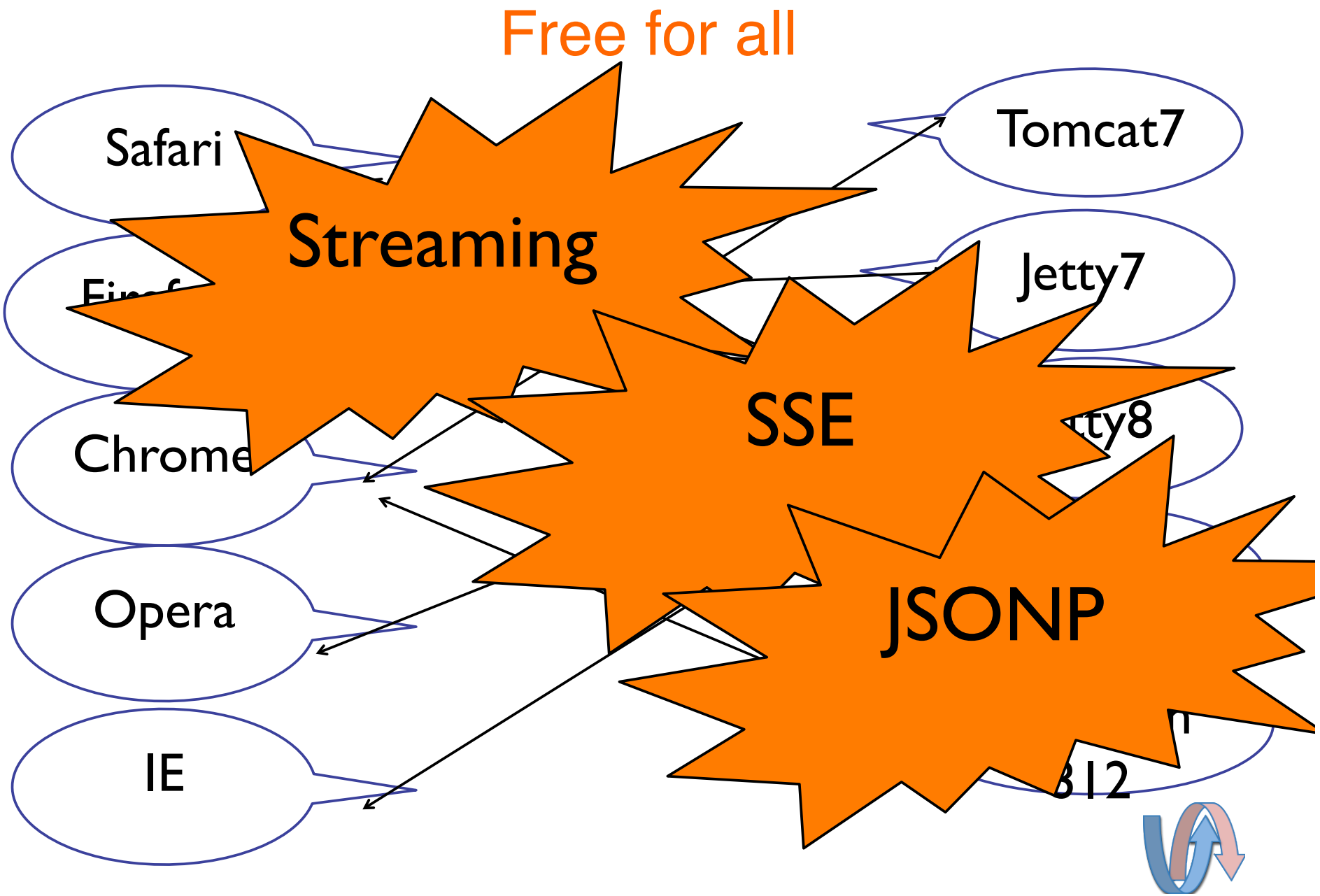


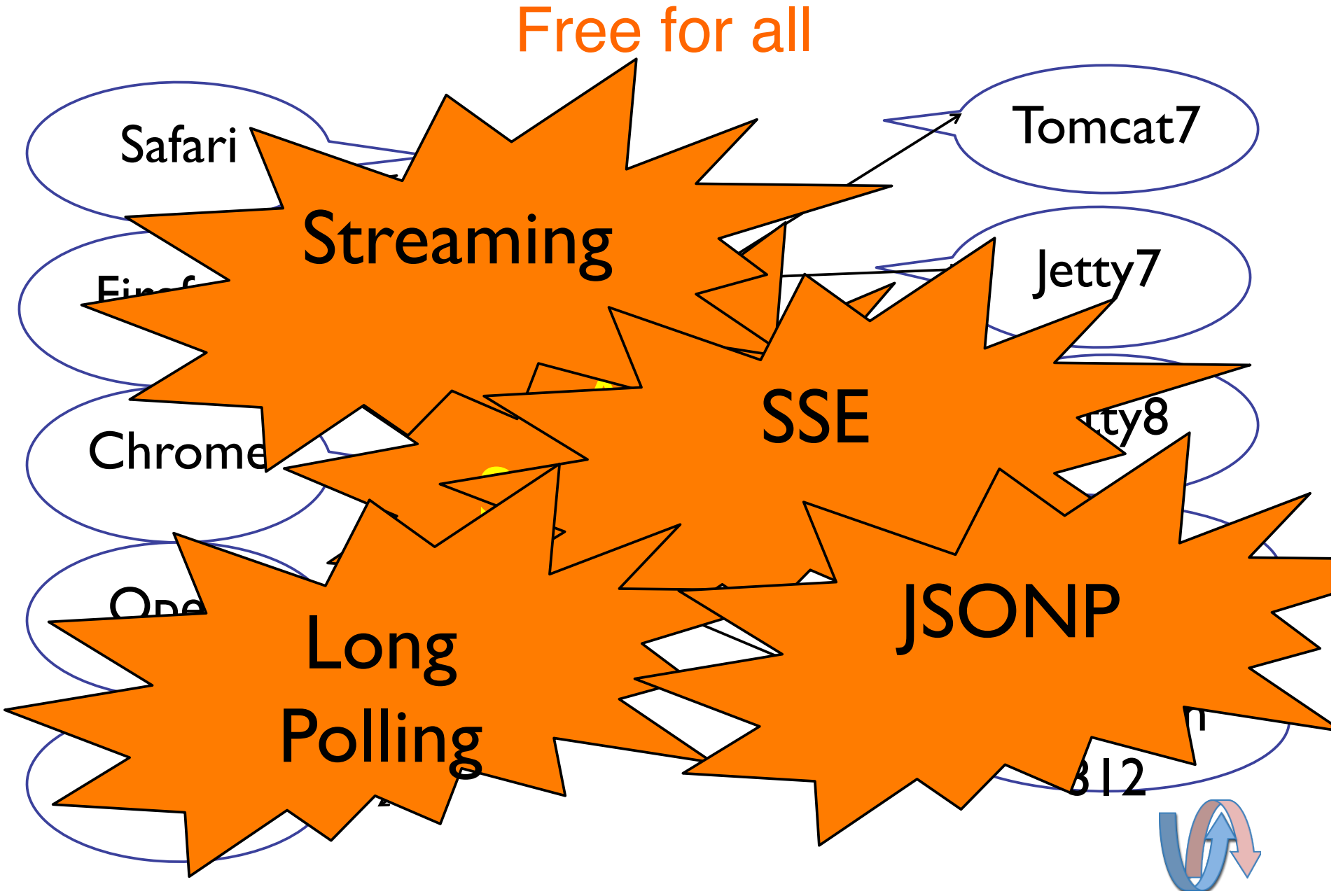
Free for all!


Safari
Firefox
Chrome
Opera
IE
Tomcat7
Jetty7
Jetty8
GlassFish3
GlassFish 312


Free for all!
Safari
Streaming
Chrome
Opera
IE
Tomcat7
Jetty7
Jetty8
GlassFish3
GlassFish 312

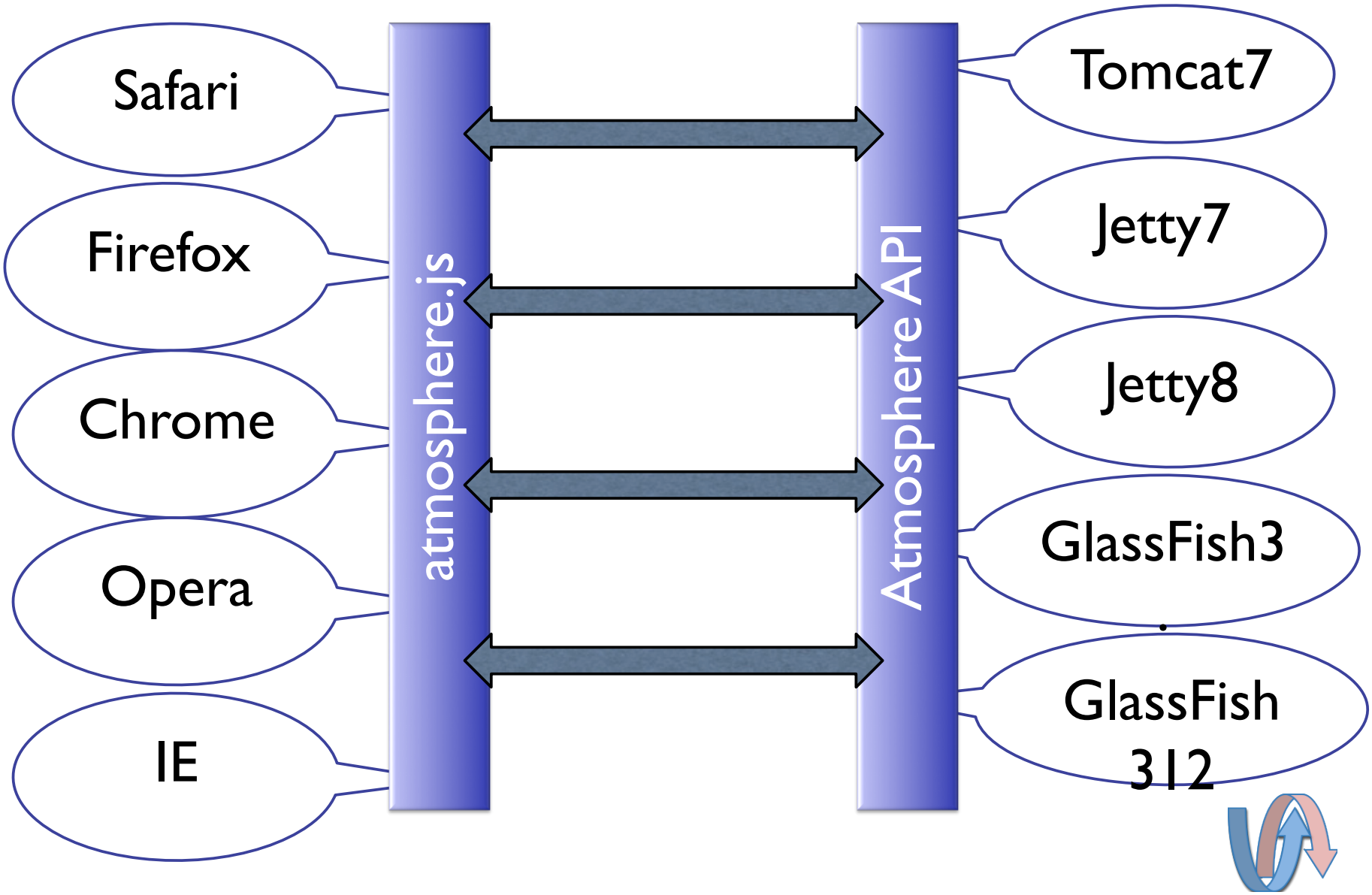

Free for all!
Safari
Streaming
Tomcat7
Jetty7
SSE
Chrome
GlassFish3
Opera
GlassFish
312
IE


Free for all
Safari
Streaming
Tomcat7
Jetty7
SSE
Chrome
JSONP
Opera
IE

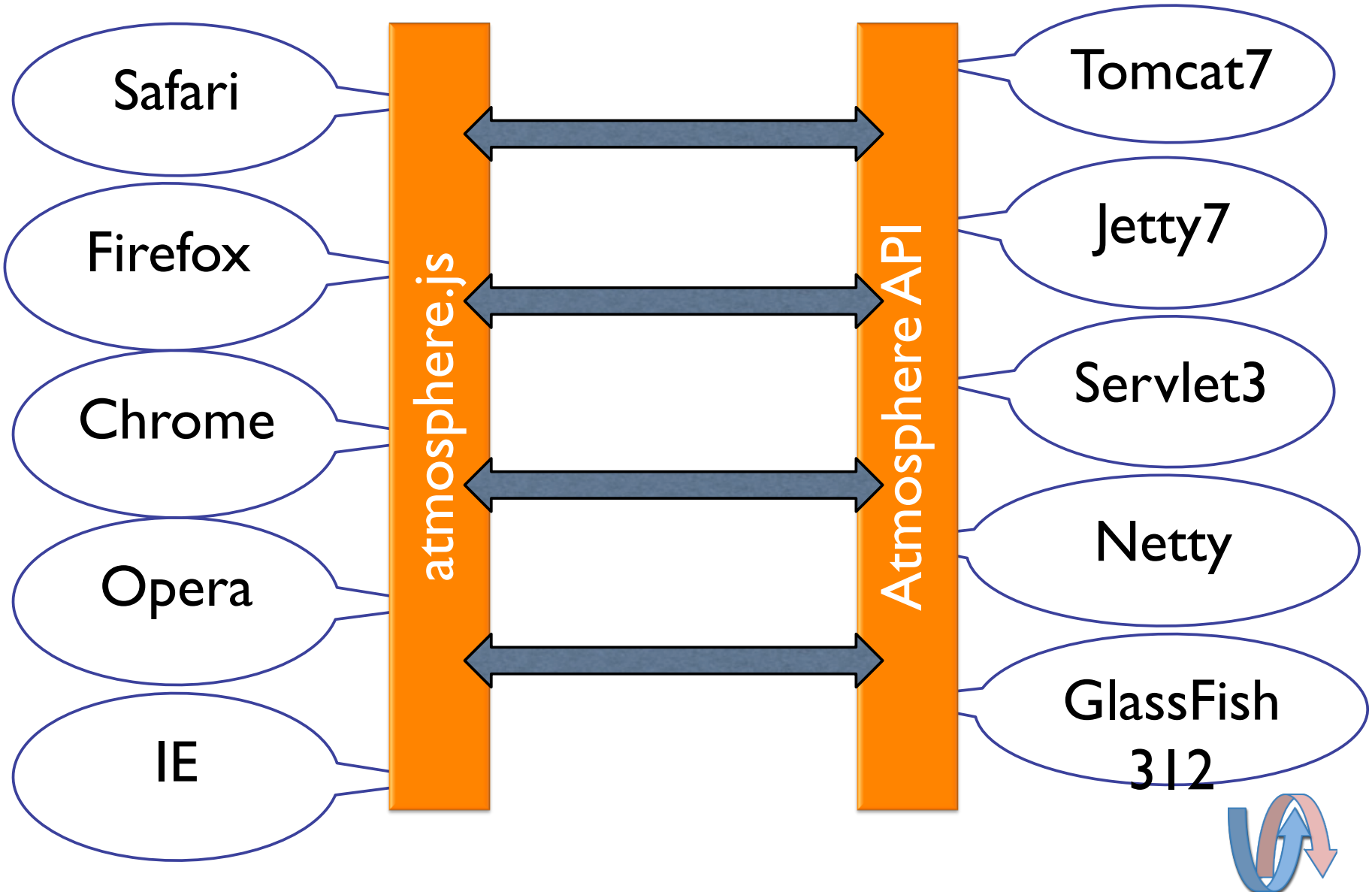

Free for all
Safari
Tomcat7
Streaming
Jetty7
SSE
Chrome
JSONP
Long
Polling


Free for all
Safari
Firefox
Chrome
Opera
IE
Tomcat7
Jetty7
Jetty8
GlassFish3
GlassFish 312
Atmosphere to the rescue!!!

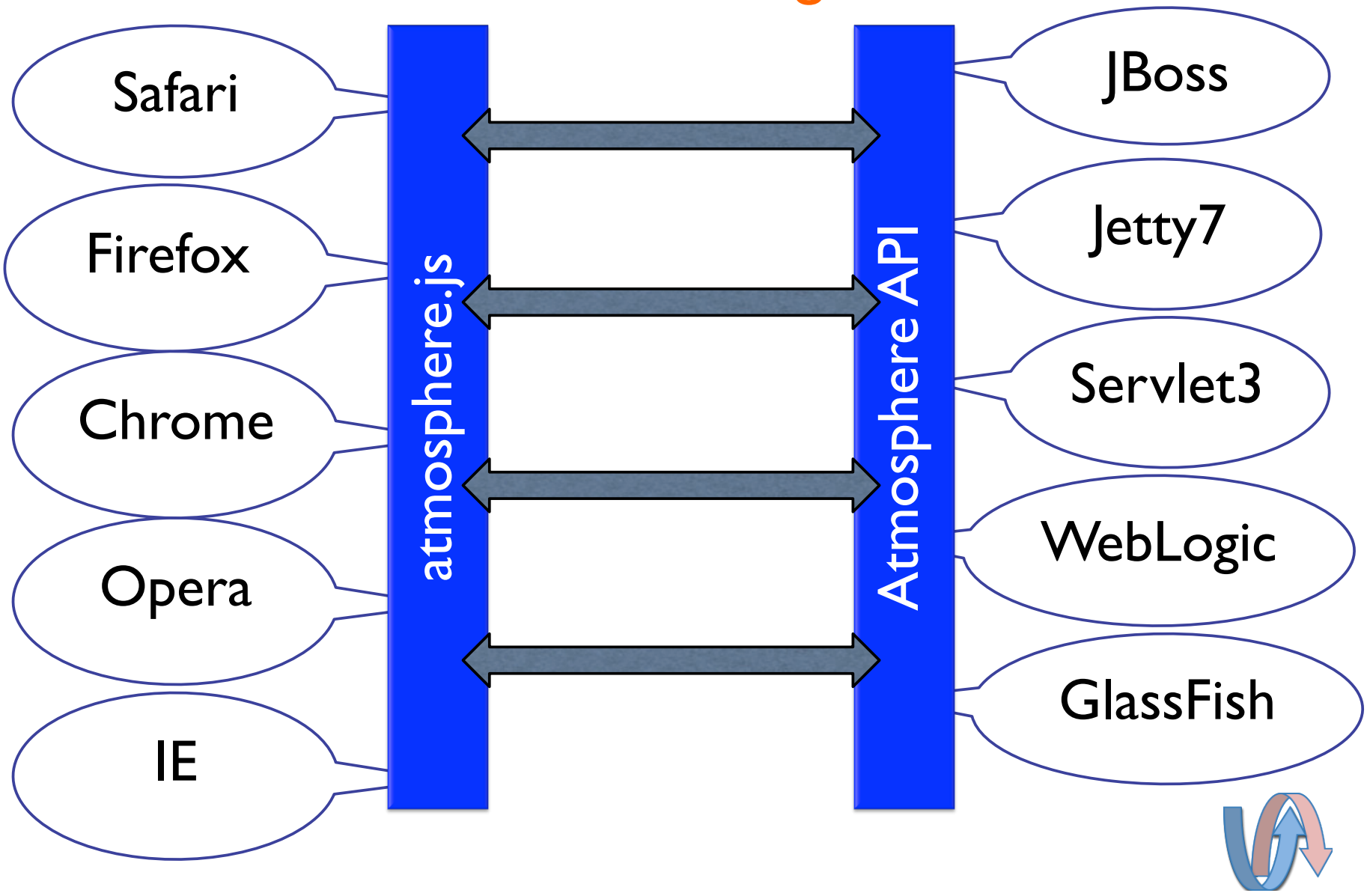
Atmosphere -WebSockets

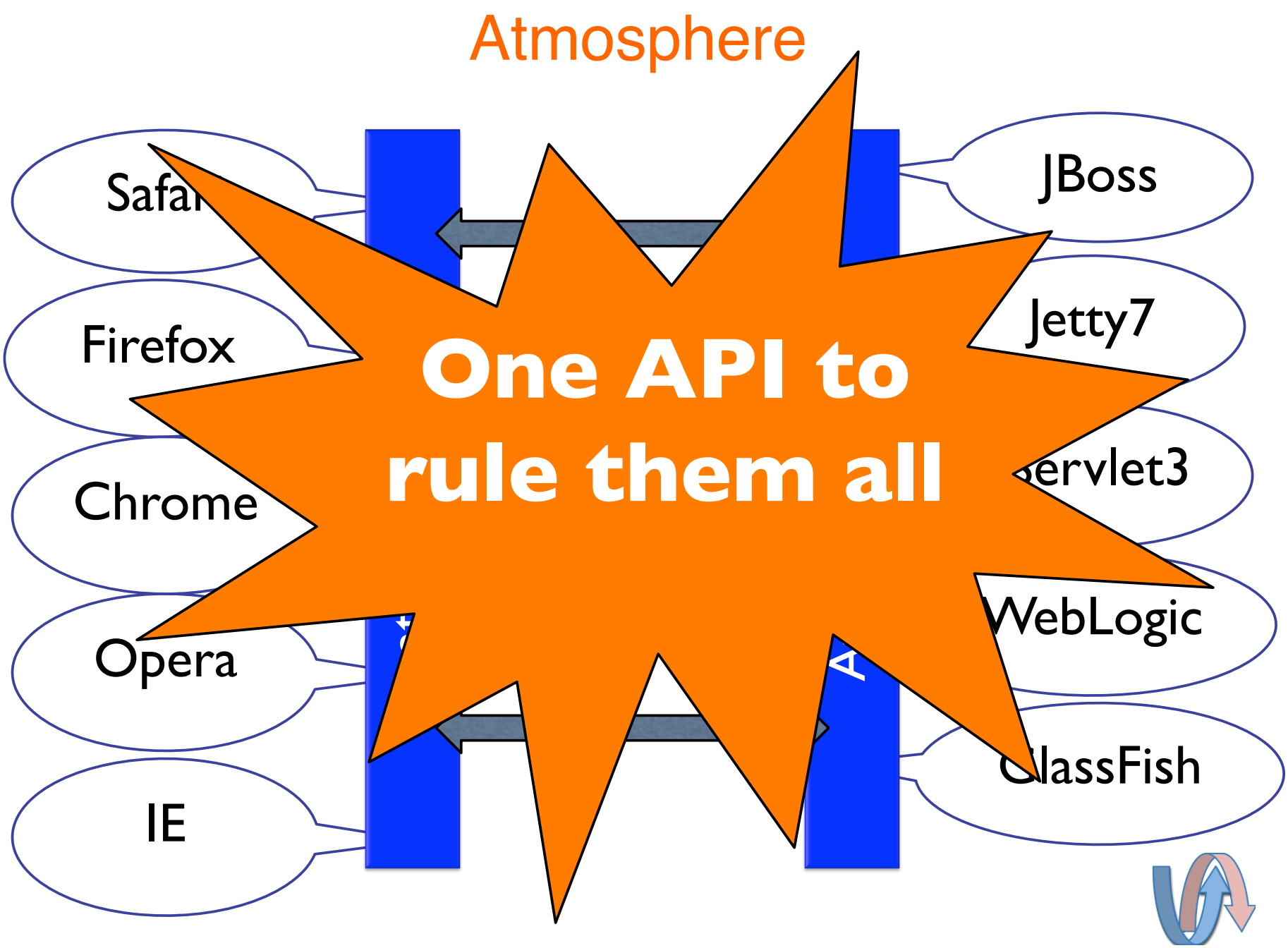


Atmosphere - HTML5 Server Side Events

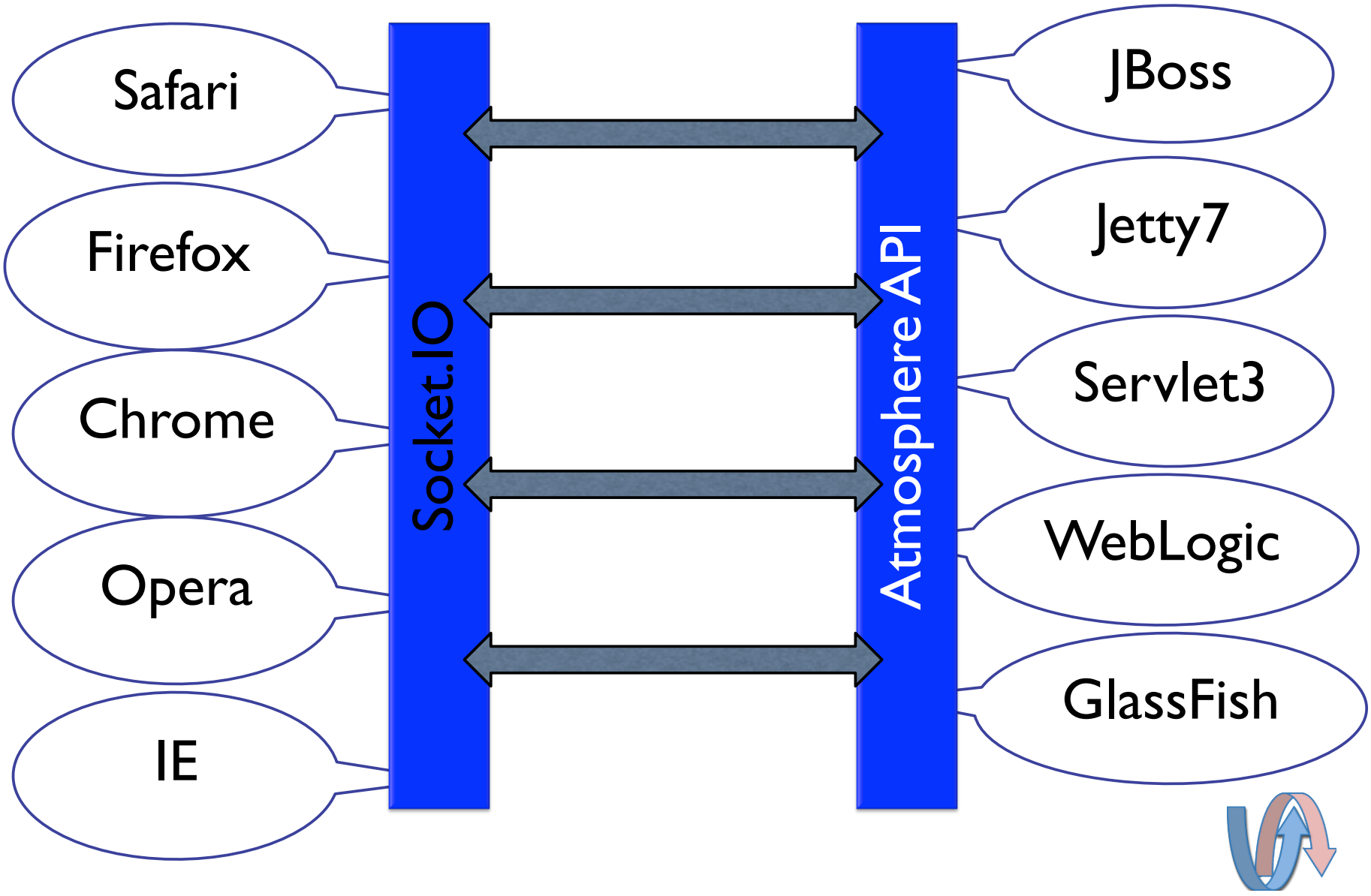


Atmosphere Long-Polling/HTTP Streaming

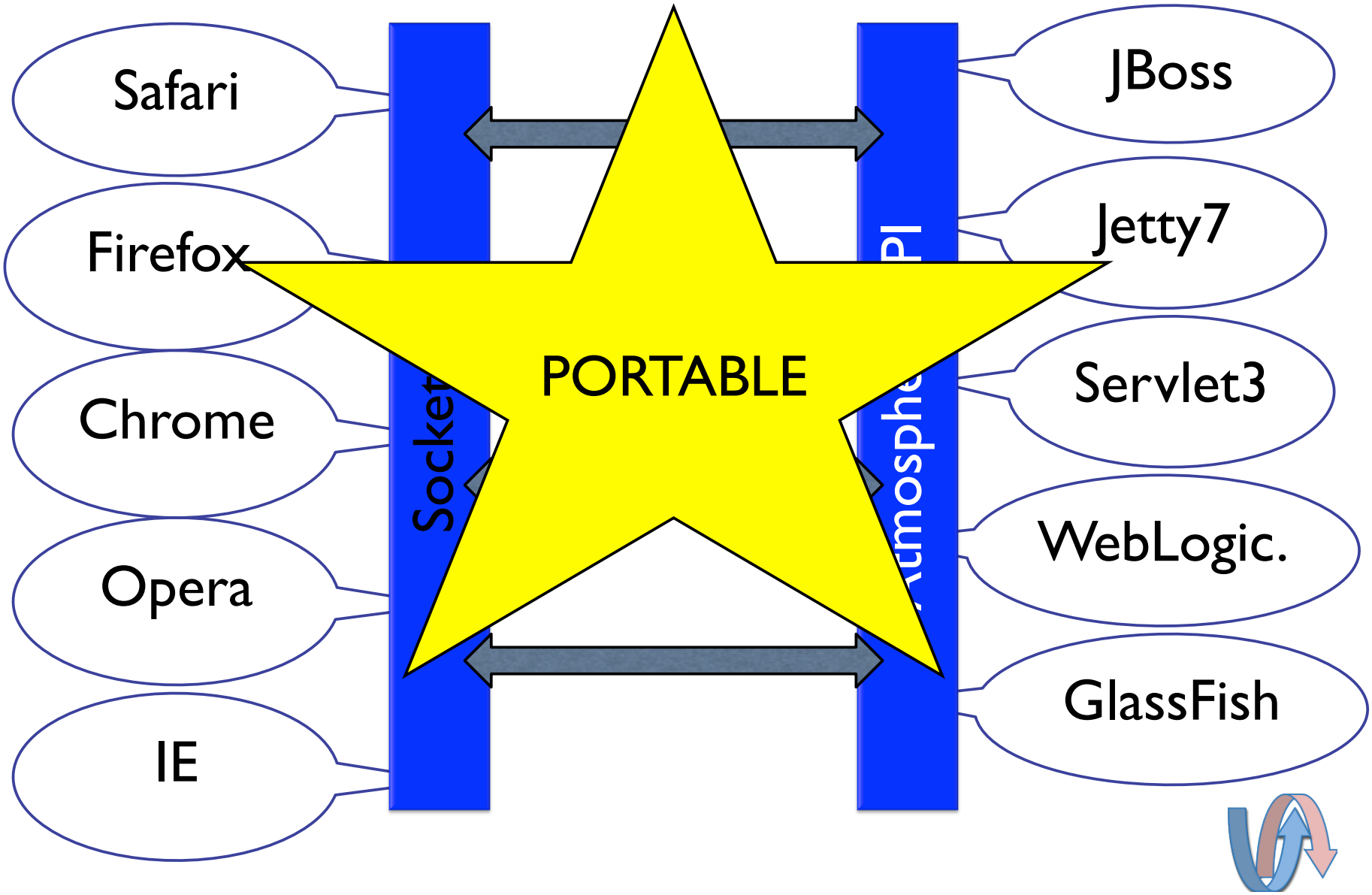



Atmosphere
Safa
Firefox
Chrome
Opera
IE
JBoss
Jetty7
ervlet3
WebLogic
lassFish
One API to
rule them all

Socket.IO, GWT, Wicket, JSF, etc.



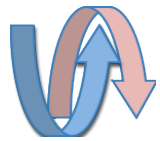
PORTABLE!


Safari
Firefox
Chrome
Opera
IE
PORTABLE
JBoss
Jetty7
Servlet3
WebLogic.
GlassFish

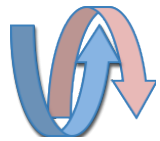
Big Mistake!!!!!

- **WebSockets is not ready for the internet, yet!**

Deploy in Production impossible

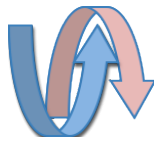


Atmosphere



Definition

- **Suspend:** open a channel, let a connection open for future events.
- **Resume:** close a channel
- **Broadcast:** push message to one or more channel, asynchronously.

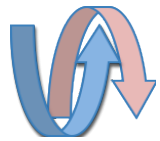


AtmosphereResource

- Represent a remote connection
- Manage lifecycle
  - Suspend
  - Resume
  - Broadcast
- Request/Response (similar to Servlet API)

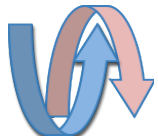
  `AtmosphereResource.getRequest()`

  `AtmosphereResource.getResponse()`
- Associated with one or more channel of communication (Broadcaster)



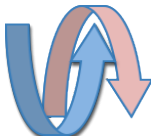
Reference

https://github.com/Atmosphere/atmosphere/wiki/Understanding-AtmosphereResource

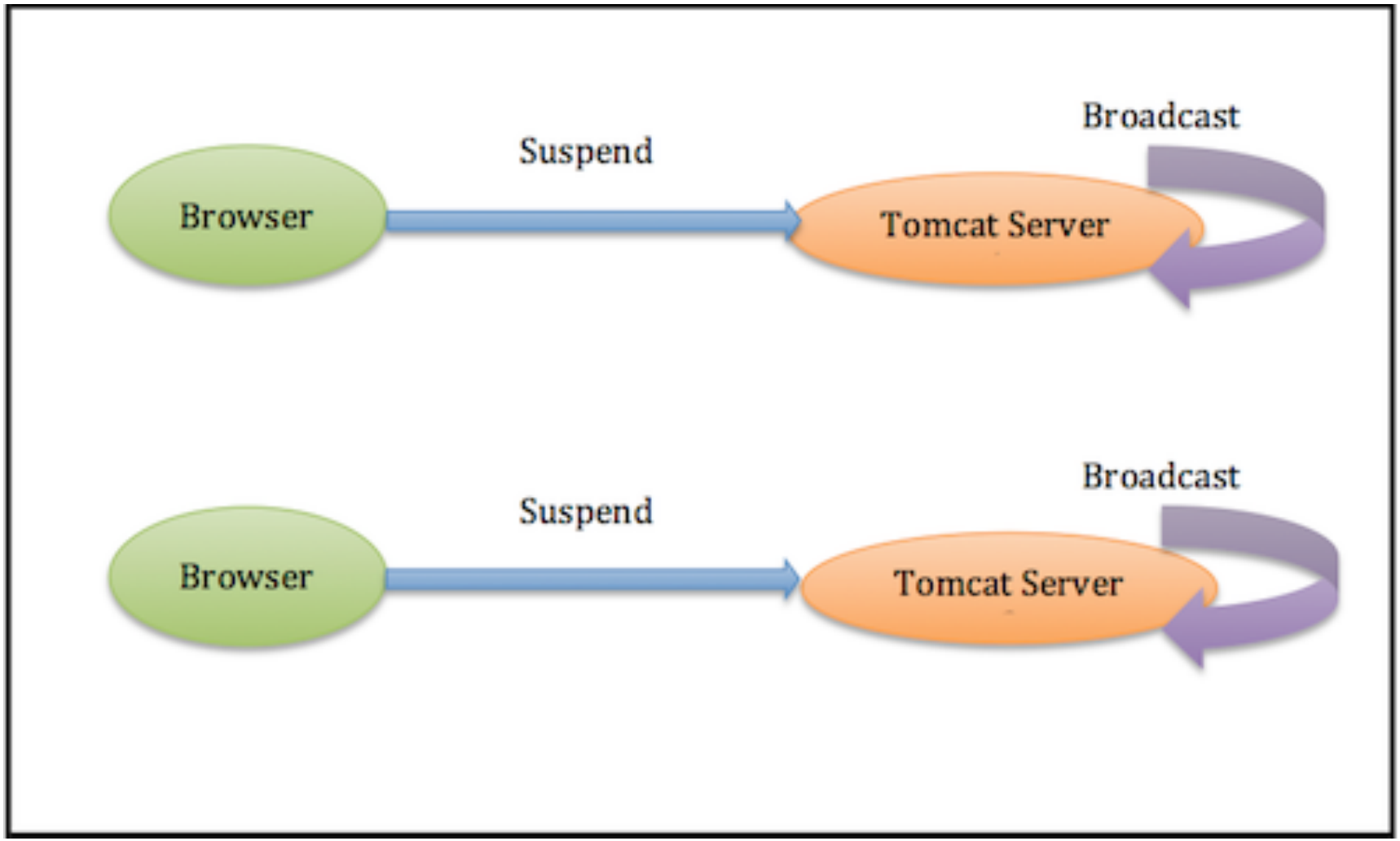


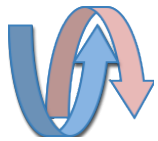
Broadcaster

- An Asynchronous Channel of communication, containing one of more AtmosphereResource.
- Similar to JMS's queue or topic
- BroadcasterFactory to create and retrieve them from any class in the application.
- MetaBroadcaster to publish events to one or more Broadcaster
- BroadcastFilter to filter, aggreagate or reject messages
- Asynchronous, Event Based, Thread Safe
- BroadcasterCache for caching message.
- Broadcaster contains zero or more AtmosphereResource

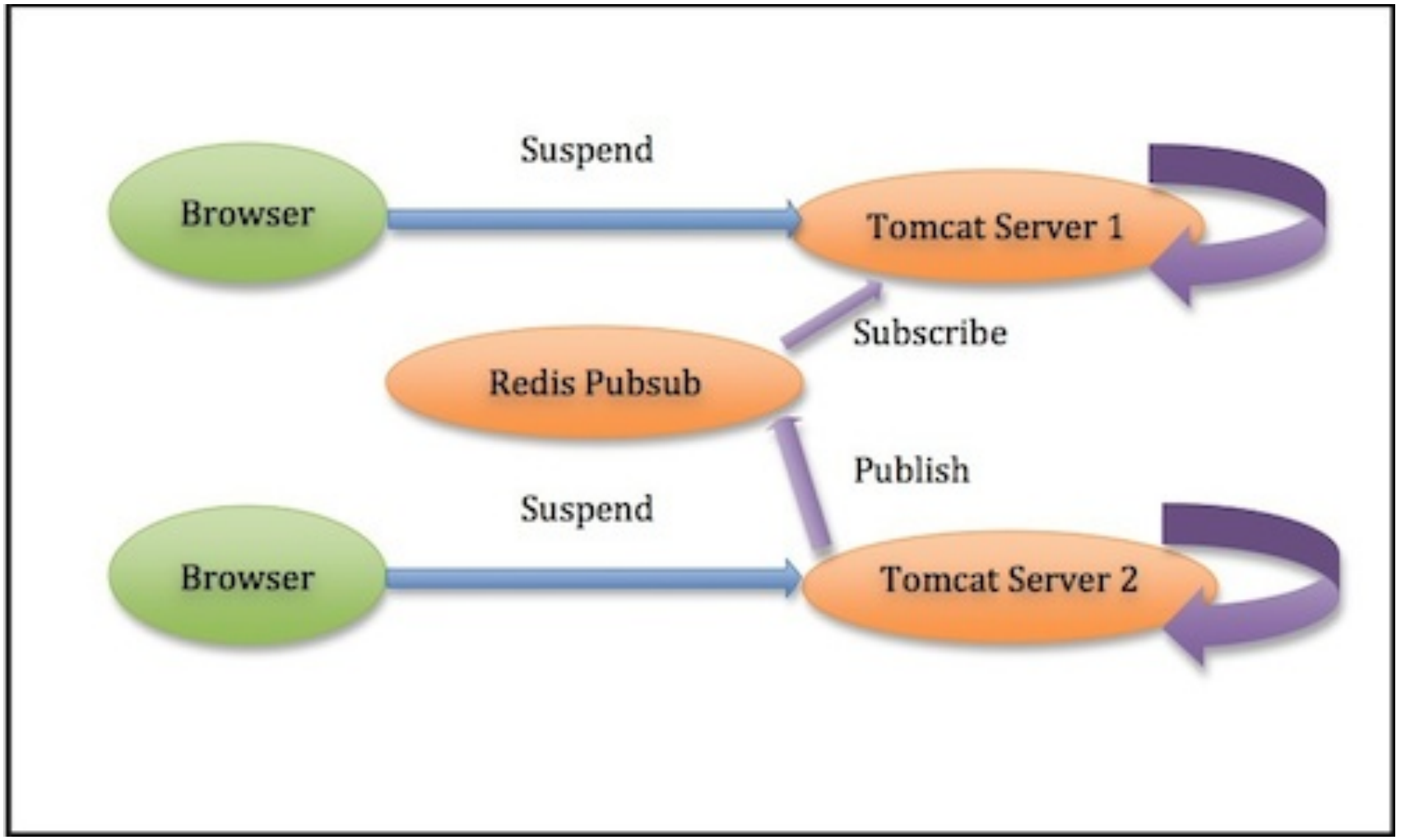


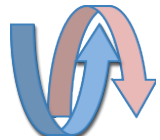
Broadcaster





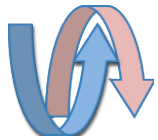
Transparent Broadcast





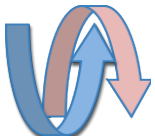
Broadcaster

- Default: in-memory
- Cloud/Cluster
  - RedisBroadcaster
  - JMSBroadcaster
  - XMPPBroadcaster
  - HazelcastBroadcaster
  - JGroupsBroascaster
- SimpleBroadcaster, JerseyBroadcaster, etc.



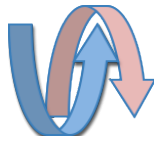
BroadcasterLifeCyclePolicy

- Make sure Broadcaster gets destroyed, recycled or alive. Prevent OOM. Confugure policy per Broadcaster.
  - NEVER: Keep all Broadcaster alive
  - EMPTY: Release resources associated when no AtmosphereResource
  - EMPTY_DESTROY: Destroy the object when no AtmosphereResource are associated
  - IDLE: Release resources associated when no broadcast happens after X time.
  - IDLE_DESTROY: Destroy the object when no broadcast happens after X time.



Reference

https://github.com/Atmosphere/atmosphere/wiki/Understanding-Broadcaster



BroadcastFilter

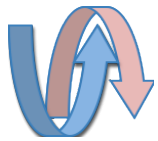
- Filter/transform/reject messages

  Resume I/O operations once needed
- For all

```
BroadcastAction filter(Object originalMessage, Object message);
```

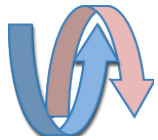
- Per request, per AtmosphereResource

```
BroadcastAction filter(AtmosphereResource atmosphereResource,
                       Object originalMessage,
                       Object message);
```



Reference

https://github.com/Atmosphere/atmosphere/wiki/Understanding-BroadcastFilter

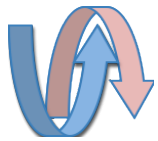


BroadcasterCache

- Track messages delivery. If a message can't be delivered, cache it and send it back once the remote client reconnect.

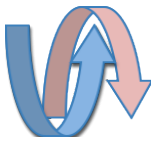
- An application can define it's own

```
void addToCache(String id, AtmosphereResource r, Object e);

List<Object> retrieveFromCache(String id, AtmosphereResource r);
```



Reference

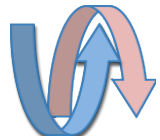
https://github.com/Atmosphere/atmosphere/wiki/Understanding-BroadcasterCache



Framework's Listener

Track AtmosphereResource lifecycle

- AtmosphereResourceEventListener
  - onPreSuspend
  - onSuspend
  - onResume
  - onDisconnect
  - onBroadcast
  - onThrowable



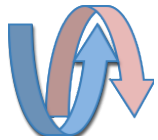
Framework's Listener

Track Broadcaster activity

- BroadcasterListener
  - onPostCreate
  - onPreDestroy
  - onBroadcast

Track broadcaster lifeCycle

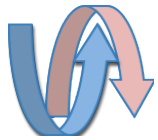
- BroadcasterLifeCycleListener
  - onEmpty
  - onIdle
  - onDestroy



Framework's Listener

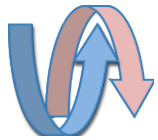
Track all activity executed by Atmosphere and all AtmosphereResource

- AsyncSupportListener
  - onSuspend
  - onResume
  - onTimeout
  - onClose
  - onDestroyed

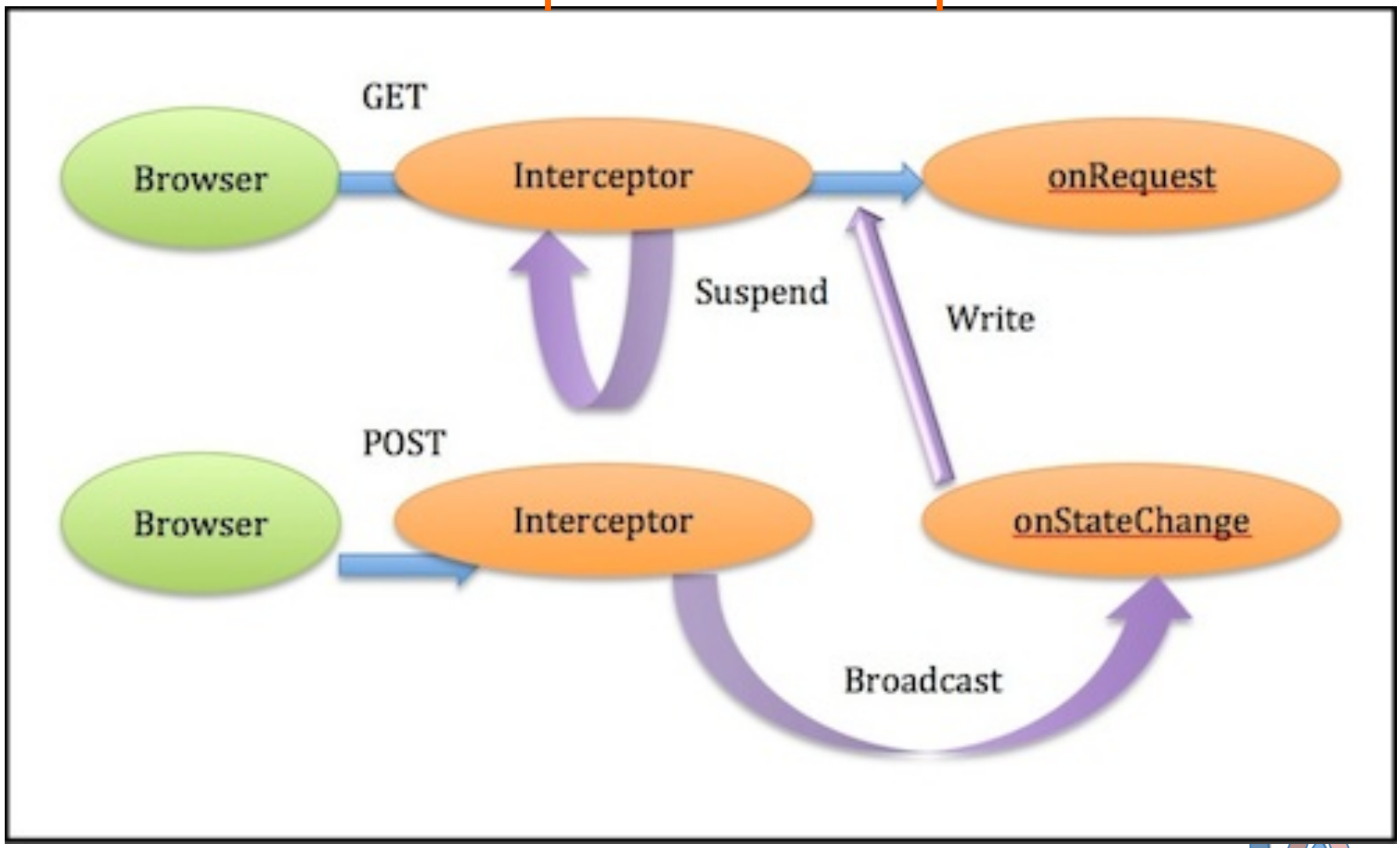


Reference

https://github.com/Atmosphere/atmosphere/wiki/Configuring-Atmosphere-Listener



AtmosphereInterceptor

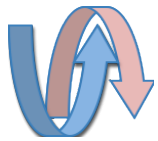


AtmosphereInterceptor

- Intercept the AtmosphereResource and pre/post process it before delivering it to AtmosphereHandler

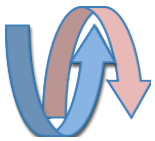
```
Action inspect(AtmosphereResource r);


void postInspect(AtmosphereResource r);
```

- All logic that apply to more than one application or not directly related to the application must go there.
- SSE, HeartBeat, OnDisconnect are all pre-defined AtmosphereInterceptor



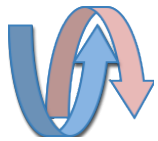
Reference

https://github.com/Atmosphere/atmosphere/wiki/Understanding-AtmosphereInterceptor



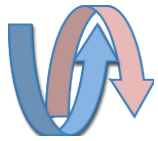
Atmosphere's Application Component

- WebSocketHandler
  - Only WebSocket (**Warning**)!
- AtmosphereHandler
  - AtmosphereGwtHandler
  - SocketIOHandler
  - **Annotations Based**
- Jersey Resource
  - All transports
- Meteor
  - All transports



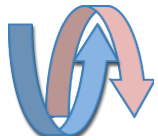
AtmosphereHandler

```
public interface AtmosphereHandler {

 void onRequest(AtmosphereResource resource)
      throws IOException;

 void onStateChange(AtmosphereResourceEvent
     event) throws IOException;

 void destroy();
}
```



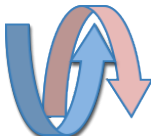
AtmosphereHandler

```
public class ChatAtmosphereHandler implements AtmosphereHandler {

    @Override
    public void onRequest(AtmosphereResource r) throws IOException {
        AtmosphereRequest req = r.getRequest();
        if (req.getMethod().equalsIgnoreCase("GET")) {
           r.suspend();
       } else if (req.getMethod().equalsIgnoreCase("POST")) {

         r.getBroadcaster()
                  .broadcast(req.getReader().readLine().trim());
        }
    }
```



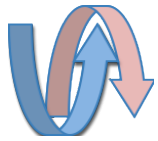
AtmosphereHandler

```
public void onStateChange(AtmosphereResourceEvent event) throws IOException {
      ….
      if (event.isSuspended()) {
            …
            res.getWriter().write(new Data(author, message).toString());
            switch (r.transport()) {
                case JSONP:
                case AJAX:
                case LONG_POLLING:
                    event.getResource().resume();
                    break;
                case WEBSOCKET :
                case STREAMING:
                    res.getWriter().flush();
                    break;
            }
        } }
```



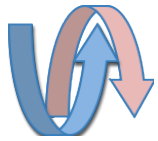
AtmosphereHandler - OnMessage

```
@AtmosphereHandlerService(
  path="/chat",
  interceptors = {AtmosphereResourceLifecycleInterceptor.class,
                  BroadcastOnPostAtmosphereInterceptor.class})
public class ChatRoom extends OnMessage<String> {
    private final ObjectMapper mapper = new ObjectMapper();

    @Override
    public void onMessage(AtmosphereResponse response, String message) {
         response.getWriter()
                     .write(mapper.writeValueAsString(
                          mapper.readValue(message, Data.class)));
    }
}
```



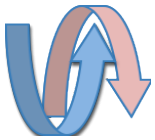
AtmosphereHandler – @Managed

```
@ManagedAtmosphereHandlerService (path = "/chat")
public class ChatAtmosphereHandler {
    private final ObjectMapper mapper = new ObjectMapper();
    @Message
    public void onMessage(AtmosphereResponse response, String message) throws
    IOException {
        response.getWriter().write(
          mapper.writeValueAsString(mapper.readValue(message, Data.class)));
    }
```



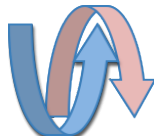
Reference

https://github.com/Atmosphere/atmosphere/wiki/Understanding-AtmosphereHandler



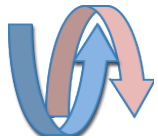
Jersey

```
@Path("/")
public class ChatResource {
    @Suspend(contentType = "application/json")
    @GET
    public String suspend() {
        return "";
    }

    @Broadcast(writeEntity = false)
    @POST
    @Produces("application/json")
    public Response broadcast(Message message) {
        return new Response(message.author, message.message);
    }

}
```



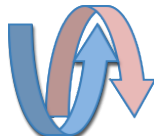
Reference

https://github.com/Atmosphere/atmosphere/wiki/Getting-Started-with-The-Atmosphere-Framework-and-WebSocket



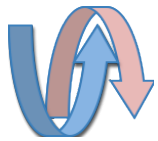
GWT

- A special AtmosphereInterceptor for serializing and deserializing GWT message
- All Atmosphere's concepts available to GWT normal application.



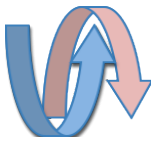
Reference

https://github.com/Atmosphere/atmosphere-extensions/wiki/Atmosphere-GWT



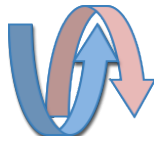
Meteor

```
@MeteorService (path = "/chat")
public class MeteorChat extends HttpServlet {
    @Override
    public void doGet(HttpServletRequest req, HttpServletResponse res) throws
    IOException {
        Meteor.build(req)
              .addListener(new AtmosphereResourceEventListenerAdapter());
    }
    @Override
    public void doPost(HttpServletRequest req, HttpServletResponse res)
    throws IOException {
      BroadcasterFactory.getDefault()
         .lookup(DefaultBroadcaster.class, "/*")
         .broadcast(new Data(author, message).toString());
    }
```



Reference

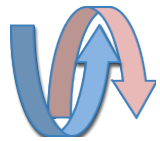
https://github.com/Atmosphere/atmosphere/wiki/Getting-Started-with-Meteor,-WebSocket-and-Long-Polling



WebSocketHandler

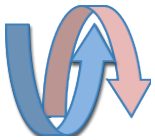
- **Only support WebSocket, mimic the Javascript client side API.**

```
void onOpen(WebSocket w);

void onClose(WebSocket w)

void onTextMessage(WebSocket w, String data)
```



Reference

https://github.com/Atmosphere/atmosphere/wiki/Understanding-WebSocketHandler



WebSocket Sub Protocol

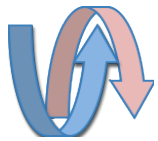
- **WebSocketProtocol**

  Define your own protocol on top of WebSocket

- Default:

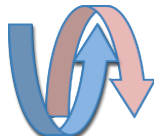
  WebSocket message => POST

- **SwaggerSocket: REST over WebSockets -> More information**



Reference

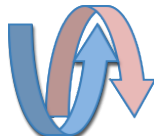
https://github.com/Atmosphere/atmosphere/wiki/Writing-WebSocket-Sub-Protocol



atmosphere.js

- **Unified API, works for all transport**

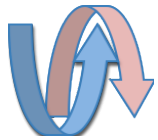
```
var request = { url: 'http://localhost:8080',
                transport: 'websocket'
                fallbackTransport: 'long-polling'
              }

request.onMessage = function() {…}
request.onOpen = function() {…}

var subSocket = jQuery.atmosphere.subscribe(request);
subSocket.push('some data');
```



atmosphere.js

create

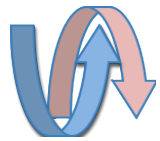
```
var socket = $.atmosphere;
var request = {
      url: 'http://localhost:8080/pubsub',
      content-type : 'application/json'
      transport : 'websocket'
}

var subSocket = socket.subscribe(request);
subsocket.push(json);
```



atmosphere.js

functions

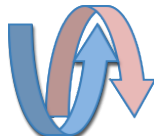
```
var socket = $.atmosphere;


socket.onOpen = function( response ){..};
socket.onMessage = function( response ){..};
socket.onError = function( response ){..};
socket.onClose = function( response ){..};
socket.onLocalMessage = function ( msg){..};
Socket.onTransportFailure
    = function ( errorMsg,  request ){..};
```



atmosphere.js

adapt

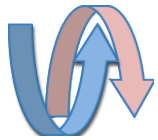
```
socket.onTransportFailure
                = function ( errorMsg, request ){

    if (window.EventSource) {
        request.fallbackTransport =
                                ”long-polling";
        transport = "see";
    }
}
```



atmosphere.js

handle messages

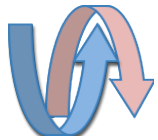
```
socket.onMessage = function ( response){

    var message= response.responseBody;
    // Handle the message
}
```



atmosphere.js

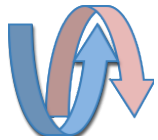
share

```
socket.onLocalMessage = function ( message){

      // Handle messages from a tabs/windows

}
```



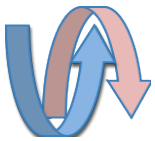
Reference

https://github.com/Atmosphere/atmosphere/wiki/jQuery.atmosphere.js-API



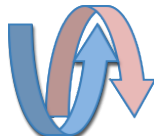
NettoSphere

- **Atmosphere running on top of the Netty Framework**
- **No JavaEE Container required.**
- **Easy to embed.**
- **Extremely Fast.**



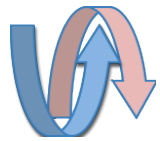
Reference

https://github.com/Atmosphere/nettosphere/blob/master/README.md



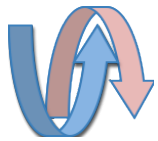
wAsync

- **Java WebSocket's, SSE, Long-Polling and Streaming Client**



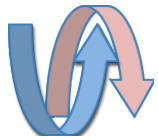
Reference

https://github.com/Atmosphere/wasync/blob/master/README.md



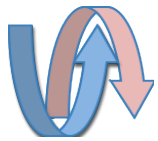
Atmosphere on Play!

- **Atmosphere running on top of the Play Framework.**



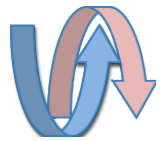
Reference

https://github.com/Atmosphere/atmosphere-play/blob/master/README.md

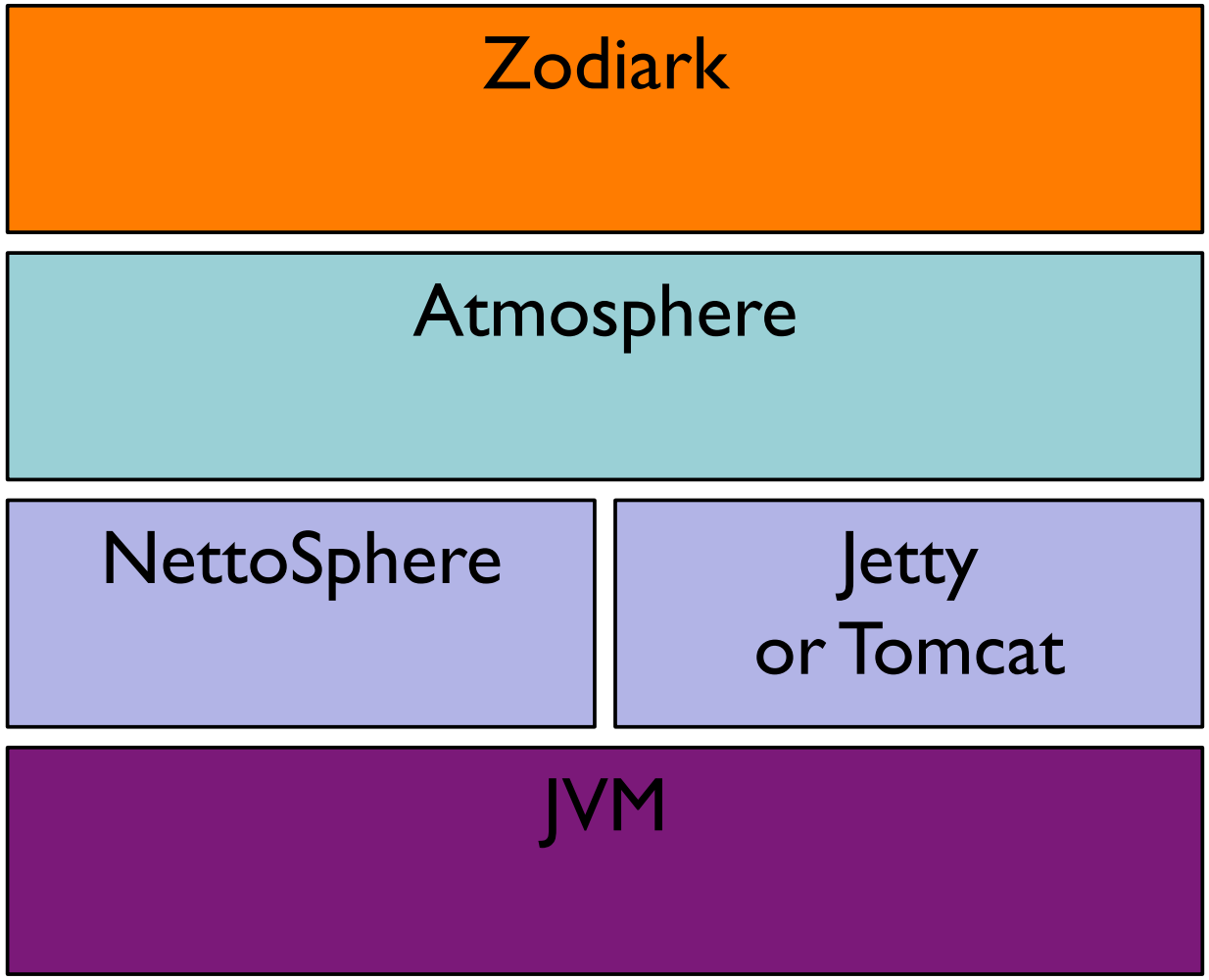


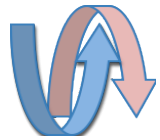
Zodiark

- **Run on Top of Atmosphere/ NettoSphere**
- **React Based Architecture**
  - **EventBus for I/O Events and Messages**
  - **Synchronous or Asynchronous Processing**

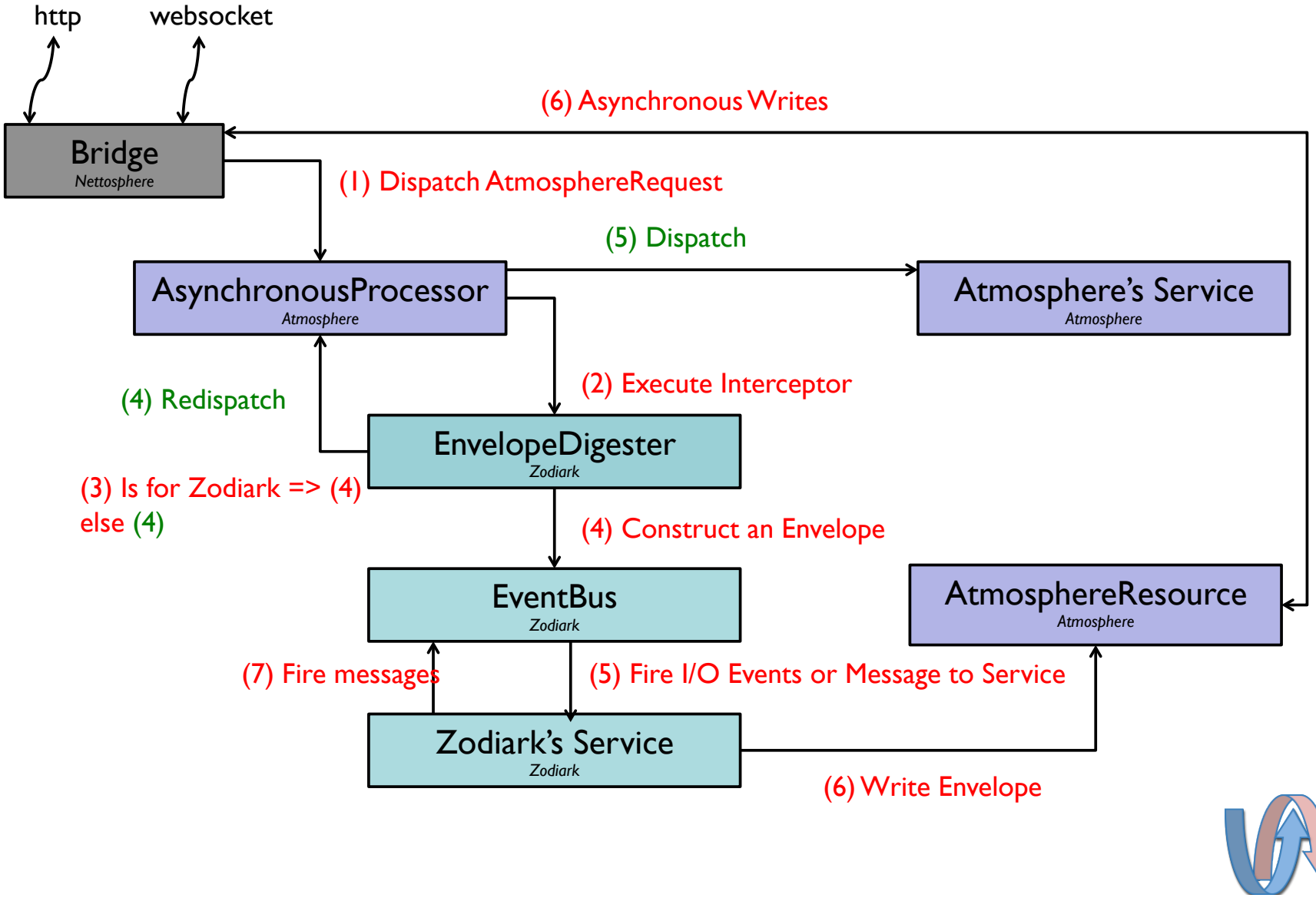


Architecture





Request Workflow



Zodiark Internal

EnvelopeDigester
Zodiark

Annotated with @AtmosphereInterceptor, this class will be deployed and its role will consist of deserializing the AtmosphereRequest's body into an Envelope. An Envelope contains the protocol information. All Endpoints must use an Envelope between them for communicating.

If the EnvelopeDigester is unable to deserialize into an Envelope, the request will be dispatched back to Atmosphere, and Atmosphere will try dispatch the request to its own services.

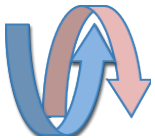
EventBus
Zodiark

Master piece of the Zodiark's Architecture. Responsible for delivering I/O events from remote endpoint to Service, as well as deliver messages between Services. Services must always communicate between them using the EventBus.

The EventBus can deliver I/O events synchronously (using the calling thread) or asynchronously. The default behavior is synchronously, but for performance reason asynchronously will eventually be enabled.

Service
Zodiark

A Service reacts to I/O events and messages. All Zodiark's features must be implemented as Service. A Service can be registered manually or programmatically using the @On annotation. PublisherService, SubscriberService, WowzaService are example of Service implementation.



Zodiark Internal

ZodiarkObjectFactory
Zodiark

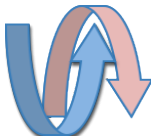
Responsible for Injecting and Creating Object in Zodiark. For example, the EventBus is getting injected for all Service. In Zodiark, no classes are created directly and instead must be created using the Context or ZodiarkObjectFactory

Chat
Atmosphere

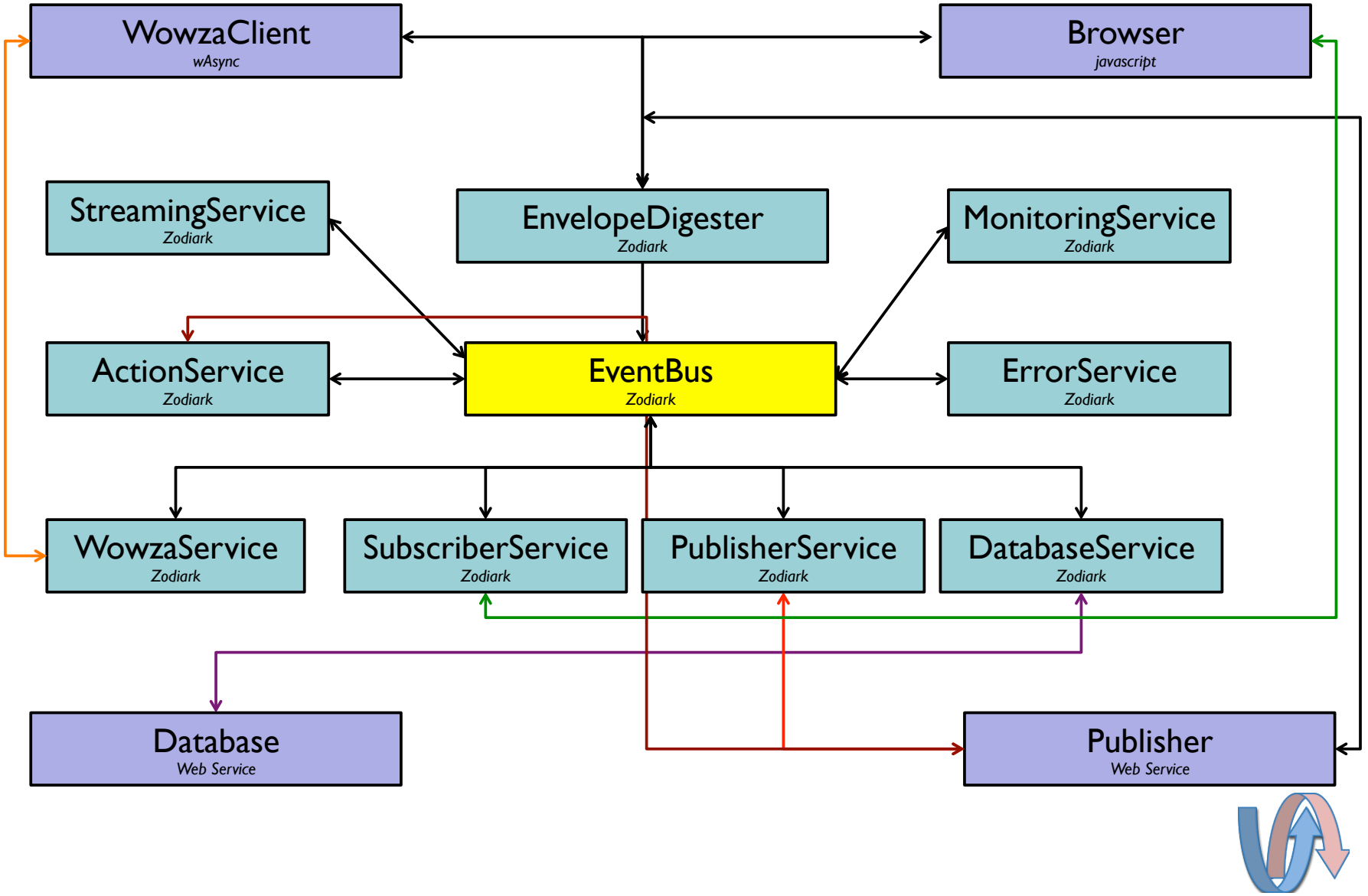
A pure Atmosphere Service used for the Zodiark Chat. The Envelope send by the Publisher and Subscriber are deserialized and the chat conversation is redispatched to Atmosphere's own Service.

ZodiarkServer
Zodiark

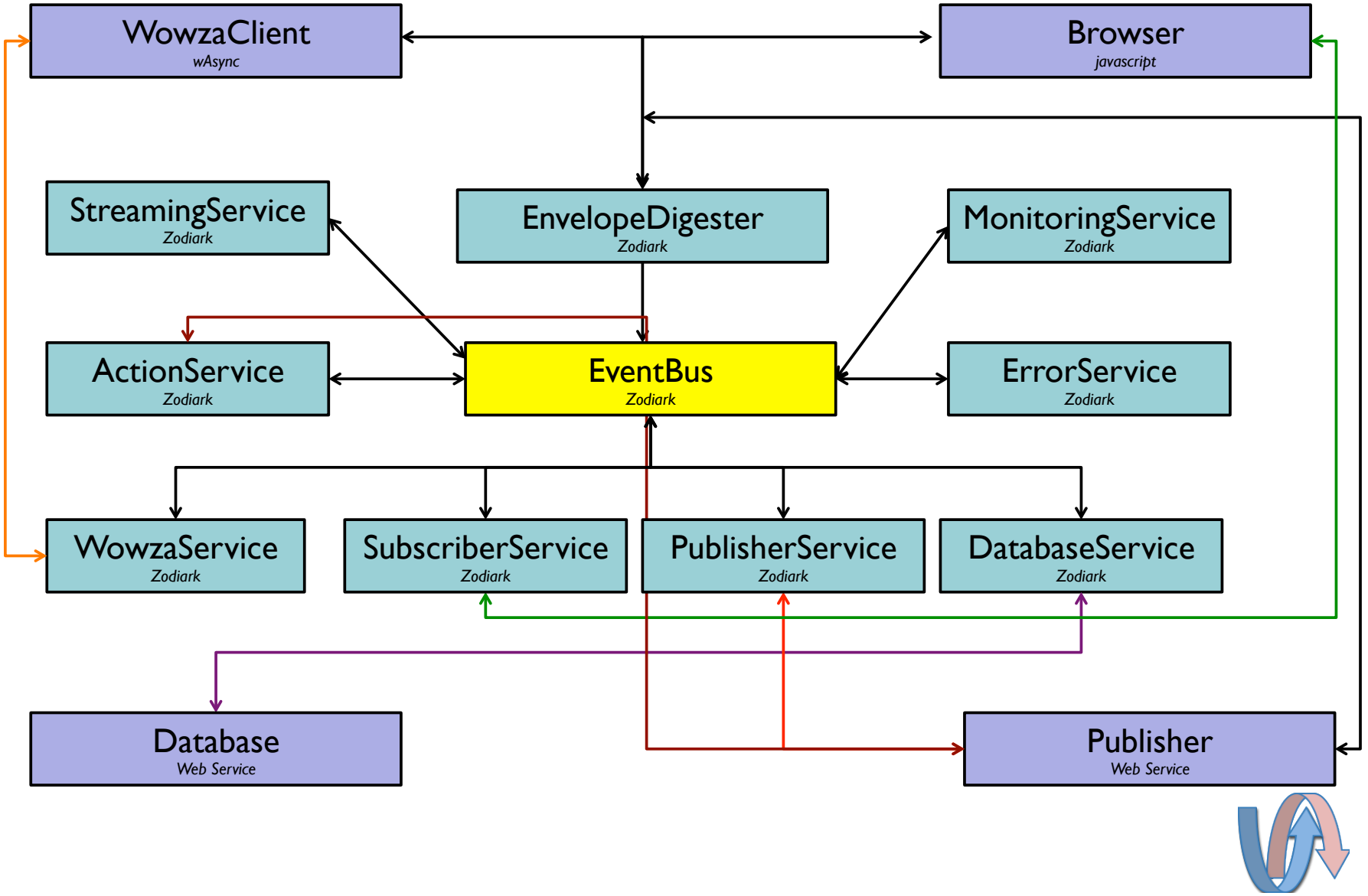
Bootstrap classes used to start NettoSphere, Atmosphere and Zodiark



Service

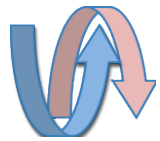

WowzaClient
wAsync
Browser
javascript
StreamingService
Zodiark
EnvelopeDigester
Zodiark
MonitoringService
Zodiark
ActionService
Zodiark
EventBus
Zodiark
ErrorService
Zodiark
WowzaService
Zodiark
SubscriberService
Zodiark
PublisherService
Zodiark
DatabaseService
Zodiark
Database
Web Service
Publisher
Web Service

Service


WowzaClient
wAsync
Browser
javascript
StreamingService
Zodiark
EnvelopeDigester
Zodiark
MonitoringService
Zodiark
ActionService
Zodiark
EventBus
Zodiark
ErrorService
Zodiark
WowzaService
Zodiark
SubscriberService
Zodiark
PublisherService
Zodiark
DatabaseService
Zodiark
Database
Web Service
Publisher
Web Service

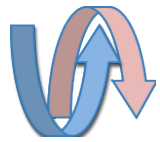
API Documentation

http://high-level.github.io/zodiark/apidocs/



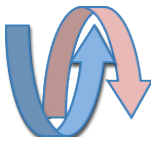
EventBus

```
public interface EventBus {

        EventBus ioEvent(Envelope e, AtmosphereResource r);

        EventBus ioEvent(Envelope e, AtmosphereResource r,
                         Reply reply);

        EventBus message(String path, Object message);

        EventBus message(String path, Object message, Reply reply);

        EventBus on(String path, Service service);

        EventBus off(String path);

}
```



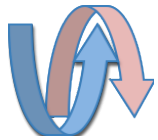
Reply

```
public interface Reply<T> {

        void ok(T response);

        void fail(T response);
}
```



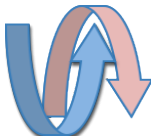
Service

```
public interface Service {

        // I/O Event
        void reactTo(Envelope e, AtmosphereResource r);

         // String
        void reactTo(String path, Object message, Reply reply);

}
```



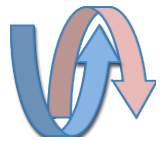
PublisherService

```
public class PublisherServiceImpl implements PublisherService, Session<PublisherEndpoint> {

    @Inject
    public EventBus eventBus;
    @Inject
    public ObjectMapper mapper;
    @Inject
    public Context context;

    @Override
    public void reactTo(String path, Object message, Reply reply) {
        switch (path) {
            ....
        }
    }

    @Override
    public void reactTo(Envelope e,AtmosphereResource r, Reply reply) {
        switch (e.getMessage().getPath()) {
            ...
        }
    }
```



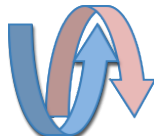
WowzaClient

- **wAsync Client extended to support the Zodiark Protocol.**
- **Support WebSockets and Streaming**
- **Fully Asynchronous**



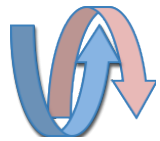
Wowza Client

```
final ZodiarkClient c = new ZodiarkClient.Builder()
                                        .path("http://127.0.0.1:8080").build();

    c.handler(new OnEnvelopHandler() {
        @Override
        public boolean onEnvelop(Envelope e) throws IOException {
            // React
            c.handler(new OnEnvelopHandler() {
                @Override
                public boolean onEnvelop(Envelope e) throws IOException {
                    ....
                    return false;
                }
            }).send(Envelope.newClientReply(e, e.getMessage()));
            return true;
        }
     })
    .open()
    .send(Envelope.newClientToServerRequest(
                            new Message(new Path("/wowza/validate/"), "OK")));
```



JavaScript Client: subscriber.js

- **Pure Javascript Client**
- **Based on atmosphere.js, with extension of the Zodiark Protocol**



JavaScript Client: subscriber.js

```
var handler = new zodiark.EnvelopeHandler();
   handler.onEnvelope = function (envelope) {
       ...
   };

   handler.onError = function (error) {
      ....
   }
   handler.onClose = function (response) {
      ...
   }

   socket = new zodiark.Builder()
                  .url(document.location.toString()).build()
                  .handler(handler).open();
```