WebSockets vs Server-Sent Events vs Long-polling

Real-Time Web September 5, 2011

Apparently social networking is the banner of the nowadays web. Everybody intends bringing some features into his projects. Some of them require immediate notification. That is getting common, if you open a page with upcoming messages (status feed, notification subsystem, friends-list), you expected them being updated as soon as a new message (status, notification, friend-making action) arrives. As you well know, original web design allowed only one-way client-server communication (one requests, another one responds), though now HTML5 working group doing their best to fix it or rather to patch it. However, the web-projects are still using long-polling trick to emulate server-client communication.

Well, now new web browser versions appear every few months. Besides they update automatically. Thus a huge number of users have the latest browser versions, which support HTML 5 communication APIs. Is that the time to put long-polling away? Let's find out.

Our test task will be something you may likely need if you have on your site any sort of user communication. That is notification of user actions. In the simplest case when the user gets a private message, the number of unread notifications increases in the user panel. We will solve the task using long-polling, Server-Sent Events and WebSockets. Then we compare the results.

First of all let's examine the common code used in the examples. We will need configuration file, a library to access DB, a model to retrieve unread notification number and to add a new notification.

Usually such communication API examples don't include any business logic, but execution delays to emulate the model working. I would like to make it close to the real application, what is meant to help when comparing memory/CPU usage on the server for each of the cases.

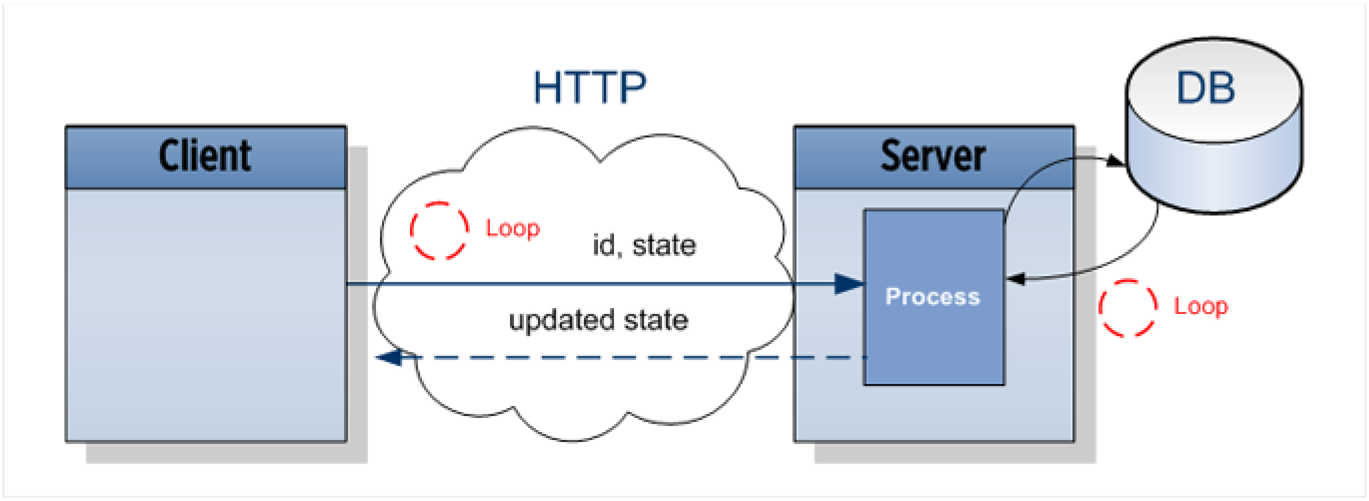
So, we will need a dump DB table:

```
CREATE TABLE IF NOT EXISTS `notification` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `recipientUid` int(10) unsigned NOT NULL,
  `eventId` int(10) unsigned NOT NULL,
  `isNew` tinyint(1) unsigned NOT NULL DEFAULT '1',
  PRIMARY KEY (`id`),
  KEY `IX_recipientUid` (`recipientUid`),
  KEY `IX_isNew` (`isNew`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 COMMENT='User notifications';
```

Here is the model:

```
<?php
class Model_UserNotification
{
   private $_db;


   public function  __construct(Lib_Db $db)
   {
        $this->_db = $db;
   }
   /**
    *
    * @param int $recipientUid
    * @return int
    */
   public function fetchNumberByRecipientUid($recipientUid)
   {
       return $this->_db->fetch("SELECT count(*) as count "
           . " FROM notification WHERE recipientUid = %d AND isNew = 1"
           , $recipientUid)->count;
   }
   /**
    *
    * @param int $recipientUid
    * @param int $eventId
    */
   public function add($recipientUid, $eventId)
   {
       $this->_db->update("INSERT INTO "
           . " notification (`id`, `recipientUid`, `eventId`, `isNew`) VALUES (NULL, '
           , $recipientUid, $eventId);
   }
   /**
    *
    * @param int $recipientUid
    */
   public function removeAll($recipientUid)
   {
       $this->_db->update("DELETE FROM "
           . " notification WHERE recipientUid = %d"
           , $recipientUid);
   }
}
```

Long pulling



How it works

Client application (browser) sends a request with event recipient id (here is the user, registered on the page) and current state (the displayed number of unread notification) to the server via HTTP. It creates an Apache process, which repeatedly checks DB until the state is changed in there. When the state eventually changed, the client gets the server response and sends next request to the server.

Implementation

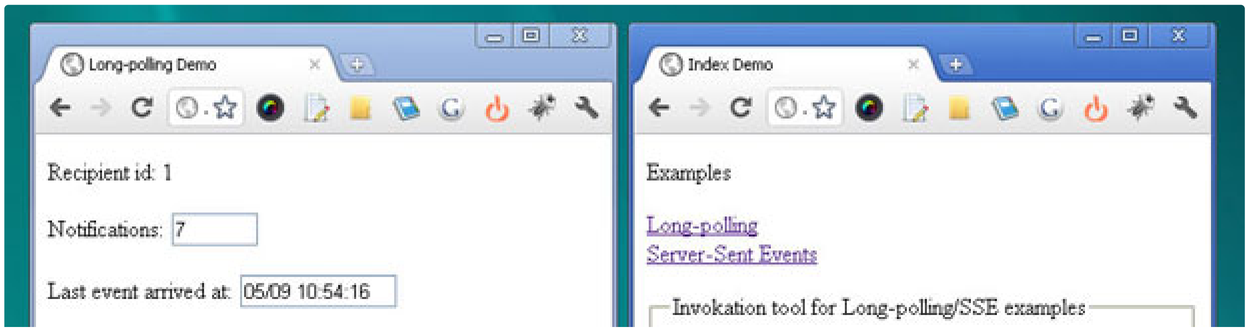
Client side contains simple HTML with two input fields to show response data (updated number of unread notifications and time of the response event). JS module sends recipient user id and current state (unread notification number) to the server as XMLHttpRequest. To make possible cross-domain communication, we use JSONP and that means the handler must of the public scope.

```
...
<p>Recipient id: <?= $recipientUid ?></p>
<p>Notifications: <input id="notificationNum" size="4" name="some" value="<?= $displaye
<p>Last event arrived at: <input id="time" size="12" name="some" value="0" /></p>

<script type="text/javascript">

(function( $ ) {

var UID = <?= $recipientUid ?>;

$.NotifierLongPolling = (function() {
    var _stateNode = $('#notificationNum'), _timeNode = $('#time');
    return {
        onMessage : function(data) {
            _stateNode.val(data.updatedNotificationNum);
            _timeNode.val(data.time);
            setTimeout($.NotifierLongPolling.send, 3000);
        },
        send : function() {
            $.ajax({
                    'url': 'server.php',
                    'type': 'POST',
                    'dataType': 'jsonp',
                    'jsonpCallback': '$.NotifierLongPolling.onMessage',
                    'data': 'recipientUid=' + UID + '%26displayedNotificationNum='
                        + _stateNode.val()
            });
        }
    }
}());

// Document is ready
$(document).bind('ready.app', function() {
   setTimeout($.NotifierLongPolling.send, 40);
});

})( jQuery );


</script>
```

Server waits 3 second than check if the updated state matches the given one. If the state has changed in the DB, the server responds otherwise it repeats the cycle.

```
//...
$recipientUid = (int)$_REQUEST["recipientUid"];
$displayedNotificationNum = (int)$_REQUEST["displayedNotificationNum"];
$secCount = 0;

do {
    sleep(IDLE_TIME);
    $updatedNotificationNum = $model->fetchNumberByRecipientUid($recipientUid);
} while ($updatedNotificationNum == $displayedNotificationNum);

header("HTTP/1.0 200");
printf ('%s({"time" : "%s", "updatedNotificationNum" : "%d"});'
    , $_REQUEST["callback"], date('d/m H:i:s'), $updatedNotificationNum);
```

Client side receives new state from the server, displays and sends to the server the new state to repeat the workflow.



Recipient id: 1

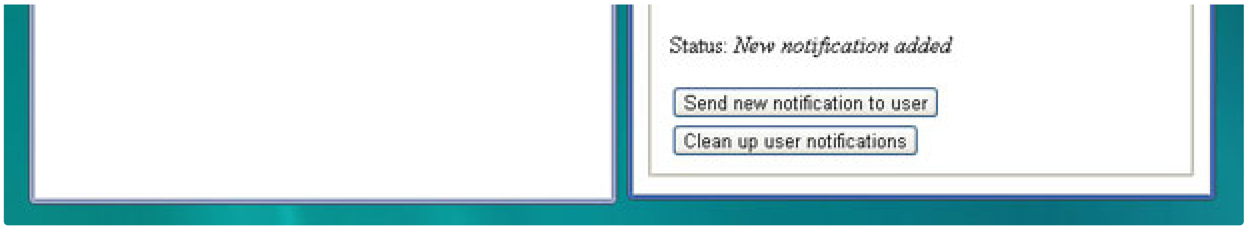
Notifications: 7

Last event arrived at: 05/09 10:54:16

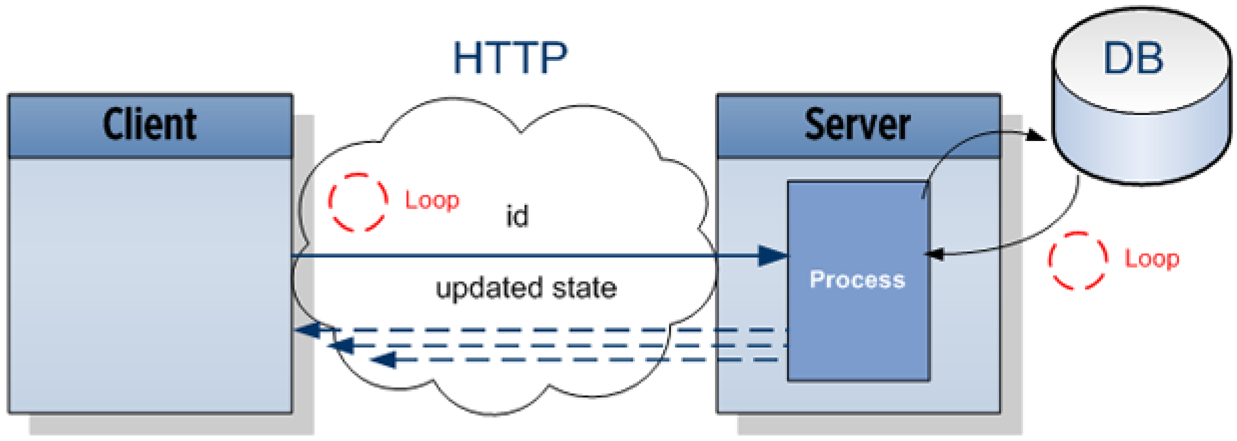
Examples

Long-polling
Server-Sent Events

Invokation tool for Long-polling/SSE examples


Status: New notification added
Send new notification to user
Clean up user notifications

Server-Sent Events



How it works

Client (browser) sends a request to the server via HTTP. It creates a process, which fetches latest state in the DB and responds back. Client gets server response and in 3 seconds sends next request to the server.

Implementation

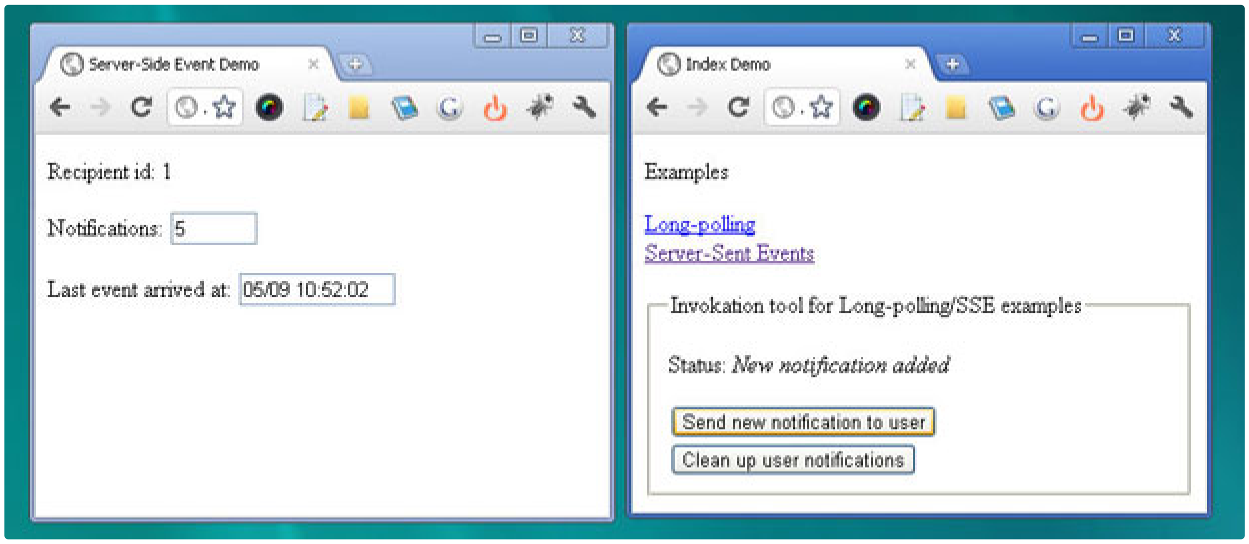
HTML on client-size has again two input fields to show response data. JS module opens EventSource and passes though the connection recipient user id.

```
...
<p>Recipient id: <?= $recipientUid ?></p>
<p>Notifications: <input id="notificationNum" size="4" name="some" value="<?= $display
<p>Last event arrived at: <input id="time" size="12" name="some" value="0" /></p>

<script type="text/javascript">

(function( $ ) {

var UID = <?= $recipientUid ?>;

NotifierSSE = (function() {
    var _stateNode = $('#notificationNum'),
        _timeNode = $('#time'),
        _src,
        _handler = {
        onMessage : function(event) {
             var data = JSON.parse(event.data);
            _stateNode.val(data.updatedNotificationNum);
            _timeNode.val(data.time);
        }
    };
    return {
        init : function () {
        _src = new EventSource("server.php?recipientUid=" + UID);
        _src.addEventListener('message', _handler.onMessage, false);
        }
    }
}());


// Document is ready
$(document).bind('ready.app', function() {
   setTimeout(NotifierSSE.init, 40);
});

})( jQuery );

</script>
```

Server responds into the data stream with the last state (unread notification number) regarding the recipient user id.

```
//...
header('Content-Type: text/event-stream');
header('Cache-Control: no-cache'); // recommended to prevent caching of event data.

$recipientUid = (int)$_REQUEST["recipientUid"];

function send($updatedNotificationNum)
{
    printf ("id: %s\n\n", PROC_ID);
    printf ('data: {"time" : "%s", "updatedNotificationNum" : "%d"}' . "\n\n"
        ,date('d/m H:i:s') , $updatedNotificationNum);
    ob_flush();
    flush();
}

while (true) {
    send($model->fetchNumberByRecipientUid($recipientUid));
    sleep(IDLE_TIME);
}
//...
```

Client gets onMessage event handler invoked, the data displayed. Browser will repeat request every 3 second unless you close the connection (close method). You can change delay between requests by passing into the data stream on the server “retry: <time in milliseconds>\n”.


Server-Side Event Demo

Recipient id: 1

Notifications: 5

Last event arrived at: 05/09 10:52:02

Index Demo

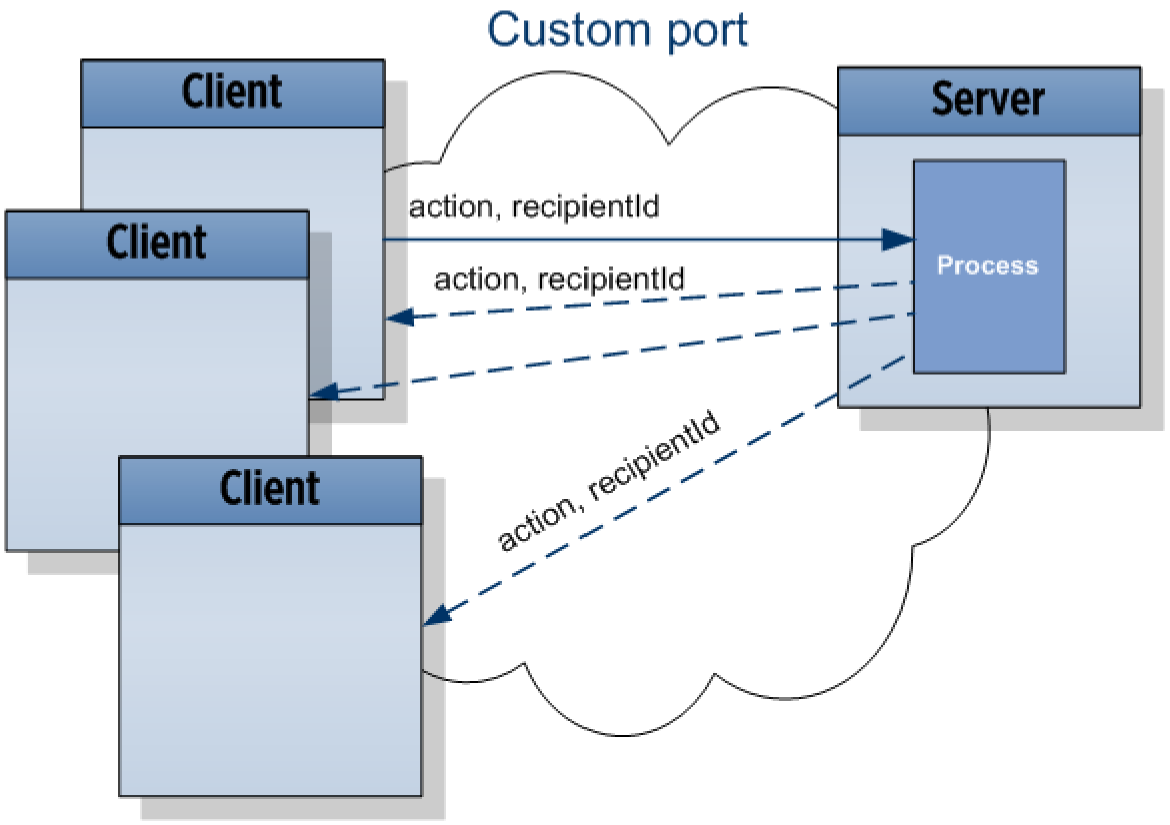
Examples

Long-polling
Server-Sent Events

Invokation tool for Long-polling/SSE examples

Status: *New notification added*

WebSockets



How it works

Client notifies web-socket server (EventMachine) of an event, giving ids of recipients. The server immediately notifies all the active clients (subscribed to that type of event). Clients process event when given recipient Id matches the client's one.

Implementation

Here we need an Event Machine supporting WebSockets. The easiest way, I see, is to deploy WaterSpout Server . You write your own controller (notification_controller.php) based on locke_controller.php.

Client opens WebSocket connection and subscribes a handler for events on /notification/updates post. Now let's add a button in the HTML which sends on /notification/presents recipient user and action ids. That will cause broadcast message on the all open connections. So every active client receives the notification. The client event handler checks if the recipient user id matches client's logged in user id and if so increment unread notification number.

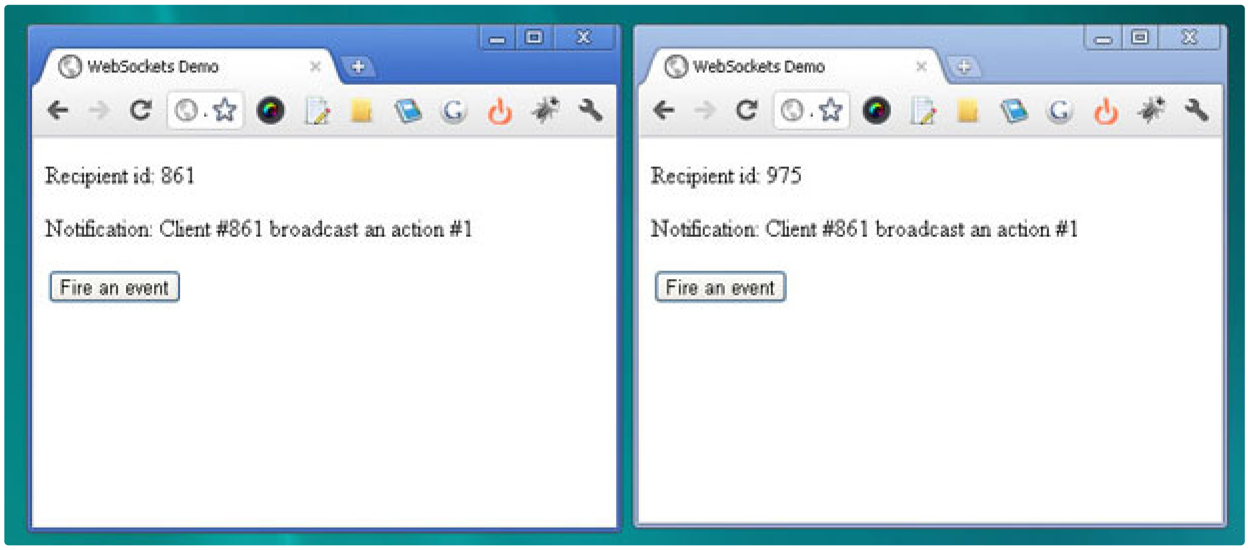
```
<p>Recipient id: <?= $recipientUid ?></p>
    <p>Notification: <span id="display"></span></p>
    <button id="test">Fire an event</button>

<script>
    realtime = new realtimeComm(window.location.host + ":20001");
    realtime.addListener('/notification/updates', function(response) {
        $('#display').html('Client #' + response.data.recipientUid + ' broadcast an act
    });

    $('#test').bind('click', this, function(e){
        e.preventDefault();

        realtime.send('/notification/presence', {
            'actionId': 1,
            'recipientUid': <?= $recipientUid ?>
        }, function() {});

    });

</script>
```

Working a bit on the controller, you can make the system responds exclusively to the client of the given recipient user id.

You've probably noticed the example working perfect on Chrome, but for Firefox 6.0 it switches for long-polling. It announced that Firefox 6 supports WebSockets, but you ain't going to find WebSocket object defined. Well, let's change WebSocket availability check in the top of realtime.js with following:

```
  if (!window.WebSocket) {
              window.WebSocket = window.MozWebSocket ? window.MozWebSocket : undefined;
   }
```

Oops, it doesn't work anyway. Firefox 6 supports hybi-07 specification version, when Chrome and WaterSpout – hybi-00 (hixie-76). So, you will hardly able the connection through reverse-proxies and gateways. Besides, the old specification was disabled in Firefox and Opera due to security issues. The only browsers now supporting the latest WebSocket specification are betas of Firefox 7/8 and Chrome 14. As for EventMachine server implementations, I found none.



Recipient id: 861

Notification: Client #861 broadcast an action #1

Recipient id: 975

Notification: Client #861 broadcast an action #1

Conclusion

Now you can download the package with the example here.

Previous
Review services getting conso...

Next
Support Request Tool like Go...

14 Comments **Dmitry Sheiko's Web Development Blog** **Disqus' Privacy Policy**

1 **Login**

Favorite 9 Tweet Share **Sort by Best**

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

johncongdon • 7 years ago

I think your explanation of SSE is a bit off. SSE is "real-time" like websockets, there is not a 3 second delay between messages.

I think what you are referring to is the fact that in the case that the SSE connection is closed, it will automatically try to reconnect in approximately 3 seconds.

3 ^ | ˅ • Reply • Share ›

Tiago ➦ johncongdon • 2 years ago

Thanks for pointing out this, now everything makes sense!

^ | ˅ • Reply • Share ›

Daniel • 8 years ago

You should be careful using the term "real-time". It's not true to call WebSockets "True real-time" only because there's no additional latency caused by connection initiation. Even WebSockets cannot guarantee real time by definition. By your definition any TCP connection would be a real time connection which is not true. In real-time systems you have to guarantee the data transmission within a defined time period. "true real-time" can be implemented using WebRTC

1 ^ | ˅ • Reply • Share ›

Stefano • 5 years ago • edited

I'm trying to run the demo and i get this error on Chrome
WebSocket connection to 'wss://mysite.com:20001/core/handshake' failed: WebSocket opening handshake timed out
anyone knows why?

^ | ˅ • Reply • Share ›

Roman Romanchuk • 5 years ago

Just unexpectedly found this on first page of google results looking on long pooling with Undertow.
Say hello to everybody around ;)

^ | ˅ • Reply • Share ›

Kelvin Muthomi • 6 years ago

Nice article! Clarified some of the grey areas I had over the three. If anyone needs help with long-polling just reply here and I will lend a 'code'.

^ | ˅ • Reply • Share ›

Bijesh Samani ➦ Kelvin Muthomi • 5 years ago • edited

Hi Kelvin,
Is there any best approach than long poling

^ | ⌄ • Reply • Share ›

Zubair Saleem ➦ Kelvin Muthomi • 6 years ago

Hi There Can you please help me out i have to complete my fyp and can't able to resolve real time notification problem..

Thanks in advance

^ | ⌄ • Reply • Share ›

Дмитрй Ярочин • 6 years ago

im developing a mobile chat application tht uses xmpp to do push notifications i want to know if using web sockets is a better option, in terms of bandwith consuption, and also as more and more pple start using my app with the server scale bettter with websockets or does xmpp handle this better

^ | ⌄ • Reply • Share ›

satish • 7 years ago

I am trying to develop a web application that requires sending periodic updates to clients. I figured SSE is a good fit for my requirement. But I am facing a problem on my server-side, where i cannot get the server to send the data as event-stream. My server configuration uses Apache as the Front-End, mainly to server static content and redirects dynamic requests to a Tomcat server, using the mod_jk connector. Eventhough i change to content-type to text/event-stream on my tomcat server and push the data into the output stream, it does not make it to the client through the apache. However, if i bypass the apache server and send the request directly to tomcat, the data is sent to the client as event-stream. Is there any configuration in apache/modjk to handle SSE correctly? Note that i have a limitation as far as the server-side architecture is concerned and i cannot use any other application/webservers other than apache and tomcat, interacting in the model described above. Please let me know if there is any way to resolve this problem in apache.

Thanks, Satish.

^ | ⌄ • Reply • Share ›

Suresh Kumar ➦ satish • 4 years ago

Try adding "\n\n" at the end of your message.

^ | ⌄ • Reply • Share ›

john philips • 7 years ago

Hi i am new to php. i have downloaded and paste the waterspout server folder in xampp's htdocs, then how to execute websockets demo.

^ | ⌄ • Reply • Share ›

shots_fired • 10 years ago

This is a nice article. I have tried to implement long polling as a fallback when web socket is not around, and i was looking for an article to enlighten me up. this one did it thanks.

Reply • Share ›

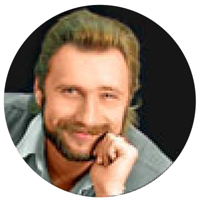
David Brear • 10 years ago

Very good write-up. I had heard about long-polling but had never implemented it nor did I know exactly how it worked. I was looking for an alternative to using web-sockets because of their current need for latest browsers. Thanks for the explanation!

Reply • Share ›

Subscribe **Add Disqus to your siteAdd DisqusAdd** **Do Not Sell My Data**

WHO'S THE DUDE?



Dmitry Sheiko is a web-developer living and working in Frankfurt am Main, DE

ELSEWHERE

Twitter Codepen
LinkedIn Slideshare
Github Stackoverflow

CREDITS

material.io/tools/icons
simpleicons.org