HACK SPARROW
Captain of the Internets. Since 1865.

Node.js UDP Server and Client Example

Posted on September 24th, 2012 under Node.js
Tags: UDP

Here is a quick tutorial on setting up a UDP server and client in Node.js. For all things UDP in Node.js, you will need to use the `dgram` library, so read it up well and good.

UDP Server

Here is a simple example of a UDP server.

```
var PORT = 33333;
var HOST = '127.0.0.1';

var dgram = require('dgram');
var server = dgram.createSocket('udp4');

server.on('listening', function () {
    var address = server.address();
    console.log('UDP Server listening on ' + address.
});

server.on('message', function (message, remote) {
    console.log(remote.address + ':' + remote.port +'

});

server.bind(PORT, HOST);
```

Things to Note

1. HOST is optional in `server.bind()`. If omitted, it will be listening on `0.0.0.0`, which might be what you want in some cases.
2. The `message` event is fired, when a UDP packet arrives destined for this server.
3. The `listening` event is fired, when the server has initialized and all ready to receive UDP packets.
4. `dgram.createSocket()` can either accept 'udp4' or 'udp6'. The former uses IPv4, the later uses IPv6.

UDP Client

And here is a simple UDP client.

```
var PORT = 33333;
var HOST = '127.0.0.1';

var dgram = require('dgram');
var message = new Buffer('My KungFu is Good!');

var client = dgram.createSocket('udp4');
client.send(message, 0, message.length, PORT, HOST, fu
    if (err) throw err;
    console.log('UDP message sent to ' + HOST +':'+ P
    client.close();
});
```

Things to Note

1. `client.send()` requires a proper Node.js Buffer object, not a plain string or number.
2. The second parameter 0, of `client.send()` is the offset in the buffer where the UDP packet starts.
3. The third parameter `message.length`, is the number of bytes we want to send from the offset in the buffer. In our case, the offset is 0, and the length is `message.length` (16 bytes), which is quite tiny and the whole buffer can be sent in a single UDP packet. This might always not be the case. For large buffers, you will need to iterate over the buffer and send it in smaller chunks of UDP packets.
4. Exceeding the allowed packet size will not result in any error. The packet will be silently dropped. That's just the nature of UDP.
5. The `err` object in the callback function of `client.send()` is going to be only of the DNS lookup kind.

6. Make sure the HOST / IP address is in conformance with the IP version you use, else your packets will not reach the destination.

There you go! A quick primer on getting started with UDP in Node.js.

Related to this post

i. TCP Socket Programming in Node.js
ii. Express.js HTTPS Server Client Example
iii. Using MySQL with Node.js
iv. JavaScript Email Validation
v. Using Redis with Node.js
vi. vhost in Express.js
vii. Node.js Restart on File Change
viii. Node.js get IP Address
ix. Web speed is slow but Torrent speed is fast – Solution!
x. How to install Node.js on Webfaction

G+1 3 Tweet

11 Responses to “Node.js UDP Server and Client Example”

Cameron says:
May 9, 2016 at 9:34 pm
WoNDeR: You can enhance the code to send a UDP message out, and the responder(s) would respond directly to the sending IP (TCP) which could confirm receipt. I would randomize the reaction (time to response) or send back the reply via HTTP GET or POST.

« Previous 1 2

Make a Comment

Your Name

Your E-Mail

Submit Comment

Follow @hacksparrow

Recent Comments

Captain on JavaScript .bind() vs .apply() and .call()
Kevin Kim on JavaScript .bind() vs .apply() and .call()
tiagojdferreira on Using Node.js to download files
Pedro Vagner on Difference between spawn and exec of Node.js child_process
Conor Doyle on The For Loop vs the For Each Loop in JavaScript
Jennifer on Node.js Module – exports vs module.exports
Gordon on Python: difference between list and tuple
kdmrobot on Mongoskin Tutorial with Examples
Captain on TCP Socket Programming in Node.js
eliot on TCP Socket Programming in Node.js

Tags

Apply Aptana array Call closure CURL database Error Express Express.js Firefox git GitHub Google Gzip Homebrew HTML HTTP HTTPS JavaScript javascript array javascript for loop Linux list lol Mac Microsoft MongoDB Mozilla MySQL mysqldump node.js NoSQL npm Object pagination PHP Python Redis Regex Sitemap string tuple Ubuntu Wordpress