

Martin Alemajoh
Arizona State University
Assignment 2

1. Understanding HTTP

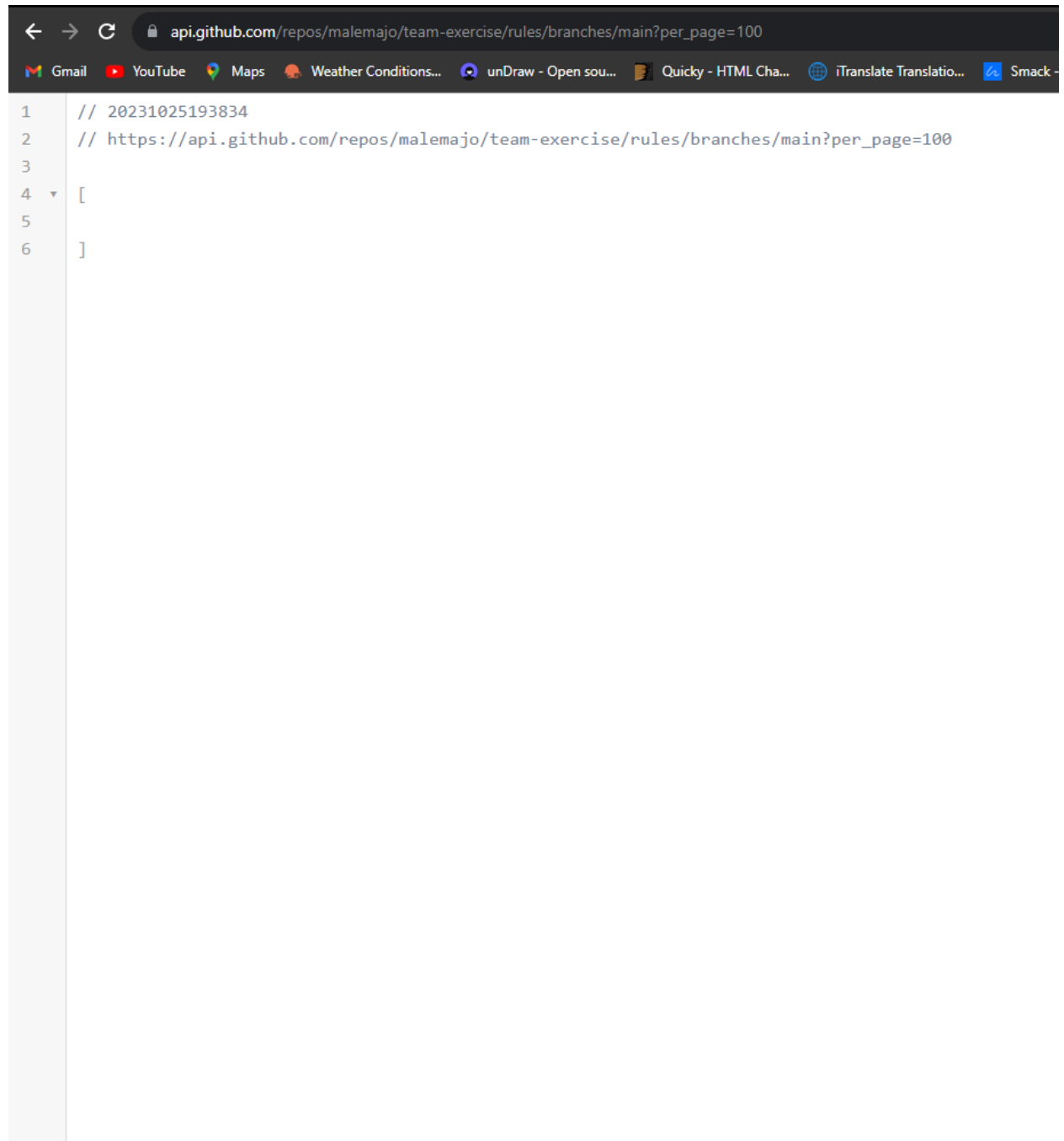
<https://api.github.com/repos/malemajoh/team-exercise/commits>

```
1 // 20231025192209
2 // https://api.github.com/repos/malemajoh/team-exercise/commits
3
4 [
5   {
6     "sha": "71b9074bb90b10b6eee0127c38a1655ceea63cfc",
7     "node_id": "C_kwDOKhtDbNoAKDcxYjkwNzRiYjkwYjEwYjZlZWUwMTI3YzN4YTE2NTVjZWVhNjNjZmM",
8     "commit": {
9       "author": {
10         "name": "Justin Salas",
11         "email": "justindsalas@gmail.com",
12         "date": "2023-10-19T02:31:37Z"
13       },
14       "committer": {
15         "name": "GitHub",
16         "email": "noreply@github.com",
17         "date": "2023-10-19T02:31:37Z"
18       },
19       "message": "Update PULLREQUEST.md\nAdded my sentence for other team",
20       "tree": {
21         "sha": "f01e3d94b9b27b3cebb868ef28bbacd3d6f7c1d",
22         "url": "https://api.github.com/repos/malemajoh/team-exercise/git/trees/f01e3d94b9b27b3cebb868ef28bbacd3d6f7c1d"
23       },
24       "url": "https://api.github.com/repos/malemajoh/team-exercise/git/commits/71b9074bb90b10b6eee0127c38a1655ceea63cfc",
25       "comment_count": 0,
26       "verification": {
27         "verified": true,
28         "reason": "valid",
29         "signature": "-----BEGIN PGP SIGNATURE-----
\n\nwsBcBAABCAAQBQJlM3UjCRBK7hj40v3rIwAAA9cIAJec0EgctODwLKSIGzK05J4E\nw50HXlGMCg3LW24h7t4nahqUwixs5S1TPvmROIuUw8kQrAUqCqHrz5Q9+fMbYma\n6C90iDZP6uQqeoYwQTBdRoHzrHyAWsCMehU+YK+2Uo5Puyp66bk/AThnXH5N\nT620CvXTNs6iTRnE/Kc2jHk3i144YHtmXJ01JuA1QGNuq/TZubjcC+oWpPiMnrk=\n=n=1PmI\n\n-----END PGP
\n\npayload": "tree f01e3d94b9b27b3cebb868ef28bbacd3d6f7c1d\nparent 3928f47a5da6d93774eca13c4e12001573d495a1\nauthor Justin Salas <justindsal
sentence for other team"
30
31     }
32   },
33   "url": "https://api.github.com/repos/malemajoh/team-exercise/commits/71b9074bb90b10b6eee0127c38a1655ceea63cfc",
34   "html_url": "https://github.com/malemajoh/team-exercise/commit/71b9074bb90b10b6eee0127c38a1655ceea63cfc",
35   "comments_url": "https://api.github.com/repos/malemajoh/team-exercise/commits/71b9074bb90b10b6eee0127c38a1655ceea63cfc/comments",
```

<https://api.github.com/users?since=1>

```
1 // 20231025193530
2 // https://api.github.com/users?since=1
3
4 [
5   {
6     "login": "defunkt",
7     "id": 2,
8     "node_id": "MDQ6VXNlcjI=",
9     "avatar_url": "https://avatars.githubusercontent.com/u/2?v=4",
10    "gravatar_id": "",
11    "url": "https://api.github.com/users/defunkt",
12    "html_url": "https://github.com/defunkt",
13    "followers_url": "https://api.github.com/users/defunkt/followers",
14    "following_url": "https://api.github.com/users/defunkt/following{/other_user}",
15    "gists_url": "https://api.github.com/users/defunkt/gists{/gist_id}",
16    "starred_url": "https://api.github.com/users/defunkt/starred{/owner}/{repo}",
17    "subscriptions_url": "https://api.github.com/users/defunkt/subscriptions",
18    "organizations_url": "https://api.github.com/users/defunkt/orgs",
19    "repos_url": "https://api.github.com/users/defunkt/repos",
20    "events_url": "https://api.github.com/users/defunkt/events{/privacy}",
21    "received_events_url": "https://api.github.com/users/defunkt/received_events",
22    "type": "User",
23    "site_admin": false
24  },
25  {
26    "login": "pjhyett",
27    "id": 3,
28    "node_id": "MDQ6VXNlcjM=",
29    "avatar_url": "https://avatars.githubusercontent.com/u/3?v=4",
30    "gravatar_id": "",
31    "url": "https://api.github.com/users/pjhyett",
32    "html_url": "https://github.com/pjhyett",
33    "followers_url": "https://api.github.com/users/pjhyett/followers",
34    "following_url": "https://api.github.com/users/pjhyett/following{/other_user}",
35    "gists_url": "https://api.github.com/users/pjhyett/gists{/gist_id}",
36    "starred_url": "https://api.github.com/users/pjhyett/starred{/owner}/{repo}",
37    "subscriptions_url": "https://api.github.com/users/pjhyett/subscriptions",
38    "organizations_url": "https://api.github.com/users/pjhyett/orgs",
```

https://api.github.com/repos/malemajo/team-exercise/rules/branches/main?per_page=100



The screenshot shows a web browser window with a REST client interface. The address bar displays the URL: `api.github.com/repos/malemajo/team-exercise/rules/branches/main?per_page=100`. Below the address bar, there is a toolbar with various icons for different services like Gmail, YouTube, Maps, etc. The main area of the browser shows a REST client interface with a list of requests on the left and the details of the selected request on the right. The selected request is a GET request to the same URL as in the address bar. The response body is empty, indicated by a single line in the response section.

```
1 // 20231025193834
2 // https://api.github.com/repos/malemajo/team-exercise/rules/branches/main?per_page=100
3
4 [
5
6 ]
```

List Users Endpoint:

URL: <https://api.github.com/users?since=1>

Method: GET

Description: This endpoint lists all GitHub users in the order that they signed up on GitHub.

Query Parameters:

since: This is an integer parameter that allows you to specify the user ID to start the listing from. In this case, since=1 indicates that the listing should start from the user with ID 1.

This endpoint will return a paginated list of users starting from the specified user ID. Each user object in the response will include information such as the user's login, ID, node_id, avatar_url, gravatar_id, url, html_url, followers_url, following_url, gists_url, starred_url, subscriptions_url, organizations_url, repos_url, events_url, received_events_url, type, site_admin, and more.

Get Branch Endpoint:

URL: https://api.github.com/repos/malemajo/team-exercise/rules/branches/main?per_page=100

Method: GET

Description: This endpoint fetches a single branch from a repository.

URL Parameters:

owner: The owner of the repository. In this case, it's malemajo.

repo: The name of the repository. In this case, it's team-exercise.

branch: The name of the branch. In this case, it's main.

Query Parameters:

per_page: This is an integer parameter that specifies the number of items per page. In this case, per_page=100 indicates that 100 items should be returned per page. However, this parameter may not have an effect on this particular endpoint as it's designed to fetch a single branch.

This endpoint will return information about the specified branch, including the name of the branch, commit details, and the protected status of the branch.

Stateless Communication:

In stateless communication, each request from a client to a server must contain all the information the server needs to fulfill the request.

There's no retention of data between transactions. Each request is processed based solely on the information that comes with it, without any awareness of previous interactions.

HTTP, without any additional configurations, is an example of a stateless protocol where each request is processed independently.

Stateless architectures are often simpler and more scalable due to the lack of inter-dependency between requests.

Stateful Communication:

In stateful communication, data from previous interactions can be stored and used to inform current transactions. This is often managed through sessions or other mechanisms.

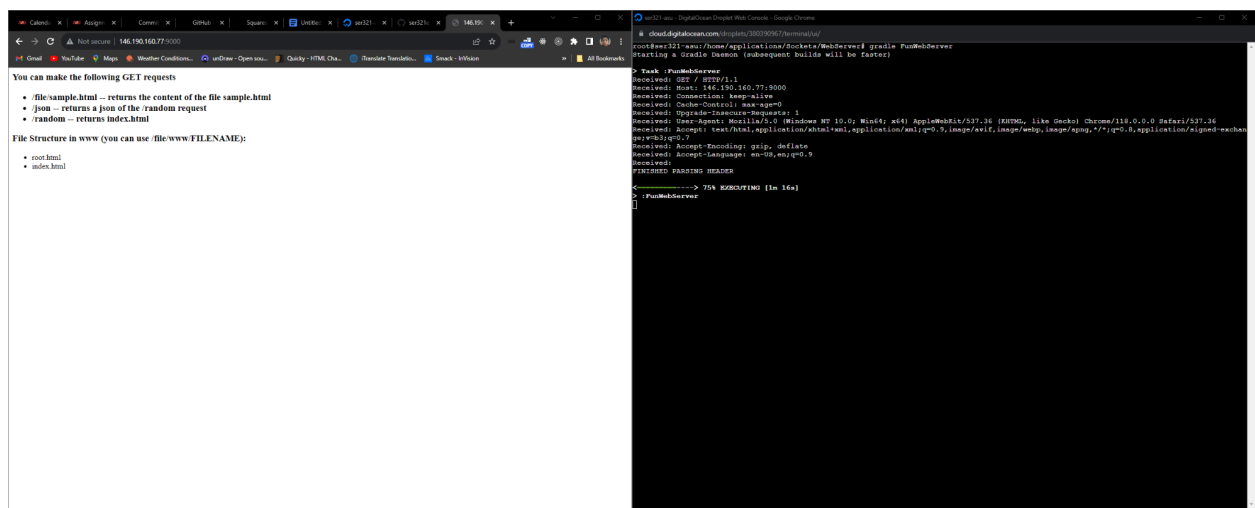
There's a retention of data between transactions, which allows for a continuity of experience.

For example, a server might remember a client's previous actions or settings, providing a more personalized or continuous experience across interactions.

Protocols like TCP are stateful as they maintain a connection and a state of communication between a client and a server, tracking interactions over time.

Stateful architectures can provide richer interactions and experiences, but they can also be more complex and potentially less scalable due to the overhead of managing state.

Running A simple Java WebServer



The screenshot shows a web browser on the left and a terminal window on the right. The browser displays a simple web page with the following content:

```
You can make the following GET requests
```

- /file/sample.html – returns the content of the file sample.html
- /json – returns a json of the random request
- /random – returns index.html

File Structure in www (you can use /file/www/FILENAME):

- root.html
- index.html

The terminal window on the right shows the command to run the web server:

```
root@ec2:~# java -jar /home/ec2-user/.gradle/caches/modules-2/files-2.1/com.googlecode.json-simple/json-simple/1.1.1/json-simple-1.1.1-jar-with-dependencies.jar
```

The terminal output shows the server starting and listening on port 8080. It receives a GET request for /file/sample.html and returns the content of the file. The output is as follows:

```
Received: GET / HTTP/1.1
Received: Host: 174.129.140.77:8080
Received: Connection: keep-alive
Received: Cache-Control: max-age=0
Received: Upgrade-Insecure-Requests: 1
Received: User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.5553.9 Safari/537.36
Received: Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/svg+xml,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Received: Accept-Encoding: gzip, deflate
Received: Accept-Language: en-US,en;q=0.9
Received:
FINISHED PARSE HEADERS
<-----> 754 REQUEST [in 16s]
> /file/sample.html
1
```

The screenshot shows the Wireshark interface with a packet list on the left and a packet details pane on the right. The packet list shows a series of TCP and HTTP packets between 192.168.1.147 and 146.190.160.77. The selected packet (No. 1740) is an HTTP 200 OK response. The packet details pane shows the structure of the response, including the status line '200 OK (text/html)' and the body content, which is a plain HTML document. The packet bytes pane shows the raw data of the response.

| No. | Time | Source | Destination | Protocol | Length | Info |
|------|-----------|----------------|----------------|----------|--------|--|
| 882 | 13.376677 | 192.168.1.147 | 146.190.160.77 | TCP | 54 | 49505 → 9000 [FIN, ACK] Seq=1 Ack=1 Win=514 Len=0 |
| 883 | 13.376980 | 192.168.1.147 | 146.190.160.77 | TCP | 66 | 49556 → 9000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM |
| 884 | 13.377176 | 192.168.1.147 | 146.190.160.77 | TCP | 66 | 49557 → 9000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM |
| 886 | 13.454144 | 146.190.160.77 | 192.168.1.147 | TCP | 66 | 9000 → 49556 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1432 SACK_PERM WS=64 |
| 887 | 13.454240 | 192.168.1.147 | 146.190.160.77 | TCP | 54 | 49556 → 9000 [ACK] Seq=1 Ack=1 Win=131584 Len=0 |
| 888 | 13.454422 | 192.168.1.147 | 146.190.160.77 | HTTP | 514 | GET / HTTP/1.1 |
| 889 | 13.455229 | 146.190.160.77 | 192.168.1.147 | TCP | 66 | 9000 → 49557 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1432 SACK_PERM WS=64 |
| 890 | 13.455288 | 192.168.1.147 | 146.190.160.77 | TCP | 54 | 49557 → 9000 [ACK] Seq=1 Ack=1 Win=131584 Len=0 |
| 892 | 13.457837 | 146.190.160.77 | 192.168.1.147 | TCP | 90 | 9000 → 49505 [PSH, ACK] Seq=1 Ack=2 Win=1004 Len=36 |
| 893 | 13.457837 | 146.190.160.77 | 192.168.1.147 | TCP | 54 | 9000 → 49505 [FIN, ACK] Seq=37 Ack=2 Win=1004 Len=0 |
| 894 | 13.457888 | 192.168.1.147 | 146.190.160.77 | TCP | 54 | 49505 → 9000 [RST, ACK] Seq=2 Ack=37 Win=0 Len=0 |
| 896 | 13.529989 | 146.190.160.77 | 192.168.1.147 | TCP | 54 | 9000 → 49556 [ACK] Seq=1 Ack=461 Win=64128 Len=0 |
| 897 | 13.548199 | 146.190.160.77 | 192.168.1.147 | TCP | 603 | 9000 → 49556 [PSH, ACK] Seq=1 Ack=461 Win=64128 Len=549 [TCP segment of a reassembled PDU] |
| 898 | 13.549002 | 146.190.160.77 | 192.168.1.147 | HTTP | 54 | HTTP/1.1 200 OK (text/html) |
| 899 | 13.549038 | 192.168.1.147 | 146.190.160.77 | TCP | 54 | 49556 → 9000 [ACK] Seq=461 Ack=551 Win=131072 Len=0 |
| 900 | 13.549098 | 192.168.1.147 | 146.190.160.77 | TCP | 54 | 49556 → 9000 [FIN, ACK] Seq=461 Ack=551 Win=131072 Len=0 |
| 943 | 13.618453 | 146.190.160.77 | 192.168.1.147 | TCP | 54 | 9000 → 49556 [ACK] Seq=551 Ack=462 Win=64128 Len=0 |
| 1737 | 25.814068 | 192.168.1.147 | 146.190.160.77 | TCP | 66 | 49564 → 9000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM |
| 1738 | 25.814674 | 192.168.1.147 | 146.190.160.77 | HTTP | 514 | GET / HTTP/1.1 |
| 1739 | 25.891878 | 146.190.160.77 | 192.168.1.147 | TCP | 66 | 9000 → 49564 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1432 SACK_PERM WS=64 |
| 1740 | 25.891878 | 146.190.160.77 | 192.168.1.147 | TCP | 603 | 9000 → 49557 [PSH, ACK] Seq=1 Ack=461 Win=64128 Len=549 [TCP segment of a reassembled PDU] |
| 1741 | 25.891878 | 146.190.160.77 | 192.168.1.147 | TCP | 54 | 9000 → 49557 [ACK] Seq=1 Ack=461 Win=64128 Len=0 |
| 1742 | 25.891878 | 146.190.160.77 | 192.168.1.147 | HTTP | 54 | HTTP/1.1 200 OK (text/html) |
| 1743 | 25.892003 | 192.168.1.147 | 146.190.160.77 | TCP | 54 | 49557 → 9000 [ACK] Seq=461 Ack=550 Win=131072 Len=0 |
| 1744 | 25.892058 | 192.168.1.147 | 146.190.160.77 | TCP | 54 | 49564 → 9000 [ACK] Seq=1 Ack=1 Win=131584 Len=0 |

Packet details for selected packet (No. 1740):

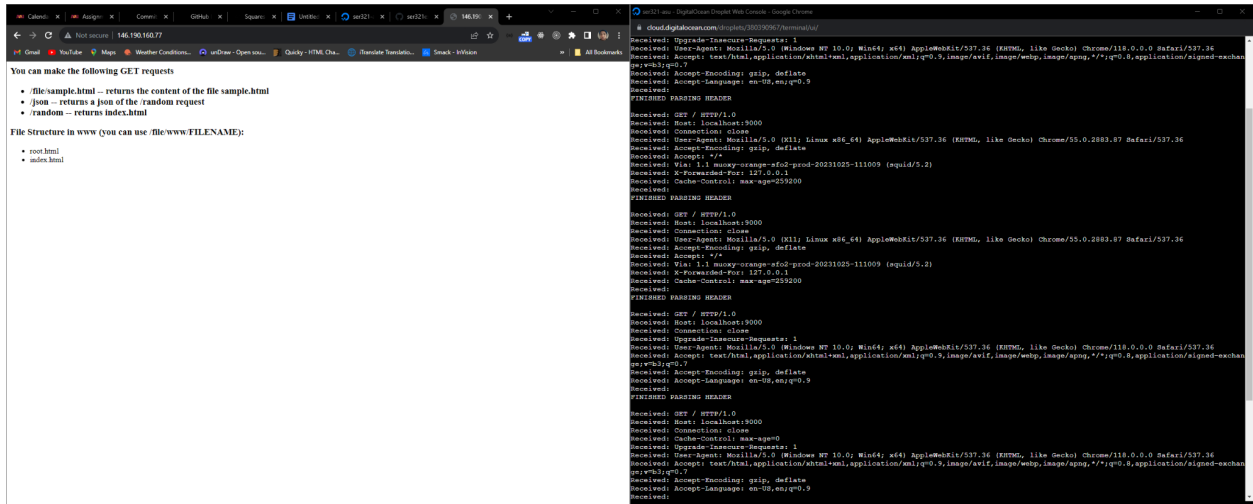
- Window size scaling factor: 64
- Checksum: 0x859e [unverified]
- Checksum Status: Unverified
- Urgent Pointer: 0
- [Timestamps]
- [2 Reassembled TCP Segments (549 bytes): #1740(549), #1742(0)]
- Hypertext Transfer Protocol
 - Line-based text data: text/html (17 lines)


```
<html>\n
<head>\n
<link rel="shortcut icon" href="data:image/x-icon;" type="image/x-icon">\n
</head>\n
<body>\n
<h3>You can make the following GET requests</h3>\n
<ul>\n
<li><a href="/file/sample.html">-- returns the content of the file sample.html</li>\n
<li><a href="/json">-- returns a json of the /random request</li>\n
<li><a href="/random">-- returns index.html</li>\n
</ul>\n
<h3>File Structure in www (you can use /file/www/FILENAME):</h3>\n
<ul>\n
<li>root.html</li>
<li>index.html</li>
</ul>\n
</body>\n
</html>
```

Packet bytes pane shows the raw data of the response, including the status line and the body content.

- ip.addr == 146.190.160.77
I chose this filter because it helps to narrow down the traffic captured by Wireshark to only show traffic between my local machine and the remote server I am interested in.
- A random image is displayed.
- / -> 200 OK
/json => 200 OK
/file/sample.html => 404 NOT FOUND
- 200 => Success Response
404 => Resource not found
- Yes. It returns the plain HTML text
- HTTPS encrypts the data
9000. No HTTP typically listens on 80 by default.
- 49793

1. <http://146.190.160.77/>
2. It is different. It uses port 80
3. It is HTTP because no SSL certificate exists on the server.



Wi-Fi

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ip.addr == 146.190.160.77

| No. | Time | Source | Destination | Protocol | Length | Info |
|--------|-------------|----------------|----------------|----------|--------|--|
| 2296.. | 2944.185206 | 192.168.1.147 | 146.190.160.77 | TCP | 66 | 51890 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM |
| 2296.. | 2944.185444 | 192.168.1.147 | 146.190.160.77 | TCP | 66 | 51891 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM |
| 2297.. | 2944.258452 | 146.190.160.77 | 192.168.1.147 | TCP | 66 | 80 → 51891 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1432 SACK_PERM WS=64 |
| 2297.. | 2944.258579 | 192.168.1.147 | 146.190.160.77 | TCP | 54 | 51891 → 80 [ACK] Seq=1 Ack=1 Win=131584 Len=0 |
| 2297.. | 2944.258742 | 192.168.1.147 | 146.190.160.77 | HTTP | 509 | GET / HTTP/1.1 |
| 2297.. | 2944.264473 | 146.190.160.77 | 192.168.1.147 | TCP | 66 | 80 → 51890 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1432 SACK_PERM WS=64 |
| 2297.. | 2944.264644 | 192.168.1.147 | 146.190.160.77 | TCP | 54 | 51890 → 80 [ACK] Seq=1 Ack=1 Win=131584 Len=0 |
| 2297.. | 2944.331199 | 146.190.160.77 | 192.168.1.147 | TCP | 54 | 80 → 51891 [ACK] Seq=1 Ack=456 Win=64128 Len=0 |
| 2297.. | 2944.335062 | 146.190.160.77 | 192.168.1.147 | TCP | 54 | [TCP Dup ACK 229750#1] 80 → 51891 [ACK] Seq=1 Ack=456 Win=64128 Len=0 |
| 2297.. | 2944.351888 | 146.190.160.77 | 192.168.1.147 | HTTP | 564 | HTTP/1.1 200 OK (text/html) |
| 2297.. | 2944.407479 | 192.168.1.147 | 146.190.160.77 | TCP | 54 | 51891 → 80 [ACK] Seq=456 Ack=511 Win=131072 Len=0 |
| 2298.. | 2944.655868 | 146.190.160.77 | 192.168.1.147 | TCP | 564 | [TCP Spurious Retransmission] 80 → 51891 [PSH, ACK] Seq=1 Ack=456 Win=64128 Len=510 |
| 2298.. | 2944.655928 | 192.168.1.147 | 146.190.160.77 | TCP | 66 | [TCP Dup ACK 229763#1] 51891 → 80 [ACK] Seq=456 Ack=511 Win=131072 Len=0 SLE=1 SRE=511 |
| 2333.. | 2989.266388 | 192.168.1.147 | 146.190.160.77 | TCP | 55 | [TCP Keep-Alive] 51890 → 80 [ACK] Seq=0 Ack=1 Win=131584 Len=1 |
| 2333.. | 2989.340892 | 146.190.160.77 | 192.168.1.147 | TCP | 66 | [TCP Window Update] 80 → 51890 [ACK] Seq=1 Ack=1 Win=64256 Len=0 SLE=0 SRE=1 |
| 2333.. | 2989.354360 | 192.168.1.147 | 146.190.160.77 | TCP | 55 | [TCP Keep-Alive] 51891 → 80 [ACK] Seq=455 Ack=511 Win=131072 Len=1 |
| 2333.. | 2989.425313 | 146.190.160.77 | 192.168.1.147 | TCP | 66 | [TCP Keep-Alive ACK] 80 → 51891 [ACK] Seq=511 Ack=456 Win=64128 Len=0 SLE=455 SRE=456 |
| 2343.. | 3004.406106 | 146.190.160.77 | 192.168.1.147 | TCP | 54 | 80 → 51890 [FIN, ACK] Seq=1 Ack=1 Win=64256 Len=0 |
| 2343.. | 3004.406168 | 192.168.1.147 | 146.190.160.77 | TCP | 54 | 51890 → 80 [ACK] Seq=1 Ack=2 Win=131584 Len=0 |
| 2355.. | 3019.362869 | 146.190.160.77 | 192.168.1.147 | TCP | 54 | 80 → 51891 [FIN, ACK] Seq=511 Ack=456 Win=64128 Len=0 |
| 2355.. | 3019.363030 | 192.168.1.147 | 146.190.160.77 | TCP | 54 | 51891 → 80 [ACK] Seq=456 Ack=512 Win=131072 Len=0 |
| 2355.. | 3020.253993 | 192.168.1.147 | 146.190.160.77 | TCP | 54 | 51890 → 80 [FIN, ACK] Seq=1 Ack=2 Win=131584 Len=0 |
| 2356.. | 3020.254037 | 192.168.1.147 | 146.190.160.77 | TCP | 54 | 51891 → 80 [FIN, ACK] Seq=456 Ack=512 Win=131072 Len=0 |
| 2356.. | 3020.324893 | 146.190.160.77 | 192.168.1.147 | TCP | 54 | 80 → 51891 [ACK] Seq=512 Ack=457 Win=64128 Len=0 |
| 2356.. | 3020.325606 | 146.190.160.77 | 192.168.1.147 | TCP | 54 | 80 → 51890 [ACK] Seq=2 Ack=2 Win=64256 Len=0 |

> Ethernet II, Src: CloudNet_9c:55:87 (30:03:c8:9c:55:87), Dst: SercommP_54:42:dc (14:7

> Internet Protocol Version 4, Src: 192.168.1.147, Dst: 146.190.160.77

> Transmission Control Protocol, Src Port: 51890, Dst Port: 80, Seq: 0, Ack: 1, Len: 1

Source Port: 51890

Destination Port: 80

[Stream index: 806]

[Conversation completeness: Complete, WITH_DATA (31)]

[TCP Segment Len: 1]

Sequence Number: 0 (relative sequence number)

Sequence Number (raw): 1329873069

[Next Sequence Number: 1 (relative sequence number)]

Acknowledgment Number: 1 (relative ack number)

Acknowledgment number (raw): 1894774109

0101 = Header Length: 20 bytes (5)

> Flags: 0x010 (ACK)

Window: 514

[Calculated window size: 131584]

[Window size scaling factor: 256]

Checksum: 0xf748 [unverified]

[Checksum Status: Unverified]

Urgent Pointer: 0

> [Timestamps]

> [SEQ/ACK analysis]

TCP payload (1 byte)

> Data (1 byte)

0000 14 7d 05 54 42 dc 30 03 c8 9c 55 87 08 00 45 00 .}.TB.0. .U...E.

0010 00 29 fc c8 40 00 00 06 08 bf c0 a8 01 93 92 be .).@.....

0020 a0 ad ca b2 00 50 4f 44 40 ad 70 ef f5 5d 50 10 .M...POD @p..]P.

0030 02 02 f7 48 00 00 00H...

Source Port (tcp.srcport), 2 bytes

Packets: 240938 • Displayed: 585 (0.2%)

Profile: Default

Weather Request Protocol:

Request

URL: host:PORT/weather?city=CITY_NAME&country=COUNTRY_CODE

Path: weather

Body Parameters:

city (String) - required

country (String) - required

Response

OK Response (200)

Example:

```
{  
  "city": "London",  
  "country": "UK",  
  "temperature": "15°C",  
  "condition": "Cloudy"  
}
```

HTTP response status codes

200 – OK, the weather data could be fetched

400 - Bad Request, either city or country parameter is missing

404 - Not Found, weather data for the specified location could not be found

Translation Request Protocol:

Request

URL: host:PORT/translate?text=TEXT&target=TARGET_LANGUAGE

Path: translate

Body Parameters:

text (String) - required

target (String) - required

Response

OK Response (200)

Example:

```
{  
  "original_text": "hello",  
  "translated_text": "hola",  
  "target_language": "es"  
}
```

HTTP response status codes

200 – OK, the text was translated successfully

400 - Bad Request, either text or target parameter is missing

404 - Not Found, translation service did not respond or target language is not supported