Active class 2: Understanding the Application Layer

Activity 1: Understanding the Application Layer using HTTP

1. Why do we need application layer protocols?

They enable applications to communicate over a network by defining rules for data exchange.

2. Examples of application layer protocols:

- HTTP: Transfers web pages.
- SMTP: Sends emails.
- FTP: Transfers files.
- DNS: Resolves domain names to IP addresses.
- IMAP: Retrieves emails from a server.

1. Key Features of HTTP

- Stateless
- Request-response model
- Uses TCP
- Supports various methods like GET, POST, PUT, DELETE

Role Play (We wrote a script for this which is as follows)

1. Client 1(Pranika):

"Client 1 initiating TCP connection with the Server."

"Client 1 sending HTTP GET request for /page1.html."

2. Server (Archit):

"Server accepting TCP connection from Client 1."

"Server responding with HTTP 200 OK and /page1.html content."

3. Client 2(Jasveena):

"Client 2 initiating TCP connection with the Server."

"Client 2 sending HTTP GET request for /page2.html."

4. Server (Archit):

"Server accepting TCP connection from Client 2."

"Server responding with HTTP 200 OK and /page2.html content."

5. Client 3(Nadiyai):

"Client 3 initiating TCP connection with the Server."

"Client 3 sending HTTP GET request for /page3.html."

6. Server (Archit):

"Server accepting TCP connection from Client 3."

"Server responding with HTTP 200 OK and /page3.html content."

With Proxy Server (Jasveena as Proxy)

1. Client 1 (Pranika):

- "Client 1 initiating TCP connection with Proxy Server."
- "Client 1 sending HTTP GET request for /page1.html."

2. Proxy Server (Jasveena):

- "Proxy Server responding with cached HTTP 200 OK and /page1.html content."

3. Client 3 (Nadia):

- "Client 3 initiating TCP connection with Proxy Server."

· "Client 3 sending HTTP GET request for /page3.html."

4. Proxy Server (Jasveena):

· "Proxy Server forwarding request to Server."

5. Server (Archit):

· "Server responding with HTTP 200 OK and /page3.html content to Proxy Server."

· Proxy Server (Jasveena):

· "Proxy Server responding with HTTP 200 OK and /page3.html content to Client 3."

Activity 2: Analysing HTTP in Wireshark

Wireshark HTTP Analysis Summary

Sequence of HTTP Message Exchange:

Begins with GET requests from 192.168.1.102 to 104.97.189.98, followed by 200 OK responses.

Persistence/Non-persistence Connection:

Persistent connection (HTTP/1.1) with multiple requests over the same connection.

Details in HTTP GET Message:

Includes headers: Host, User-Agent, Accept, Accept-Encoding, Connection, Upgrade-Insecure-Requests.

Frame Details:

Ethernet II: MAC addresses.

IPv4: Source (192.168.1.102) and destination (104.97.189.98) IP addresses.

TCP: Source and destination ports, sequence numbers.

HTTP: Request/response details and headers.

HTTP Version Used by Browser:

HTTP/1.1

Response Message from Server:

HTTP/1.1 200 OK

HTTP Version on Server:

HTTP/1.1

IP Address of Computer:

192.168.1.102

IP Address of Server:

104.97.189.98

Status Code from Server:

200 OK

Wireshark HTTP Analysis Summary (Second Website)

Sequence of HTTP Message Exchange:

Like the previous capture, the sequence starts with GET requests from the client (192.168.1.102) to the server (47.246.42.236), followed by 200 OK responses.

Persistence/Non-persistence Connection:

Persistent connection (HTTP/1.1) is observed, with multiple requests using the same TCP connection.

Details in HTTP GET Message:

Headers in the GET message include Host, User-Agent, Accept, Accept-Encoding, Connection.

Frame Details:

Ethernet II: MAC addresses.

IPv4: Source (192.168.1.102) and destination (47.246.42.236) IP addresses.

TCP: Source and destination ports, sequence numbers.

HTTP: Request/response details and headers.

HTTP Version Used by Browser:

HTTP/1.1

Response Message from Server:

HTTP/1.1 200 OK

HTTP Version on Server:

HTTP/1.1

IP Address of Your Computer:

192.168.1.102

IP Address of the Server:

47.246.42.236

Status Code from Server:

200 OK

What happens when you use "https"? Can you analyse the responses?

HTTPS secures communication, making detailed analysis of HTTP content in Wireshark impossible without decryption.

Active class 3: DNS

Activity 1:

What is the core Internet function provided by DNS?

Answer: The core function of DNS (Domain Name System) is to translate human-readable domain names (e.g., www.example.com) into IP addresses (e.g., 192.0.2.1), allowing users to access websites using easy-to-remember names.

Why do we need DNS?

Answer: We need DNS to simplify the process of accessing websites and other resources on the Internet by using human-friendly domain names instead of numerical IP addresses, making the Internet more accessible and user-friendly.

What is the layer that DNS belongs to?

Answer: DNS operates at the Application layer (Layer 7) of the OSI model.

Do you think a single DNS server is enough to support the entire network? Justify your answer. Provide an alternate solution if we have any.

Answer: No, a single DNS server is not enough to support the entire network because it would create a single point of failure, lead to high latency, and be unable to handle the volume of DNS requests globally.

Alternate Solution: A distributed and hierarchical system of DNS servers is used globally, including root DNS servers, TLD (Top-Level Domain) servers, and authoritative DNS servers, to ensure redundancy, load balancing, and scalability.

Role Play

Client 1 Accesses the Webpage

Step 1: Client 1 Initiates the Request

Pranika (Client 1):

"I want to access deakin.edu.au. First, I need to find the IP address. I'll send a DNS query to the DNS server."

Step 2: Client 1 Queries the DNS Server

Pranika (Client 1):

"DNS server, what's the IP address for deakin.edu.au?"

Step 3: DNS Server Responds to Client 1

Archit (DNS Server):

"I don't have the IP address cached. Let me query the Root, TLD, and Authoritative DNS servers."

Archit (DNS Server):

"The IP address for deakin.edu.au is 203.0.113.5."

Step 4: Client 1 Sends HTTP Request to Web Server

Pranika (Client 1):

"Web server at IP 203.0.113.5, please send me the webpage for deakin.edu.au."

Step 5: Web Server Responds to Client 1

Jasveena (Web Server):

"Here is the webpage for deakin.edu.au."

Step 6: Client 1 Receives the Webpage

Pranika (Client 1):

"Thank you, I have received the webpage for deakin.edu.au."

Second Client Accesses the Same Webpage

Step 7: Client 2 Initiates the Request

Nadia (Client 2):

"I also want to access deakin.edu.au. I'll send a DNS query to the DNS server."

Step 8: Client 2 Queries the DNS Server

nadia (Client 2):

"DNS server, what's the IP address for deakin.edu.au?"

Step 9: DNS Server Responds to Client 2

Archit (DNS Server):

"I have the IP address cached. The IP address for deakin.edu.au is 203.0.113.5."

Step 10: Client 2 Sends HTTP Request to Web Server

nadia (Client 2):

"Web server at IP 203.0.113.5, please send me the webpage for deakin.edu.au."

Step 11: Web Server Responds to Client 2

Jasveena (Web Server):

"Here is the webpage for deakin.edu.au."

Step 12: Client 2 Receives the Webpage

nadia (Client 2):

“I have received the webpage for deakin.edu.au."

Activity 2:

1. Identify the IP address of deakin.edu.au using nslookup2

The IP addresses for deakin.edu.au are 128.184.204.21 and 128.184.20.21. 2

. Identify authoritative DNS servers for harvard.edu

- Authoritative DNS servers: a10-66.akam.net, a7-65.akam.net, a11-67.akam.net, a6-66.akam.net, a1-171.akam.net, a16-64.akam.net.
- Authoritative answers: IP addresses for these DNS servers are listed.

3.Trace DNS in Wireshark

- Empty the DNS cache:

Command for MacOS: sudo killall -HUP mDNSResponder

- Capture packets in Wireshark:

Analysis:

- DNS Query and Response Messages:
- Transport Layer Protocol: UDP
- DNS Query Destination Port: 53
- DNS Response Source Port: 53
- Query IP Address: 192.168.1.1
- DNS Query Type: Standard query (A)
- DNS Response: Contains the IP address 208.112.52.122

2. Local DNS Server

Command Used: scutil –dns

Overall answer

DNS Query Results for deakin.edu.au:

- Non-authoritative IP addresses: 128.184.204.21 and 128.184.20.21.

Authoritative DNS Servers for harvard.edu:

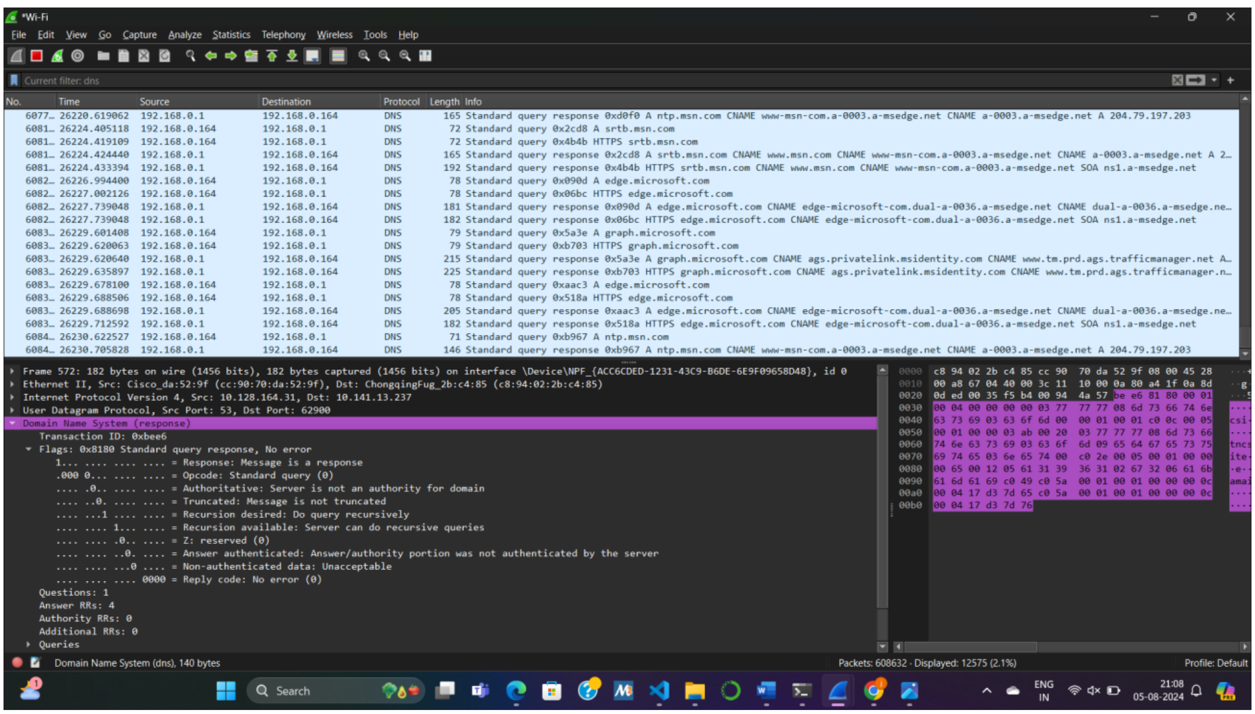
- Servers: a10-66.akam.net, a7-65.akam.net, a11-67.akam.net, a6-66.akam.net, a1-171.akam.net, a16-64.akam.net.
- IP addresses for these servers provided.

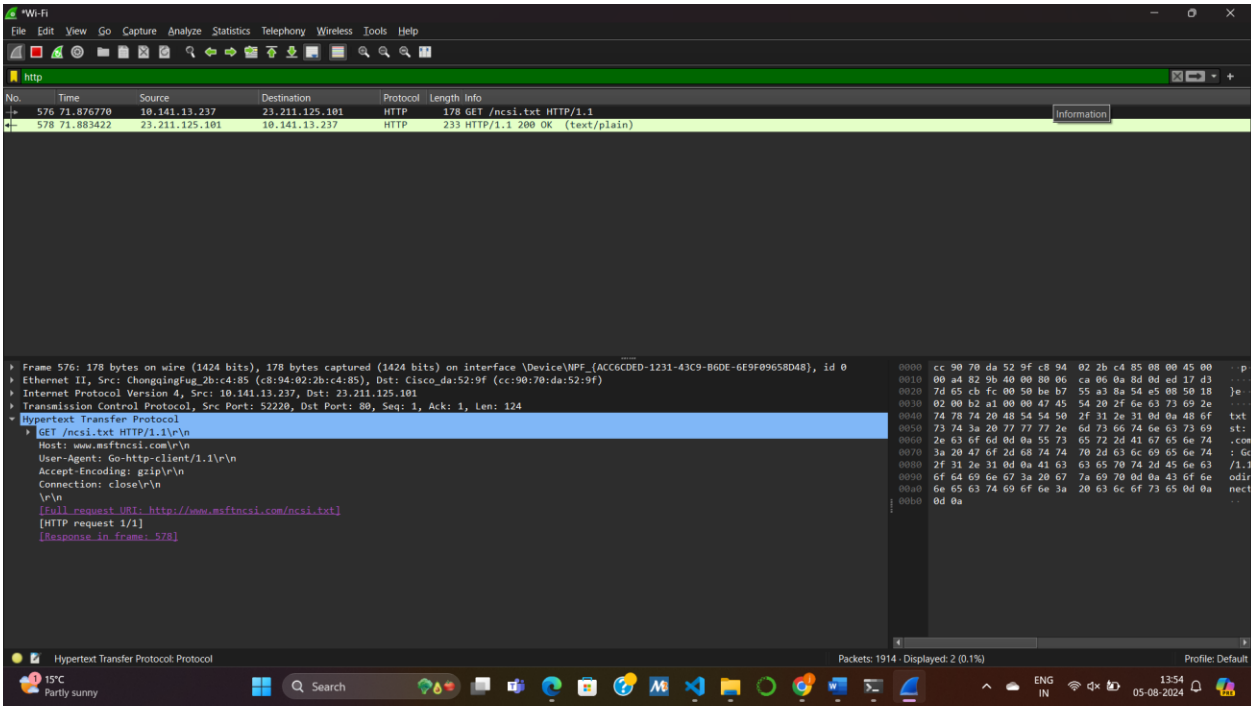
Wireshark Analysis:

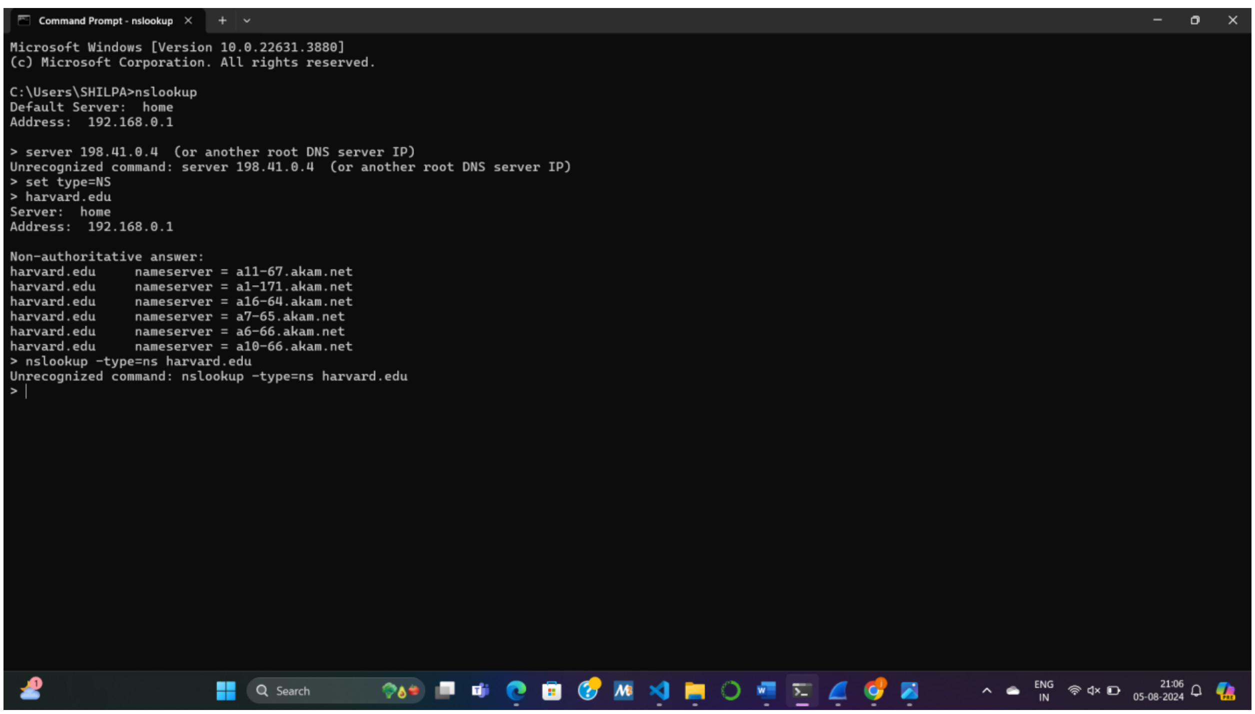
- Transport Layer Protocol: UDP.
- DNS Query Destination Port: 53.
- DNS Response Source Port: 53.
- Query IP Address: 192.168.1.1.
- Local DNS Server IP Address: 192.168.1.1.
- DNS Query Type: Standard query (A).
- DNS Response: Contains the IP address 208.112.52.122.

DNS Queries for Images:

Typically, browsers cache DNS responses. Therefore, multiple requests for resources on the same domain do not trigger additional DNS queries.







(this is a screenshot from my command prompt, but the IP address written is what we inferred from when we did from the group activity.)

EXERCISE 1

Explore different applications that are developed based on the client-server architecture and peer to peer architecture. Note down your answers before you continue.

Client-Server Architecture Applications:

1. **Web Browsing**: Web browsers (e.g., Chrome) access web servers (e.g., Apache) to view websites.
2. **Email**: Email clients (e.g., Outlook) communicate with email servers (e.g., Gmail) for sending and receiving messages.
3. **File Transfer**: FTP clients (e.g., FileZilla) upload/download files from FTP servers.
4. **Database Access**: Database tools (e.g., MySQL Workbench) interact with database servers (e.g., MySQL) for data queries.
5. **Remote Desktop**: Remote desktop apps (e.g., Microsoft Remote Desktop) connect to remote servers for control.

Peer-to-Peer (P2P) Architecture Applications:

1. **File Sharing**: Applications (e.g., BitTorrent) allow direct file sharing between peers.
2. **VoIP**: Services (e.g., Skype) enable direct voice/video calls between users.
3. **Messaging**: Apps (e.g., Signal) facilitate direct, encrypted messaging between users.
4. **Distributed Computing**: Projects use unused computing power from peers for research.

EXERCISE2:

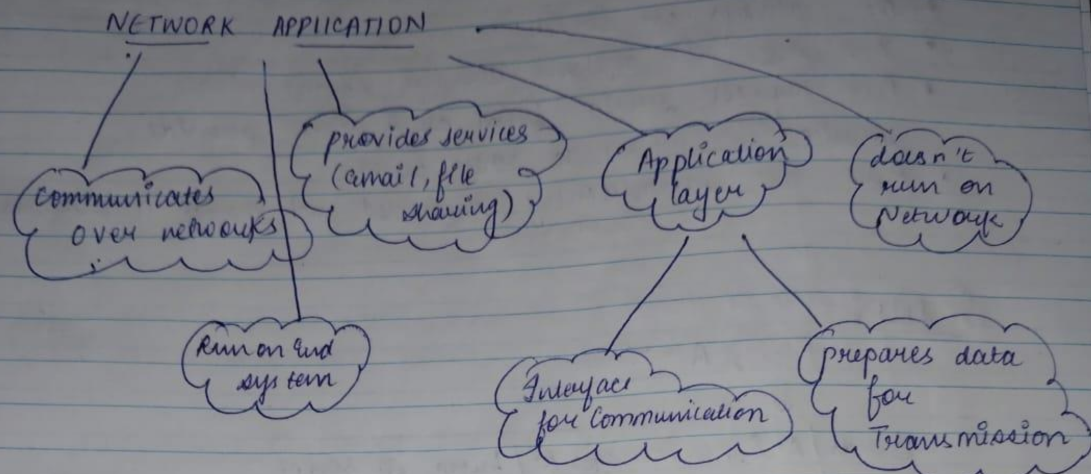
List down four network applications you use on daily basis and their application and transport protocols.

1. **Web Browsing**
   - **Application Protocol**: HTTP/HTTPS
   - **Transport Protocol**: TCP
2. **Email**
   - **Application Protocol**: SMTP/IMAP/POP3

- **Transport Protocol**: TCP

3. **File Transfer**
   - **Application Protocol**: FTP/SFTP
   - **Transport Protocol**: TCP
4. **VoIP Calls**
   - **Application Protocol**: SIP/RTP
   - **Transport Protocol**: UDP

APPLICATION LAYER



Today's network applications use two main architectures.
- the client - server architecture
- the peer-to-peer (P2P) architecture

Network Application Architectures :-

* Client - Server Architecture :-

Server :- * Always-on host
* Permanent IP address.

Clients :- * Contact and communicate with server
* May be intermittently connected.
* May have dynamic IP addresses.
* Do not communicate directly with other clients.
Examples : HTTP (web), IMAP (email), File transfer.

* Peer to Peer Architecture :-

- No always-on server
- Arbitrary end systems directly communicate.
- Peer request service from other peers, provide service in return to other peers.
- Peers are intermittently connected and change IP addresses.

Sockets :-

Process A

↓

Sends / Receives messages to / from its socket.

↓ (send message).

Socket - A

Analogy : Door

↓

Socket : B

(Analogy : Door)

↓ Recieve message

Process B

* Application Process
(eg: web app)

↓ uses

Socket

↓ Transfers messages

Transport layer
Protocol (TL)

↓ Transfers messages across sockets

TCP (connection)
UDP (connection-less)

\# The HTTP protocol uses messages to communicate.

- There are two types of HTTP messages.
  - Request Messages
  - Response Messages

INTERNET TRANSPORT PROTOCOLS SERVICES

TCP service :-

* "Reliable Transport" between sending and receiving process
* "Congestion control" throttle sender when network overloaded.
* "Connection-oriented" setup required between client and server processes
* "does not provide", timing, minimum throughout guarentee, security.

UDP service:

* "Unreliable data transfer" between sending and recieving process
* "Does not provide" reliability, flow control, congestion control, timing, throughput guarentee, security, or connection setup.

* Application Process
(eg: web app)

↓ uses

Socket

↓ Transfers messages

Transport layer
Protocol (TL)

↓ Transfers messages across sockets

TCP (connection)
UDP (connection-less)

The HTTP protocol uses messages to communicate.

· There are two types of HTTP messages.
  — Request Messages
  — Response Messages

☆ Issues with persistent and non-persistent HTTP :-

Non-persistent HTTP issues :-
- requires 2 RTT's per object
- Os overhead for each TCP connection
- Browsers often open multiple parallel TCP connections to fetch referenced object in parallel.

Persistent HTTP :-

- First version of HTTP
- server leaves connection open after sending response
- subsequent HTTP messages between same client/server sent over open connection.

•

Client		Server
TCP Handshake Establish TCP connection
TLS Handshake client/server Hello, certificate, key Exchange, change Cipher Spec.
HTTP Request / Response message (encrypted messages)
Time		Time

DNS Services and Structures :-

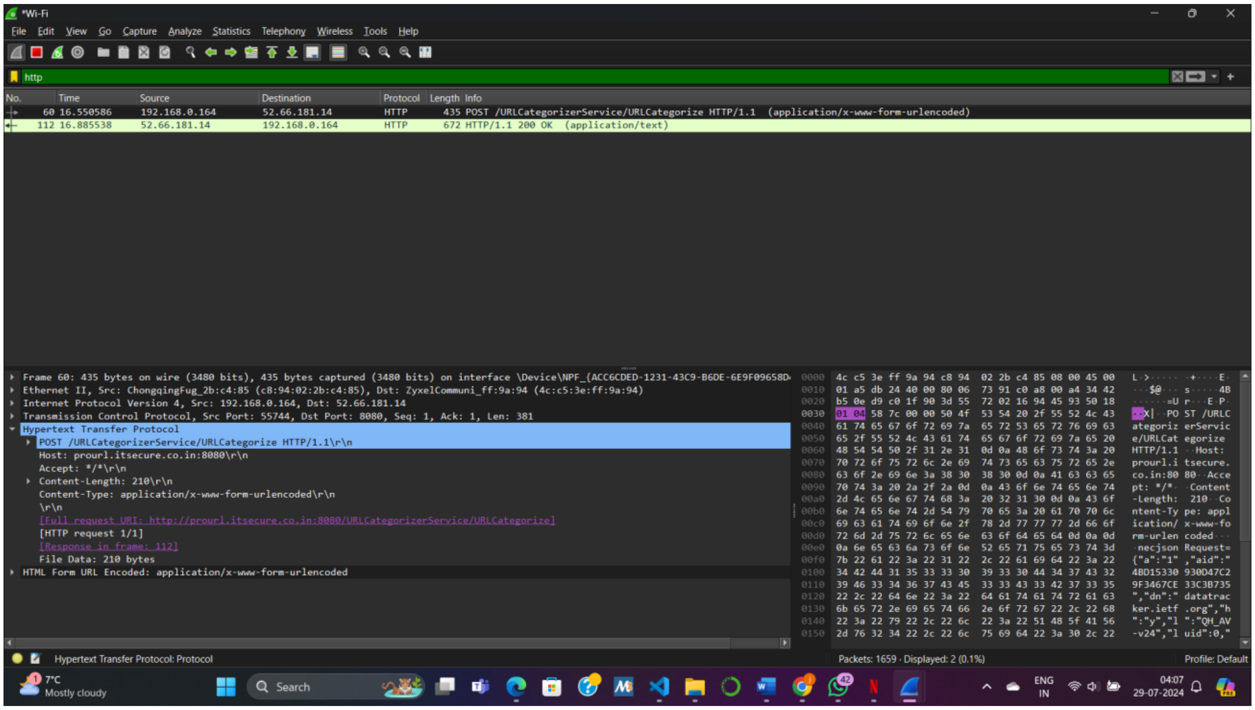
DNS service

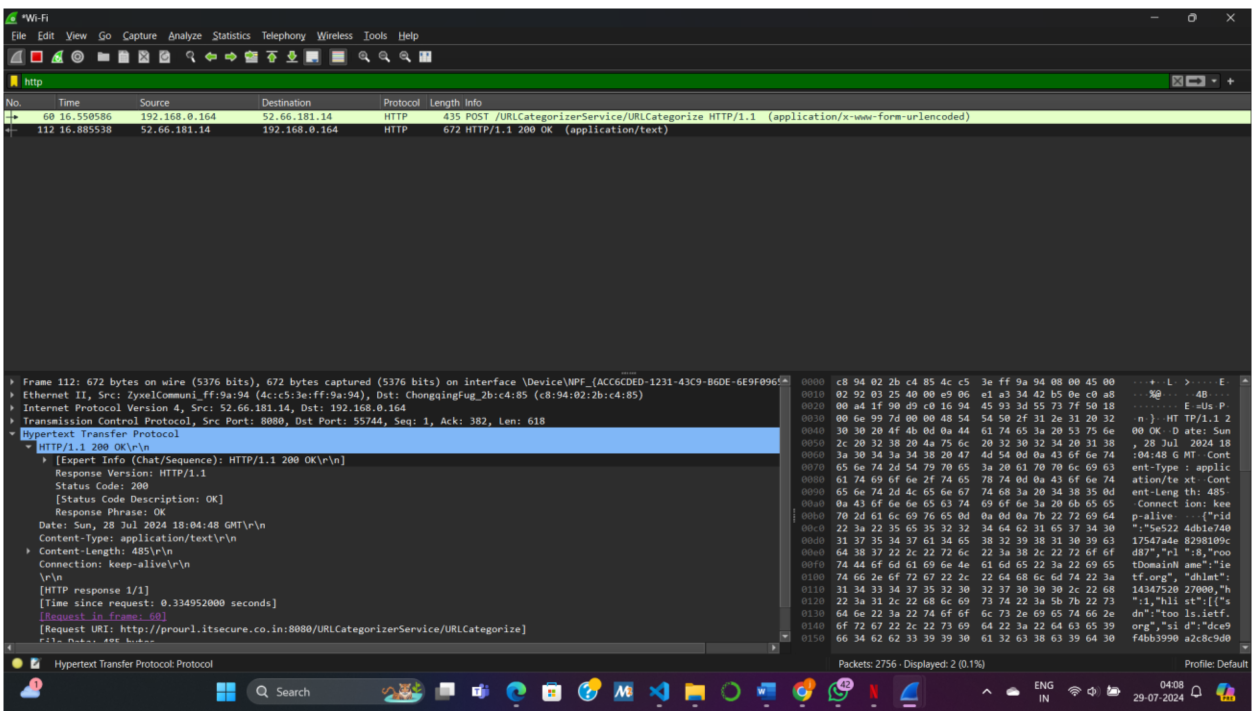
- Hostname to IP address translation
- Host aliasing
    - Canonical host name, alias names
- Mail server aliasing.
- Load distribution
- Replicate Web servers: many IP addresses correspond to one name.

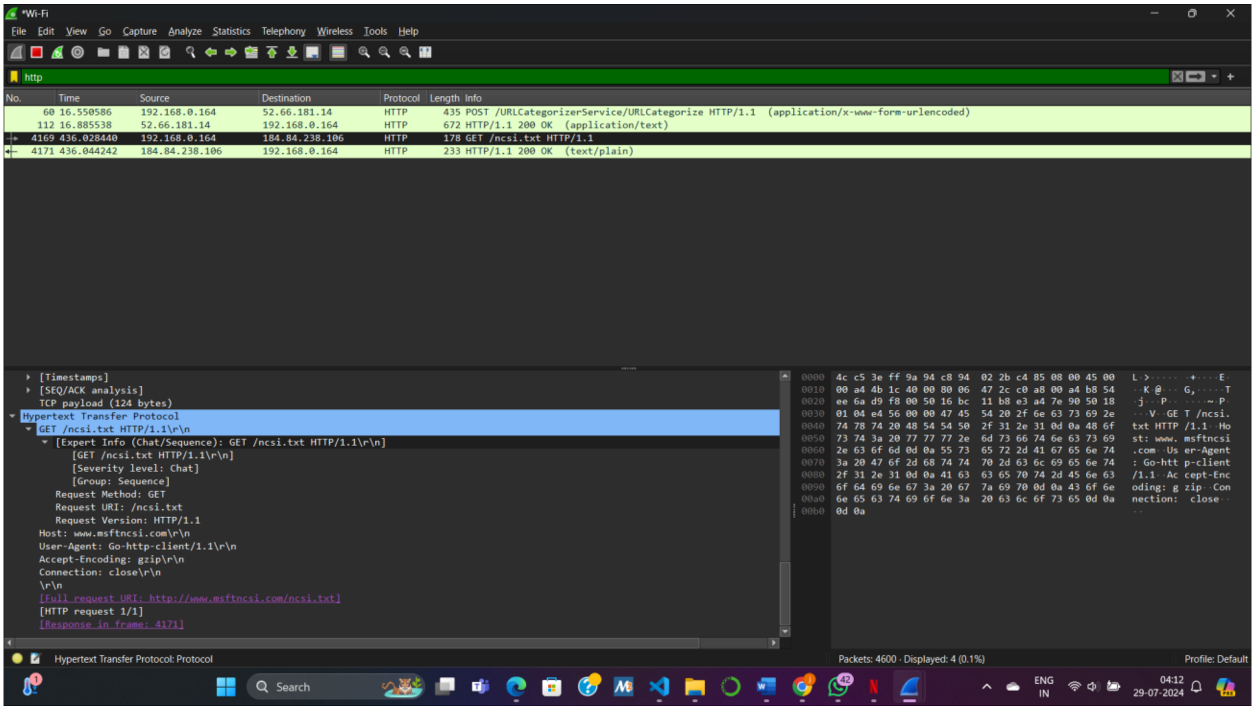
local DNS name Servers

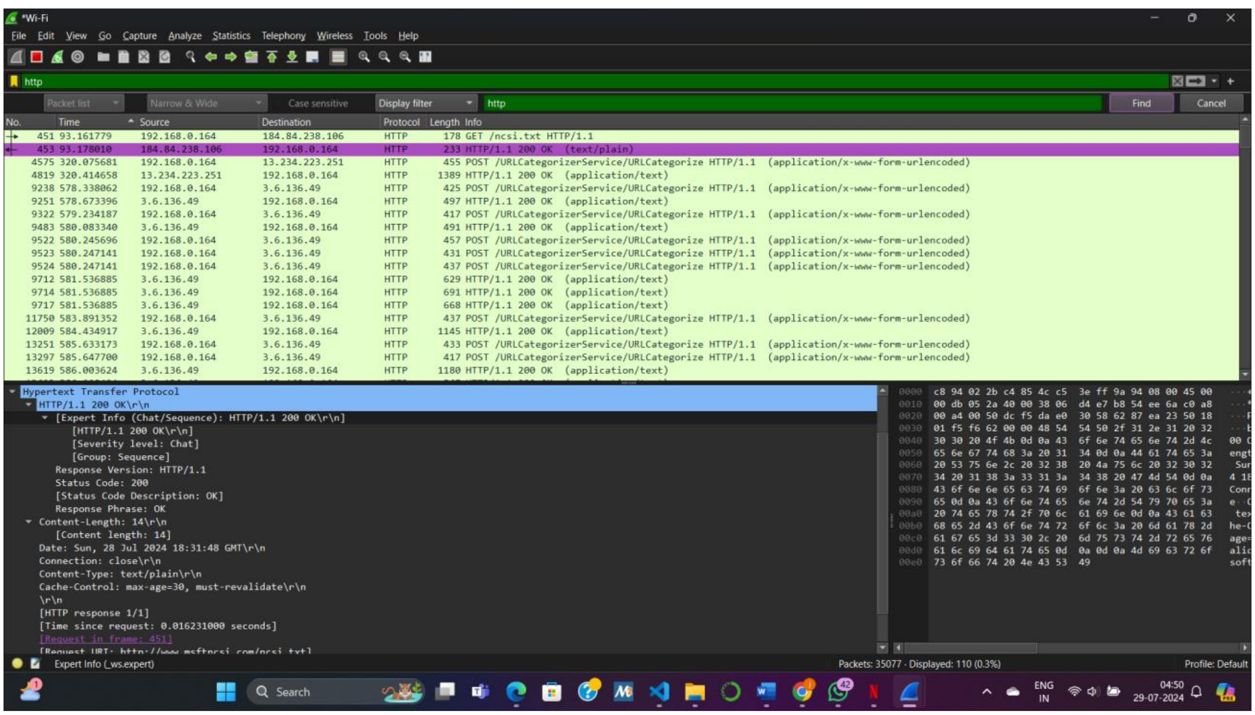
★ When host makes DNS query, it is sent to its local DNS server

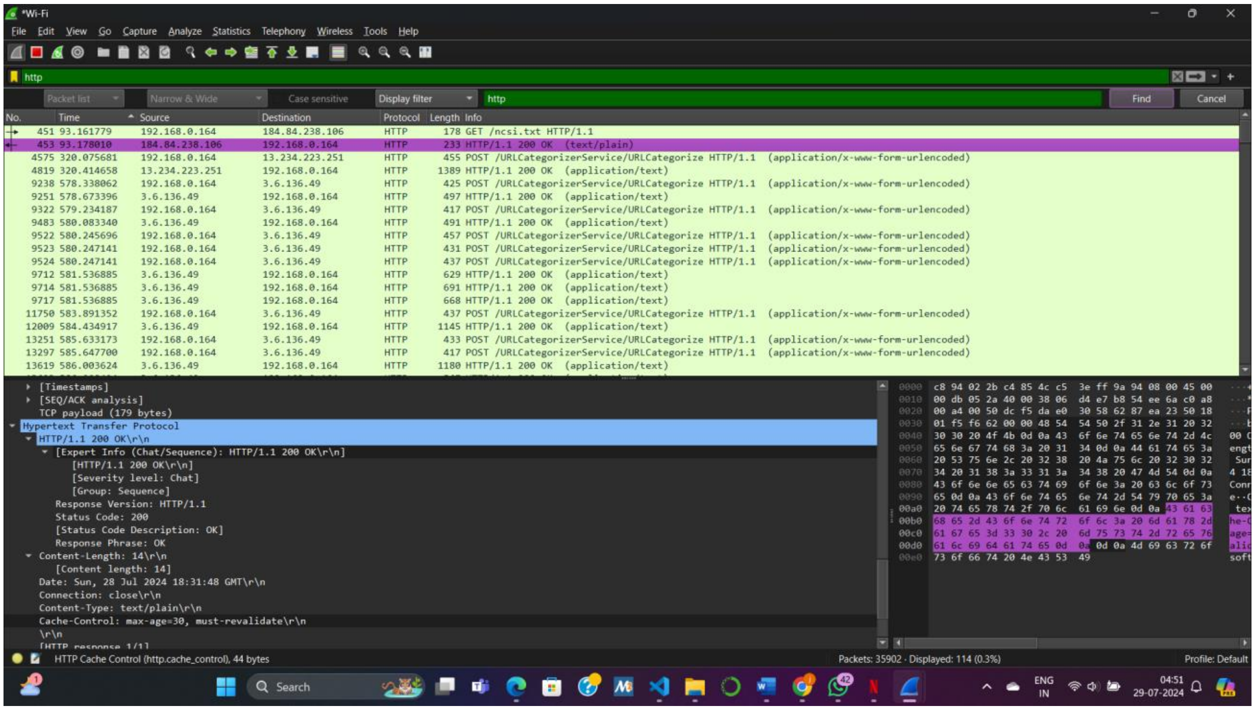
★ local DNS server doesn't strictly belong to hierarchy.


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http
No. Time Source Destination Protocol Length Info
60 16.550586 192.168.0.164 52.66.181.14 HTTP 435 POST /URLCategorizerService/URLCategorize HTTP/1.1 (application/x-www-form-urlencoded)
112 16.885538 52.66.181.14 192.168.0.164 HTTP 672 HTTP/1.1 200 OK (application/text)
Frame 60: 435 bytes on wire (3480 bits), 435 bytes captured (3480 bits) on interface \Device\NPF_{ACC6CDED-1231-43C9-B6DE-6E9F09658D
Ethernet II, Src: ChongqingFug_2b:c4:85 (c8:94:02:2b:c4:85), Dst: ZyxelCommuni_ff:9a:94 (4c:c5:3e:ff:9a:94)
Internet Protocol Version 4, Src: 192.168.0.164, Dst: 52.66.181.14
Transmission Control Protocol, Src Port: 55744, Dst Port: 8080, Seq: 1, Ack: 1, Len: 381
Hypertext Transfer Protocol
POST /URLCategorizerService/URLCategorize HTTP/1.1\r\n
Host: prourl.itsecure.co.in:8080\r\n
Accept: */*\r\n
Content-Length: 210\r\n
Content-Type: application/x-www-form-urlencoded\r\n
\r\n
[Full request URI: http://prourl.itsecure.co.in:8080/URLCategorizerService/URLCategorize]
[HTTP request 1/1]
[Response in frame: 112]
File Data: 210 bytes
HTML Form URL Encoded: application/x-www-form-urlencoded
Hypertext Transfer Protocol: Protocol
Packets: 1659 · Displayed: 2 (0.1%)
Profile: Default
7°C
Mostly cloudy
Search
ENG
IN
04:07
29-07-2024


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http
No. Time Source Destination Protocol Length Info
60 16.550586 192.168.0.164 52.66.181.14 HTTP 435 POST /URLCategorizerService/URLCategorize HTTP/1.1 (application/x-www-form-urlencoded)
112 16.885538 52.66.181.14 192.168.0.164 HTTP 672 HTTP/1.1 200 OK (application/text)
Frame 112: 672 bytes on wire (5376 bits), 672 bytes captured (5376 bits) on interface \Device\NPF_{ACC6CDED-1231-43C9-B6DE-6E9F096
Ethernet II, Src: ZyxelCommuni_ff:9a:94 (4c:c5:3e:ff:9a:94), Dst: ChongqingFug_2b:c4:85 (c8:94:02:2b:c4:85)
Internet Protocol Version 4, Src: 52.66.181.14, Dst: 192.168.0.164
Transmission Control Protocol, Src Port: 8080, Dst Port: 55744, Seq: 1, Ack: 382, Len: 618
Hypertext Transfer Protocol
HTTP/1.1 200 OK\r\n
[Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
Response Version: HTTP/1.1
Status Code: 200
[Status Code Description: OK]
Response Phrase: OK
Date: Sun, 28 Jul 2024 18:04:48 GMT\r\n
Content-Type: application/text\r\n
Content-Length: 485\r\n
Connection: keep-alive\r\n
\r\n
[HTTP response 1/1]
[Time since request: 0.334952000 seconds]
[Request in frame: 60]
[Request URI: http://prourl.itsecure.co.in:8080/URLCategorizerService/URLCategorize]
Hypertext Transfer Protocol: Protocol
Packets: 2756 · Displayed: 2 (0.1%)
Profile: Default
Search
ENG
IN
04:08
29-07-2024


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http
No. Time Source Destination Protocol Length Info
60 16.550586 192.168.0.164 52.66.181.14 HTTP 435 POST /URLCategorizerService/URLCategorize HTTP/1.1 (application/x-www-form-urlencoded)
112 16.885538 52.66.181.14 192.168.0.164 HTTP 672 HTTP/1.1 200 OK (application/text)
4169 436.028440 192.168.0.164 184.84.238.106 HTTP 178 GET /ncsi.txt HTTP/1.1
4171 436.044242 184.84.238.106 192.168.0.164 HTTP 233 HTTP/1.1 200 OK (text/plain)
[Timestamps]
[SEQ/ACK analysis]
TCP payload (124 bytes)
Hypertext Transfer Protocol
GET /ncsi.txt HTTP/1.1\r\n
[Expert Info (Chat/Sequence): GET /ncsi.txt HTTP/1.1\r\n]
[GET /ncsi.txt HTTP/1.1\r\n]
[Severity level: Chat]
[Group: Sequence]
Request Method: GET
Request URI: /ncsi.txt
Request Version: HTTP/1.1
Host: www.msftncsi.com\r\n
User-Agent: Go-http-client/1.1\r\n
Accept-Encoding: gzip\r\n
Connection: close\r\n
\r\n
[Full request URI: http://www.msftncsi.com/ncsi.txt]
[HTTP request 1/1]
[Response in frame: 4171]
Hypertext Transfer Protocol: Protocol
Packets: 4600 · Displayed: 4 (0.1%)
Profile: Default
Search
ENG IN
04:12
29-07-2024


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http
Packet list Narrow & Wide Case sensitive Display filter http Find Cancel
No. Time Source Destination Protocol Length Info
451 93.161779 192.168.0.164 184.84.238.106 HTTP 178 GET /ncsi.txt HTTP/1.1
453 93.178010 184.84.238.106 192.168.0.164 HTTP 233 HTTP/1.1 200 OK (text/plain)
4575 320.075681 192.168.0.164 13.234.223.251 HTTP 455 POST /URLCategorizerService/URLCategorize HTTP/1.1 (application/x-www-form-urlencoded)
4819 320.414658 13.234.223.251 192.168.0.164 HTTP 1389 HTTP/1.1 200 OK (application/text)
9238 578.338062 192.168.0.164 3.6.136.49 HTTP 425 POST /URLCategorizerService/URLCategorize HTTP/1.1 (application/x-www-form-urlencoded)
9251 578.673396 3.6.136.49 192.168.0.164 HTTP 497 HTTP/1.1 200 OK (application/text)
9322 579.234187 192.168.0.164 3.6.136.49 HTTP 417 POST /URLCategorizerService/URLCategorize HTTP/1.1 (application/x-www-form-urlencoded)
9483 580.083340 3.6.136.49 192.168.0.164 HTTP 491 HTTP/1.1 200 OK (application/text)
9522 580.245696 192.168.0.164 3.6.136.49 HTTP 457 POST /URLCategorizerService/URLCategorize HTTP/1.1 (application/x-www-form-urlencoded)
9523 580.247141 192.168.0.164 3.6.136.49 HTTP 431 POST /URLCategorizerService/URLCategorize HTTP/1.1 (application/x-www-form-urlencoded)
9524 580.247141 192.168.0.164 3.6.136.49 HTTP 437 POST /URLCategorizerService/URLCategorize HTTP/1.1 (application/x-www-form-urlencoded)
9712 581.536885 3.6.136.49 192.168.0.164 HTTP 629 HTTP/1.1 200 OK (application/text)
9714 581.536885 3.6.136.49 192.168.0.164 HTTP 691 HTTP/1.1 200 OK (application/text)
9717 581.536885 3.6.136.49 192.168.0.164 HTTP 668 HTTP/1.1 200 OK (application/text)
11750 583.891352 192.168.0.164 3.6.136.49 HTTP 437 POST /URLCategorizerService/URLCategorize HTTP/1.1 (application/x-www-form-urlencoded)
12009 584.434917 3.6.136.49 192.168.0.164 HTTP 1145 HTTP/1.1 200 OK (application/text)
13251 585.633173 192.168.0.164 3.6.136.49 HTTP 433 POST /URLCategorizerService/URLCategorize HTTP/1.1 (application/x-www-form-urlencoded)
13297 585.647700 192.168.0.164 3.6.136.49 HTTP 417 POST /URLCategorizerService/URLCategorize HTTP/1.1 (application/x-www-form-urlencoded)
13619 586.003624 3.6.136.49 192.168.0.164 HTTP 1180 HTTP/1.1 200 OK (application/text)
Hypertext Transfer Protocol
HTTP/1.1 200 OK\r\n
[Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
[HTTP/1.1 200 OK\r\n]
[Severity level: Chat]
[Group: Sequence]
Response Version: HTTP/1.1
Status Code: 200
[Status Code Description: OK]
Response Phrase: OK
Content-Length: 14\r\n
[Content length: 14]
Date: Sun, 28 Jul 2024 18:31:48 GMT\r\n
Connection: close\r\n
Content-Type: text/plain\r\n
Cache-Control: max-age=30, must-revalidate\r\n
\r\n
[HTTP response 1/1]
[Time since request: 0.016231000 seconds]
[Request in frame: 451]
Expert Info (_ws.expert)
Packets: 35077 · Displayed: 110 (0.3%)
Profile: Default
Search
ENG IN
04:50
29-07-2024



*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http
Packet list
Narrow & Wide
Case sensitive
Display filter
http
Find
Cancel
No. Time Source Destination Protocol Length Info
451 93.161779 192.168.0.164 184.84.238.106 HTTP 178 GET /ncsi.txt HTTP/1.1
453 93.178010 184.84.238.106 192.168.0.164 HTTP 233 HTTP/1.1 200 OK (text/plain)
4575 320.075681 192.168.0.164 13.234.223.251 HTTP 455 POST /URLCategorizerService/URLCategorize HTTP/1.1 (application/x-www-form-urlencoded)
4819 320.414658 13.234.223.251 192.168.0.164 HTTP 1389 HTTP/1.1 200 OK (application/text)
9238 578.338062 192.168.0.164 3.6.136.49 HTTP 425 POST /URLCategorizerService/URLCategorize HTTP/1.1 (application/x-www-form-urlencoded)
9251 578.673396 3.6.136.49 192.168.0.164 HTTP 497 HTTP/1.1 200 OK (application/text)
9322 579.234187 192.168.0.164 3.6.136.49 HTTP 417 POST /URLCategorizerService/URLCategorize HTTP/1.1 (application/x-www-form-urlencoded)
9483 580.083340 3.6.136.49 192.168.0.164 HTTP 491 HTTP/1.1 200 OK (application/text)
9522 580.245696 192.168.0.164 3.6.136.49 HTTP 457 POST /URLCategorizerService/URLCategorize HTTP/1.1 (application/x-www-form-urlencoded)
9523 580.247141 192.168.0.164 3.6.136.49 HTTP 431 POST /URLCategorizerService/URLCategorize HTTP/1.1 (application/x-www-form-urlencoded)
9524 580.247141 192.168.0.164 3.6.136.49 HTTP 437 POST /URLCategorizerService/URLCategorize HTTP/1.1 (application/x-www-form-urlencoded)
9712 581.536885 3.6.136.49 192.168.0.164 HTTP 629 HTTP/1.1 200 OK (application/text)
9714 581.536885 3.6.136.49 192.168.0.164 HTTP 691 HTTP/1.1 200 OK (application/text)
9717 581.536885 3.6.136.49 192.168.0.164 HTTP 668 HTTP/1.1 200 OK (application/text)
11750 583.891352 192.168.0.164 3.6.136.49 HTTP 437 POST /URLCategorizerService/URLCategorize HTTP/1.1 (application/x-www-form-urlencoded)
12009 584.434917 3.6.136.49 192.168.0.164 HTTP 1145 HTTP/1.1 200 OK (application/text)
13251 585.633173 192.168.0.164 3.6.136.49 HTTP 433 POST /URLCategorizerService/URLCategorize HTTP/1.1 (application/x-www-form-urlencoded)
13297 585.647700 192.168.0.164 3.6.136.49 HTTP 417 POST /URLCategorizerService/URLCategorize HTTP/1.1 (application/x-www-form-urlencoded)
13619 586.003624 3.6.136.49 192.168.0.164 HTTP 1180 HTTP/1.1 200 OK (application/text)
[Timestamps]
[SEQ/ACK analysis]
TCP payload (179 bytes)
Hypertext Transfer Protocol
HTTP/1.1 200 OK\r\n
[Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
[HTTP/1.1 200 OK\r\n]
[Severity level: Chat]
[Group: Sequence]
Response Version: HTTP/1.1
Status Code: 200
[Status Code Description: OK]
Response Phrase: OK
Content-Length: 14\r\n
[Content length: 14]
Date: Sun, 28 Jul 2024 18:31:48 GMT\r\n
Connection: close\r\n
Content-Type: text/plain\r\n
Cache-Control: max-age=30, must-revalidate\r\n
\r\n
HTTP Cache Control (http.cache_control), 44 bytes
Packets: 35902 · Displayed: 114 (0.3%)
Profile: Default
Search
ENG
IN
04:51
29-07-2024