Web requests

CS252

August 23, 2018

Lifecycle of a web-request

We will see the major steps required for a machine to connect to a network and successfully request a web-page on its browser.

The major steps involved are as follows.

1. Getting an IP address using DHCP, UDP, IP, Ethernet.

Lifecycle of a web-request

We will see the major steps required for a machine to connect to a network and successfully request a web-page on its browser.

The major steps involved are as follows.

1. Getting an IP address using DHCP, UDP, IP, Ethernet.
2. Getting to know the gateway MAC address and the webserver IP address: ARP, DNS

Lifecycle of a web-request

We will see the major steps required for a machine to connect to a network and successfully request a web-page on its browser.

The major steps involved are as follows.

1. Getting an IP address using DHCP, UDP, IP, Ethernet.
2. Getting to know the gateway MAC address and the webserver IP address: ARP, DNS
3. Requesting the web-page: TCP and HTTP.

Stage I: DHCP, UDP, IP, Ethernet

1. The machine connects to the network. It knows its MAC address, does not have an IP address.

Stage I: DHCP, UDP, IP, Ethernet

1. The machine connects to the network. It knows its MAC address, does not have an IP address.

2. It sends a *DHCP request*, a UDP protocol packet.

Stage I: DHCP, UDP, IP, Ethernet

1. The machine connects to the network. It knows its MAC address, does not have an IP address.

2. It sends a *DHCP request*, a UDP protocol packet.

3. The DHCP request is enclosed in an IP packet with source 0.0.0.0 (unknown IP address) and destination 255.255.255.255 (broadcast). The IP packet is enclosed in an ethernet frame with the source id that of the machine, and destination ff:ff:ff:ff:ff:ff (broadcast).

Stage I: DHCP, UDP, IP, Ethernet

1. The machine connects to the network. It knows its MAC address, does not have an IP address.

2. It sends a *DHCP request*, a UDP protocol packet.

3. The DHCP request is enclosed in an IP packet with source 0.0.0.0 (unknown IP address) and destination 255.255.255.255 (broadcast). The IP packet is enclosed in an ethernet frame with the source id that of the machine, and destination ff:ff:ff:ff:ff:ff (broadcast).

4. The ethernet switch of the LAN broadcasts on all outgoing ports and the packet gets to the router.

Stage I: DHCP, UDP, IP, Ethernet

1. The machine connects to the network. It knows its MAC address, does not have an IP address.

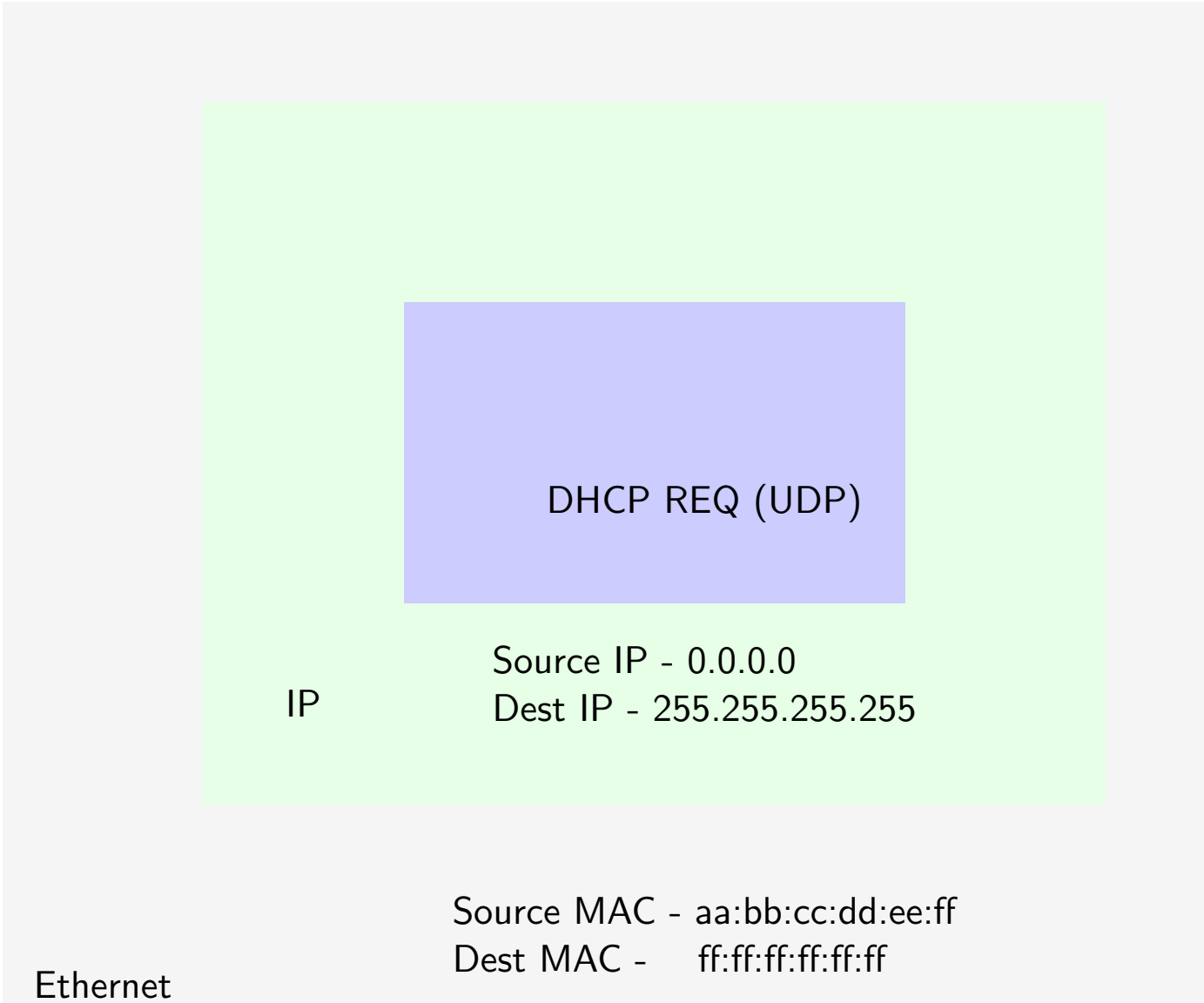
2. It sends a *DHCP request*, a UDP protocol packet.

3. The DHCP request is enclosed in an IP packet with source 0.0.0.0 (unknown IP address) and destination 255.255.255.255 (broadcast). The IP packet is enclosed in an ethernet frame with the source id that of the machine, and destination ff:ff:ff:ff:ff:ff (broadcast).

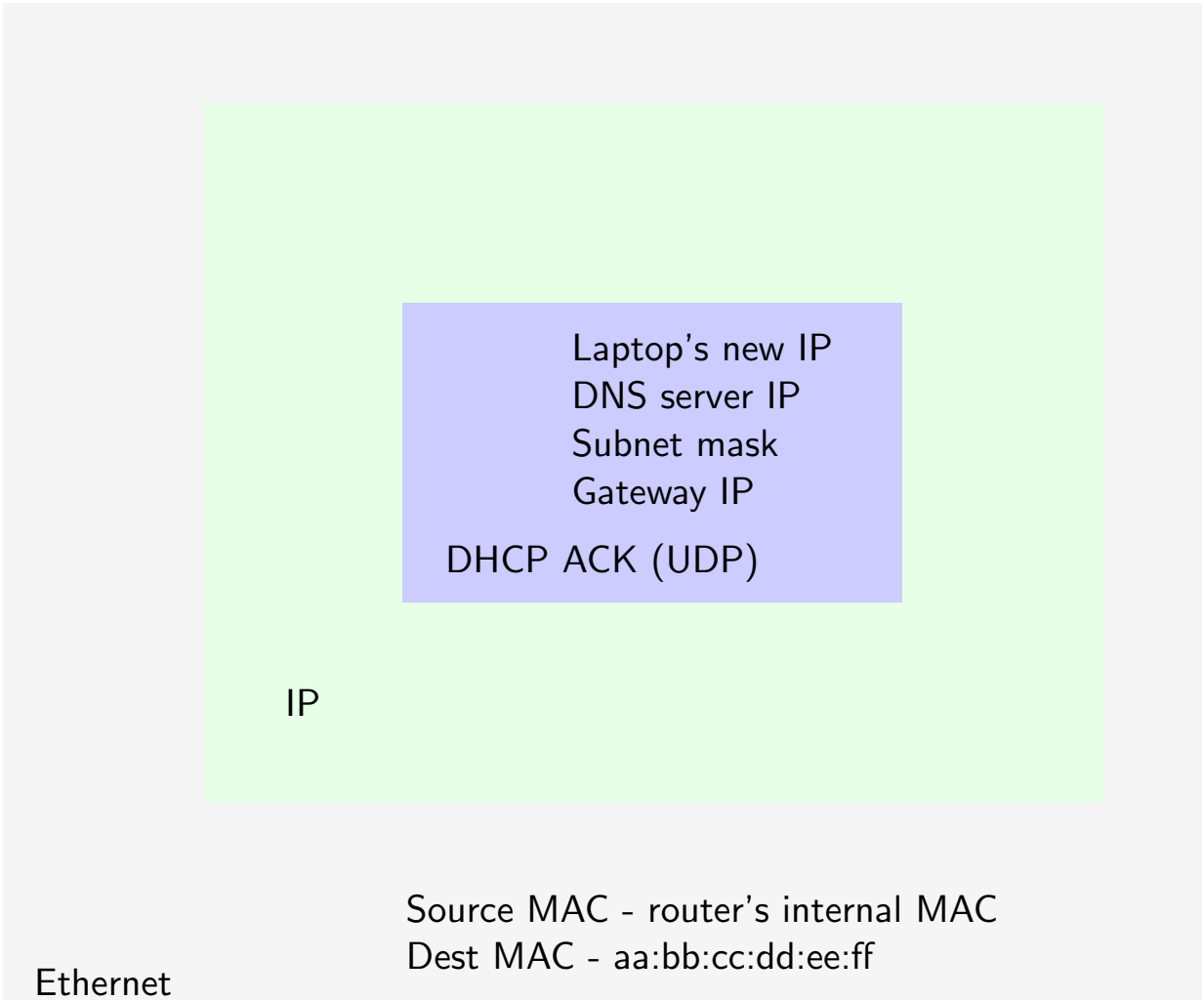
4. The ethernet switch of the LAN broadcasts on all outgoing ports and the packet gets to the router.

5. The router running the DHCP server (on port 67) assigns an IP address and sends a DHCP ACK response containing the machine's assigned IP address, the IP address of the *DNS* server, the IP address for the default *gateway*. It is put in an IP datagram, inside an ethernet frame with source mac address equal to the router's internal interface, and destination MAC address that of the laptop.

DHCP request over IP



DHCP response over IP



How does the ethernet switch receive this packet? How is it forwarded to the laptop?

Stage II: ARP, DNS

The machine knows its own IP address, its own MAC address, the address of the DNS server, the subnet mask, and the IP address of the default gateway. (see also: adapter TCP settings in Windows)

1. The user on the laptop opens a browser and types `http://www.google.com` in the location bar.

Stage II: ARP, DNS

The machine knows its own IP address, its own MAC address, the address of the DNS server, the subnet mask, and the IP address of the default gateway. (see also: adapter TCP settings in Windows)

1. The user on the laptop opens a browser and types `http://www.google.com` in the location bar.

2. Now we try to construct the DNS query message, a UDP packet, inside an IP packet with source=laptop's IP and destination=DNS server's IP. It now has to send the packet to the gateway. But we don't know the MAC address of the default gateway.

Stage II: ARP, DNS

The machine knows its own IP address, its own MAC address, the address of the DNS server, the subnet mask, and the IP address of the default gateway. (see also: adapter TCP settings in Windows)

1. The user on the laptop opens a browser and types `http://www.google.com` in the location bar.

2. Now we try to construct the DNS query message, a UDP packet, inside an IP packet with source=laptop's IP and destination=DNS server's IP. It now has to send the packet to the gateway. But we don't know the MAC address of the default gateway.

3. So we send an ARP packet to know the gateway MAC address. From the ARP reply, we get the gateway MAC address.

Stage II: ARP, DNS

The machine knows its own IP address, its own MAC address, the address of the DNS server, the subnet mask, and the IP address of the default gateway. (see also: adapter TCP settings in Windows)

1. The user on the laptop opens a browser and types `http://www.google.com` in the location bar.

2. Now we try to construct the DNS query message, a UDP packet, inside an IP packet with source=laptop's IP and destination=DNS server's IP. It now has to send the packet to the gateway. But we don't know the MAC address of the default gateway.

3. So we send an ARP packet to know the gateway MAC address. From the ARP reply, we get the gateway MAC address.

4. Now the packet in step 2 can be completed. We send the DNS query to the local DNS server, and get the IP address of `www.google.com`

Stage III: TCP, HTTP

1. The browser creates a TCP socket to `www.google.com`. Using a three-way-handshake, we establish a connection at port 80 between the Google HTTP server and the laptop.

Stage III: TCP, HTTP

1. The browser creates a TCP socket to `www.google.com`. Using a three-way-handshake, we establish a connection at port 80 between the Google HTTP server and the laptop.

2. Using an HTTP GET message through the socket, the TCP segment is sent to www.google.com in a datagram.

Stage III: TCP, HTTP

1. The browser creates a TCP socket to `www.google.com`. Using a three-way-handshake, we establish a connection at port 80 between the Google HTTP server and the laptop.

2. Using an HTTP GET message through the socket, the TCP segment is sent to www.google.com in a datagram.

3. Using HTTP response, the HTTP server sends the webpage in the body of the response message to the TCP socket.

Some Basic Tools

1. ping: send ICMP_ECHO messages to specified hosts. Used to test whether the host is reachable.

Some Basic Tools

1. ping: send ICMP_ECHO messages to specified hosts. Used to test whether the host is reachable.
2. traceroute: find the hops used to reach a host.

Some Basic Tools

1. ping: send ICMP_ECHO messages to specified hosts. Used to test whether the host is reachable.
2. traceroute: find the hops used to reach a host.
3. dig: DNS lookup.

Some Basic Tools

1. ping: send ICMP_ECHO messages to specified hosts. Used to test whether the host is reachable.
2. traceroute: find the hops used to reach a host.
3. dig: DNS lookup.
4. tcpdump: e.g. `tcpdump -c 40000 -e` to examine traffic on a network interface.

Some Basic Tools

1. ping: send ICMP_ECHO messages to specified hosts. Used to test whether the host is reachable.
2. traceroute: find the hops used to reach a host.
3. dig: DNS lookup.
4. tcpdump: e.g. `tcpdump -c 40000 -e` to examine traffic on a network interface.
5. netstat: examine the state of the Networking subsystem. e.g. `netstat | grep tcp`

curl

1. Simple tool used to transfer data from and to any server.

curl

1. Simple tool used to transfer data from and to any server.
2. Commonly used to access HTTP servers

curl

1. Simple tool used to transfer data from and to any server.
2. Commonly used to access HTTP servers
3. Our lab next week will focus on using curl effectively

Simple curl GET usage

1. Getting an HTML page from a server

Simple curl GET usage

1. Getting an HTML page from a server
2. curl URL

Simple curl GET usage

1. Getting an HTML page from a server
2. curl URL
3. Saving curl output to a file

Simple curl GET usage

1. Getting an HTML page from a server
2. curl URL
3. Saving curl output to a file
4. curl -o URL

Simple curl GET usage

1. Getting an HTML page from a server
2. curl URL
3. Saving curl output to a file
4. curl -o URL
5. Getting a file from an FTP server

Simple curl GET usage

1. Getting an HTML page from a server
2. curl URL
3. Saving curl output to a file
4. curl -o URL
5. Getting a file from an FTP server
6. curl ftp://username:password@ftp.server:21/path/to/file.tar.gz

Simple curl POST usage

1. Sending data to a server

Simple curl POST usage

1. Sending data to a server
2. curl -d 'name=nisheeth', -d 'age=35' http://server/

Simple curl POST usage

1. Sending data to a server
2. curl -d 'name=nisheeth', -d 'age=35' http://server/
3. Default content-type is application/x-www-form-urlencoded

Simple curl POST usage

1. Sending data to a server
2. curl -d 'name=nisheeth', -d 'age=35' http://server/
3. Default content-type is application/x-www-form-urlencoded
4. Can change it

Simple curl POST usage

1. Sending data to a server
2. curl -d 'name=nisheeth', -d 'age=35' http://server/
3. Default content-type is application/x-www-form-urlencoded
4. Can change it
5. curl -d '"user": "name": "nisheeth"' -H "Content-Type: application/json" http://server

Simple curl POST usage

1. Sending data to a server
2. curl -d 'name=nisheeth', -d 'age=35' http://server/
3. Default content-type is application/x-www-form-urlencoded
4. Can change it
5. curl -d '"user": "name": "nisheeth"' -H "Content-Type: application/json" http://server
6. Can read in file as input

Simple curl POST usage

1. Sending data to a server
2. curl -d 'name=nisheeth', -d 'age=35' http://server/
3. Default content-type is application/x-www-form-urlencoded
4. Can change it
5. curl -d '"user": "name": "nisheeth"' -H "Content-Type: application/json" http://server
6. Can read in file as input
7. curl -d @sample-data.txt http://server/

Interacting with APIs

1. API = Application Programming Interface

Interacting with APIs

1. API = Application Programming Interface
2. Web APIs let developers use functionality from other web services in their own apps

Interacting with APIs

1. API = Application Programming Interface
2. Web APIs let developers use functionality from other web services in their own apps
3. Looks like a simple web request, but with additional API-specific parameters

Interacting with APIs

1. API = Application Programming Interface
2. Web APIs let developers use functionality from other web services in their own apps
3. Looks like a simple web request, but with additional API-specific parameters
4. Most commonly returned data format is JSON

A sample API - eRail.in

1. Registering on their website gets you an API key - its a random string bound to your IP address

A sample API - eRail.in

1. Registering on their website gets you an API key - its a random string bound to your IP address
2. Respect provider's usage limits: GetStations 10, GetTrains 4, Max-requests 5

A sample API - eRail.in

1. Registering on their website gets you an API key - its a random string bound to your IP address
2. Respect provider's usage limits: GetStations 10, GetTrains 4, Max-requests 5
3. man API $\rightarrow$ route, parameter list

A sample API - eRail.in

1. Registering on their website gets you an API key - its a random string bound to your IP address
2. Respect provider's usage limits: GetStations 10, GetTrains 4, Max-requests 5
3. man API $\rightarrow$ route, parameter list
4. API calls are structured like URLs using routes and parameters

A sample API - eRail.in

1. Registering on their website gets you an API key - its a random string bound to your IP address
2. Respect provider's usage limits: GetStations 10, GetTrains 4, Max-requests 5
3. man API $\rightarrow$ route, parameter list
4. API calls are structured like URLs using routes and parameters
5. Get all stations by requesting
   $http://api.erail.in/stations/?key=API\_KEY$

Sample eRail.in calls

1. http://api.erail.in/trains/?key=API_KEY&stnfrom=NDLS&stnto=B

Sample eRail.in calls

1. http://api.erail.in/trains/?key=API_KEY&stnfrom=NDLS&stnto=B
2. http://api.erail.in/route/?key=API_KEY&trainno=12138

Sample eRail.in calls

1. http://api.erail.in/trains/?key=API_KEY&stnfrom=NDLS&stnto=B
2. http://api.erail.in/route/?key=API_KEY&trainno=12138
3. http://api.erail.in/fullroute/?key=API_KEY&trainno=12138

Sample eRail.in calls

1. http://api.erail.in/trains/?key=API_KEY&stnfrom=NDLS&stnto=B
2. http://api.erail.in/route/?key=API_KEY&trainno=12138
3. http://api.erail.in/fullroute/?key=API_KEY&trainno=12138
4. http://api.erail.in/pnr?key=API_KEY&pnr=4857412584

Sample eRail.in calls

1. http://api.erail.in/trains/?key=API_KEY&stnfrom=NDLS&stnto=B
2. http://api.erail.in/route/?key=API_KEY&trainno=12138
3. http://api.erail.in/fullroute/?key=API_KEY&trainno=12138
4. http://api.erail.in/pnr?key=API_KEY&pnr=4857412584
5. http://api.erail.in/live/?key=API_KEY&trainno=12138&stnfrom=N SEP-2018

Output format - JSON

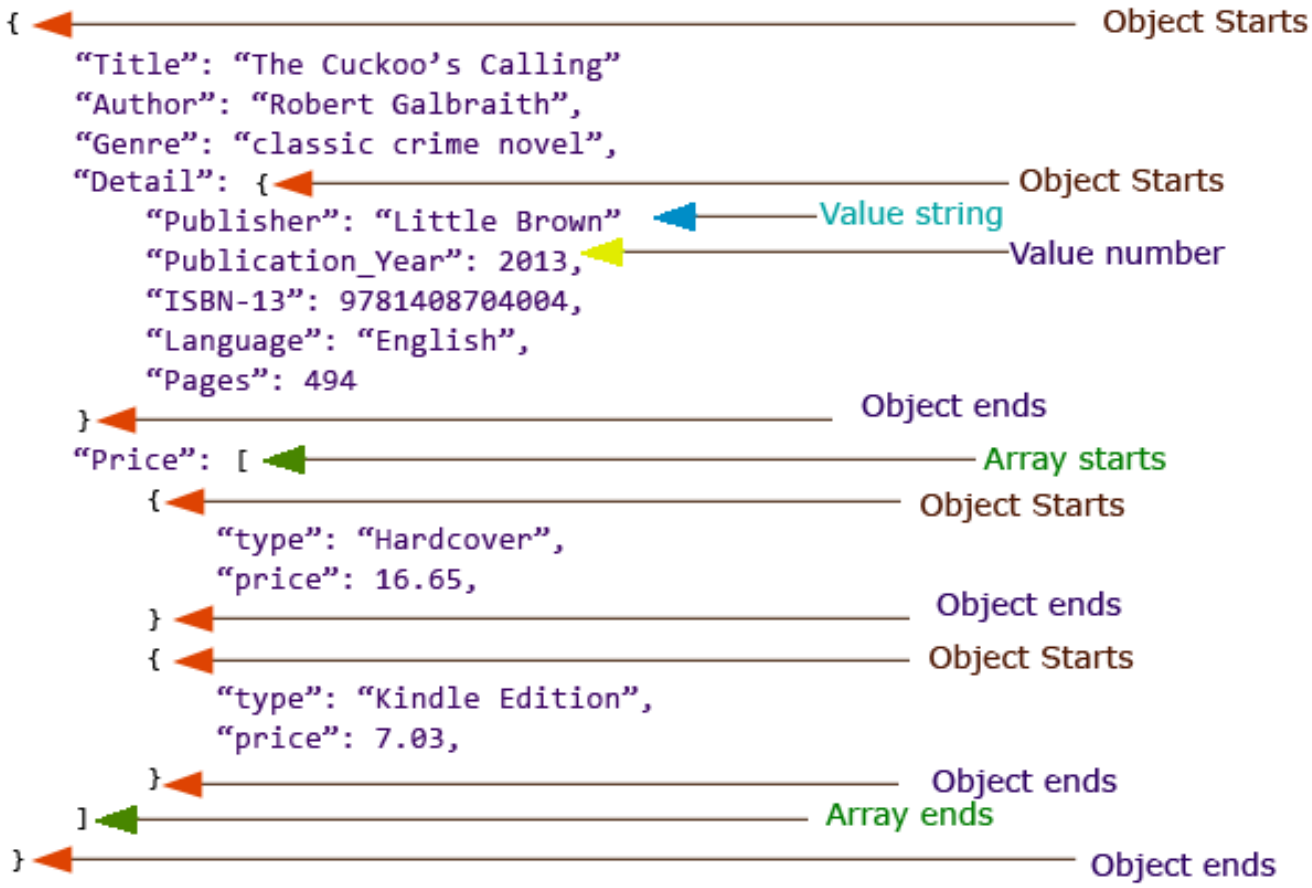


Figure: A sample JSON string

Lab this week

1. Experiment using GET requests with curl

Lab this week

1. Experiment using GET requests with curl
2. Use curl to get iitk.ac.in, cse.iitk.ac.in, oars.cc.iitk.ac.in in headers, listing the servers and caching policies they use.

Lab this week

1. Experiment using GET requests with curl
2. Use curl to get iitk.ac.in, cse.iitk.ac.in, oars.cc.iitk.ac.in in headers, listing the servers and caching policies they use.
3. Post to the course feedback Google Form using curl

Lab this week

1. Experiment using GET requests with curl
2. Use curl to get iitk.ac.in, cse.iitk.ac.in, oars.cc.iitk.ac.in in headers, listing the servers and caching policies they use.
3. Post to the course feedback Google Form using curl
4. Use API calls to the eTrain API to find the quickest train from CNB to your hometown on Fridays

Lab this week

1. Experiment using GET requests with curl
2. Use curl to get iitk.ac.in, cse.iitk.ac.in, oars.cc.iitk.ac.in in headers, listing the servers and caching policies they use.
3. Post to the course feedback Google Form using curl
4. Use API calls to the eTrain API to find the quickest train from CNB to your hometown on Fridays
5. Do something fun with some other API of your choice

Lab this week

1. Experiment using GET requests with curl
2. Use curl to get iitk.ac.in, cse.iitk.ac.in, oars.cc.iitk.ac.in in headers, listing the servers and caching policies they use.
3. Post to the course feedback Google Form using curl
4. Use API calls to the eTrain API to find the quickest train from CNB to your hometown on Fridays
5. Do something fun with some other API of your choice
6. We will do Assignment 1 during the first lab hour in the coming week