Computer Network I

Reti di Calcolatori I

Università di Napoli Federico II – Scuola Politecnica e delle Scienze di Base
Corso di Laurea in Informatica

Riccardo Caccavale

(riccardo.caccavale@unina.it)

UNIVERSITÀ DEGLI STUDI DI NAPOLI
FEDERICO II

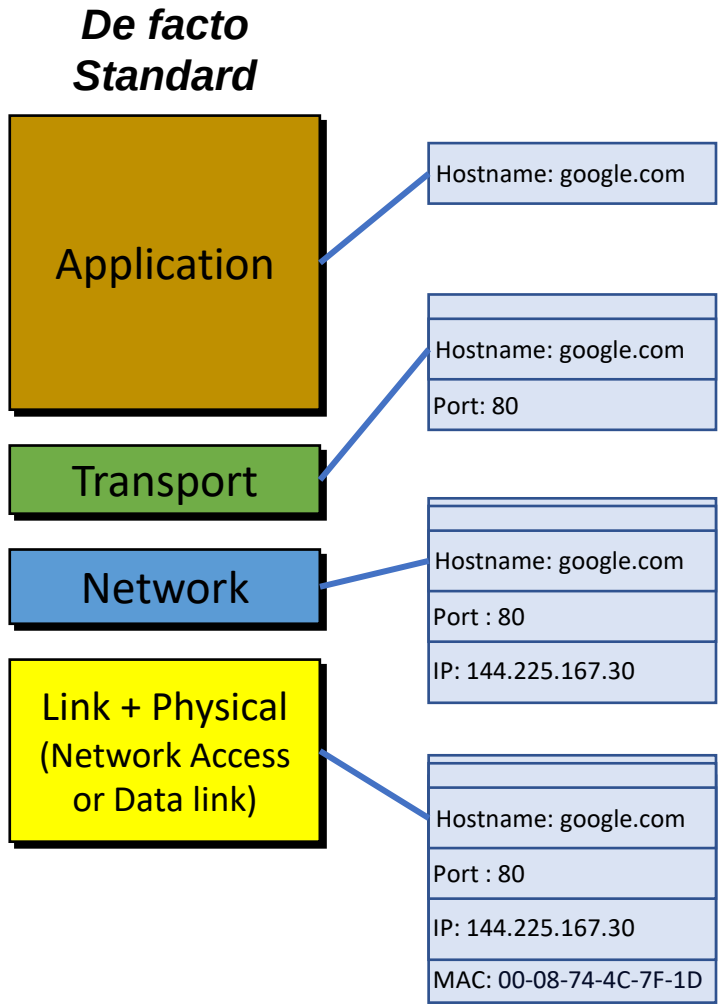
DIETI

UNINA

Stack Overview

There and Back Again

- As we have just seen, **several protocols in the TCP/IP stack collaborate** in allowing messages to travel back-and-forth a network.
- **Each layer-specific protocol adds a piece of information** to messages (through encapsulation and decapsulation) regulating one or more aspects of the communication (services offered).
- It is somehow difficult to get the “big picture” out of the single protocols, we will now recap the whole process by proposing a **web page request scenario**.

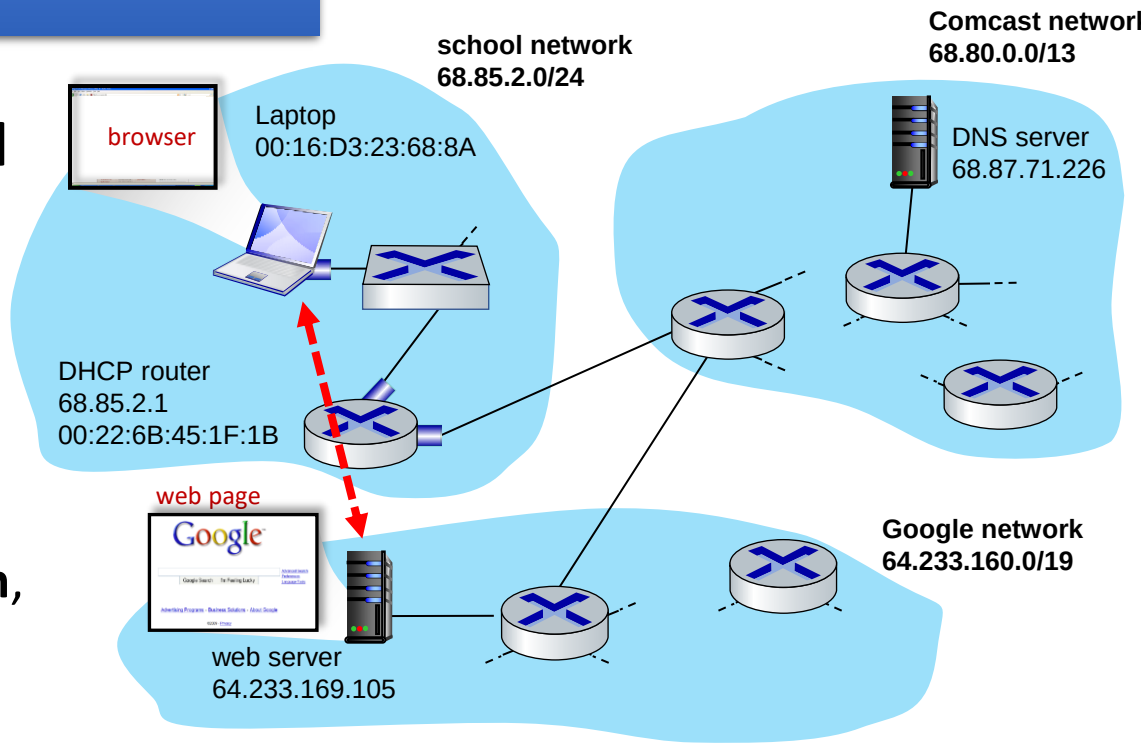


Example of the different addresses/identifiers considered in each layer of the stack.

Stack Overview

Web Page Example

- Let's assume that **a user (Bob) wants to access the google.com web page from the institutional network** (school).
  - To do so, Bob **connects the laptop to the school's network through an Ethernet cable**.
- In this example we assume a **typical configuration** for the school's network:
  - Ethernet **sockets (on walls) are connected to a switch**, which is **connected to the school's DHCP-enabled router**.
  - The **school's router is connected to an ISP** (e.g., comcast.net) that provides also **DNS service**.
- For the sake of simplicity, we also assume there is **no NAT service** from the router, all **connections are Ethernet**, and **packets are never lost**.



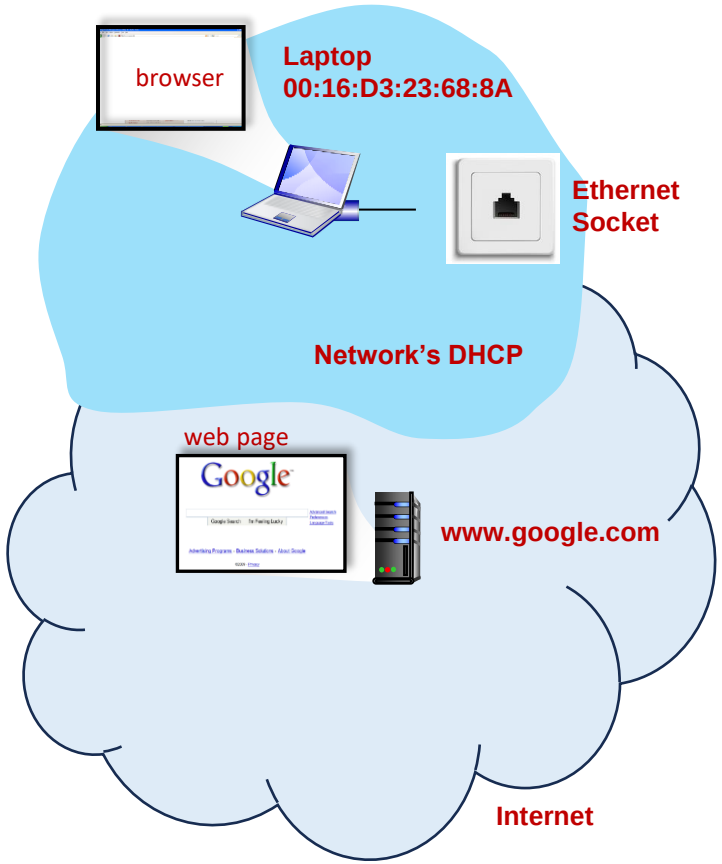
Stack Overview

Web Page Example

- On start, the **user knows very little about the network** topology, the way the LAN is connected to Internet, or the devices involved.

- What Bob knows is that:
  - **There is a www.google.com website somewhere on Internet** that Bob wants to reach.
  - **The local network (LAN) is somehow connected to Internet**.
  - The **laptop can be connected to the LAN** through an Ethernet cable (there is a socket on the wall).
  - The **LAN has an active DHCP**.

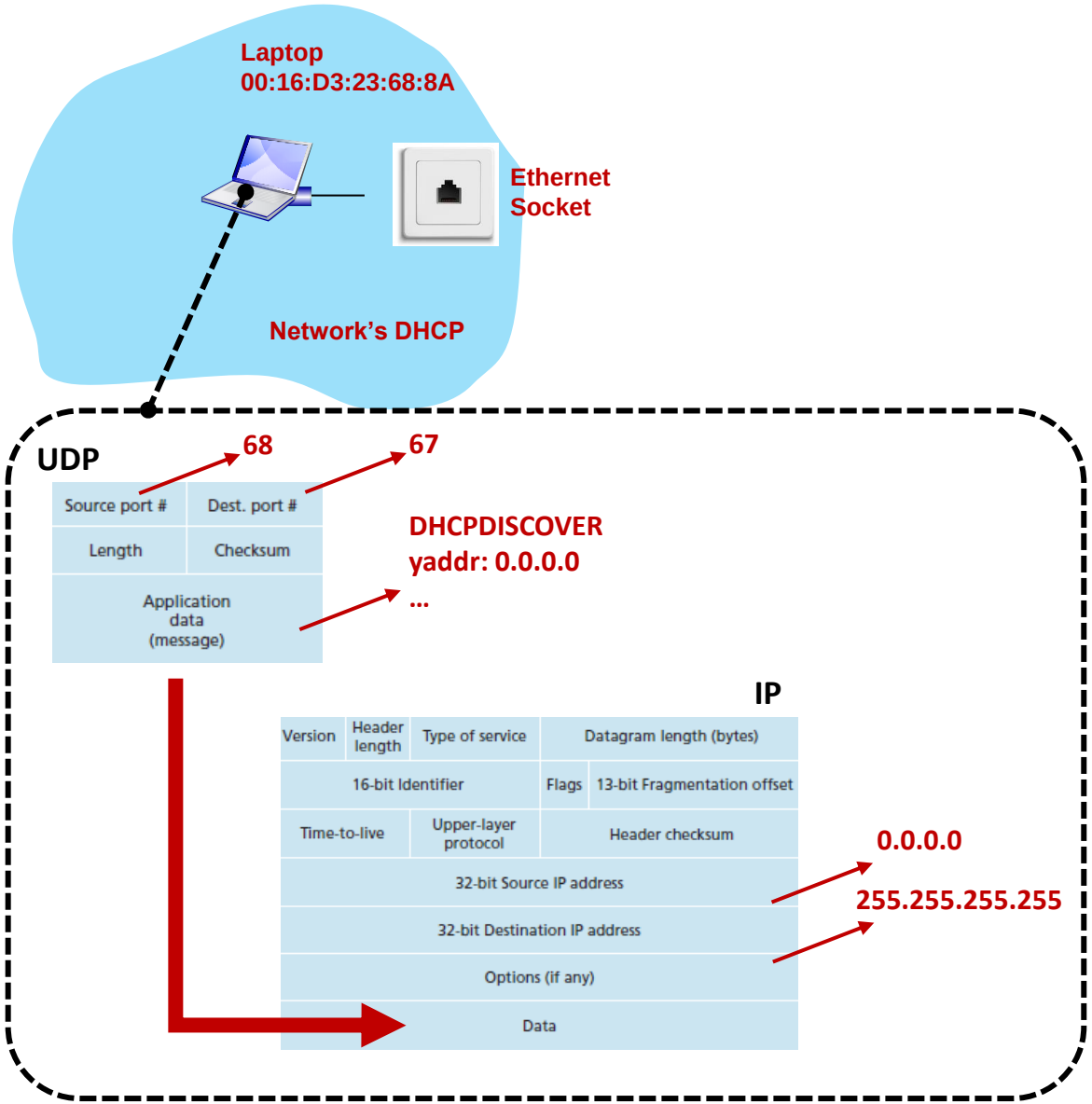
- **Once the laptop is physically connected** to the network (through the Ethernet cable) it has to:
  - **Join the network** (get network information from DHCP).
  - **Get the IP address of the website** (DNS).
  - **Get the Google's webpage** (HTTP).



Stack Overview

DHCP

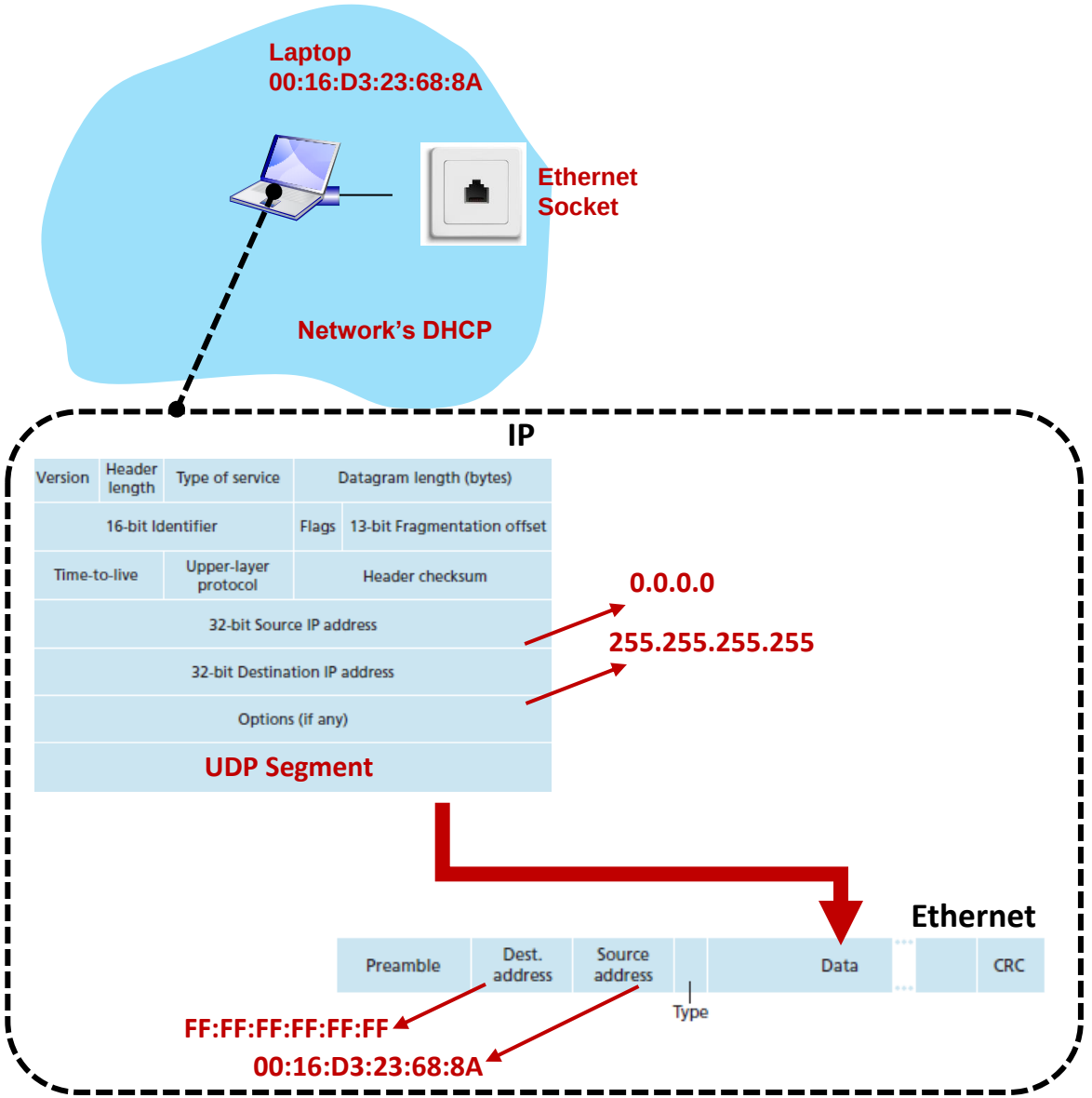
- The first step is for the laptop to join the network by **requesting a host IP, gateway and DNS** to the DHCP:
  1. The **operating system (OS) on the laptop creates a DHCP discovery message** and puts this message **within a UDP segment** with destination port 67 (DHCP server) and source port 68 (DHCP client).
     - The **UDP segment is then placed within an IP datagram** with a broadcast IP destination address (255.255.255.255) and a source IP address of 0.0.0.0 (no host IP yet).



Stack Overview

DHCP

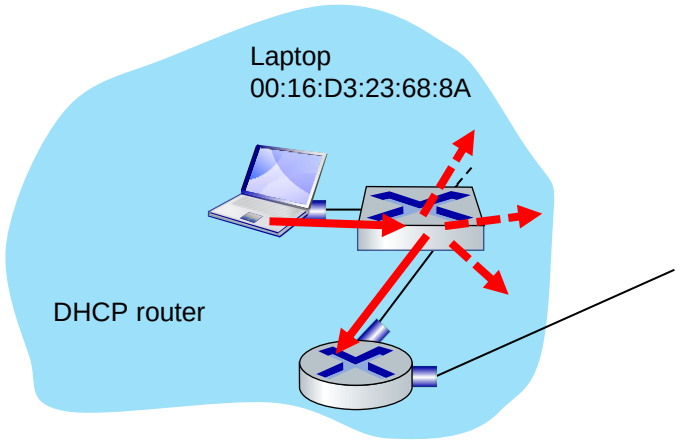
2. The **IP datagram** containing the DHCP discovery message **is then placed within an Ethernet frame** having FF:FF:FF:FF:FF:FF (broadcast) as destination MAC addresses (it will reach all devices on the switch and, hopefully, the DHCP) and laptop's MAC address 00:16:D3:23:68:8A as source.



Stack Overview

DHCP

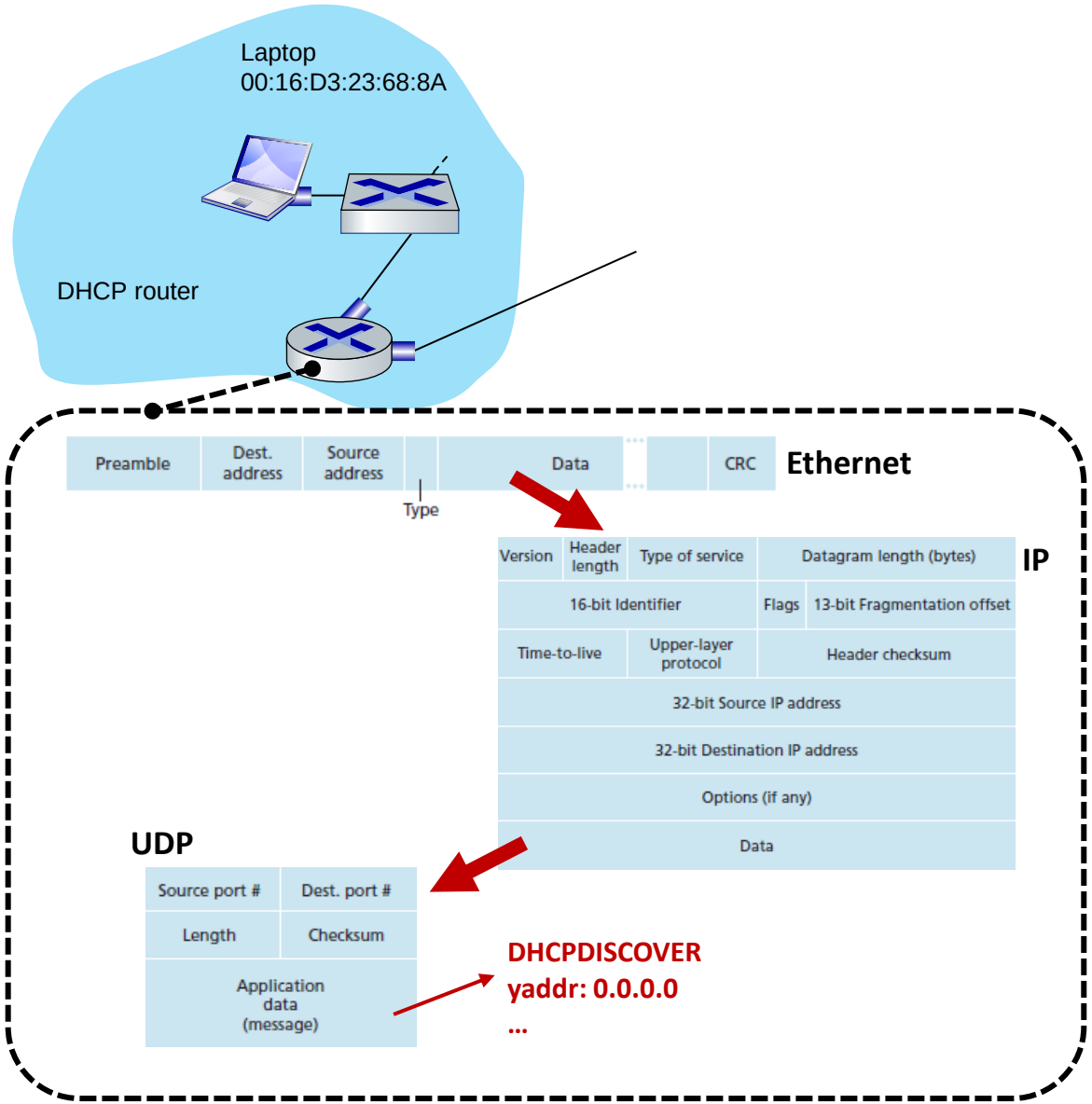
3. The **Ethernet frame containing the DHCP message is sent to the Ethernet switch**. The **switch broadcasts the frame** on all outgoing ports (including the port connected to the router).



Stack Overview

DHCP

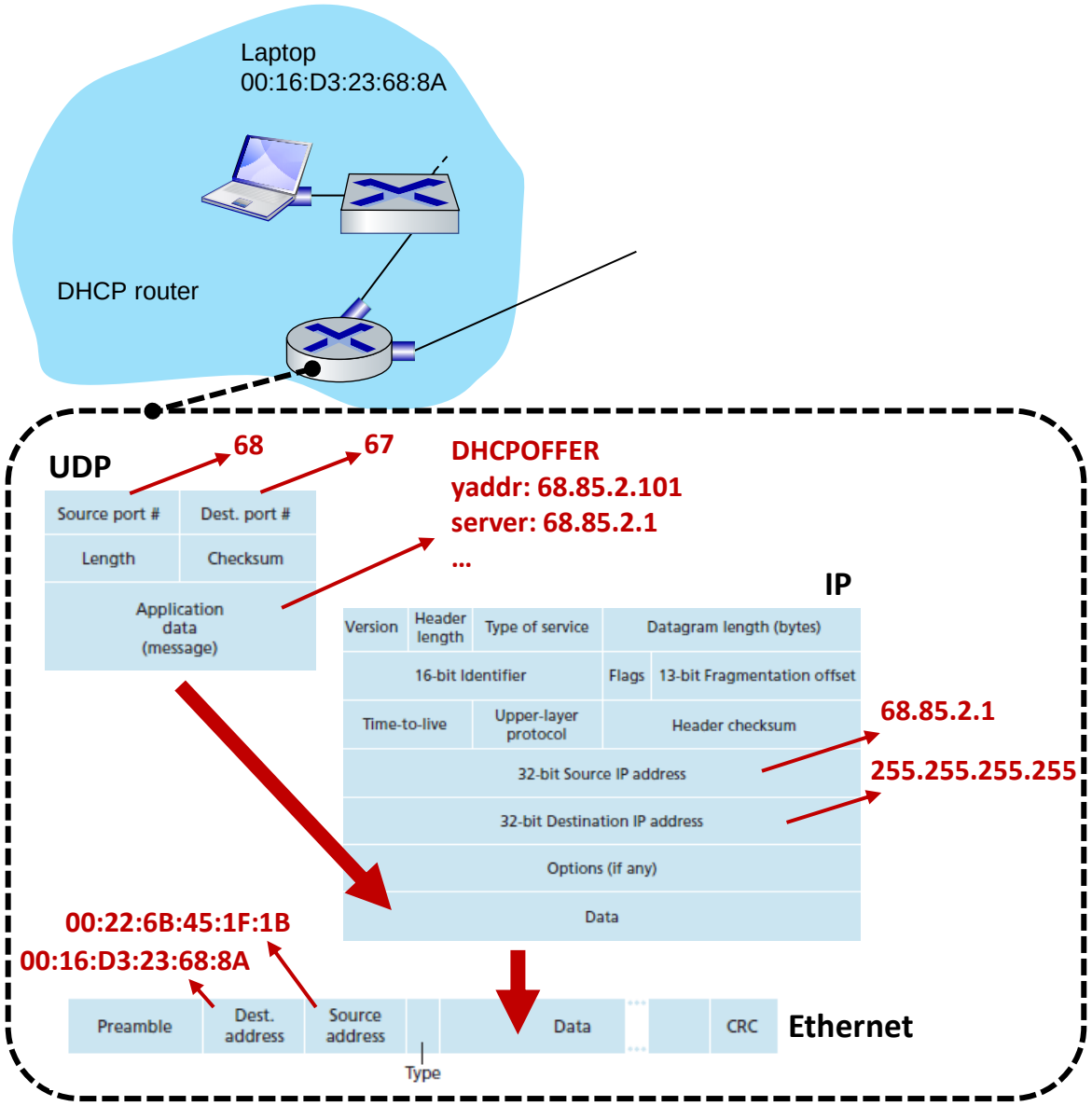
3. The **Ethernet frame containing the DHCP message is sent to the Ethernet switch**. The **switch broadcasts the frame** on all outgoing ports (including the port connected to the router).
4. The **router receives the Ethernet frame** containing the DHCP discovery on its interface (with MAC address 00:22:6B:45:1F:1B), **the message is decapsulated**:
   - The **frame is accepted because it has broadcast destination MAC** address, then the IP datagram is extracted.
   - The **datagram is accepted because it has broadcast IP** destination address, then the UDP segment is extracted.
   - The **UDP segment is demultiplexed** and the payload is received by the DHCP process on the router.



Stack Overview

DHCP

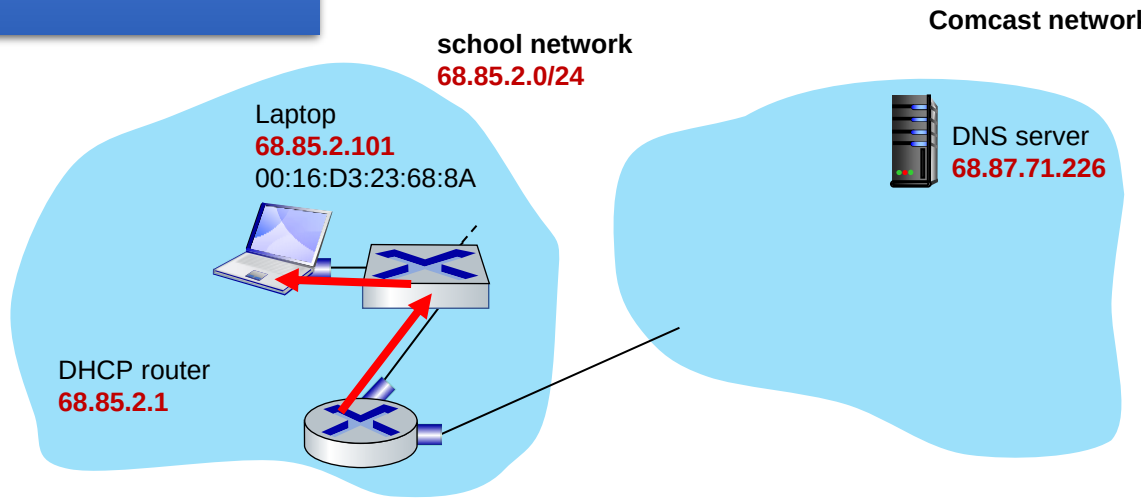
5. Let's now assume that **the DHCP router can allocate IP addresses in the CIDR block 68.85.2.0/24** (which is a subnetwork provided by the ISP):
   - The **DHCP server offers the IP** address 68.85.2.101 to the laptop.
   - The **DHCP server creates a DHCP offer message** containing:
     - The **offered IP address** and mask (68.85.2.101/24).
     - The **IP address of the DNS** server (68.87.71.226).
     - The **IP address for the gateway** (68.85.2.1).
   - The message is **encapsulated**:
     - A **UDP segment** is created with source port 67 and destination port 68.
     - The **segment is put into an IP datagram having broadcast as destination** address and 68.80.2.1 as source address.
     - The **segment is put into an Ethernet frame having 00:22:6B:45:1F:1B as source MAC** address (the router's LAN-interface) and **00:16:D3:23:68:8A as destination MAC** address (the laptop).



Stack Overview

DHCP

6. The Ethernet **frame containing the DHCP offer is sent** (unicast) by the router to the switch, which forwards it to the laptop checking the destination MAC address.
7. Bob's **laptop receives the Ethernet frame** with the DHCP offer and **decapsulates** it:
   - **Frame accepted** due to **correct MAC address**, then IP datagram is extracted.
   - **IP datagram accepted** due to **broadcast IP** address, then UDP segment is extracted.
   - DHCP **offer is demultiplexed** to the client DHCP process of the OS.
   - The **OS accepts the offer and sends back a DHCP request message** (skipped for brevity).
   - When **a DHCP ACK message is eventually received** from a new UDP segment, the **OS sets the received network information** (IP, gateway and DNS). Now the laptop has joined the network.



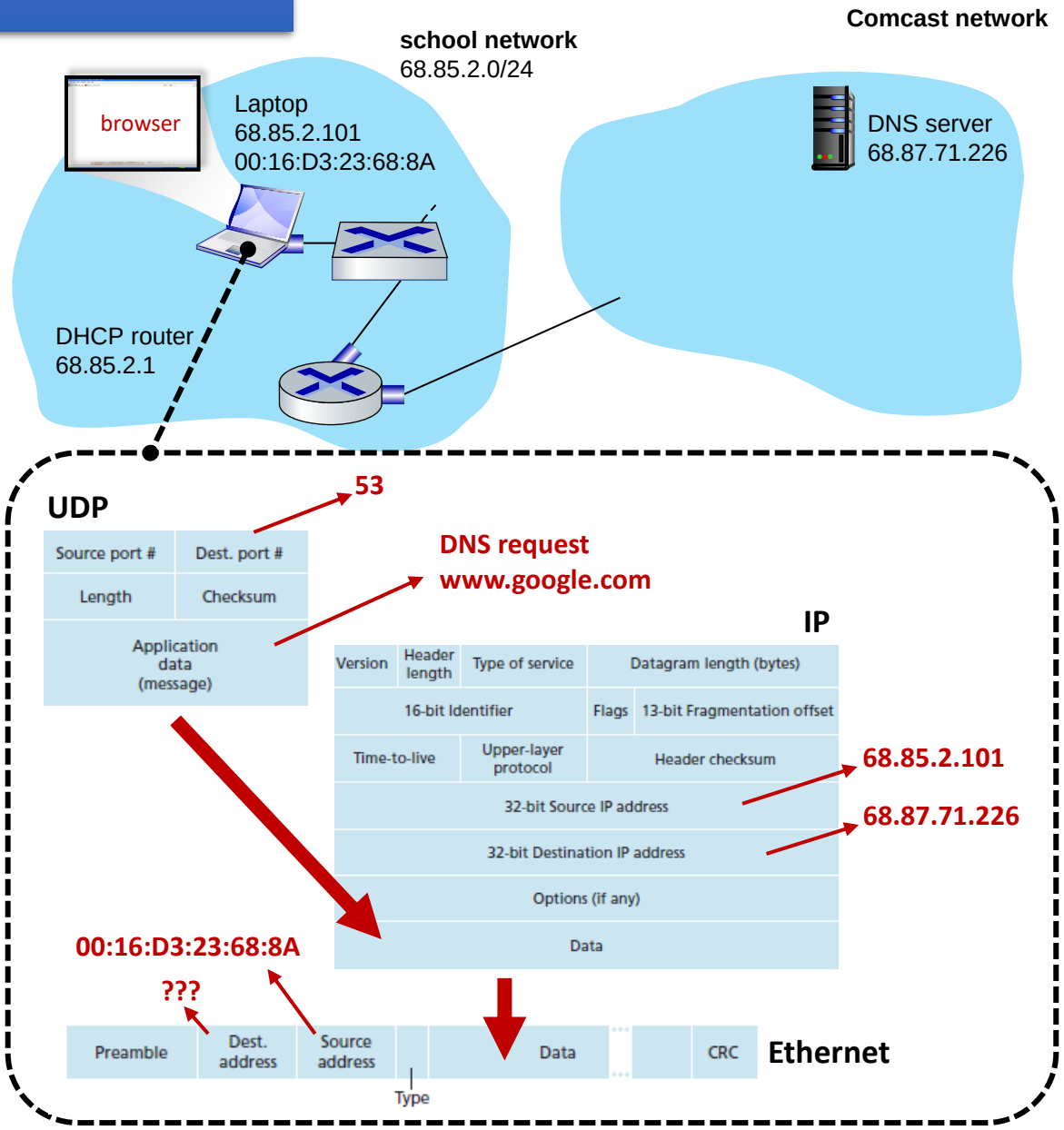
At the beginning, we only knew the MAC address of the laptop, **now we know**:

- The **IP of the laptop**.
- The **IP of the router** (gateway).
- The **network configuration** (mask).
- The **IP of the DNS**.

Stack Overview

DNS and ARP

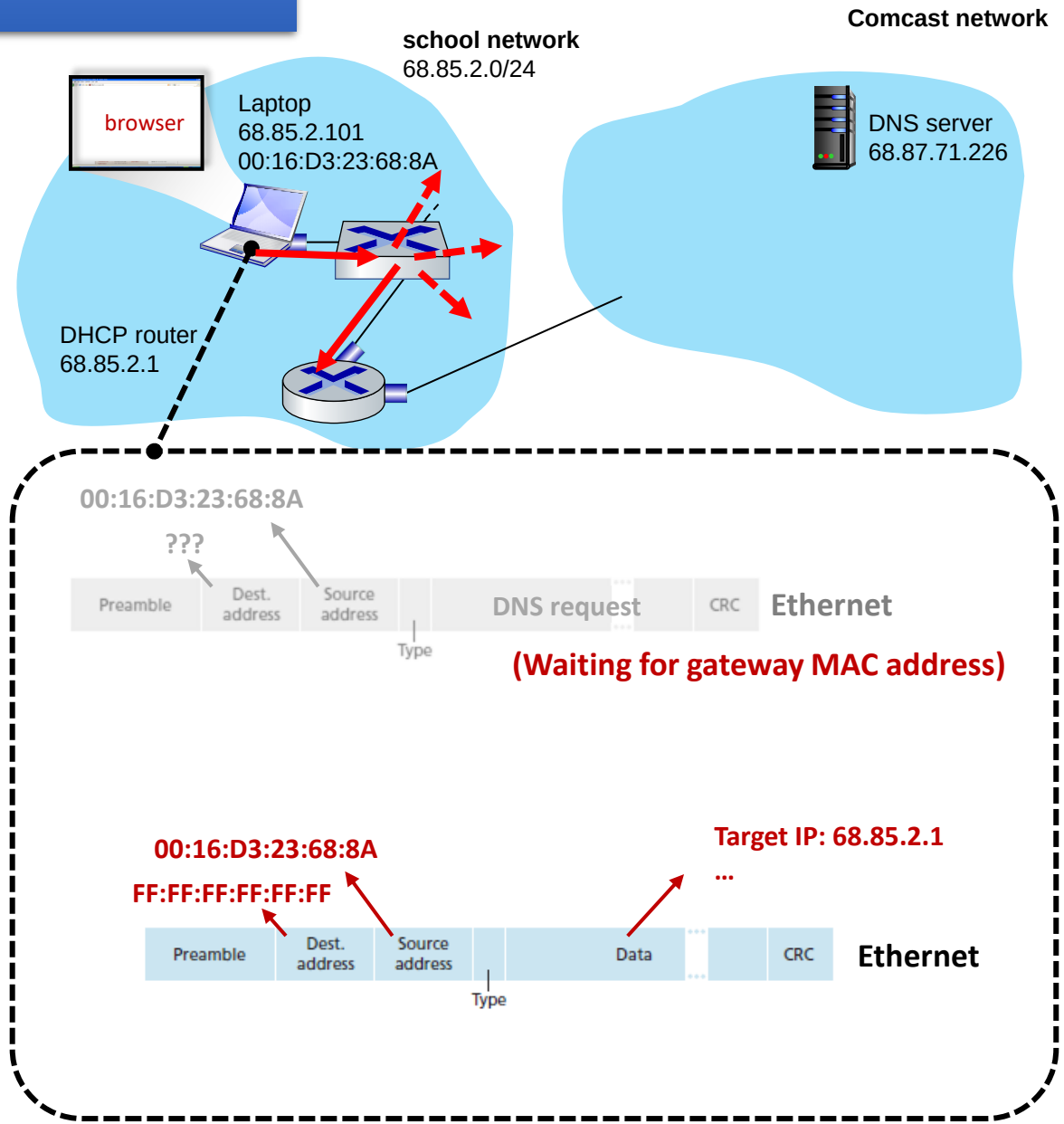
- Now **the user decides to open the browser and to navigate to the Google's home page** (www.google.com), hence the IP of the web server must be retrieved from the DNS:
  8. The **browser invokes an OS routine to get the server's IP** (as in gethostbyname function):
     - The **OS creates a DNS query** message for the "www.google.com" hostname.
     - The **DNS message is encapsulated within a UDP segment** having 53 as destination port (DNS server).
     - The **UDP segment is placed within an IP datagram** having 68.87.71.226 (IP of DNS retrieved from DHCP) as destination address and 68.85.2.101 as source IP address (self).
  9. The IP datagram containing the **DNS query must be encapsulated into an Ethernet frame**. To do so **we need the MAC address of the gateway**, hence an **ARP query** must be created.
     - Note that even if **we know the IP of the gateway** from the DHCP, **we are not aware of the MAC** address.



Stack Overview

ARP

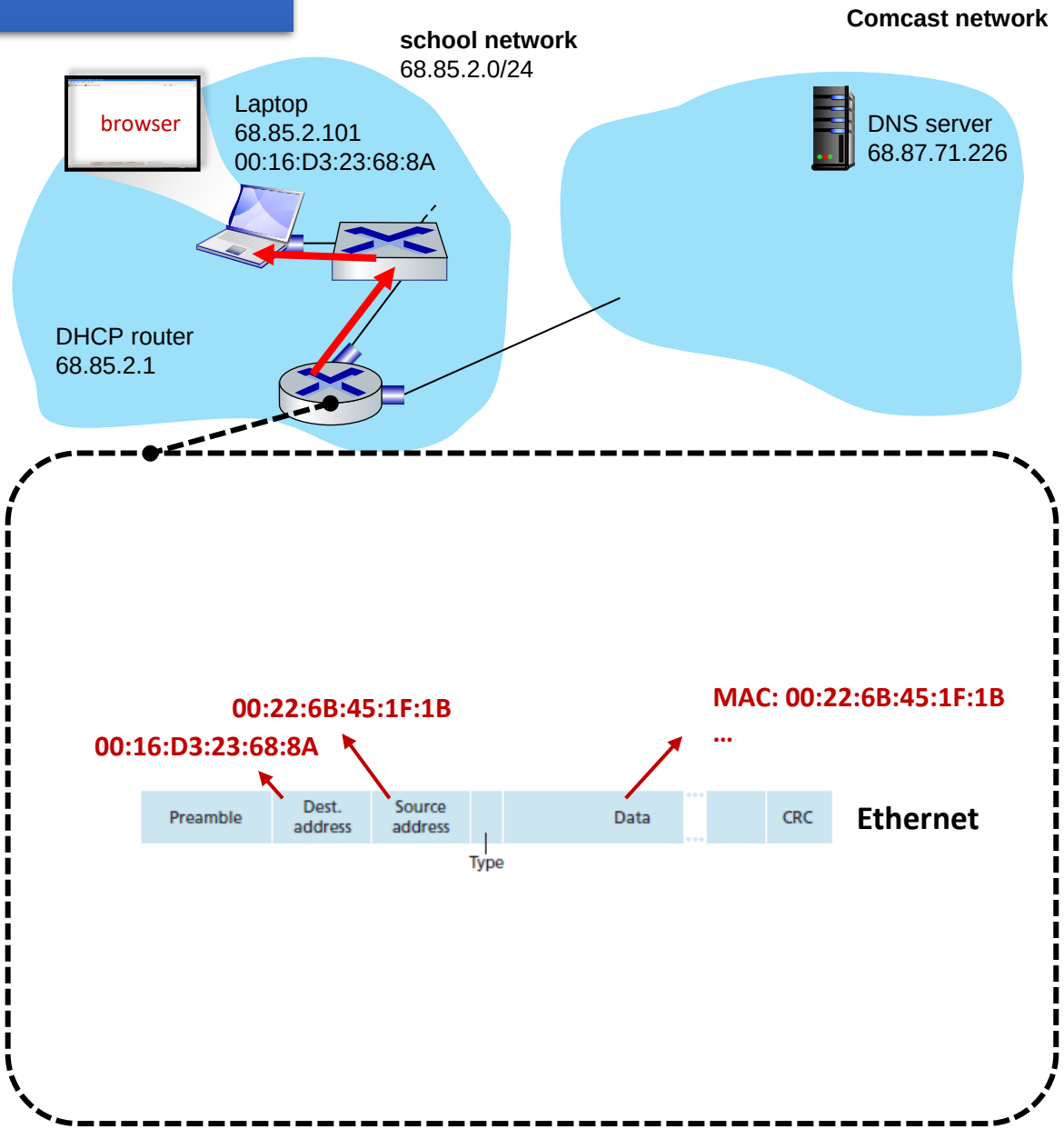
10. The **OS creates an ARP query** frame having a **target IP address of 68.85.2.1** (the default gateway) and broadcast (FF:FF:FF:FF:FF:FF) as destination MAC address.
    - The **frame is sent to the switch**, which delivers it **to all connected devices**, including the **gateway router**.



Stack Overview

ARP

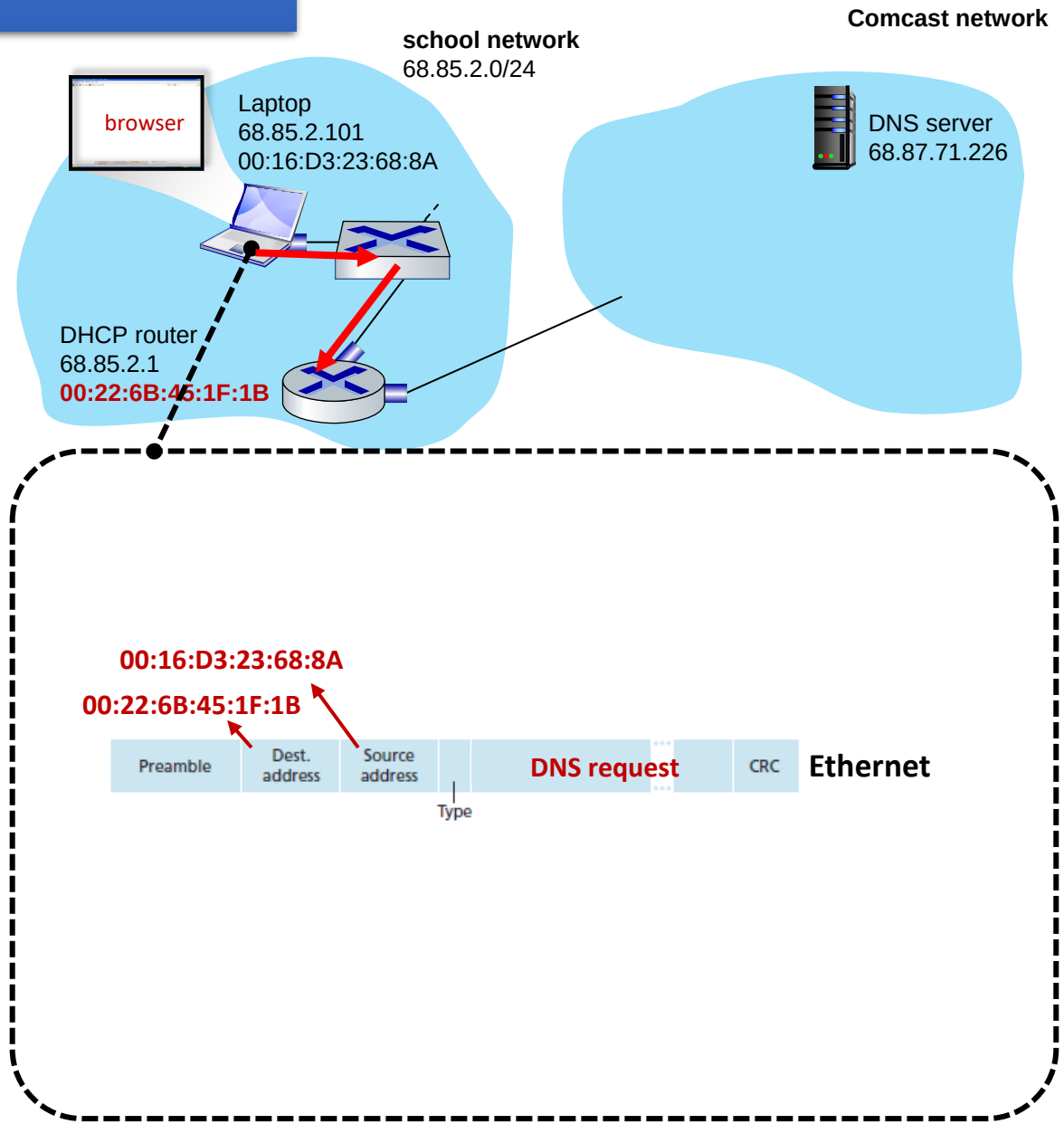
11. The **router receives the frame** on its LAN-side interface and finds that the target IP address of 68.85.2.1 in the **ARP message matches the IP address of the interface**:
    - The gateway **router prepares an ARP reply**, indicating that that the MAC address 00:22:6B:45:1F:1B corresponds to the IP address 68.85.2.1.
    - The **ARP reply message is put into an Ethernet frame** having 00:16:D3:23:68:8A as destination address (laptop address).
    - The **frame is sent to the switch**, which delivers it **to the laptop**.



Stack Overview

ARP

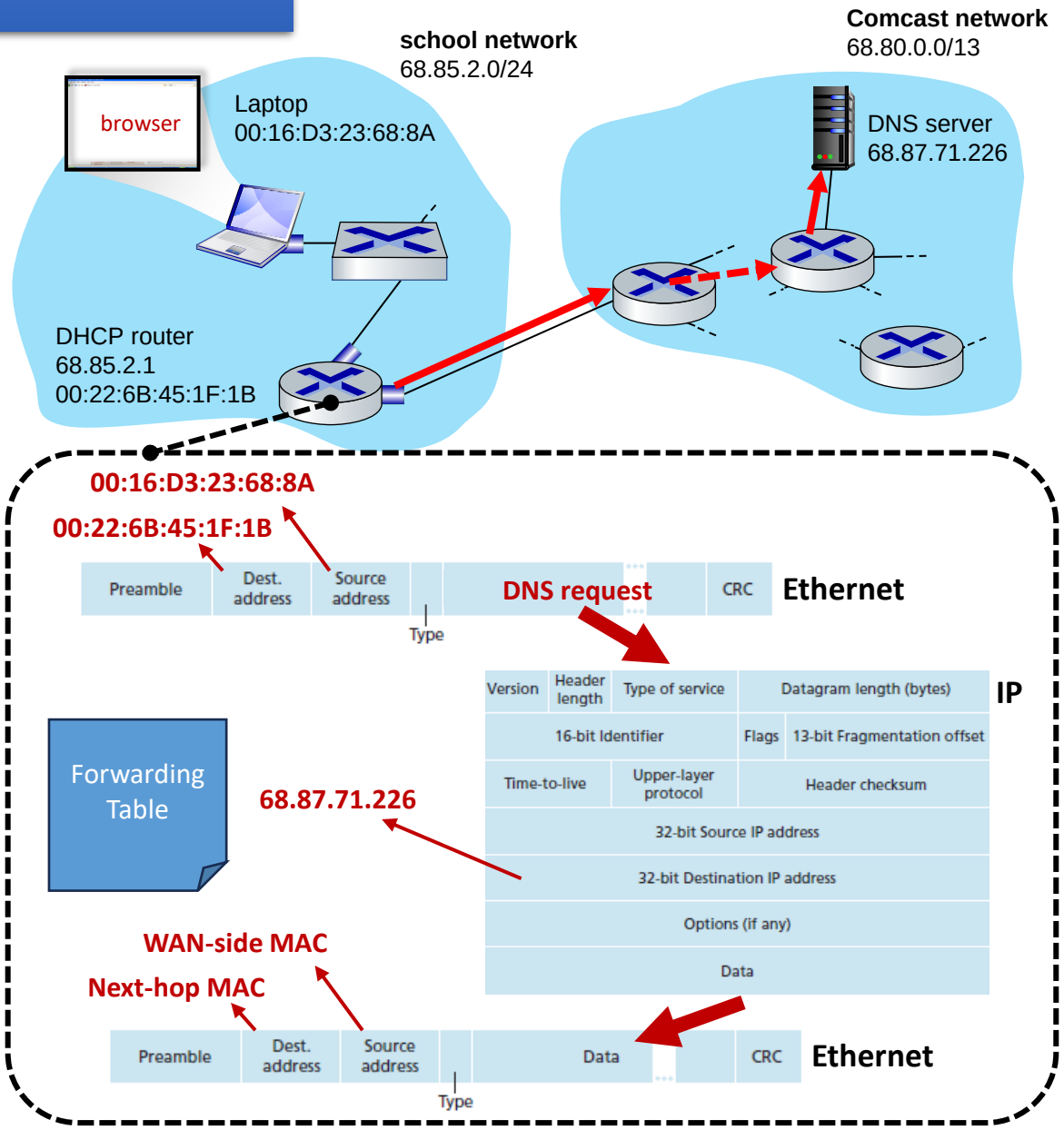
12. The **laptop receives the frame** containing the ARP reply message and extracts the MAC address of the gateway router (00:22:6B:45:1F:1B).
13. The **laptop can now send the Ethernet frame containing the DNS query** to the gateway's MAC address.
    - Note that, in this case, the **IP datagram** will have a **destination IP address of 68.87.71.226** (the DNS server), and a **destination MAC address of 00:22:6B:45:1F:1B** (the gateway router).



Stack Overview

DNS

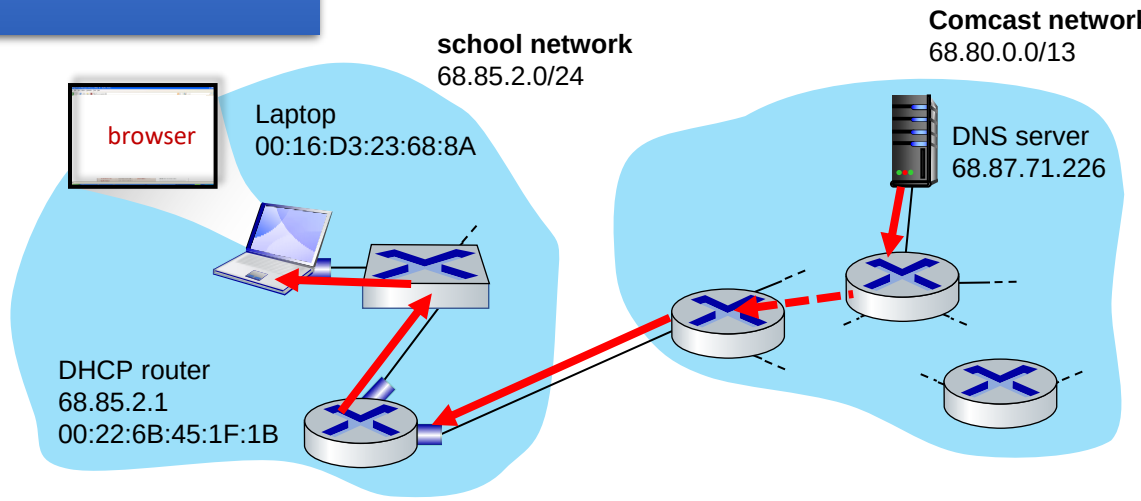
14. The gateway **router receives the Ethernet frame** and decapsulates it:
    - The **router extracts the IP datagram** from the frame and retrieve the destination IP of the DNS.
    - The **router looks up the destination IP** (68.87.71.226) and determines from its forwarding table that the **datagram should be sent to the first router of the ISP network** (leftmost router in the Comcast network).
    - The IP **datagram is placed inside a link-layer frame** appropriate for the link connecting the school's router to the leftmost Comcast router and the frame is sent over this link.
15. The **first router of the ISP receives the frame** and extracts the IP datagram:
    - The **router checks the destination address** (68.87.71.226) and retrieves from the forwarding table (created through a link-state or distance-vector algorithm) the outgoing interface.
    - A **new frame is then created and sent to the next router** through the selected interface.
    - **This process may be iterated** several times before the DNS server is reached.



Stack Overview

DNS

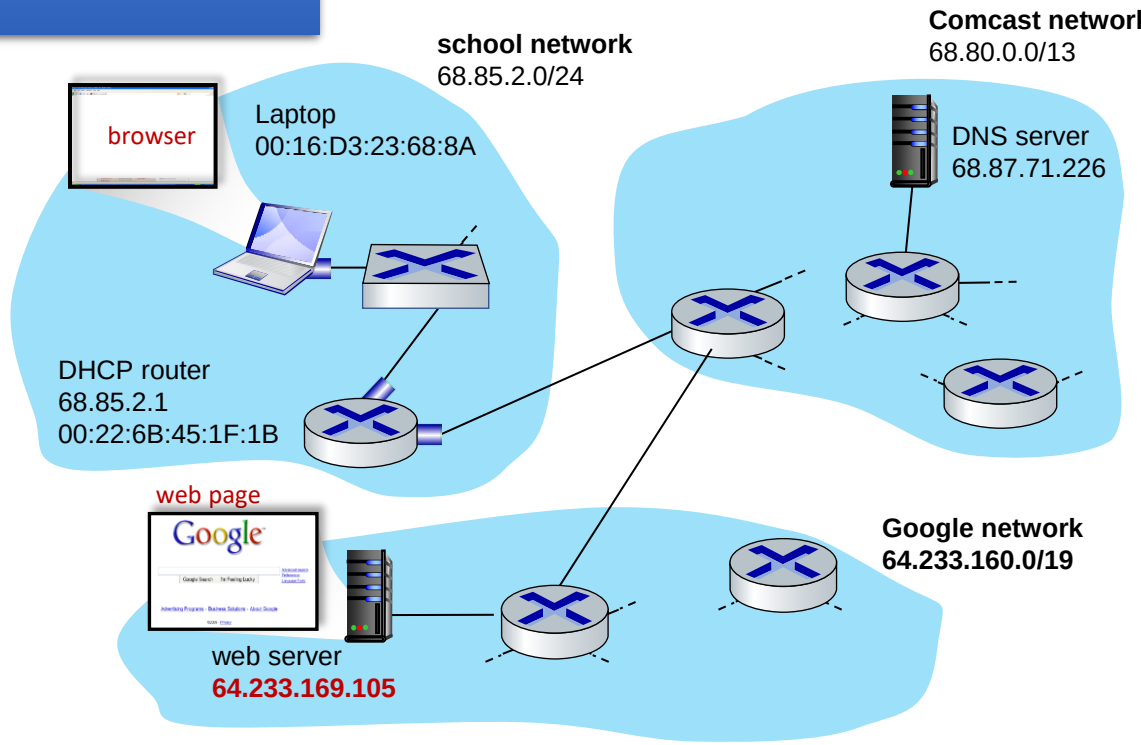
16. Eventually the IP datagram containing **the DNS query arrives at the DNS** server:
    - The DNS server extracts the DNS query message, **looks up the name www.google.com from its database** and finds the IP address 64.233.169.105 (Google's server).
        - Notice **that here we are assuming a cached IP**, otherwise additional requests to authoritative servers should be sent.
    - The DNS **server creates a DNS reply message** containing the retrieved IP address, and places it **in a UDP segment**.
    - The **segment is put in an IP datagram** having destination IP of 68.85.2.101 (the laptop) and then into the appropriate frame.
    - This **message will be forwarded back** through the Comcast network to the school's router and from there, via the switch to the laptop.



Stack Overview

DNS

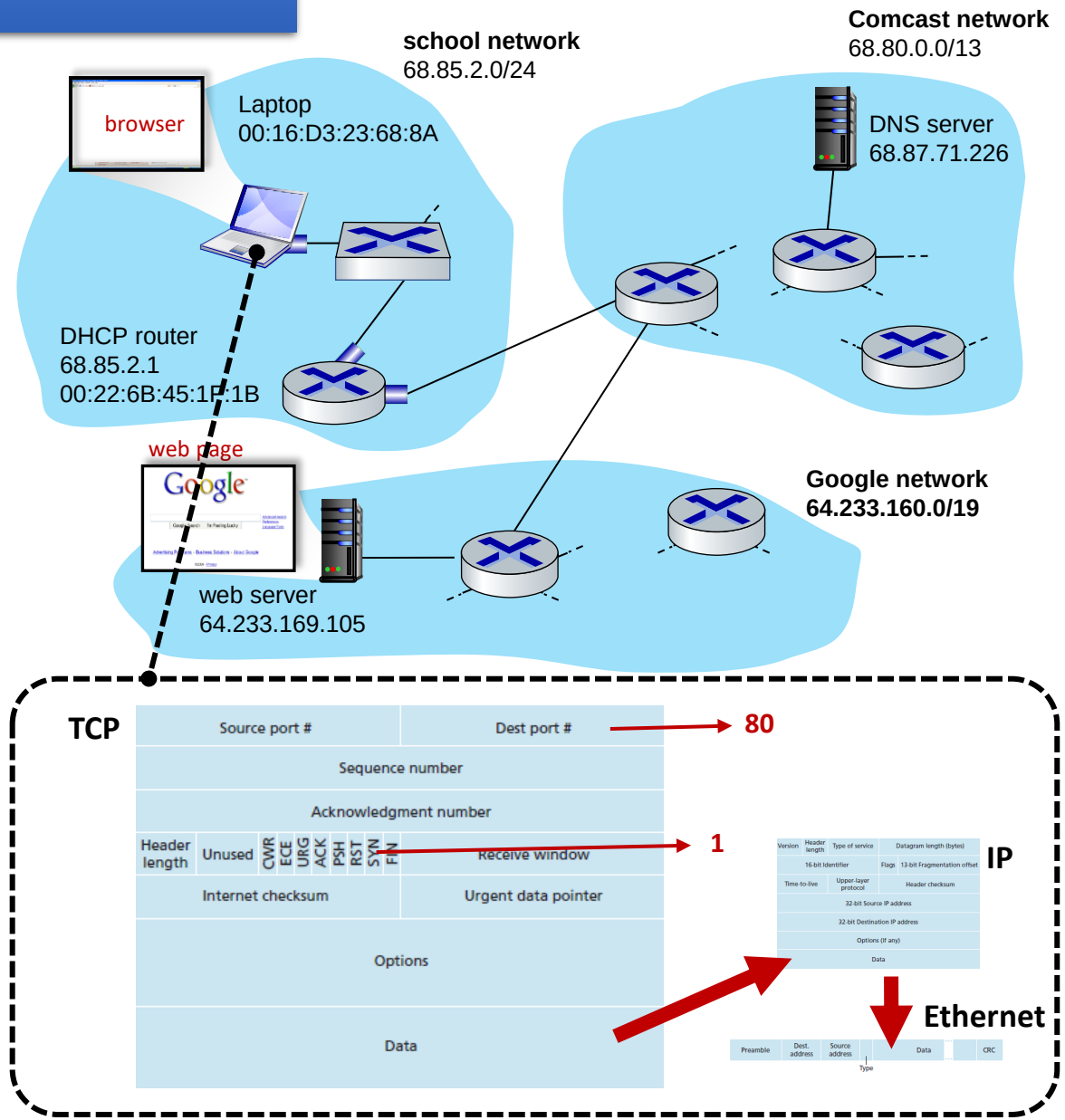
16. Eventually the IP datagram containing **the DNS query arrives at the DNS** server:
    - The DNS server extracts the DNS query message, **looks up the name www.google.com from its database** and finds the IP address 64.233.169.105 (Google's server).
        - Notice **that here we are assuming a cached IP**, otherwise additional requests to authoritative servers should be sent.
    - The DNS **server creates a DNS reply message** containing the retrieved IP address, and places it **in a UDP segment**.
    - The **segment is put in an IP datagram** having destination IP of 68.85.2.101 (the laptop) and then into the appropriate frame.
    - This **message will be forwarded back** through the Comcast network to the school's router and from there, via the switch to the laptop.
17. The **OS of the laptop extracts the target IP** address from the DNS message. **Now we know how to reach the Google web server**.



Stack Overview

HTTP

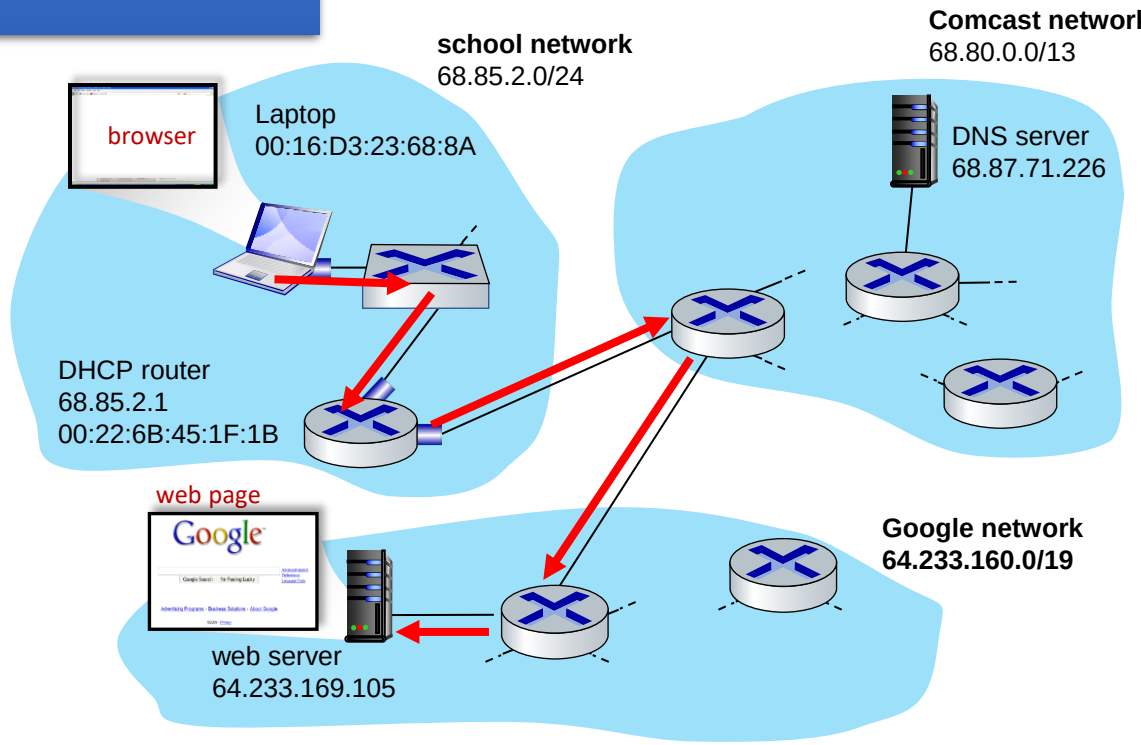
- Knowing the IP of the server **we can now create a HTTP GET request** for the google home page:
  18. The **browser creates a TCP socket**, so the **OS performs a three-way handshake** with the Google web server:
      - The **OS creates a TCP SYN segment** with destination port 80 (HTTP).
      - The **segment is encapsulated inside an IP datagram** having a destination IP address of 64.233.169.105 (www.google .com).
      - The **datagram is put inside a frame** with a destination MAC address of 00:22:6B:45:1F:1B (the gateway router) and sent to the switch.



Stack Overview

HTTP

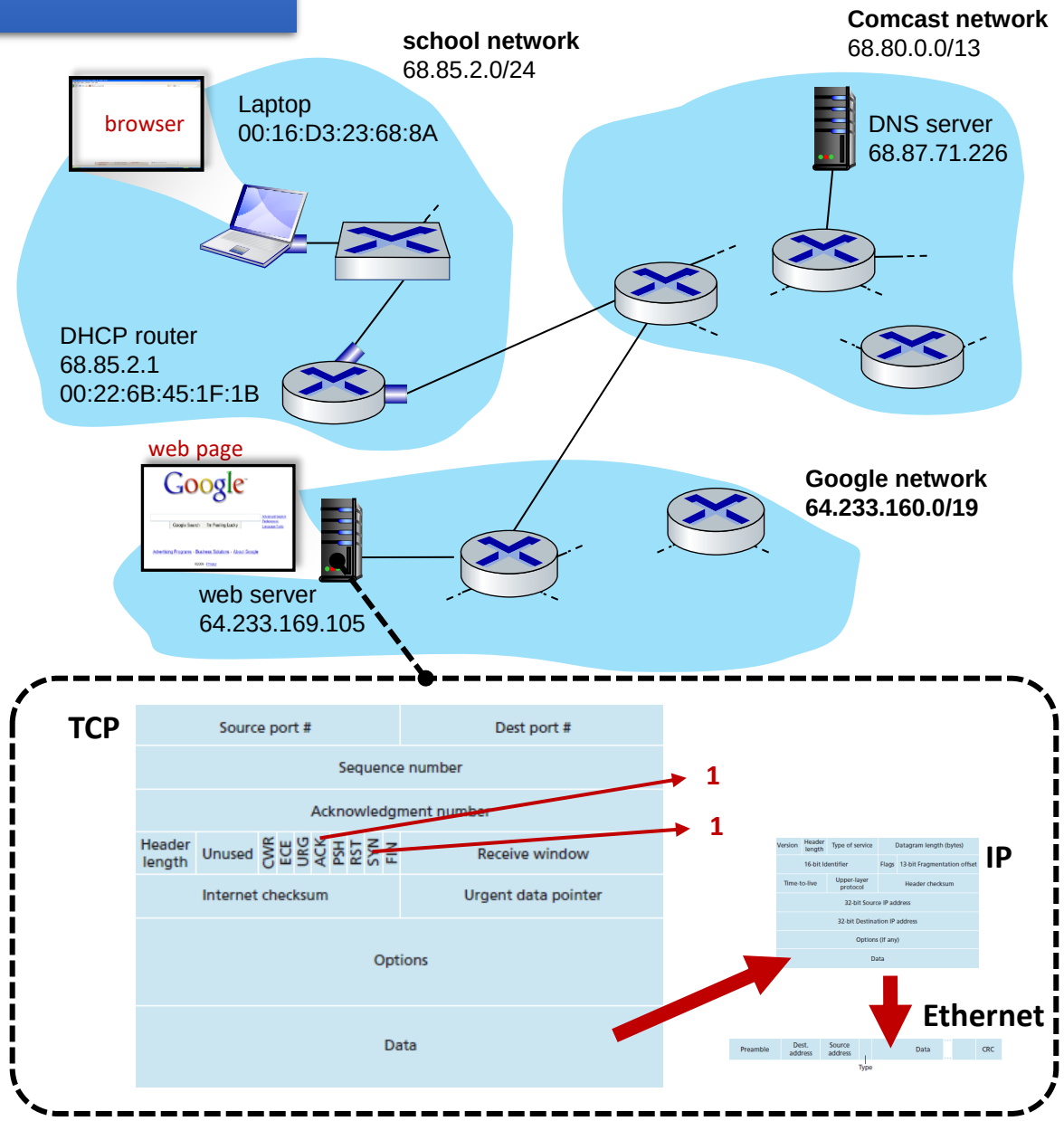
- Knowing the IP of the server **we can now create a HTTP GET request** for the google home page:
  18. The **browser creates a TCP socket**, so the **OS performs a three-way handshake** with the Google web server:
      - The **OS creates a TCP SYN segment** with destination port 80 (HTTP).
      - The **segment is encapsulated inside an IP datagram** having a destination IP address of 64.233.169.105 (www.google .com).
      - The **datagram is put inside a frame** with a destination MAC address of 00:22:6B:45:1F:1B (the gateway router) and sent to the switch.
  19. All **routers (from local, ISP, and Google networks) forward the datagram** to the web server using their forwarding table.
      - Notice that **frames can be modified along the path depending on the specific links**.



Stack Overview

HTTP

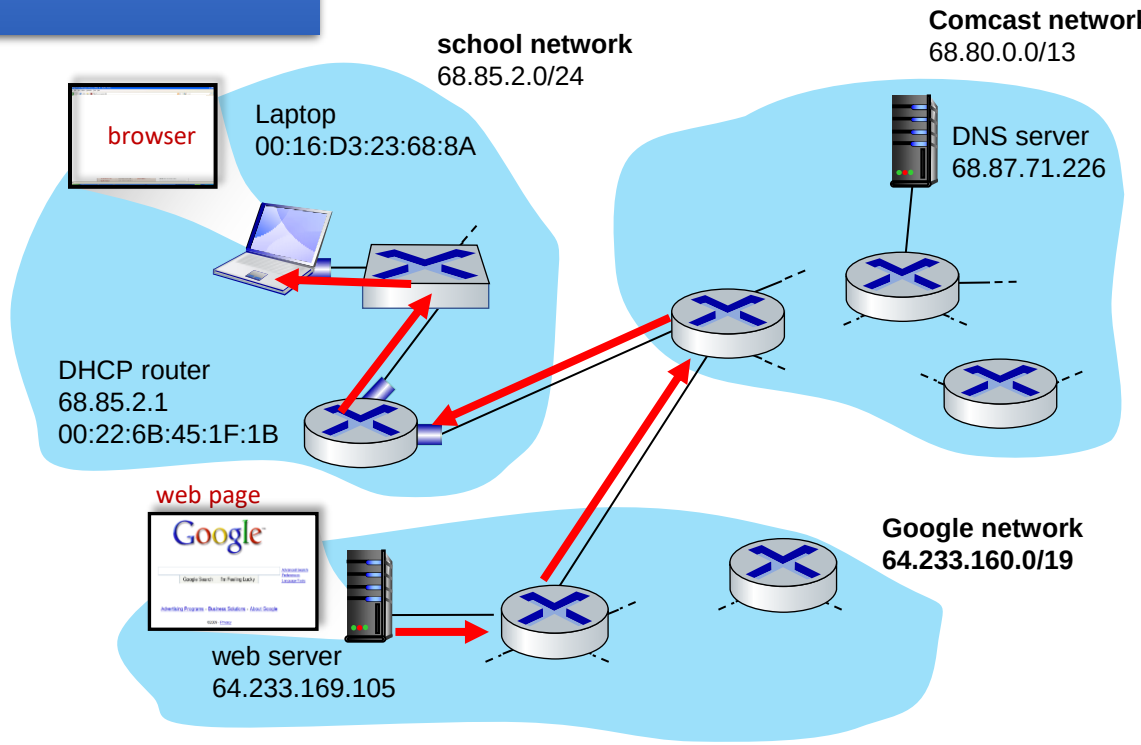
20. The frame containing the **SYN segment eventually arrives at the server**:
    - The **datagram is extracted** from the frame.
    - The **segment is extracted** from the datagram and **demultiplexed** to the welcome socket associated with port 80.
    - A **connection-specific socket is created** between the Google web server and the laptop.
    - A **TCP SYNACK segment is generated**, placed inside a datagram having destination address of 68.85.2.101 (laptop).
    - The **segment is placed into an appropriate link-layer frame** to travel the link connecting www.google.com to its first-hop router.



Stack Overview

HTTP

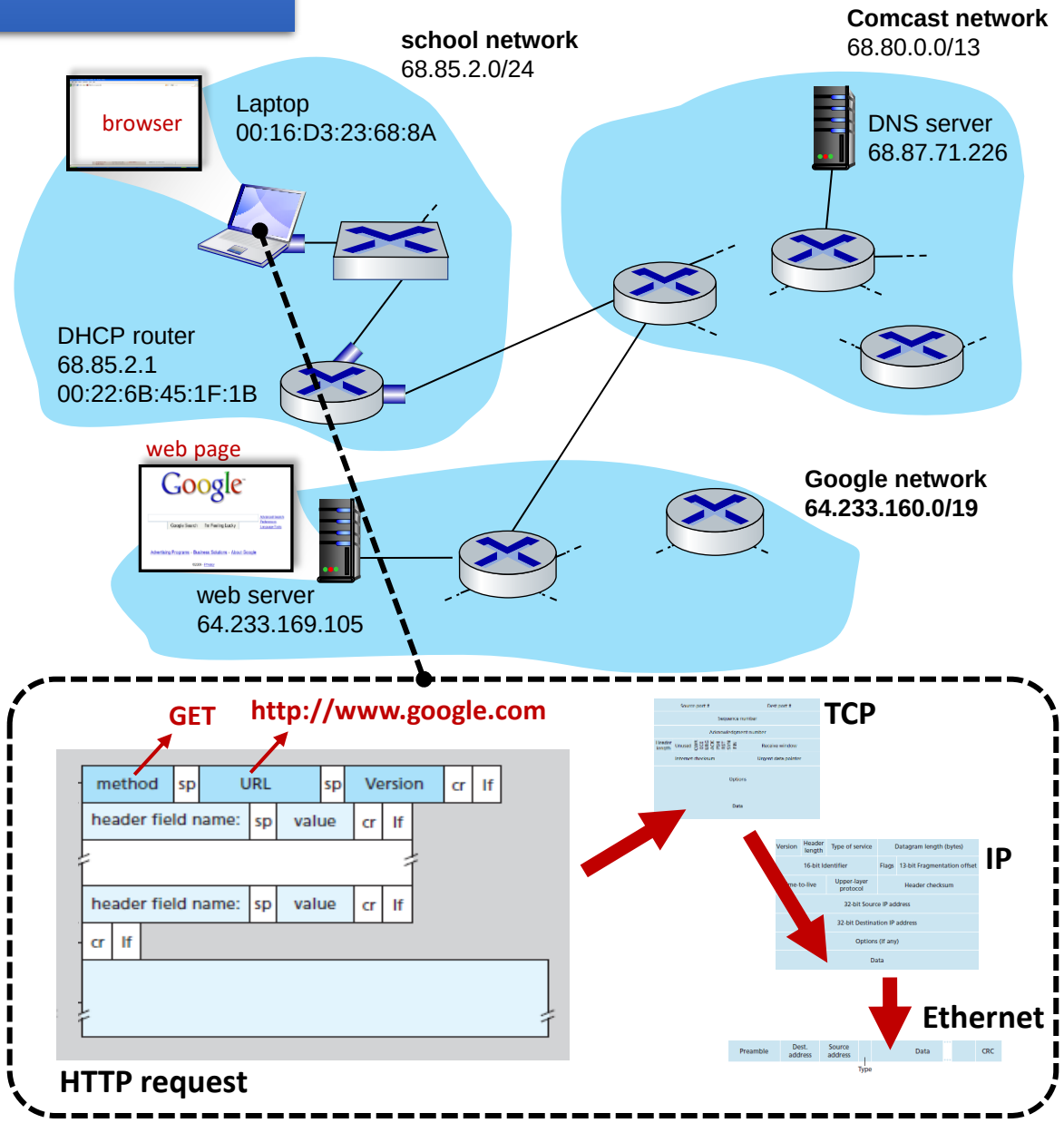
20. The frame containing the **SYN segment eventually arrives at the server**:
    - The **datagram is extracted** from the frame.
    - The **segment is extracted** from the datagram and **demultiplexed** to the welcome socket associated with port 80.
    - A **connection-specific socket is created** between the Google web server and the laptop.
    - A **TCP SYNACK segment is generated**, placed inside a datagram having destination address of 68.85.2.101 (laptop).
    - The **segment is placed into an appropriate link-layer frame** to travel the link connecting www.google.com to its first-hop router.
21. The **TCP SYNACK eventually arrives at the laptop**:
    - The **datagram is demultiplexed** by the OS to the TCP socket created in step 18.
    - The **socket enters the "connection established" state**.



Stack Overview

HTTP

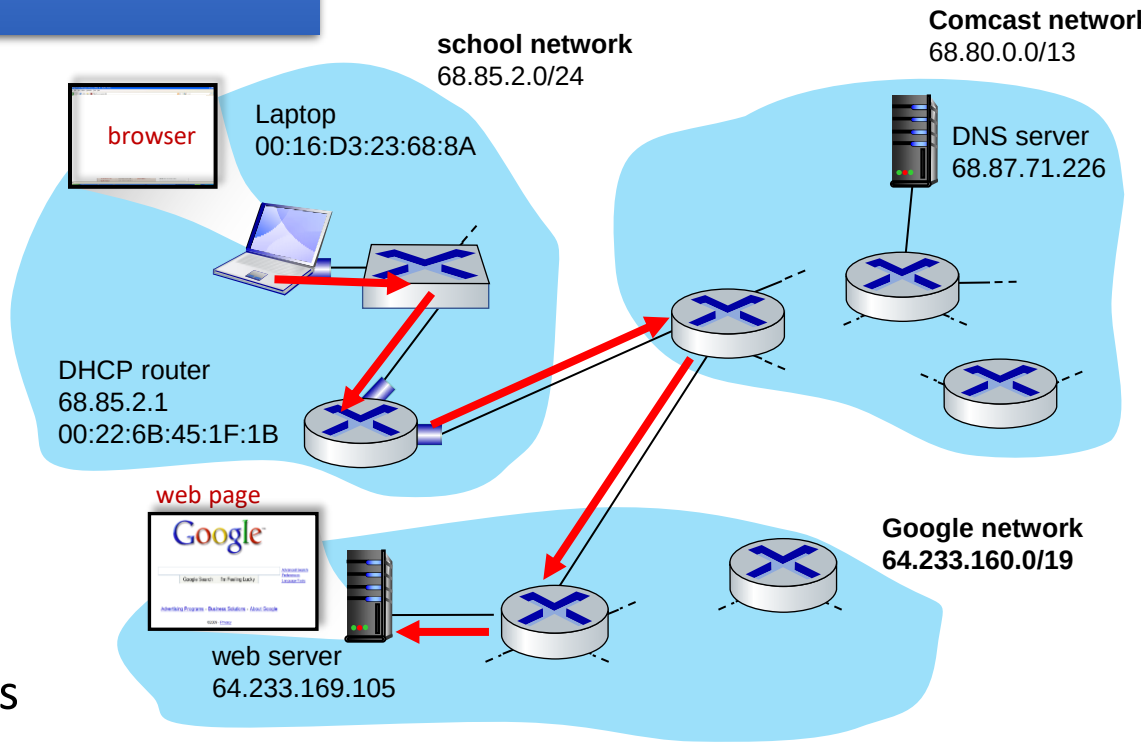
22. The **socket on the laptop now ready to send bytes** to web server:
    - The **browser creates the HTTP GET message** containing the URL to be fetched.
    - The **message is written into the socket**, and the GET request becomes the **payload of a TCP segment**.
    - The TCP **segment is placed in a datagram and then into a frame** (encapsulation).



Stack Overview

HTTP

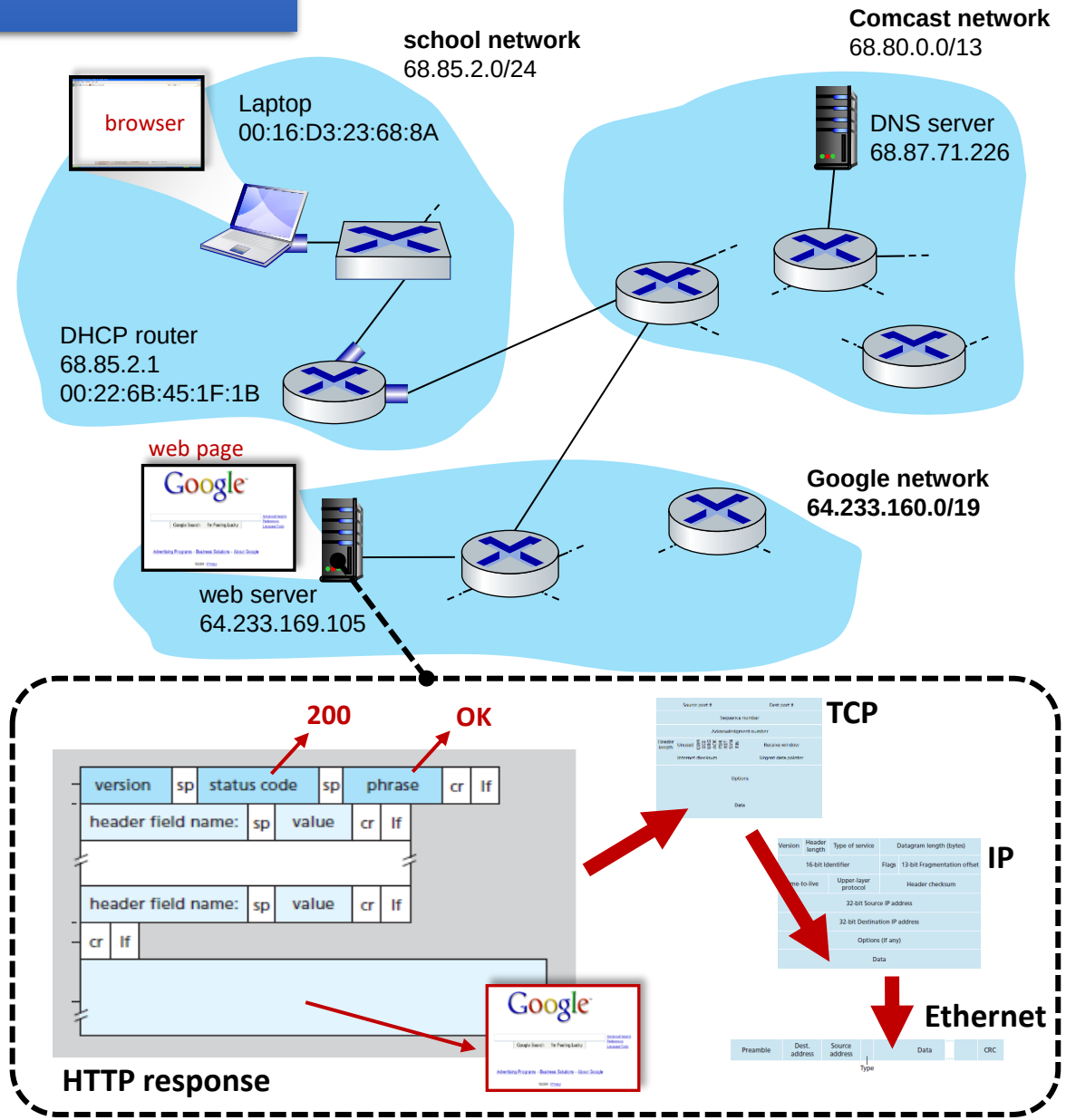
22. The **socket on the laptop now ready to send bytes** to web server:
    - The **browser creates the HTTP GET message** containing the URL to be fetched.
    - The **message is written into the socket**, and the GET request becomes the **payload of a TCP segment**.
    - The TCP **segment is placed in a datagram and then into a frame** (encapsulation).
    - The **message is delivered to www.google.com** (as in steps 18-20).



Stack Overview

HTTP

23. The Web **server at www.google.com receives the GET message** from the TCP socket:
    - The **message is decapsulated and demultiplexed**, then the **GET request is interpreted**.
    - The **server creates an HTTP response message**, having **the Google home page in the body**.
    - The **message is written into the TCP socket**.
    - The **server's OS encapsulates the message** into a segment, a datagram, and finally a frame.



Stack Overview

HTTP

24. The datagram containing the **HTTP reply message is forwarded through the 3 networks**, and arrives at the laptop:
    - The **laptop's OS decapsulates the message**, which is **demultiplexed to the browser**.
    - The **browser reads the HTTP response** from the socket.
    - The **browser extracts the html code from the body** of the HTTP response
    - The **browser finally displays the Web page**!