CSCI 466: Networks

Putting it all together

Reese Pearsall

Fall 2024

Announcements

Quiz 4 on Friday (no lecture)

- IP Addresses
- Subnets
- NAT
- Routing (LS, DV, OSPF, BGP)
- Link Layer
- Mac Addresses
- Multiple Access Protocols
- ARP
- Ethernet
- Switches

Wireshark Lab 3 due on Monday (10/21)

OSI Model

Application Layer

Presentation Layer *

Session Layer *

Transport Layer

Network Layer

Data Link Layer

Physical Layer

Application Layer

Messages from Network Applications

↓

Physical Layer

Bits being transmitted over a copper wire

**In the textbook, they condense it to a 5-layer model, but 7 layers is what is most used*

OSI Model

Application Layer

Presentation Layer

Session Layer *

Transport Layer

Network Layer

Data Link Layer

Physical Layer

We will cover these briefly in the coming weeks

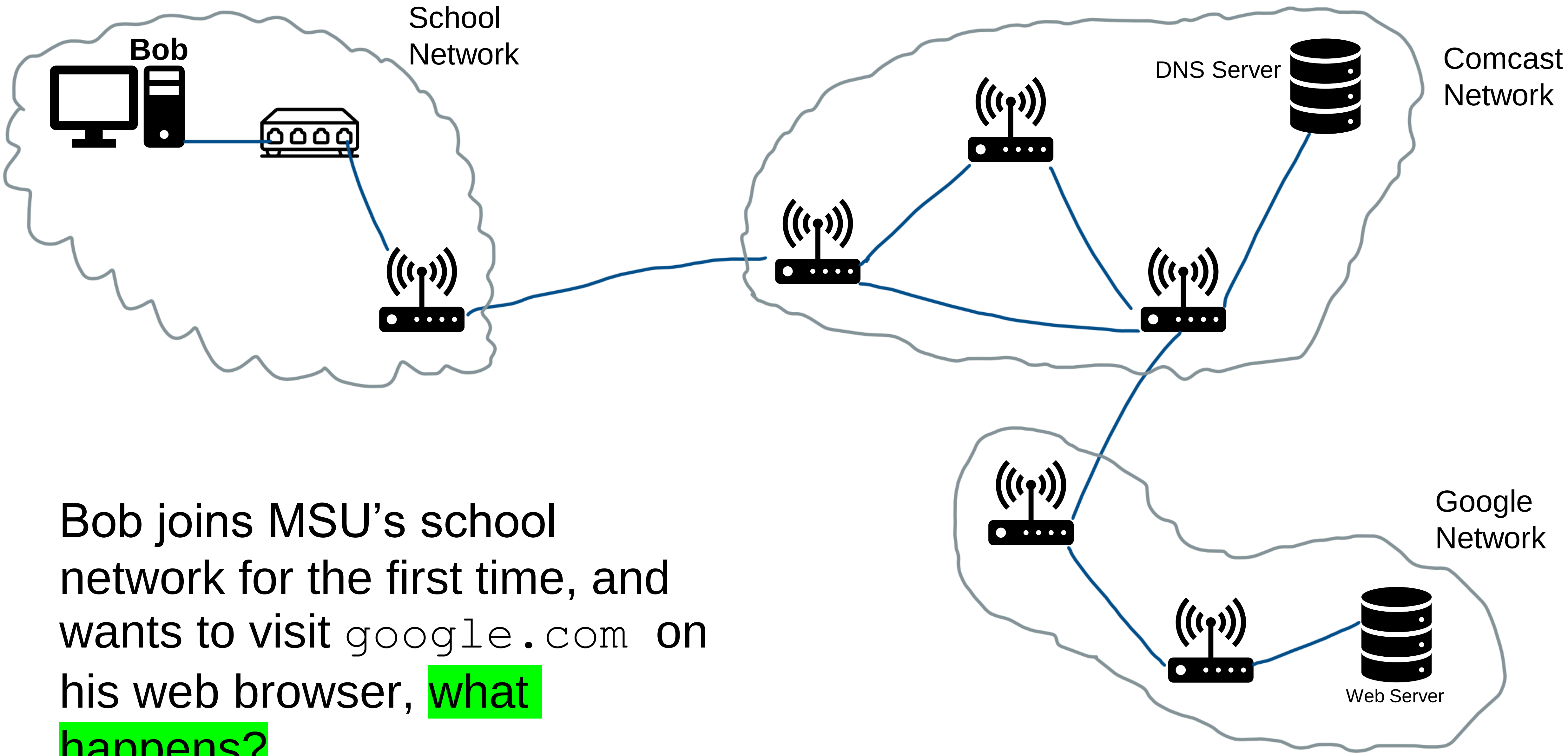
Application Layer

Messages from Network Applications

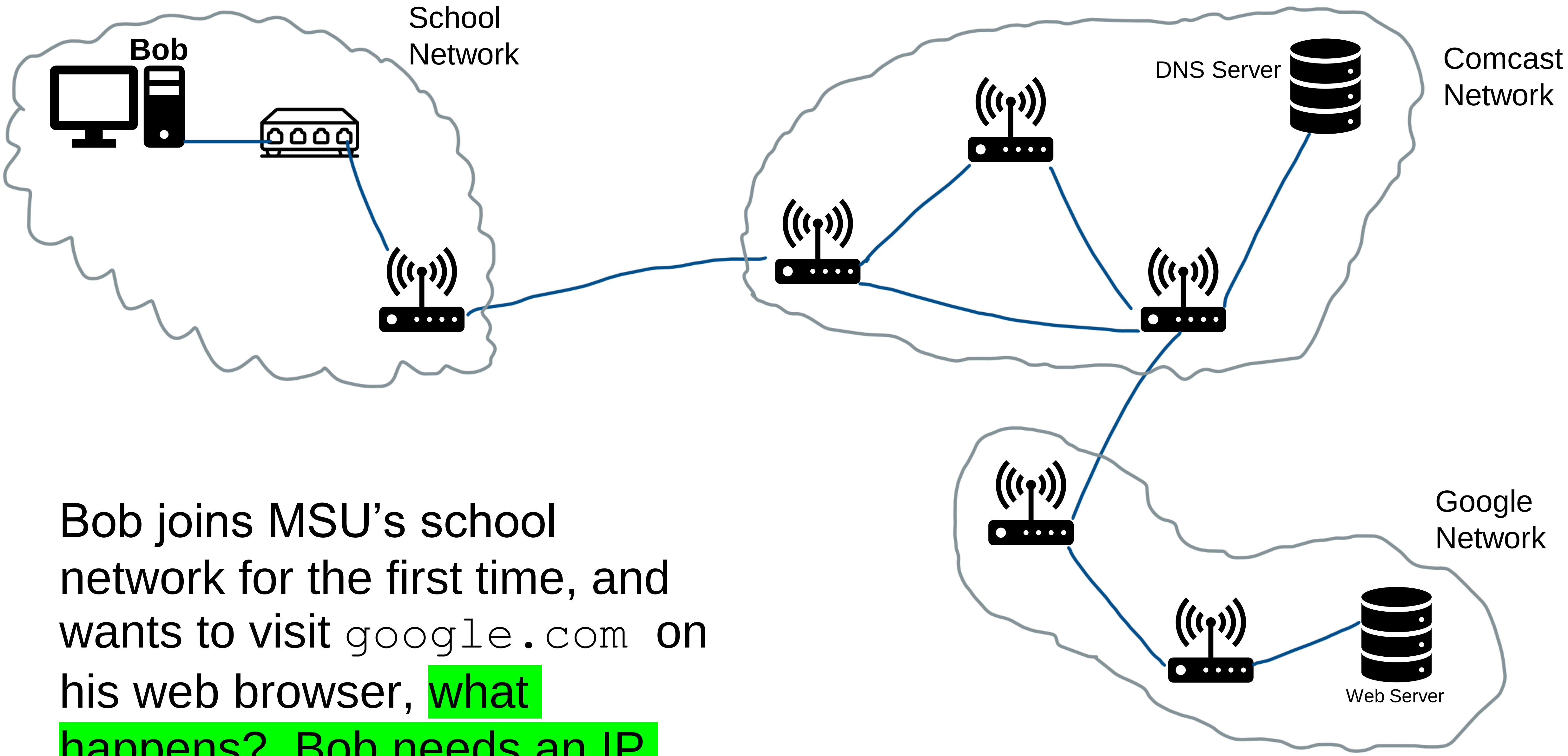
Physical Layer

Bits being transmitted over a copper wire

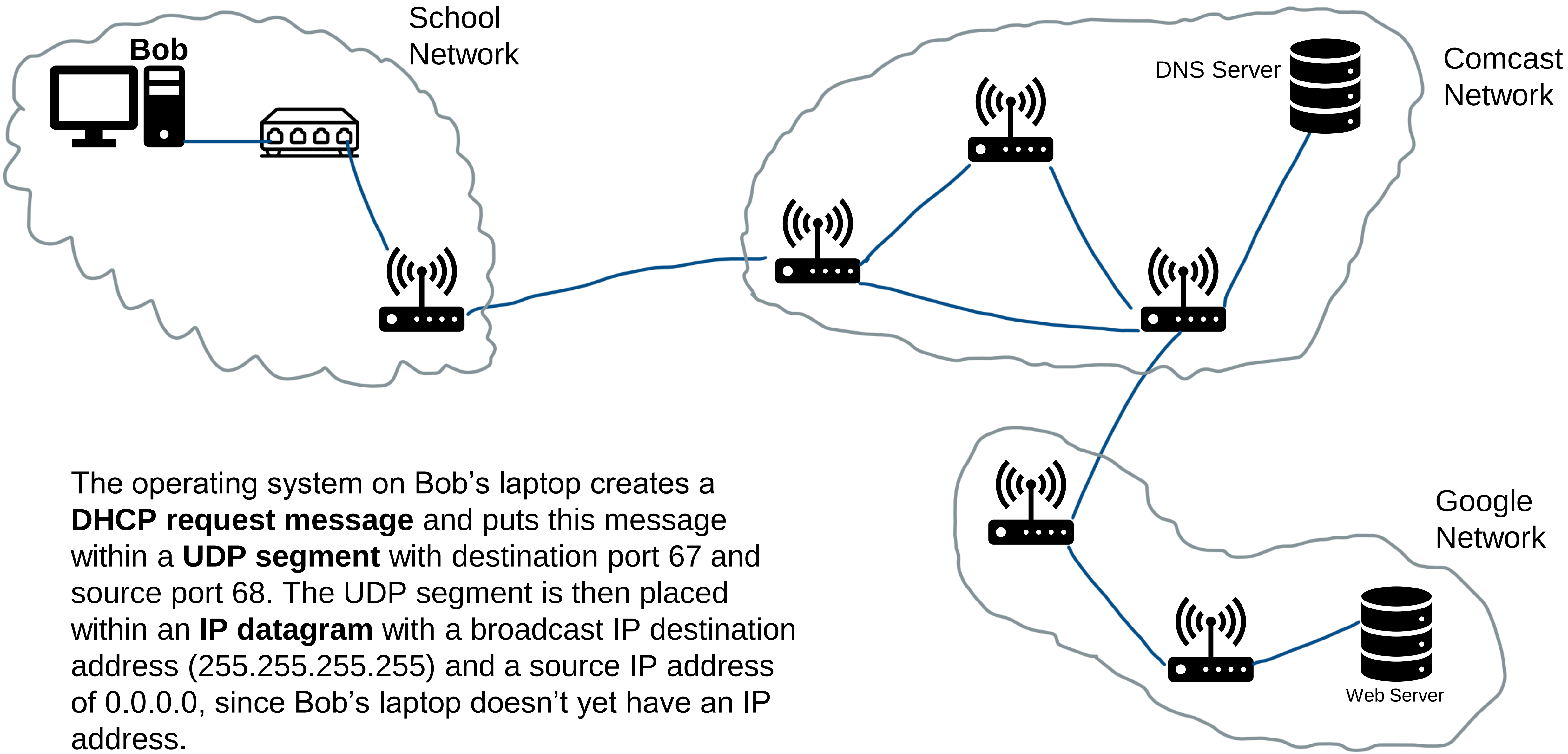
**In the textbook, they condense it to a 5-layer model, but 7 layers is what is most used*



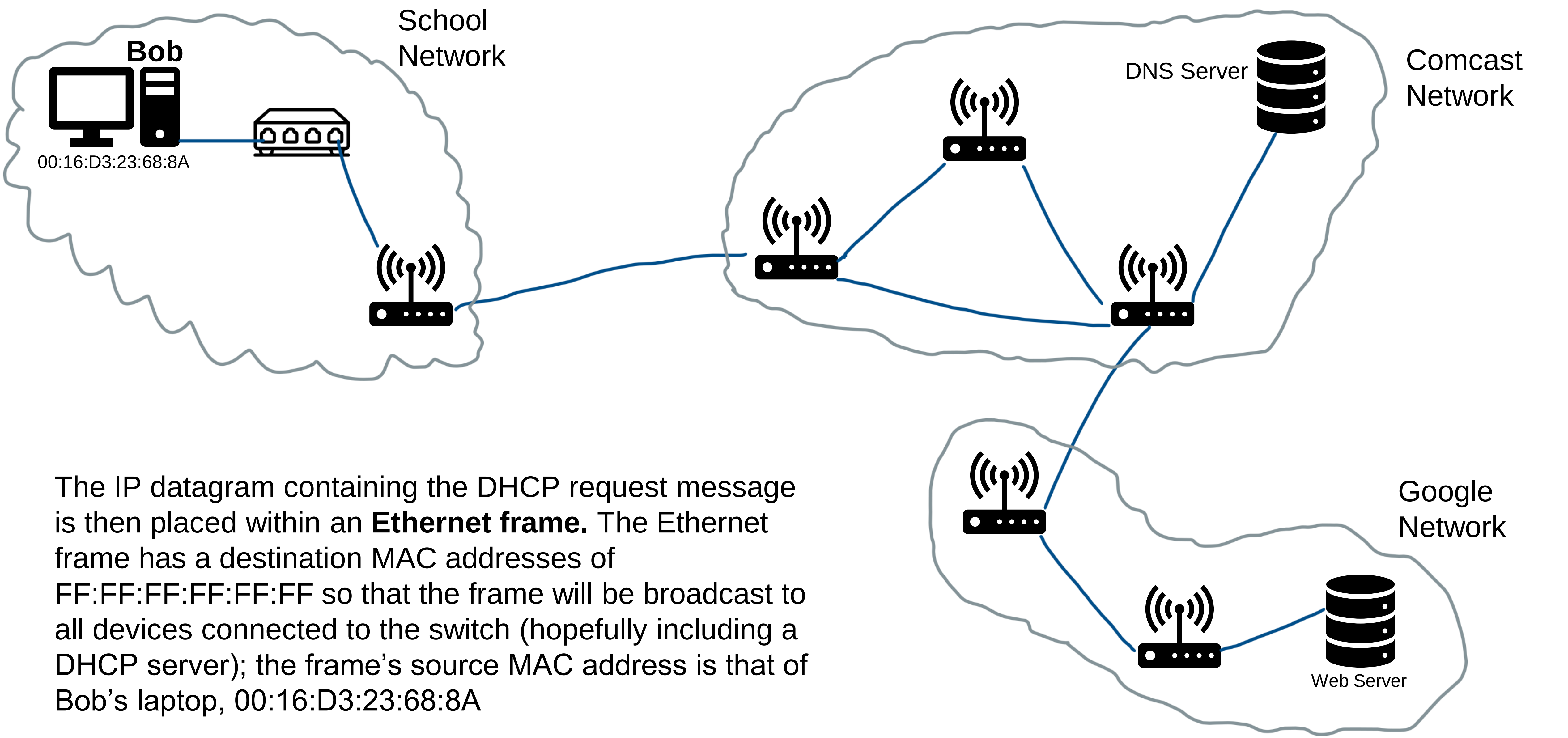
Bob joins MSU's school network for the first time, and wants to visit `google.com` on his web browser, what happens?



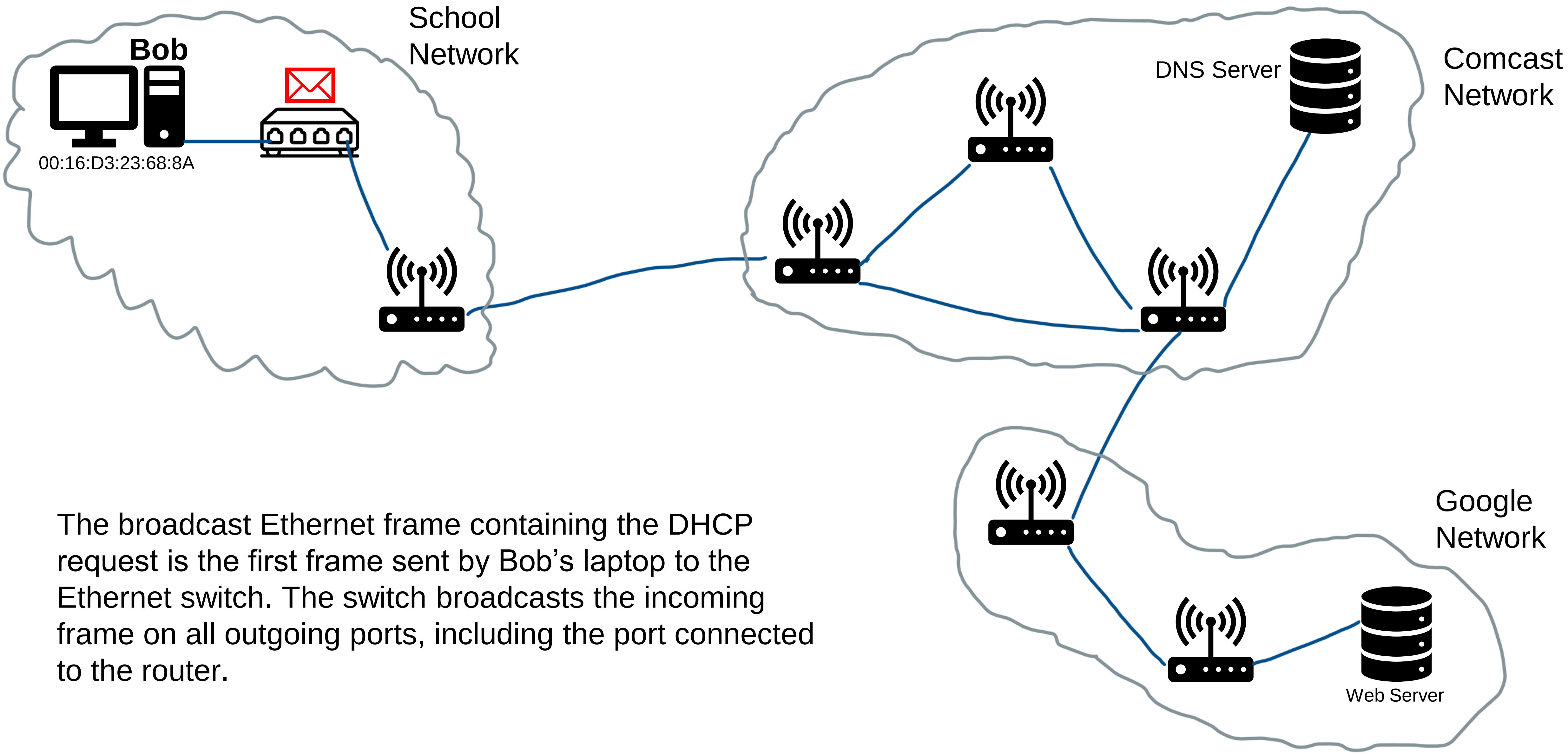
Bob joins MSU's school network for the first time, and wants to visit `google.com` on his web browser, what happens? Bob needs an IP address!



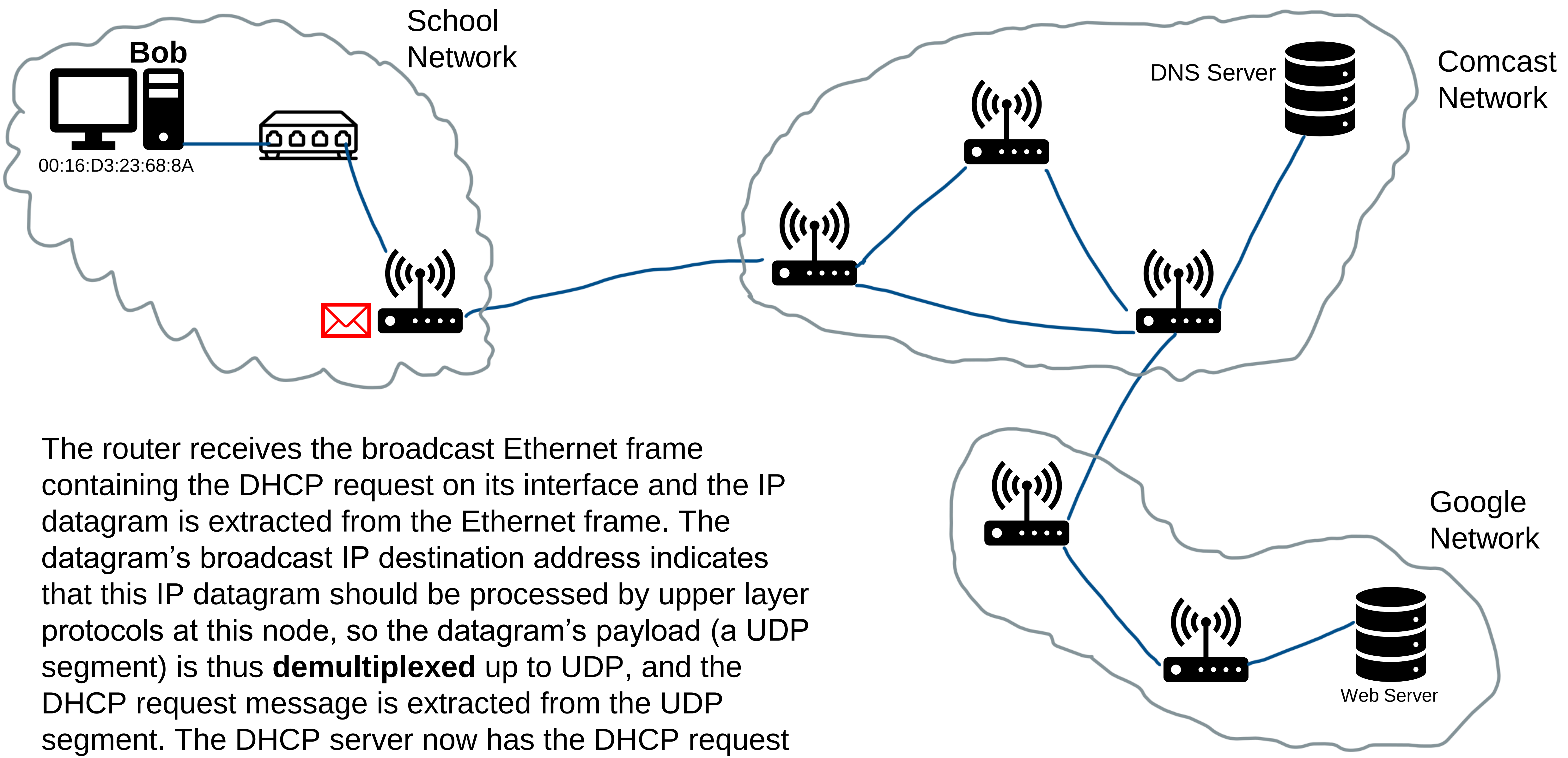
The operating system on Bob's laptop creates a **DHCP request message** and puts this message within a **UDP segment** with destination port 67 and source port 68. The UDP segment is then placed within an **IP datagram** with a broadcast IP destination address (255.255.255.255) and a source IP address of 0.0.0.0, since Bob's laptop doesn't yet have an IP address.



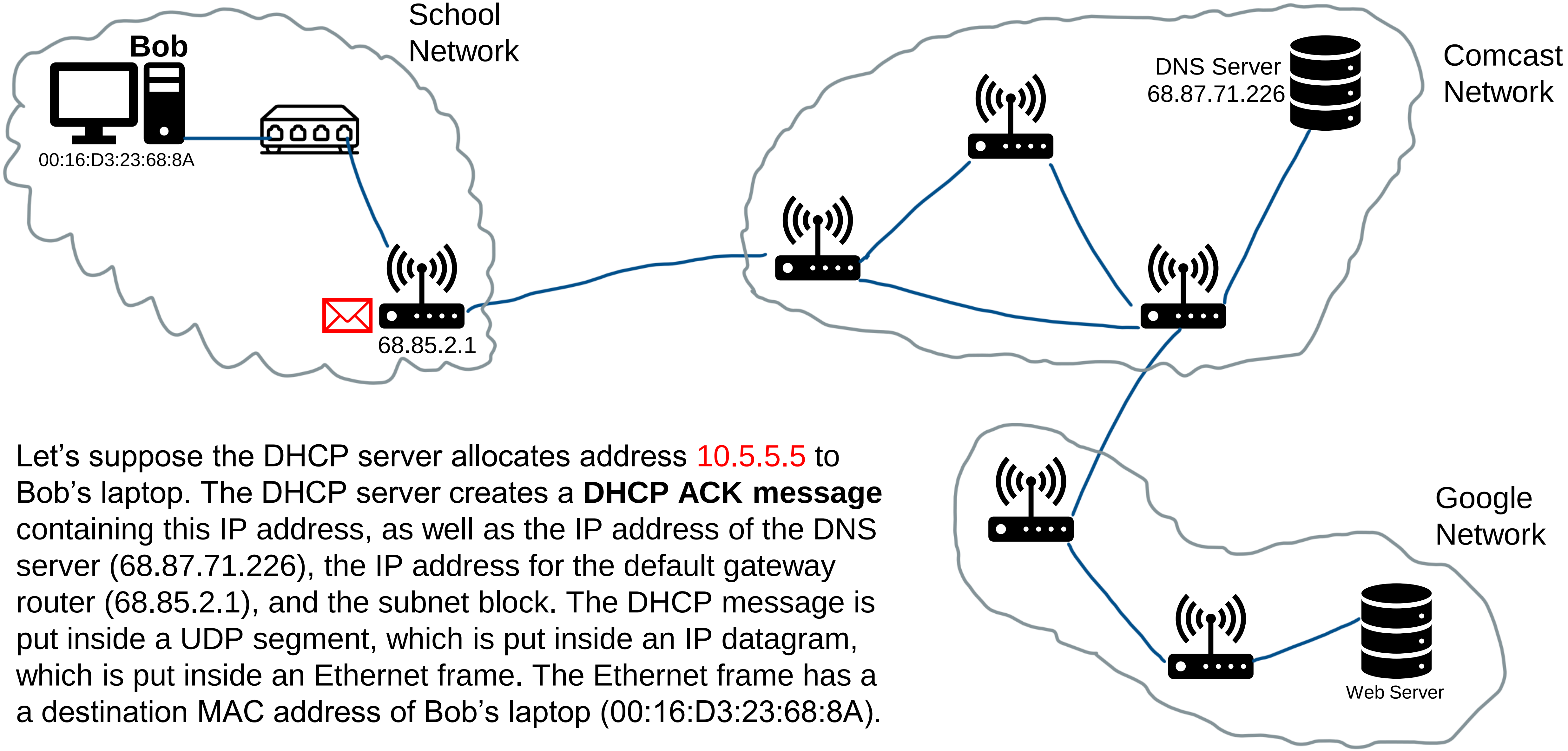
The IP datagram containing the DHCP request message is then placed within an **Ethernet frame.** The Ethernet frame has a destination MAC addresses of FF:FF:FF:FF:FF:FF so that the frame will be broadcast to all devices connected to the switch (hopefully including a DHCP server); the frame's source MAC address is that of Bob's laptop, 00:16:D3:23:68:8A



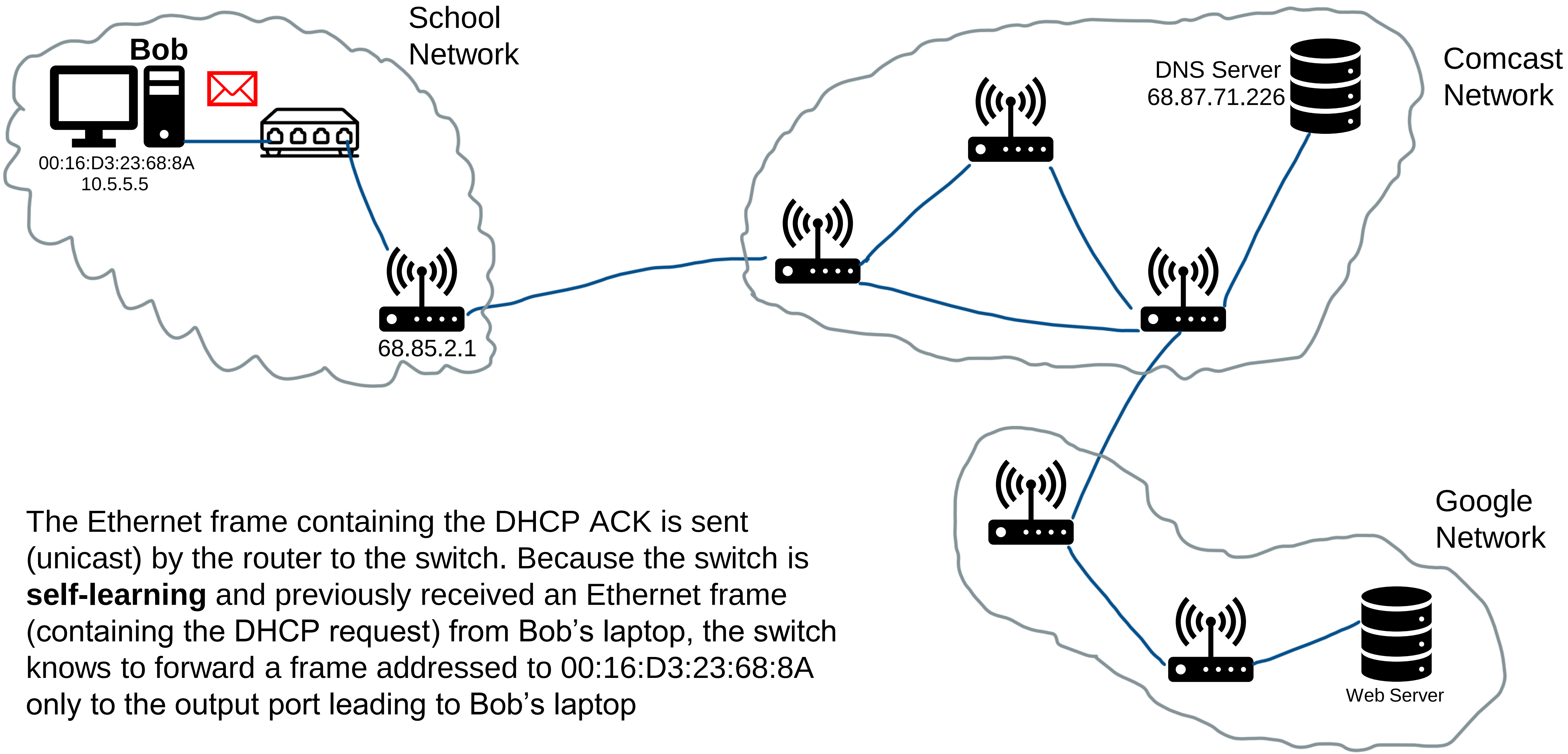
The broadcast Ethernet frame containing the DHCP request is the first frame sent by Bob's laptop to the Ethernet switch. The switch broadcasts the incoming frame on all outgoing ports, including the port connected to the router.



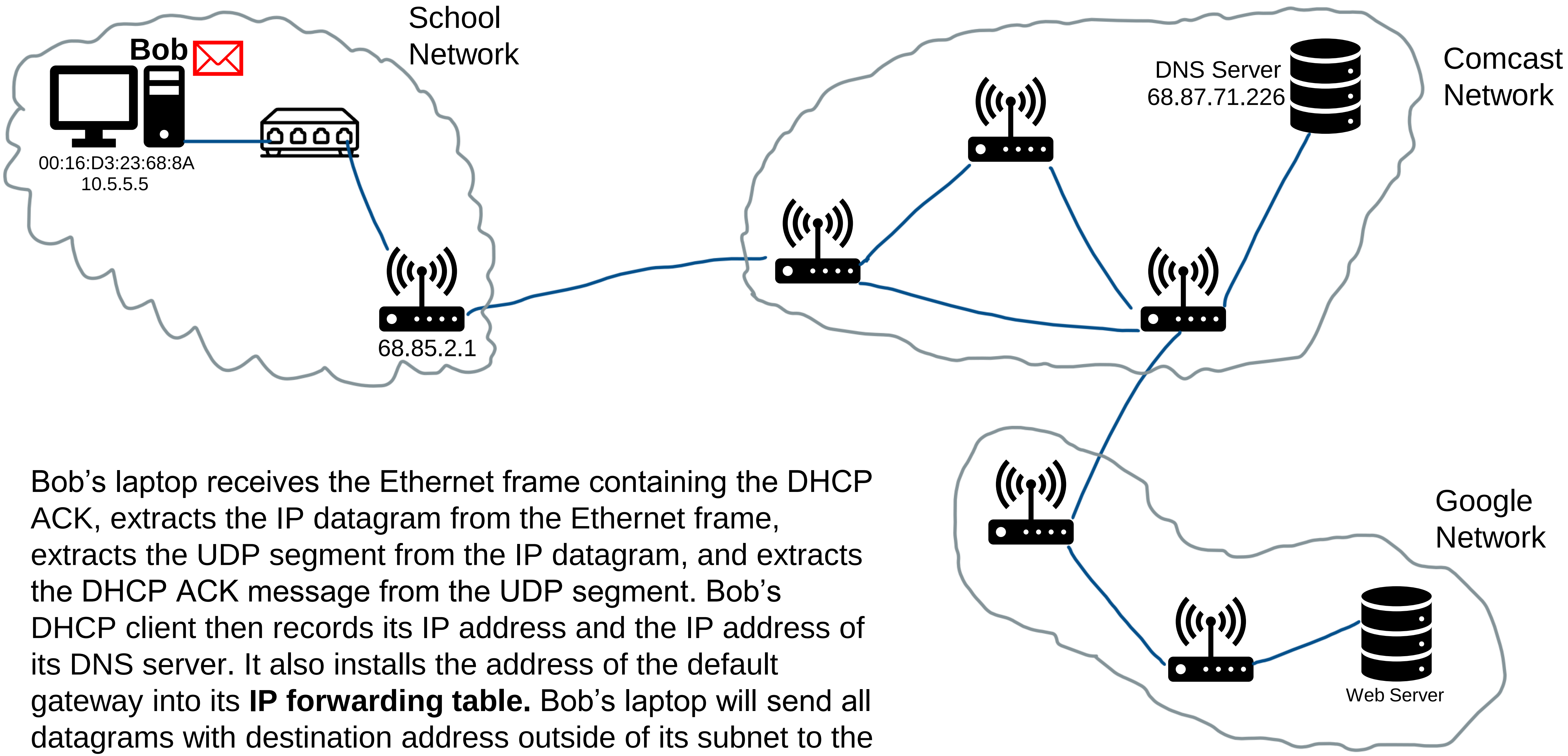
The router receives the broadcast Ethernet frame containing the DHCP request on its interface and the IP datagram is extracted from the Ethernet frame. The datagram's broadcast IP destination address indicates that this IP datagram should be processed by upper layer protocols at this node, so the datagram's payload (a UDP segment) is thus **demultiplexed** up to UDP, and the DHCP request message is extracted from the UDP segment. The DHCP server now has the DHCP request message.



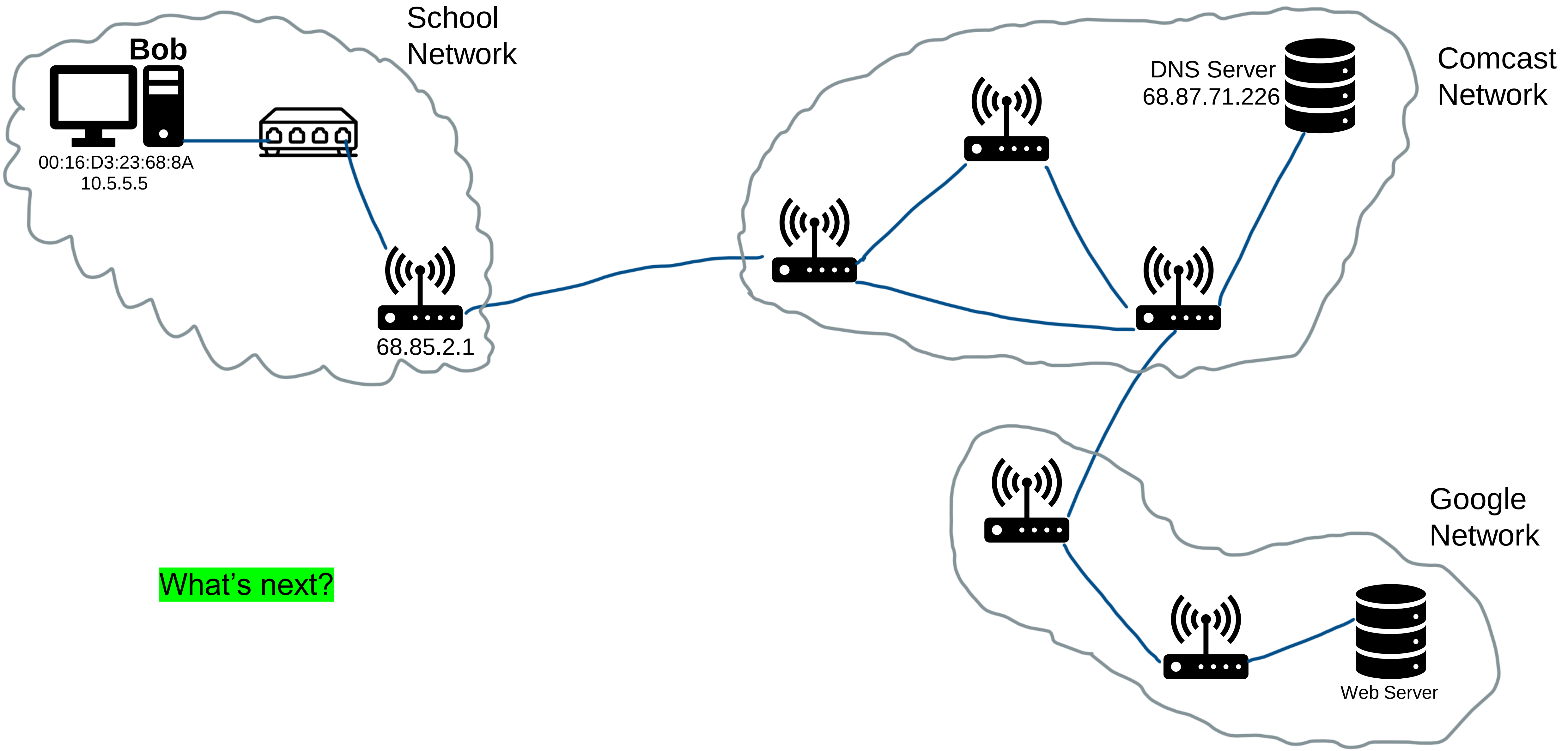
Let’s suppose the DHCP server allocates address 10.5.5.5 to Bob’s laptop. The DHCP server creates a **DHCP ACK message** containing this IP address, as well as the IP address of the DNS server (68.87.71.226), the IP address for the default gateway router (68.85.2.1), and the subnet block. The DHCP message is put inside a UDP segment, which is put inside an IP datagram, which is put inside an Ethernet frame. The Ethernet frame has a a destination MAC address of Bob’s laptop (00:16:D3:23:68:8A).



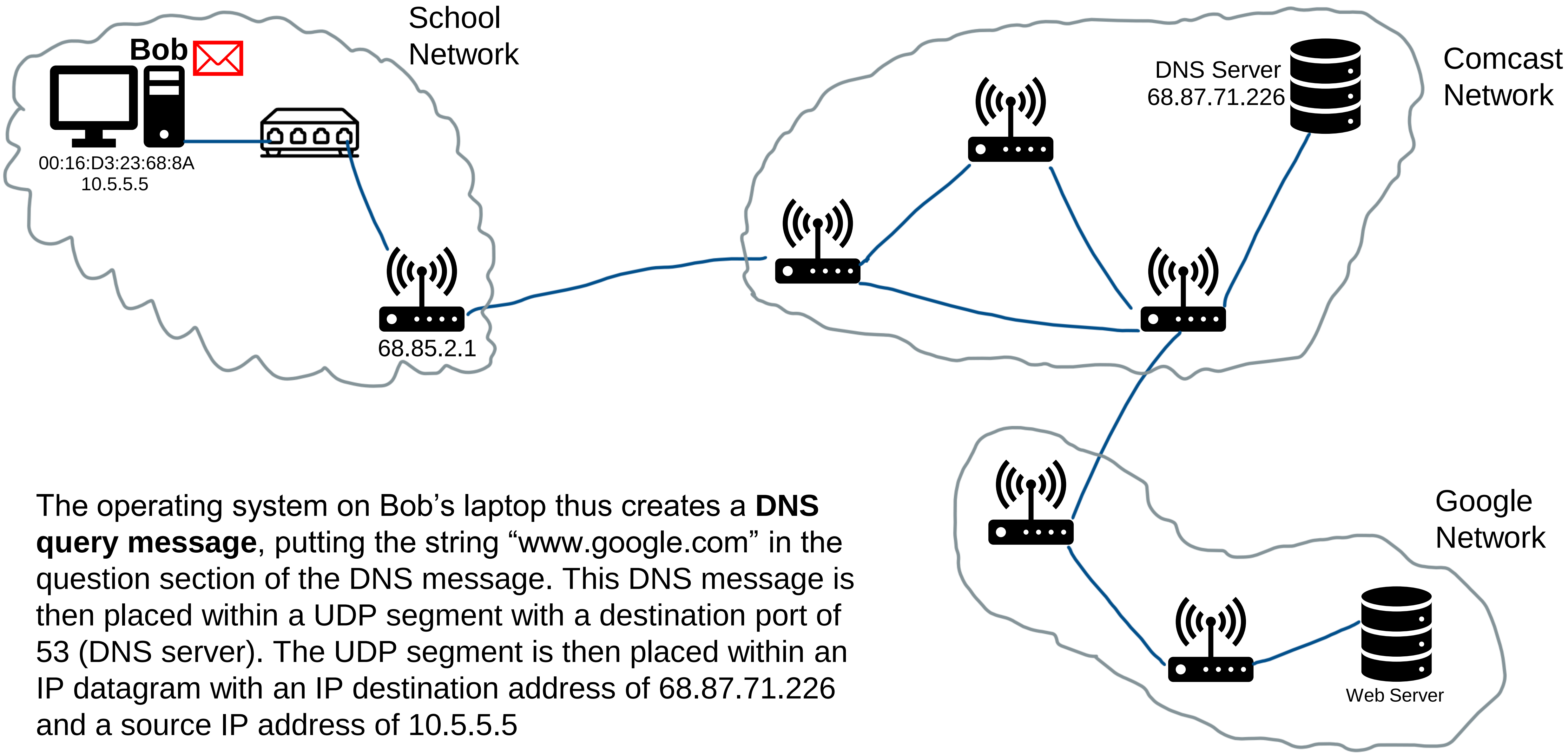
The Ethernet frame containing the DHCP ACK is sent (unicast) by the router to the switch. Because the switch is **self-learning** and previously received an Ethernet frame (containing the DHCP request) from Bob's laptop, the switch knows to forward a frame addressed to 00:16:D3:23:68:8A only to the output port leading to Bob's laptop



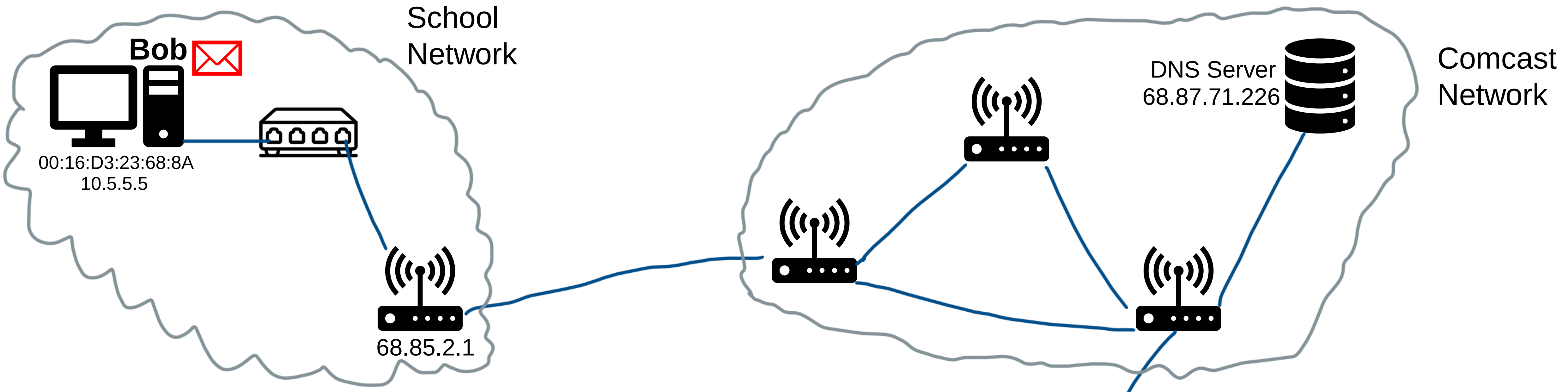
Bob's laptop receives the Ethernet frame containing the DHCP ACK, extracts the IP datagram from the Ethernet frame, extracts the UDP segment from the IP datagram, and extracts the DHCP ACK message from the UDP segment. Bob's DHCP client then records its IP address and the IP address of its DNS server. It also installs the address of the default gateway into its **IP forwarding table.** Bob's laptop will send all datagrams with destination address outside of its subnet to the default gateway.

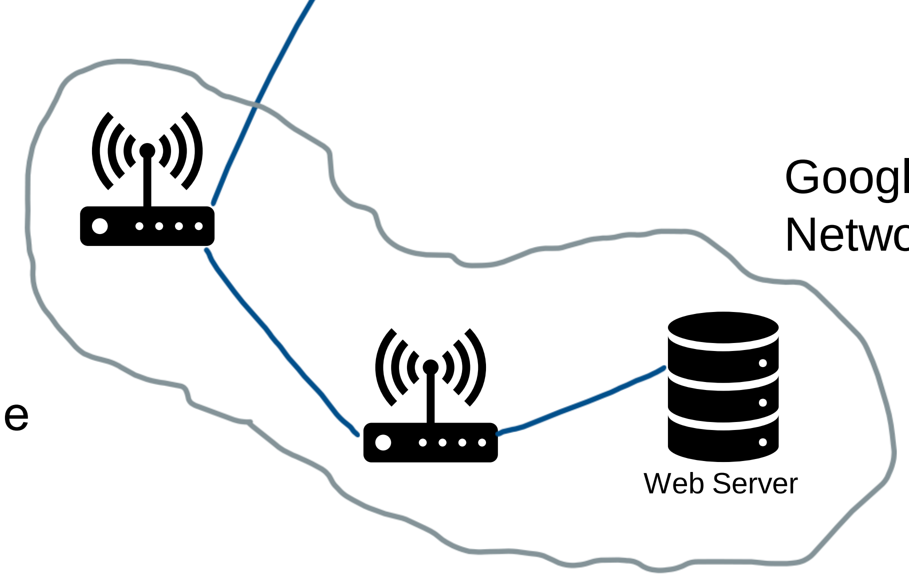

School
Network
Bob
00:16:D3:23:68:8A
10.5.5.5
68.85.2.1
DNS Server
68.87.71.226
Comcast
Network
Google
Network
Web Server

What's next?

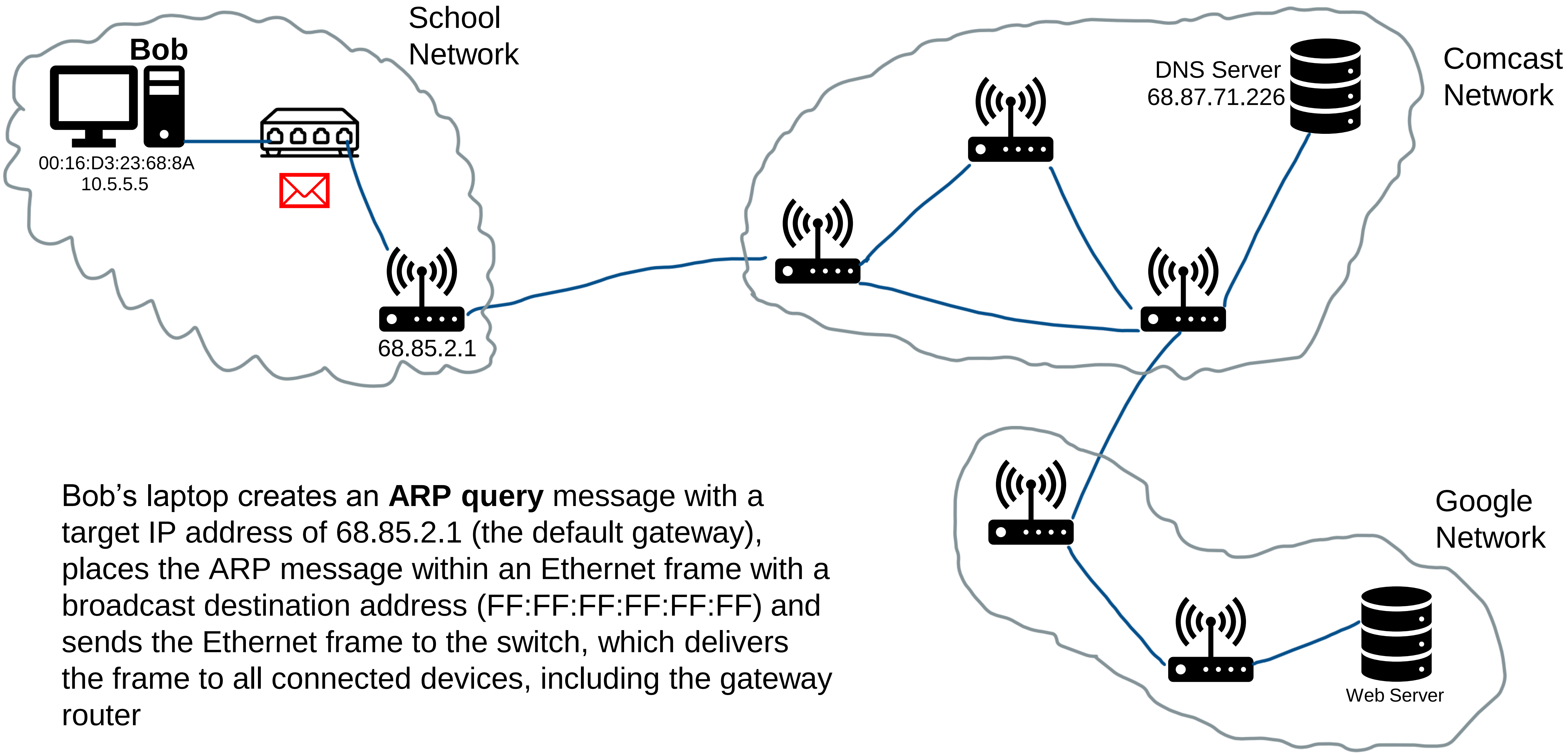


The operating system on Bob’s laptop thus creates a **DNS query message**, putting the string “www.google.com” in the question section of the DNS message. This DNS message is then placed within a UDP segment with a destination port of 53 (DNS server). The UDP segment is then placed within an IP datagram with an IP destination address of 68.87.71.226 and a source IP address of 10.5.5.5

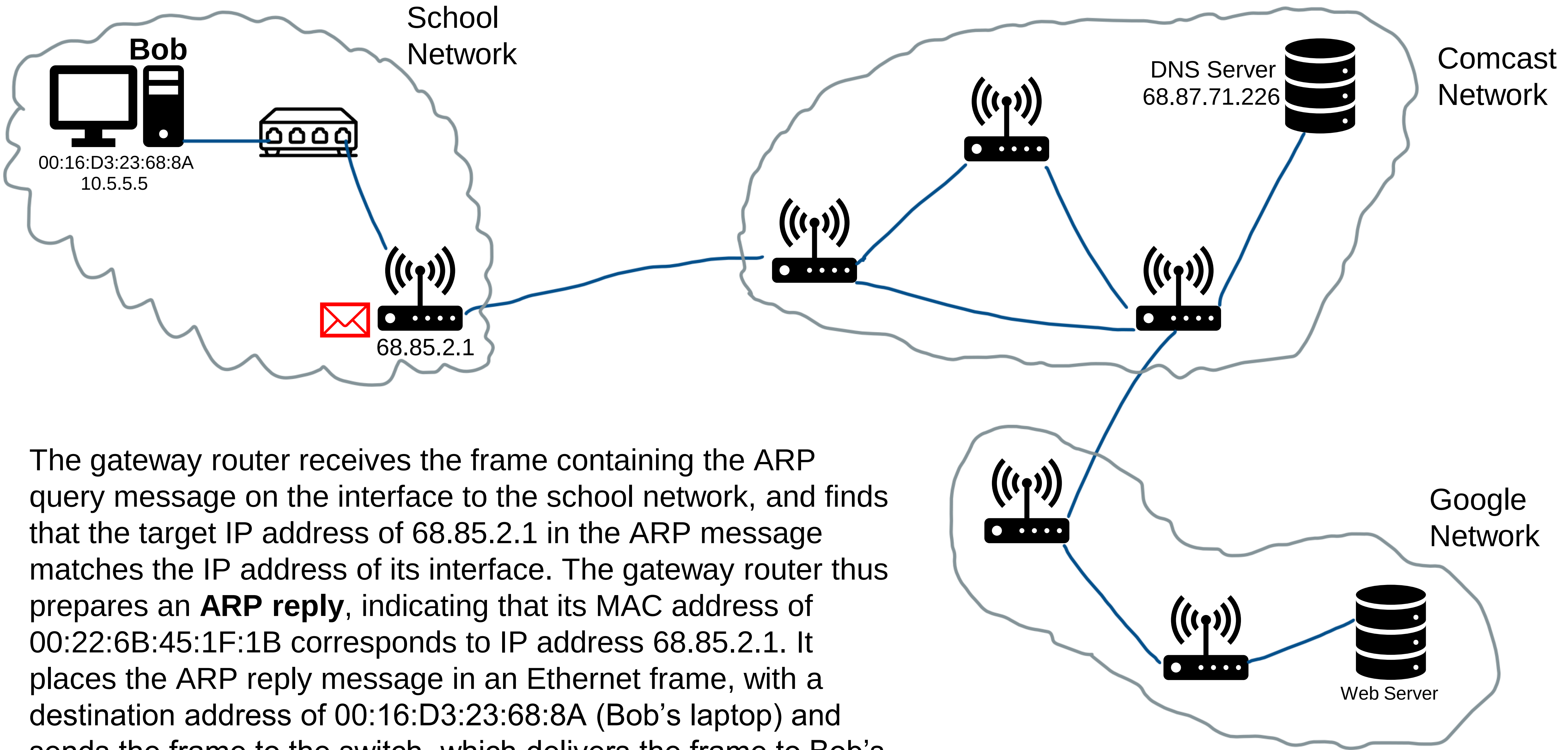




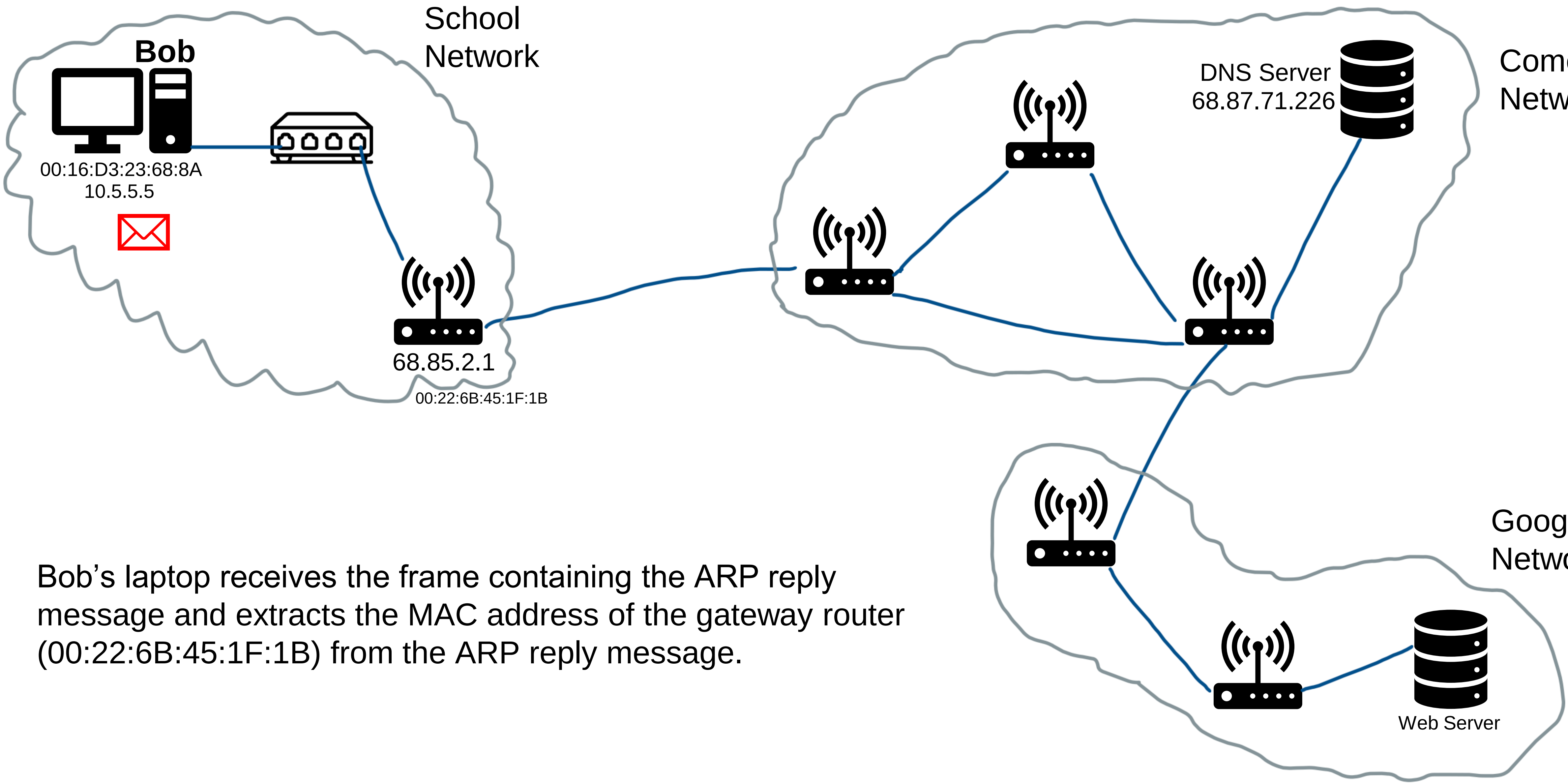
Bob’s laptop then places the datagram containing the DNS query message in an Ethernet frame. This frame will be sent (addressed, at the link layer) to the gateway router in Bob’s school’s network. However, even though Bob’s laptop knows the IP address of the school’s gateway router (68.85.2.1) via the DHCP ACK message, it doesn’t know the gateway router’s MAC address. In order to obtain the MAC address of the gateway router, Bob’s laptop will need to use the **ARP protocol**



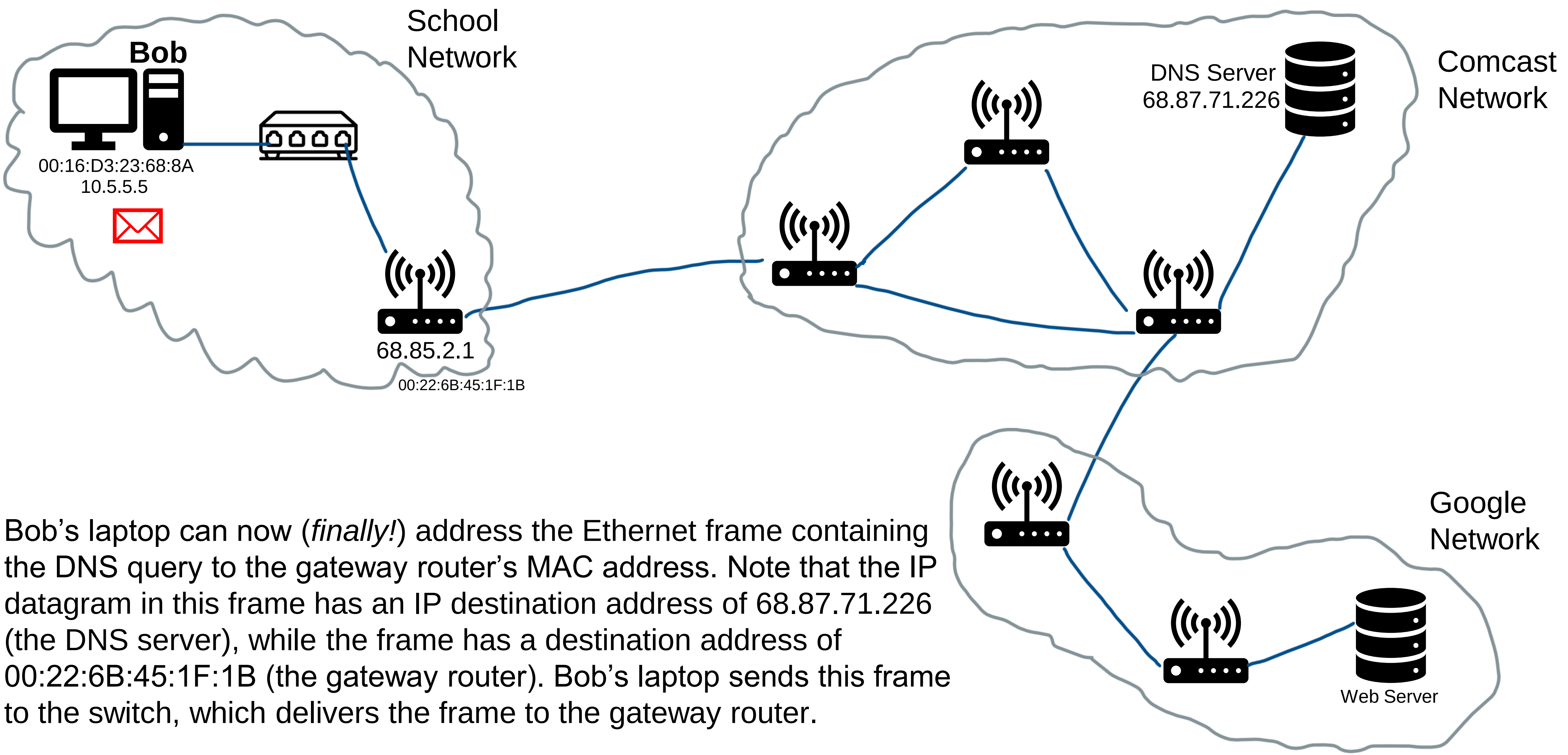
Bob's laptop creates an **ARP query** message with a target IP address of 68.85.2.1 (the default gateway), places the ARP message within an Ethernet frame with a broadcast destination address (FF:FF:FF:FF:FF:FF) and sends the Ethernet frame to the switch, which delivers the frame to all connected devices, including the gateway router



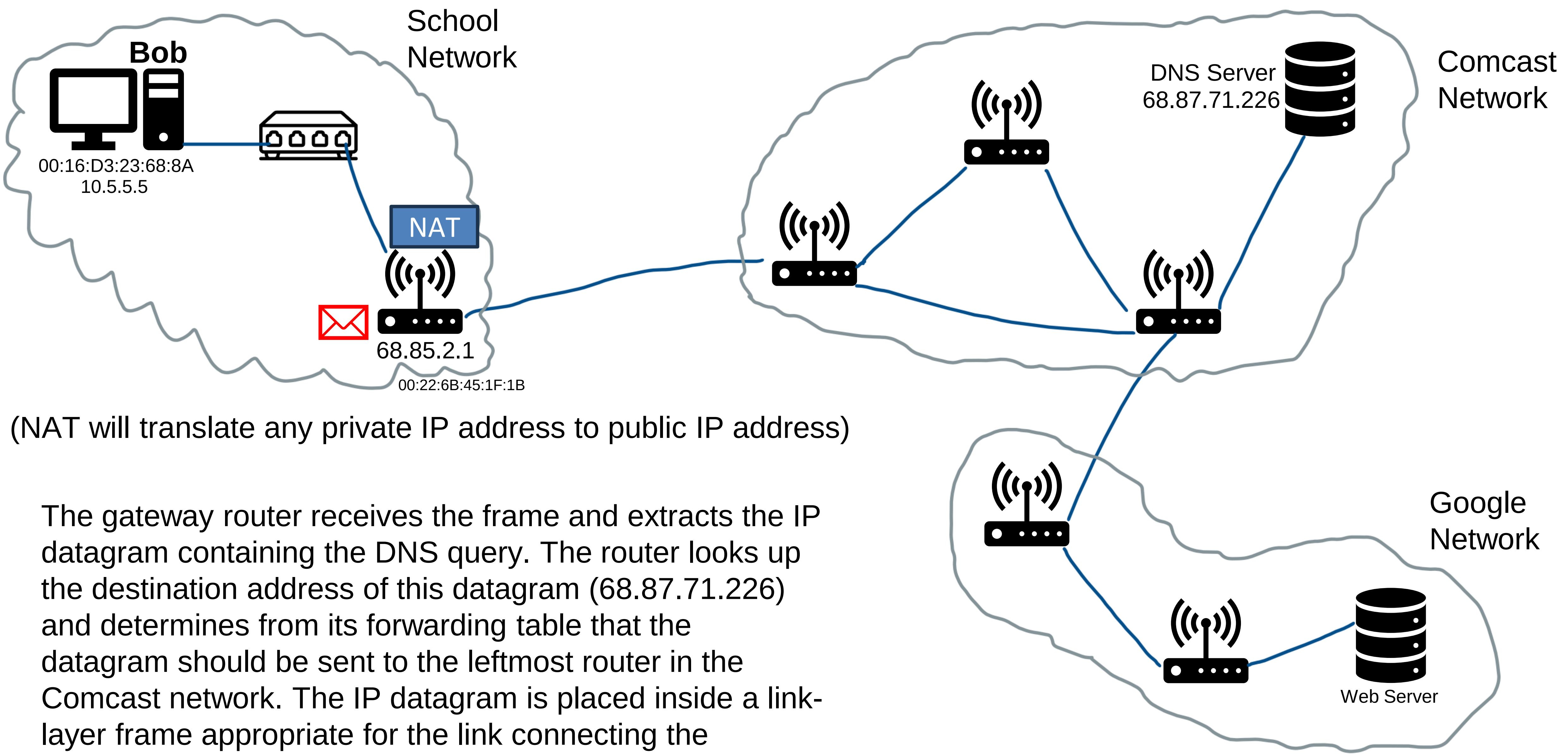
The gateway router receives the frame containing the ARP query message on the interface to the school network, and finds that the target IP address of 68.85.2.1 in the ARP message matches the IP address of its interface. The gateway router thus prepares an **ARP reply**, indicating that its MAC address of 00:22:6B:45:1F:1B corresponds to IP address 68.85.2.1. It places the ARP reply message in an Ethernet frame, with a destination address of 00:16:D3:23:68:8A (Bob's laptop) and sends the frame to the switch, which delivers the frame to Bob's laptop



Bob's laptop receives the frame containing the ARP reply message and extracts the MAC address of the gateway router (00:22:6B:45:1F:1B) from the ARP reply message.

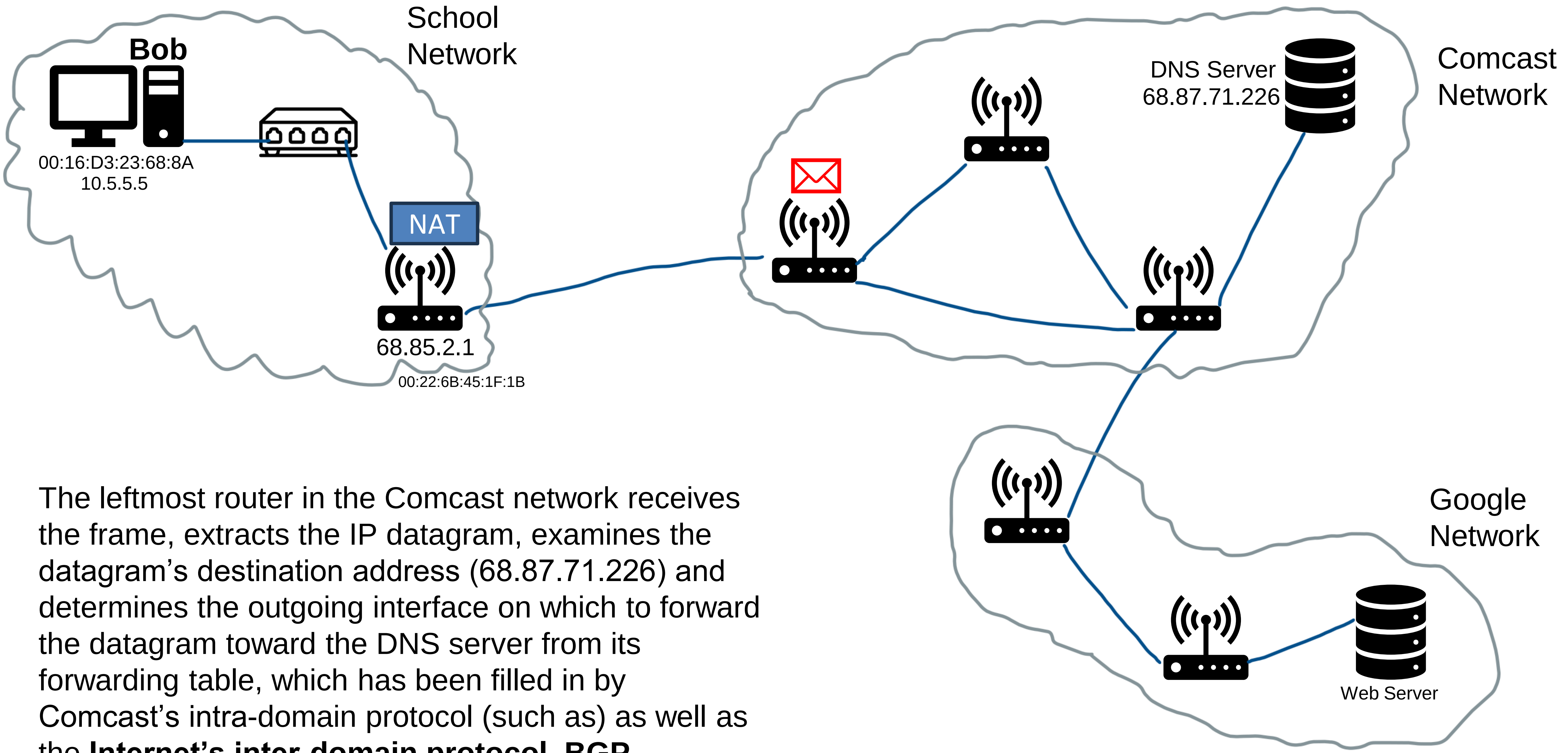


Bob's laptop can now (*finally!*) address the Ethernet frame containing the DNS query to the gateway router's MAC address. Note that the IP datagram in this frame has an IP destination address of 68.87.71.226 (the DNS server), while the frame has a destination address of 00:22:6B:45:1F:1B (the gateway router). Bob's laptop sends this frame to the switch, which delivers the frame to the gateway router.

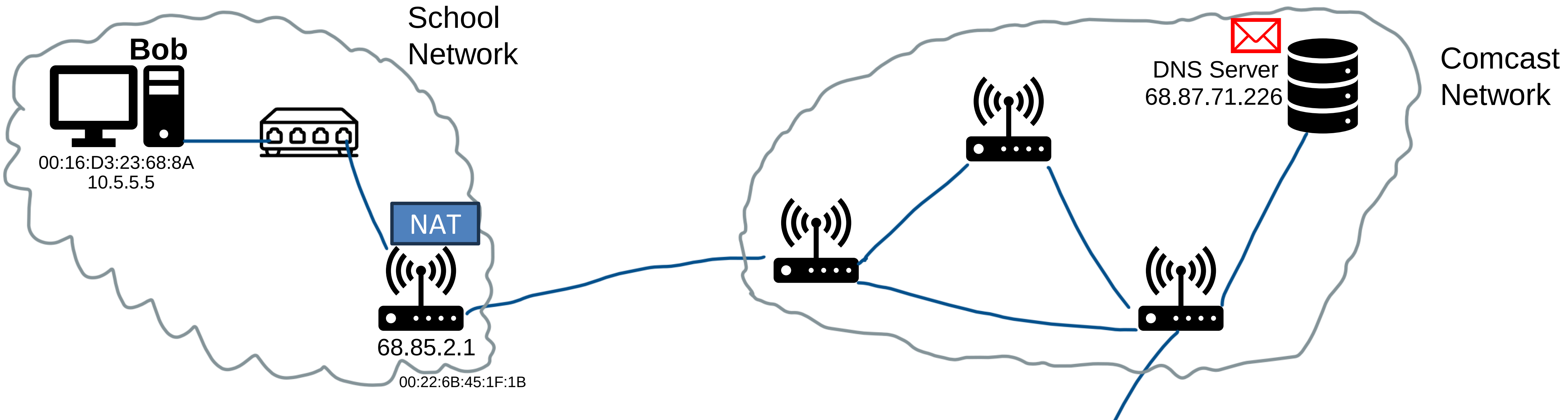


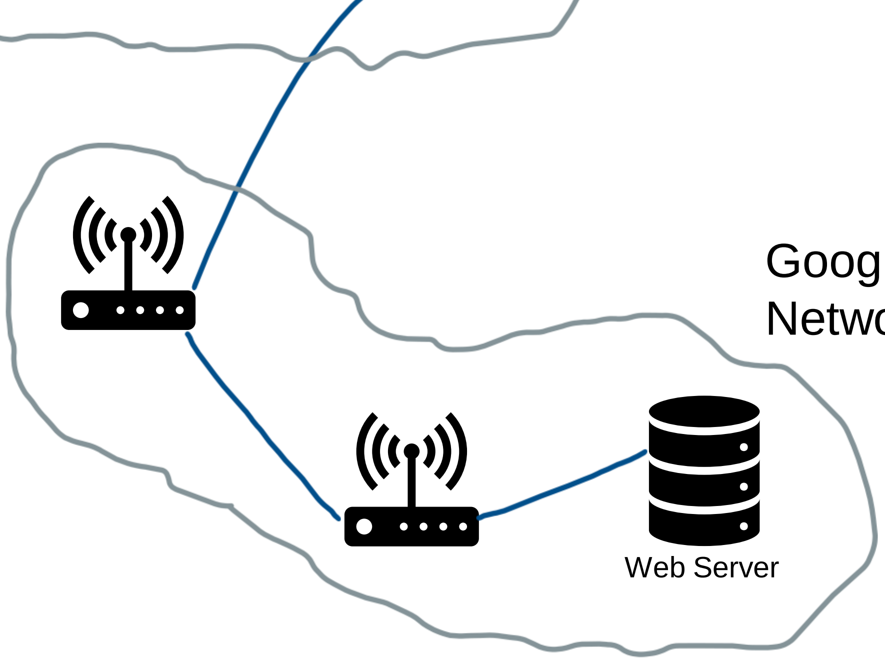
(NAT will translate any private IP address to public IP address)

The gateway router receives the frame and extracts the IP datagram containing the DNS query. The router looks up the destination address of this datagram (68.87.71.226) and determines from its forwarding table that the datagram should be sent to the 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.

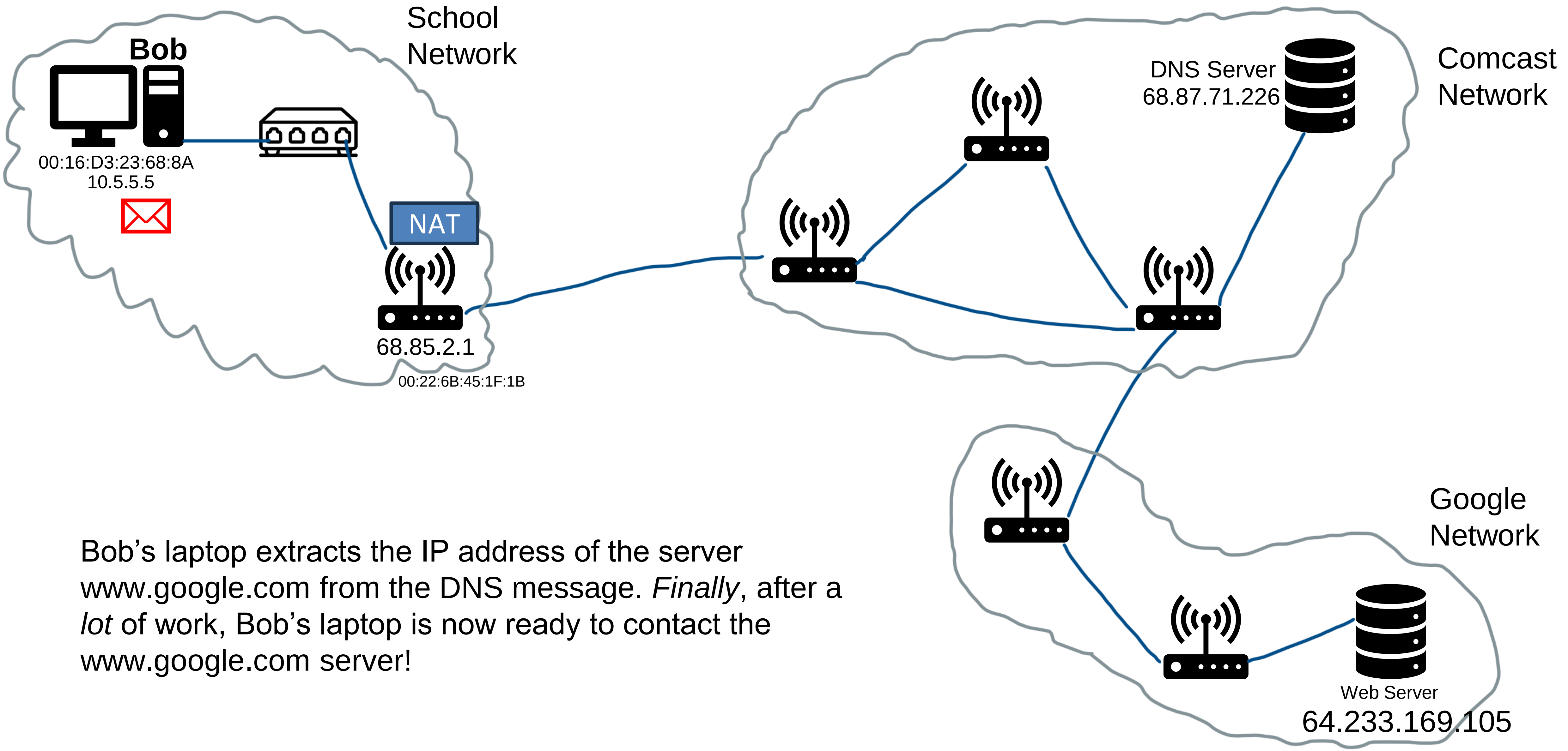


The leftmost router in the Comcast network receives the frame, extracts the IP datagram, examines the datagram's destination address (68.87.71.226) and determines the outgoing interface on which to forward the datagram toward the DNS server from its forwarding table, which has been filled in by Comcast's intra-domain protocol (such as) as well as the **Internet's inter-domain protocol**, **BGP**

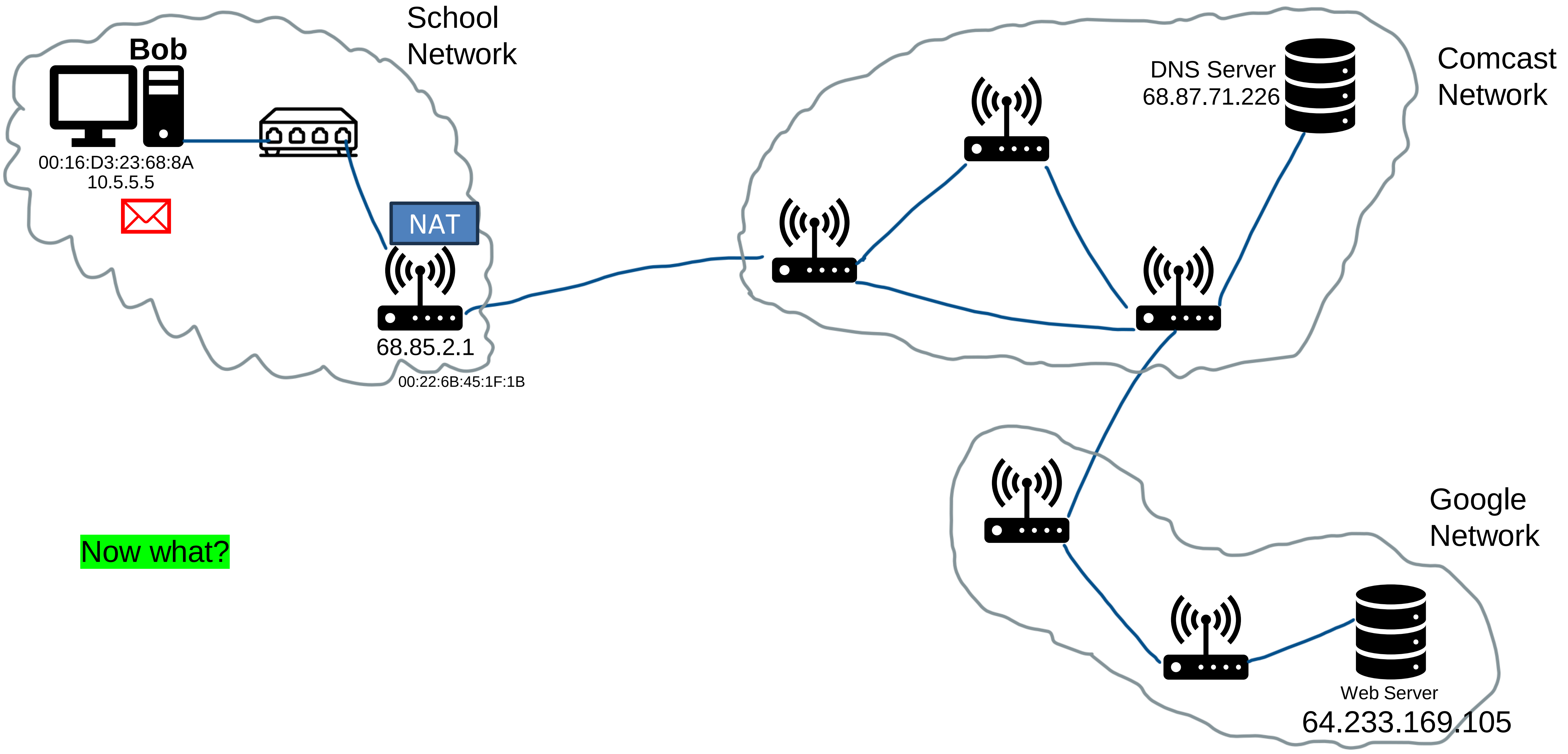




Eventually the IP datagram containing the DNS query arrives at the DNS server. The DNS server extracts the DNS query message, looks up the name www.google.com in its DNS database and finds the **DNS resource record** that contains the IP address (64.233.169.105) for www.google.com. (assuming that it is currently cached in the DNS server). Recall that this cached data originated in the **authoritative DNS server** for google.com. The DNS server forms a **DNS reply message** containing this hostname-to-IP-address mapping, and places the DNS reply message in a UDP segment, and the segment within an IP datagram addressed to Bob's public IP (68.85.2.1). This datagram will be forwarded back through the Comcast network to the school's router and from there, via the Ethernet switch to Bob's laptop

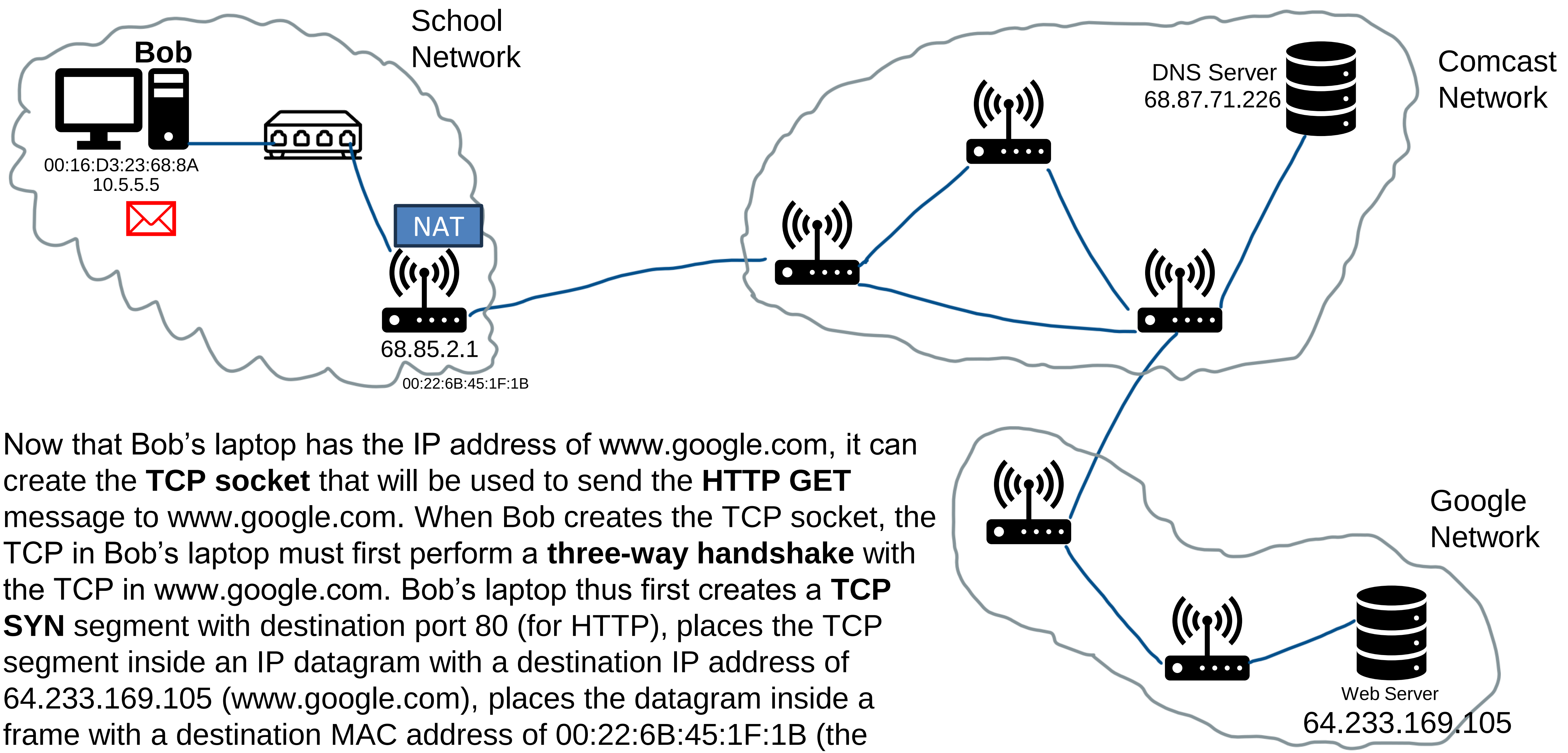


Bob's laptop extracts the IP address of the server www.google.com from the DNS message. *Finally*, after a *lot* of work, Bob's laptop is now ready to contact the www.google.com server!

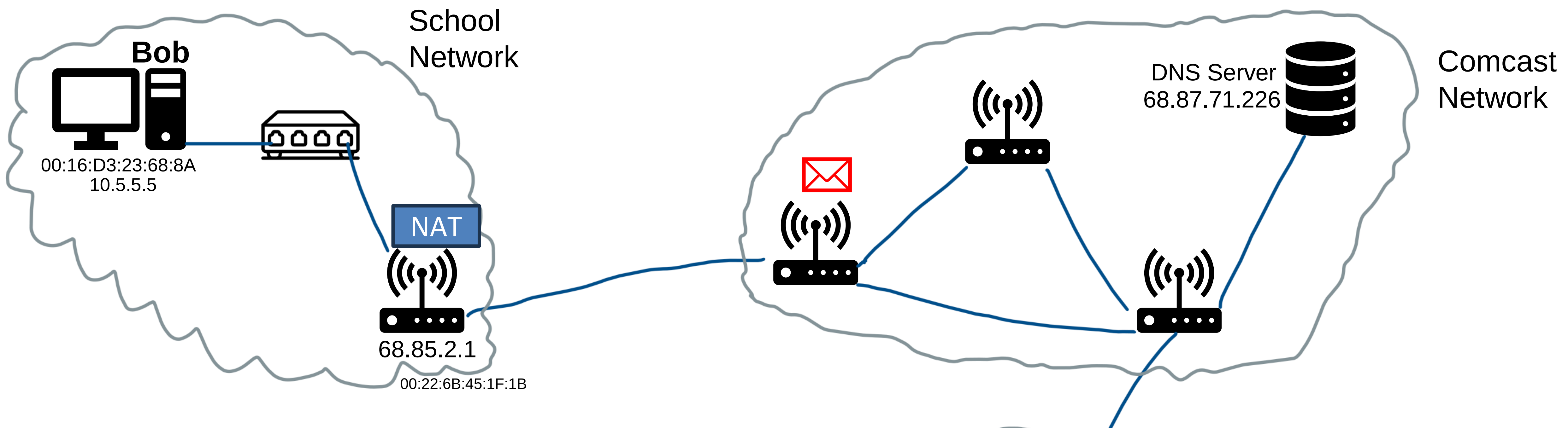


School
Network
Bob
00:16:D3:23:68:8A
10.5.5.5
NAT
68.85.2.1
00:22:6B:45:1F:1B
DNS Server
68.87.71.226
Comcast
Network
Google
Network
Web Server
64.233.169.105

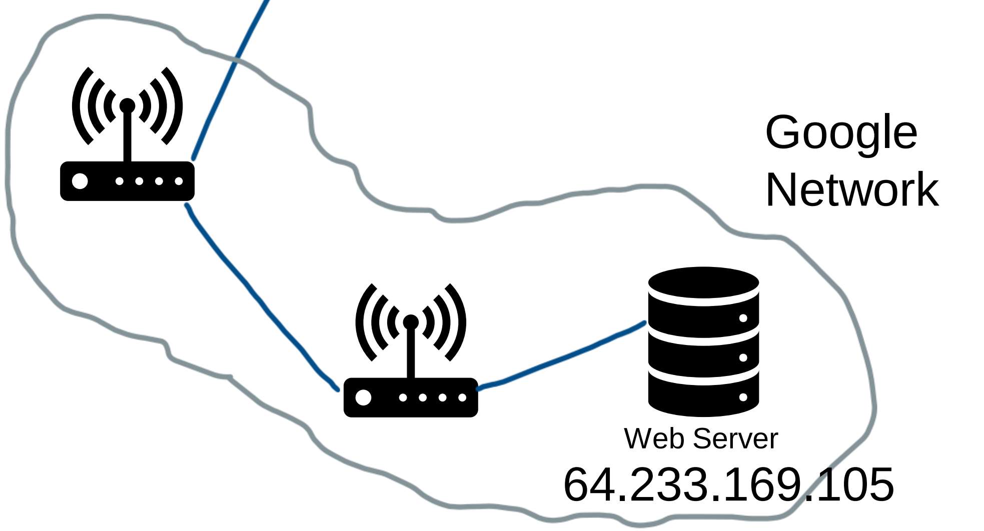
Now what?

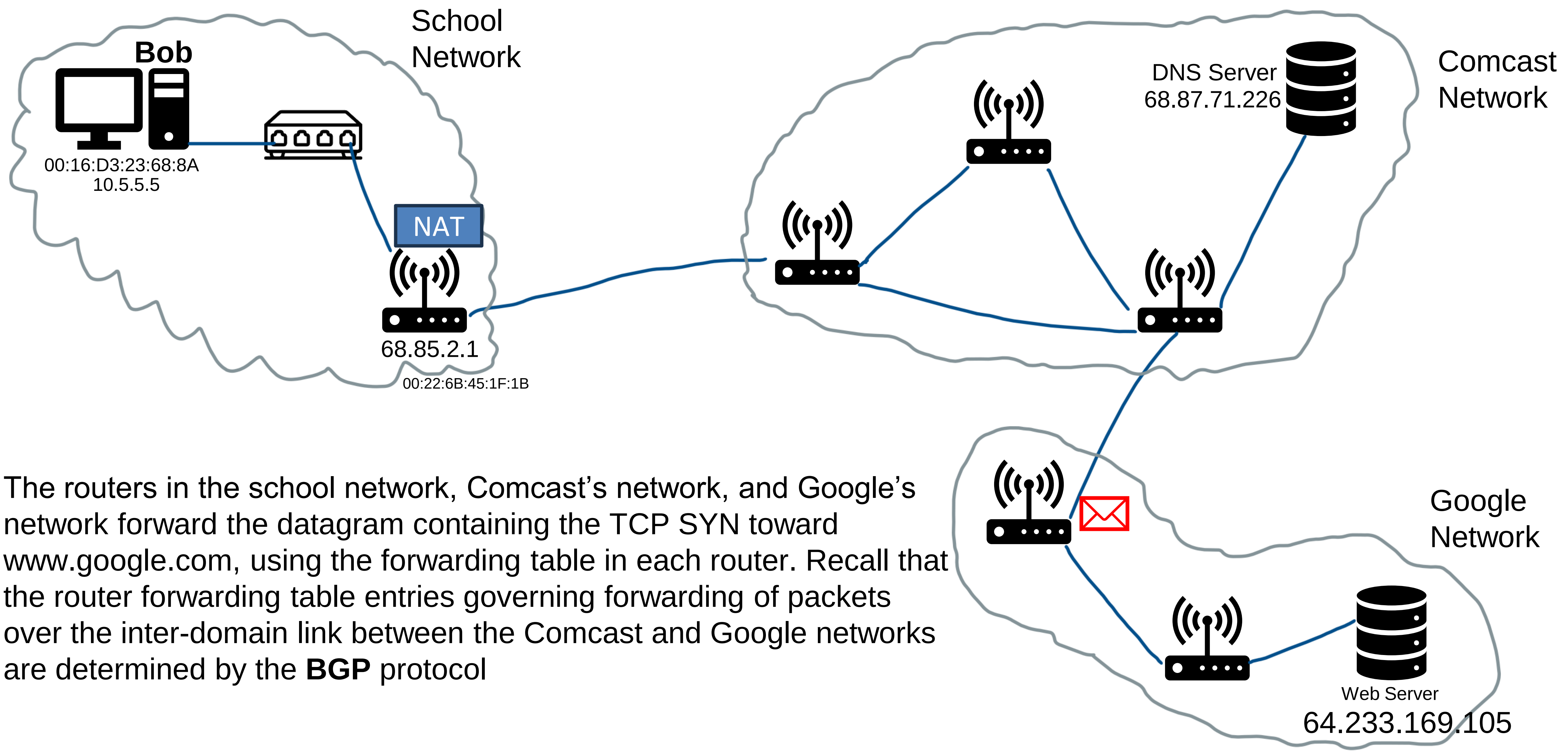


Now that Bob's laptop has the IP address of www.google.com, it can create the **TCP socket** that will be used to send the **HTTP GET** message to www.google.com. When Bob creates the TCP socket, the TCP in Bob's laptop must first perform a **three-way handshake** with the TCP in www.google.com. Bob's laptop thus first creates a **TCP SYN** segment with destination port 80 (for HTTP), places the TCP segment inside an IP datagram with a destination IP address of 64.233.169.105 (www.google.com), places the datagram inside a frame with a destination MAC address of 00:22:6B:45:1F:1B (the gateway router) and sends the frame to the switch

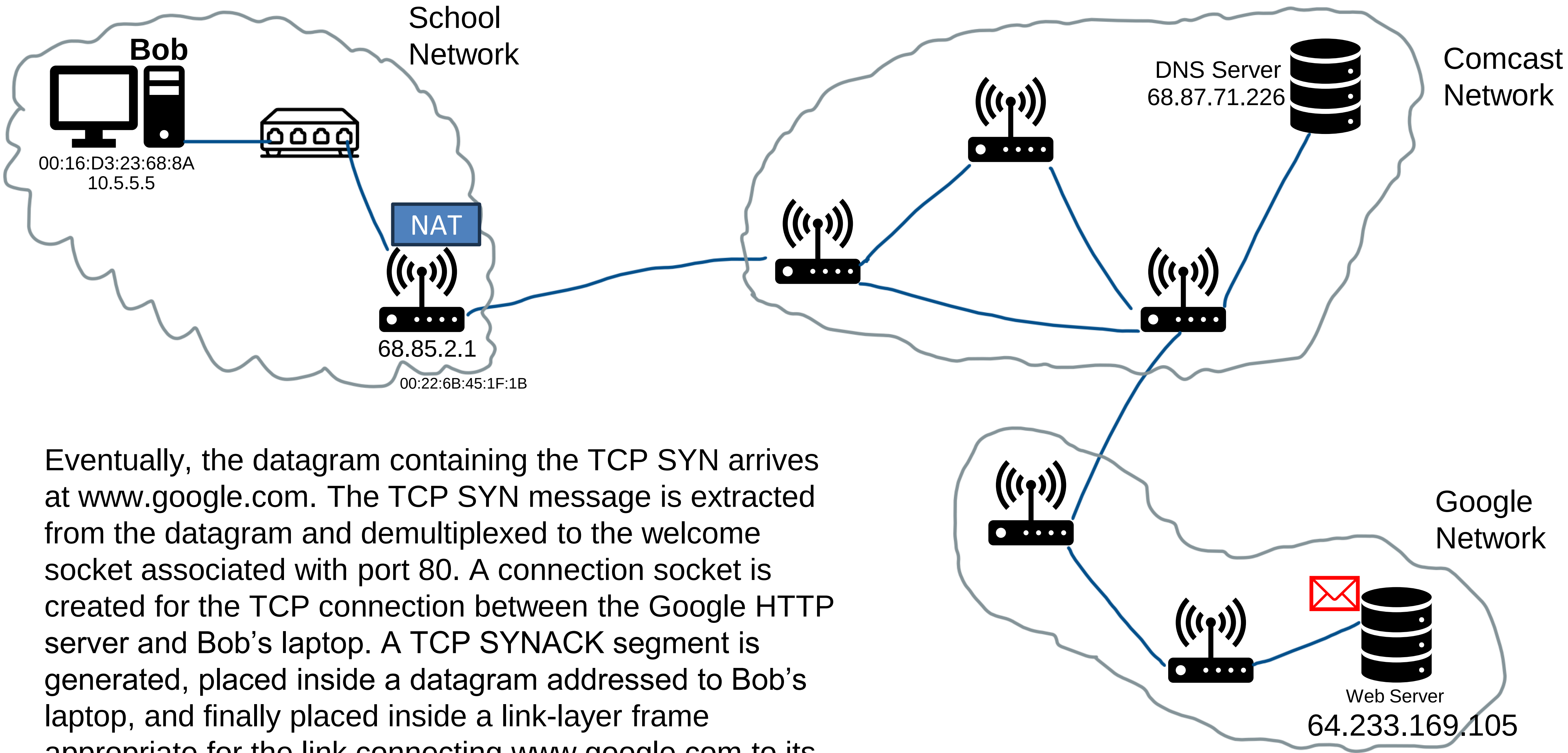


The routers in the school network, Comcast's network, and Google's network forward the datagram containing the TCP SYN toward www.google.com, using the forwarding table in each router. Recall that the router forwarding table entries governing forwarding of packets over the inter-domain link between the Comcast and Google networks are determined by the **BGP** protocol

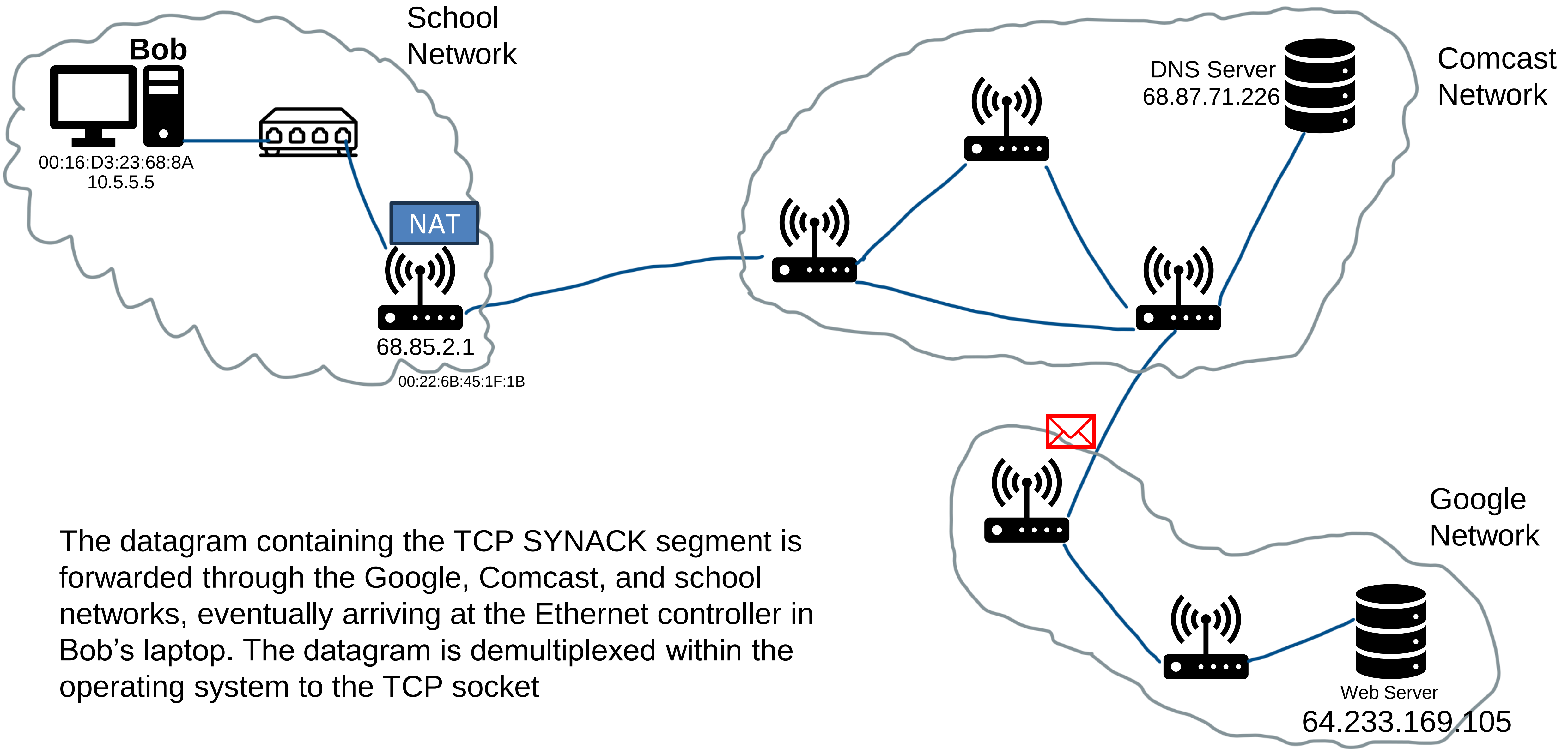




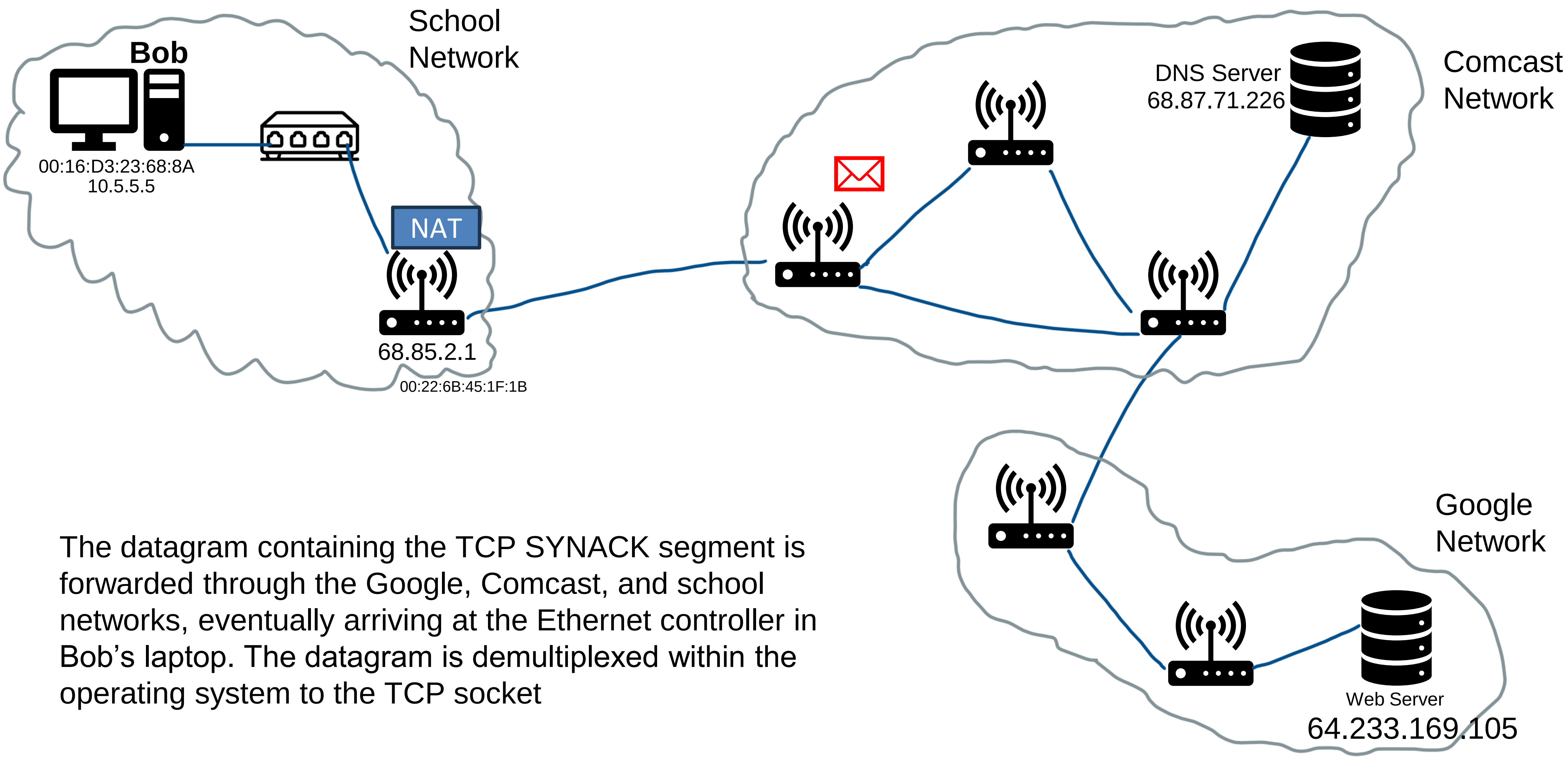
The routers in the school network, Comcast's network, and Google's network forward the datagram containing the TCP SYN toward www.google.com, using the forwarding table in each router. Recall that the router forwarding table entries governing forwarding of packets over the inter-domain link between the Comcast and Google networks are determined by the **BGP** protocol



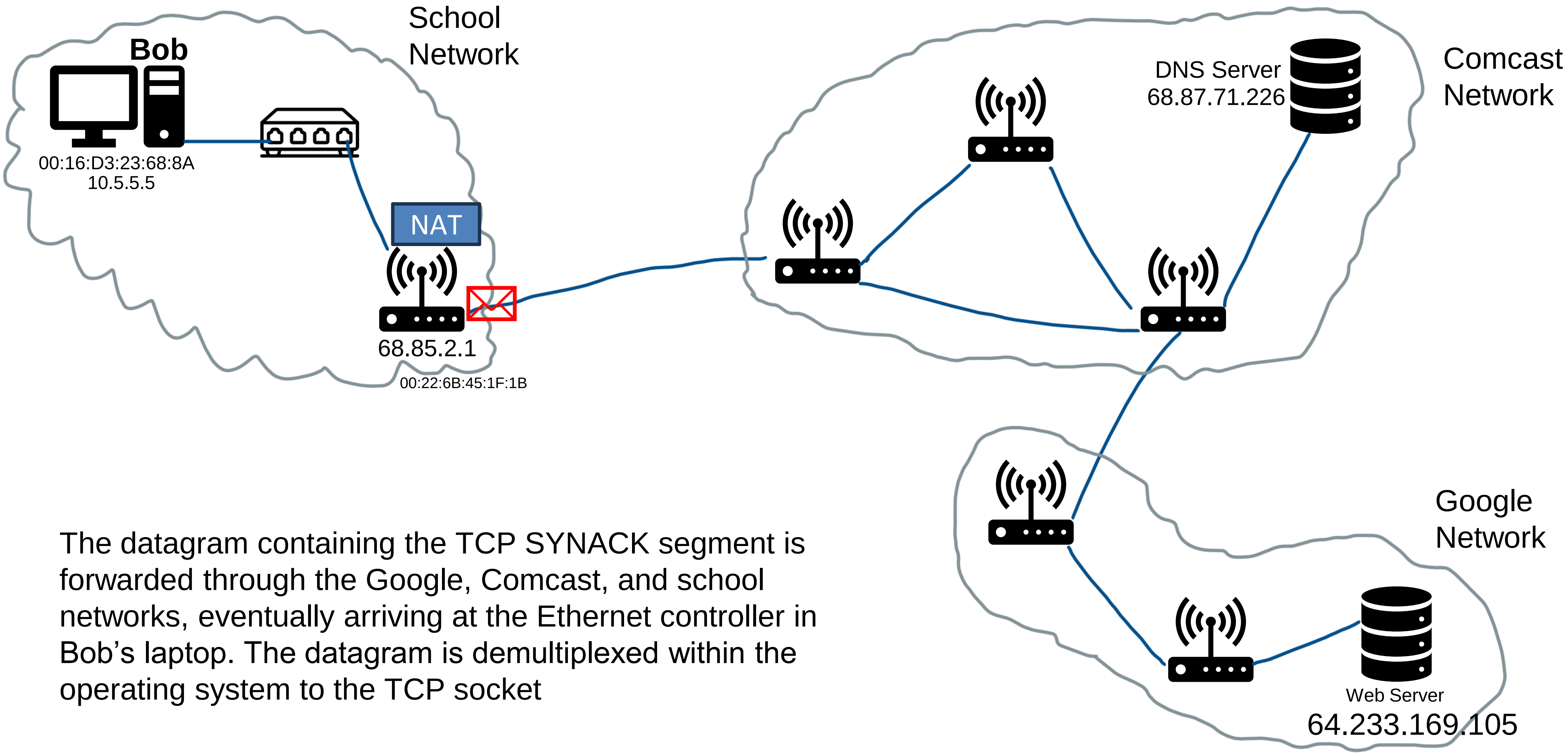
Eventually, the datagram containing the TCP SYN arrives at www.google.com. The TCP SYN message is extracted from the datagram and demultiplexed to the welcome socket associated with port 80. A connection socket is created for the TCP connection between the Google HTTP server and Bob's laptop. A TCP SYNACK segment is generated, placed inside a datagram addressed to Bob's laptop, and finally placed inside a link-layer frame appropriate for the link connecting www.google.com to its first-hop router.



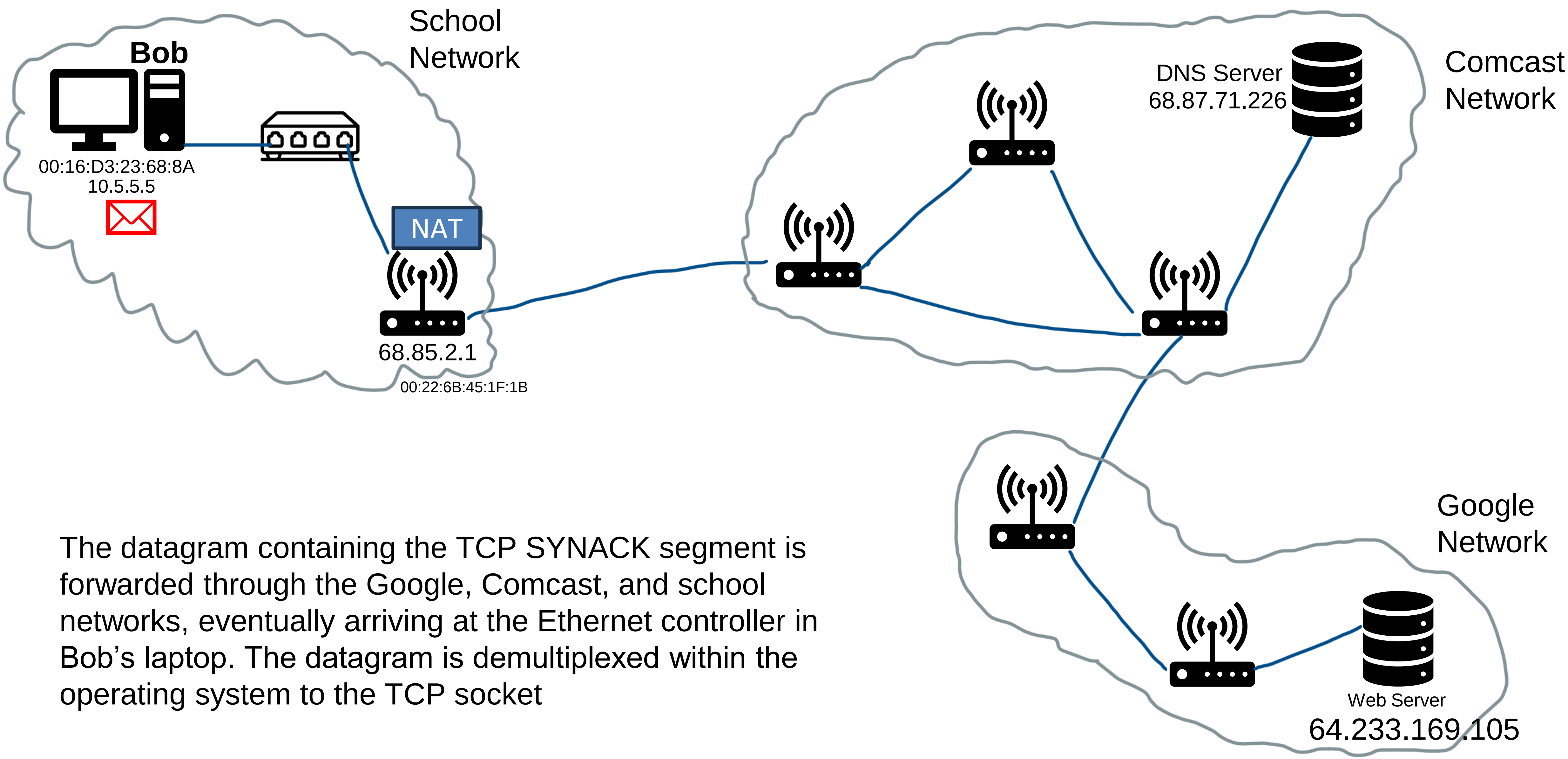
The datagram containing the TCP SYNACK segment is forwarded through the Google, Comcast, and school networks, eventually arriving at the Ethernet controller in Bob’s laptop. The datagram is demultiplexed within the operating system to the TCP socket



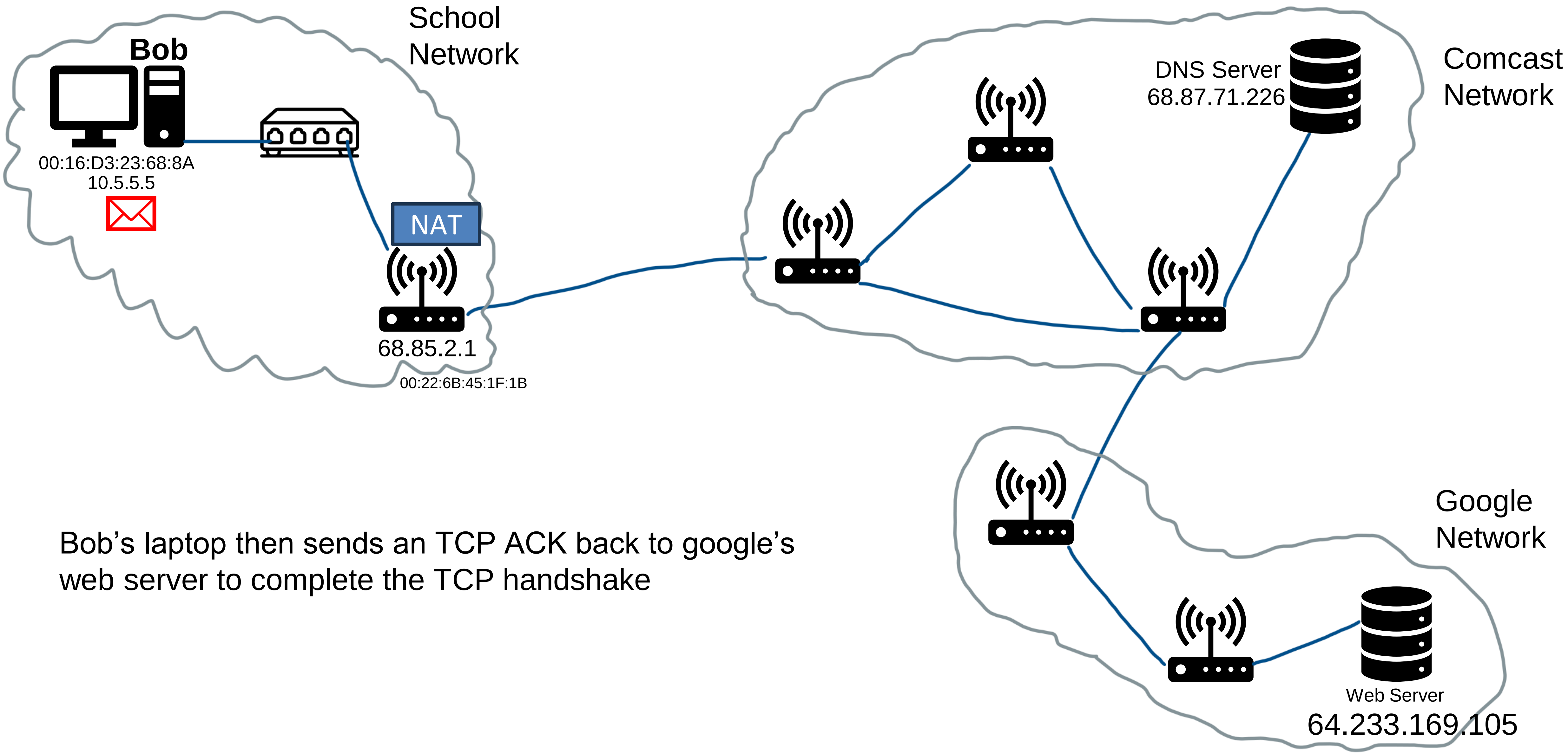
The datagram containing the TCP SYNACK segment is forwarded through the Google, Comcast, and school networks, eventually arriving at the Ethernet controller in Bob’s laptop. The datagram is demultiplexed within the operating system to the TCP socket



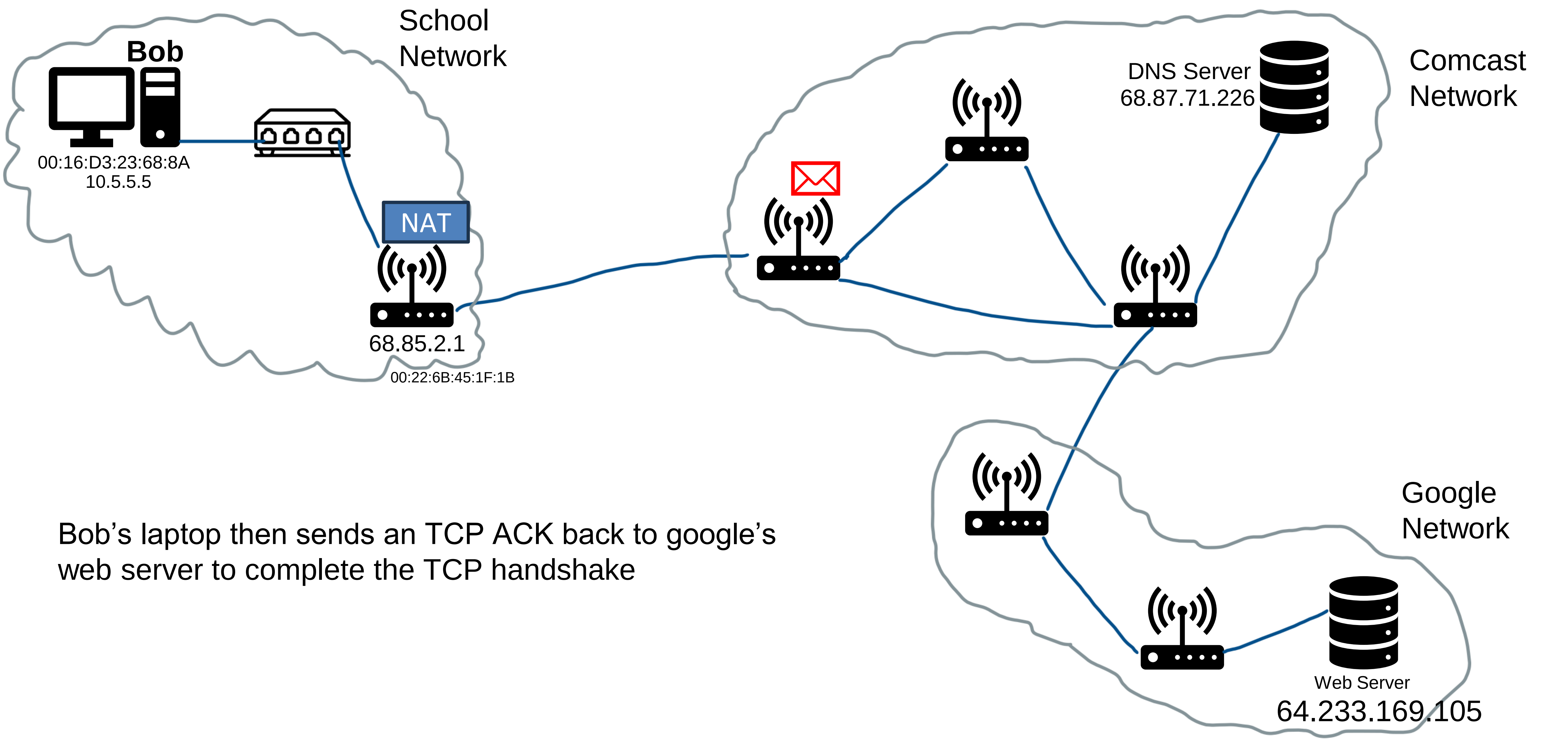
The datagram containing the TCP SYNACK segment is forwarded through the Google, Comcast, and school networks, eventually arriving at the Ethernet controller in Bob's laptop. The datagram is demultiplexed within the operating system to the TCP socket



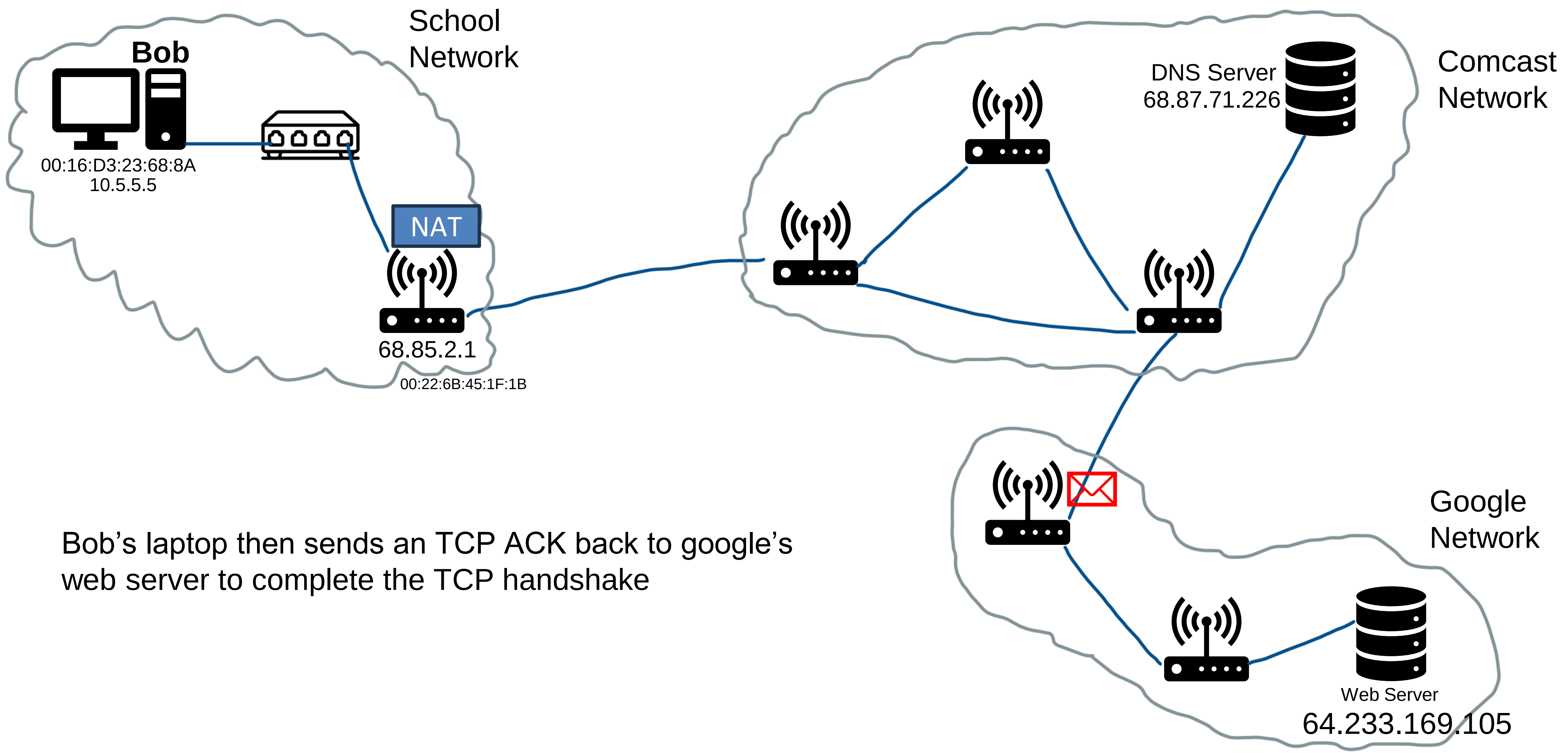
The datagram containing the TCP SYNACK segment is forwarded through the Google, Comcast, and school networks, eventually arriving at the Ethernet controller in Bob's laptop. The datagram is demultiplexed within the operating system to the TCP socket



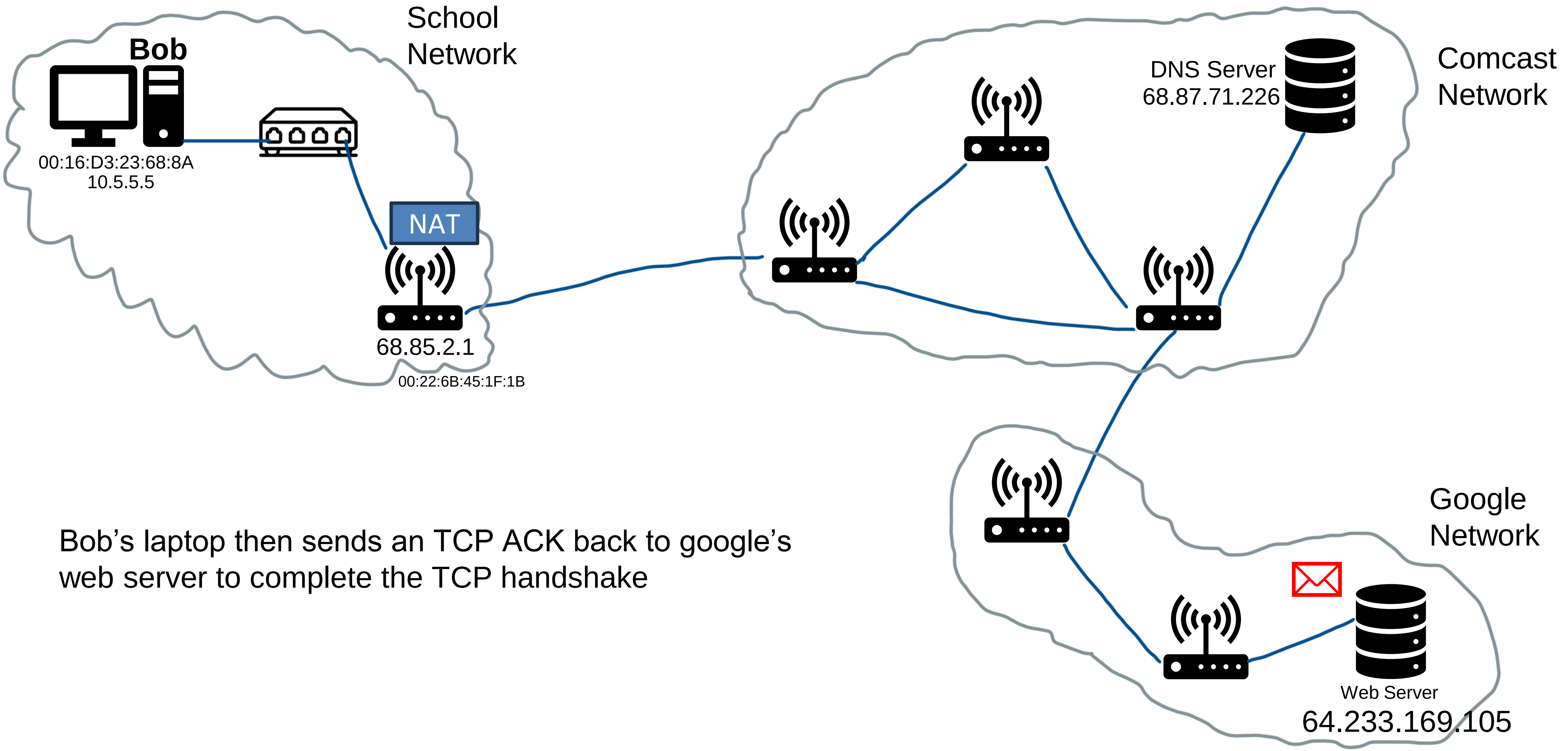
Bob's laptop then sends an TCP ACK back to google's web server to complete the TCP handshake



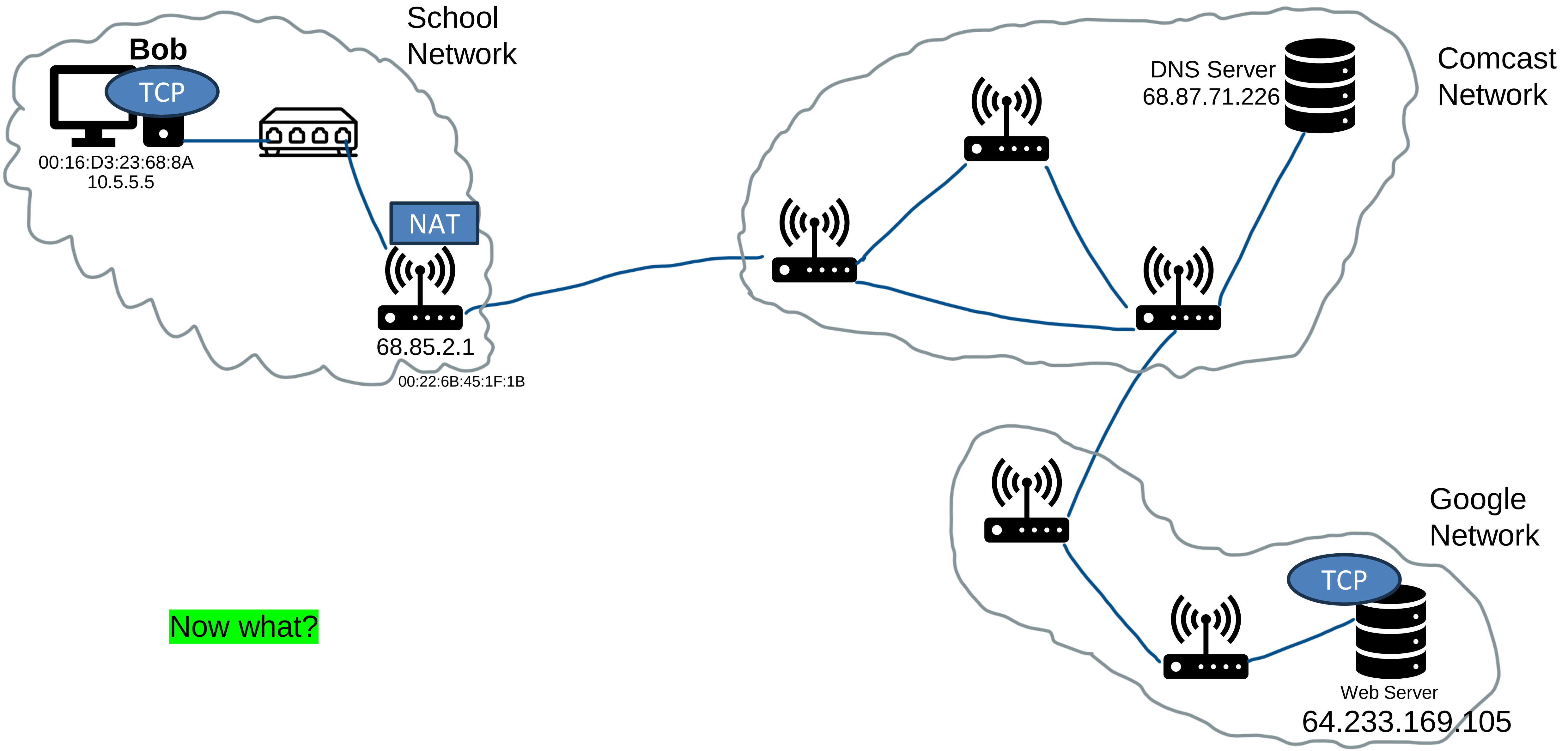
Bob’s laptop then sends an TCP ACK back to google’s web server to complete the TCP handshake



Bob’s laptop then sends an TCP ACK back to google’s web server to complete the TCP handshake

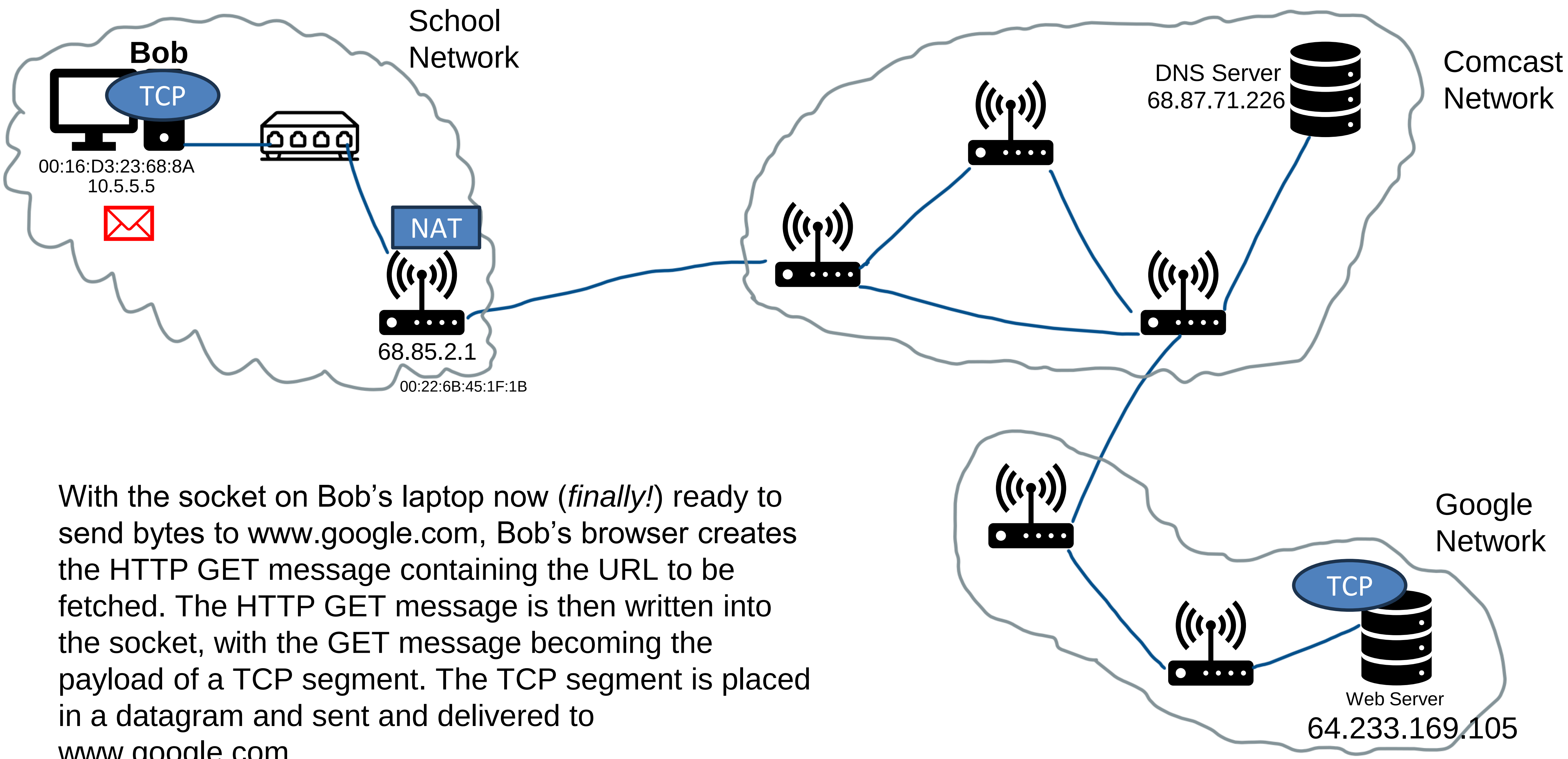


Bob's laptop then sends an TCP ACK back to google's web server to complete the TCP handshake

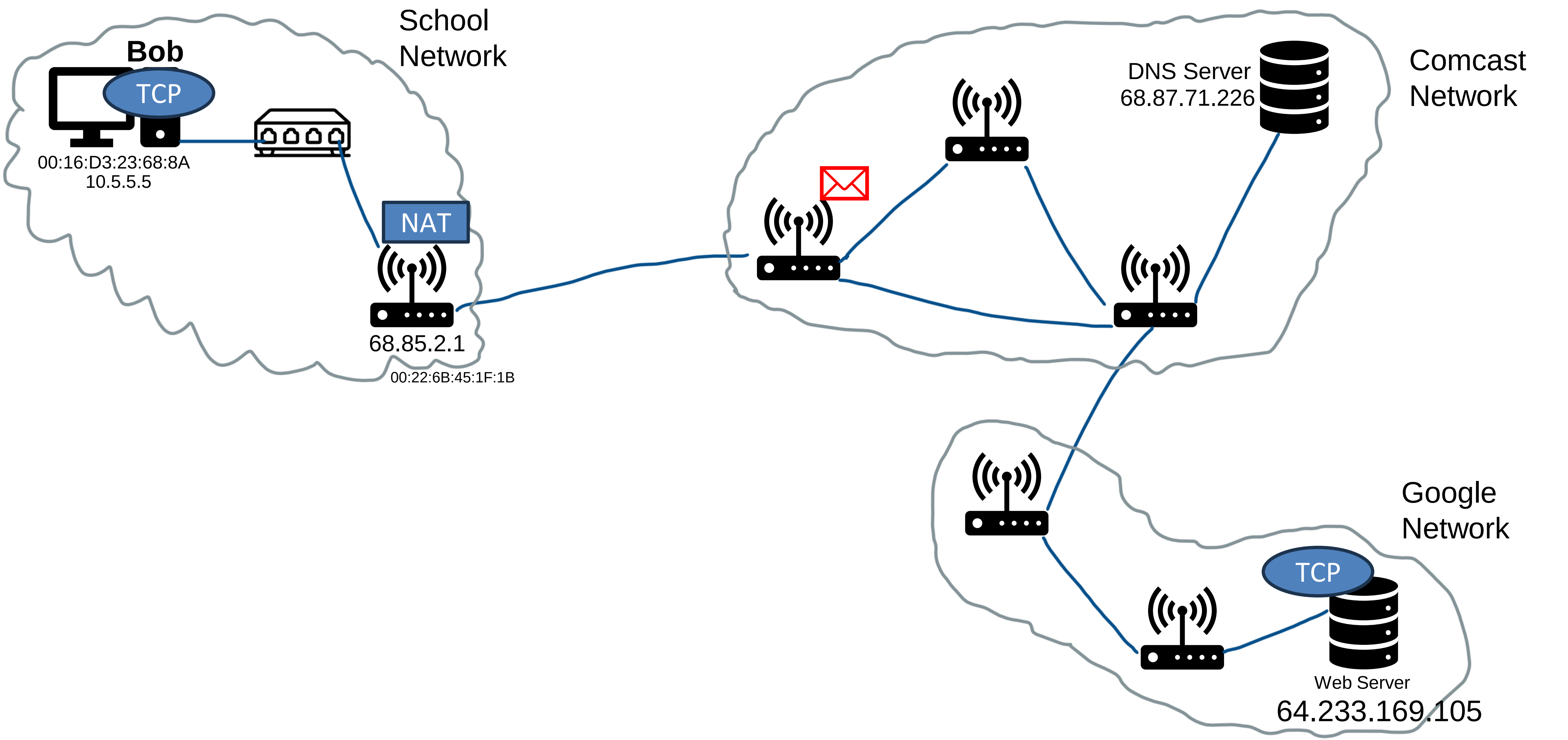


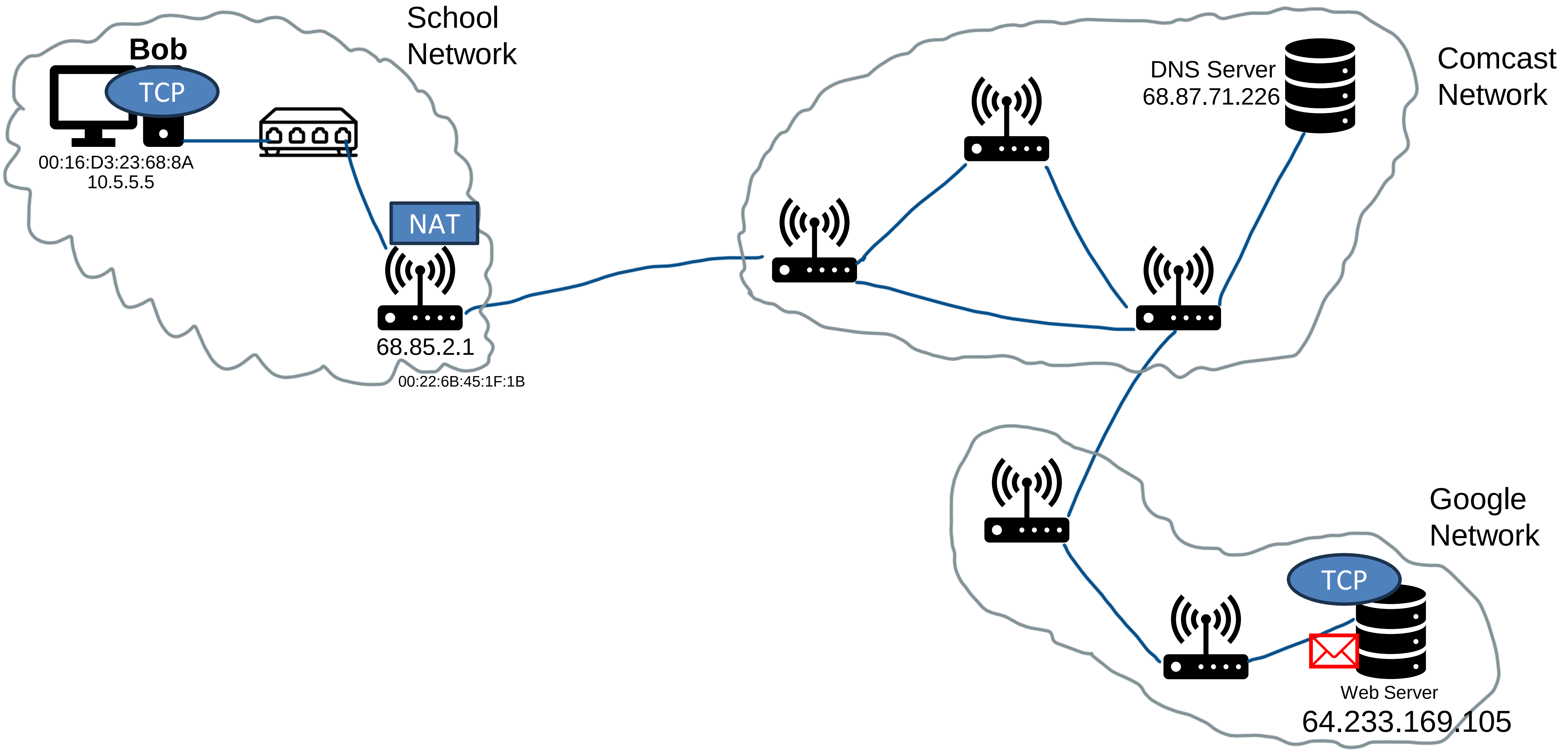
School
Network
Bob
TCP
00:16:D3:23:68:8A
10.5.5.5
NAT
68.85.2.1
00:22:6B:45:1F:1B
DNS Server
68.87.71.226
Comcast
Network
Google
Network
TCP
Web Server
64.233.169.105

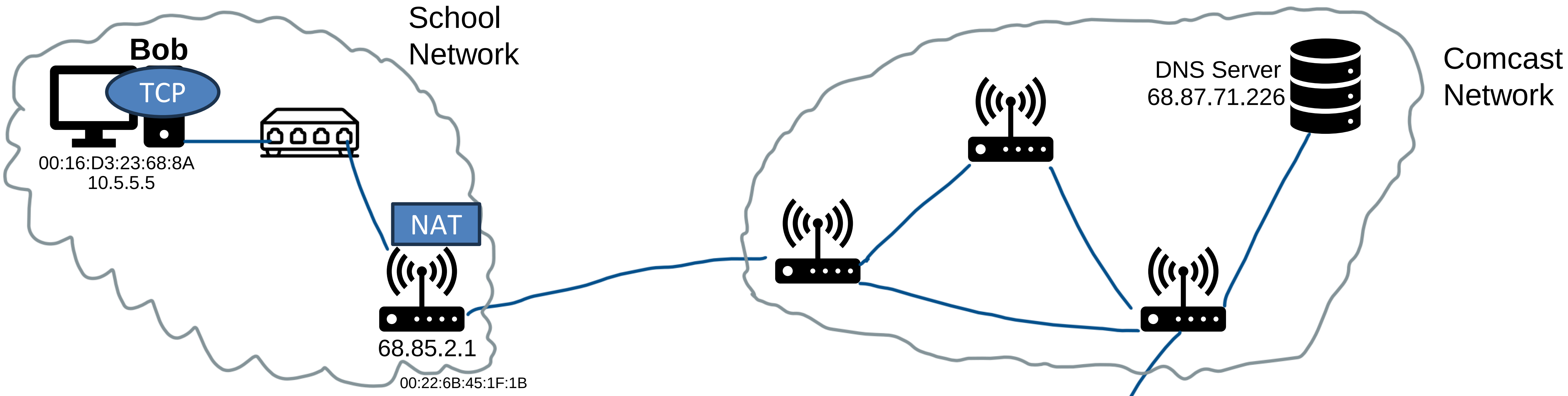
Now what?

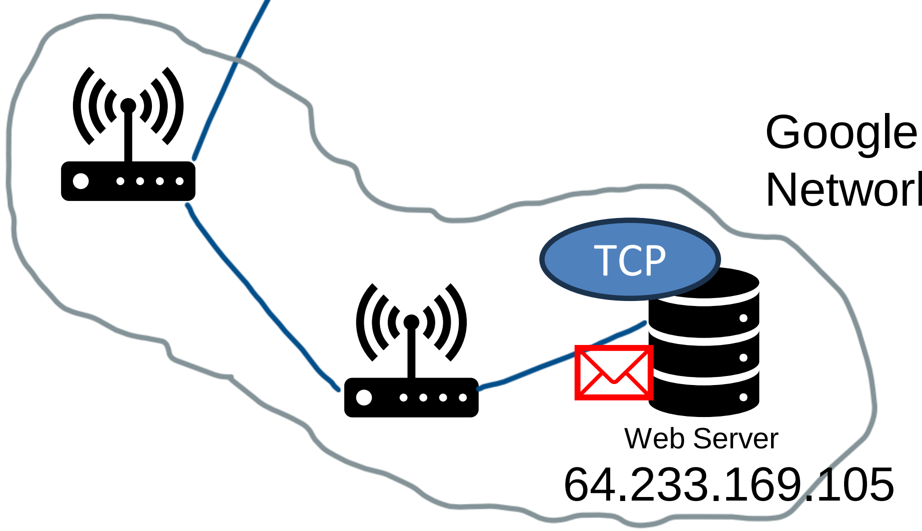


With the socket on Bob's laptop now (*finally!*) ready to send bytes to www.google.com, Bob's browser creates the HTTP GET message containing the URL to be fetched. The HTTP GET message is then written into the socket, with the GET message becoming the payload of a TCP segment. The TCP segment is placed in a datagram and sent and delivered to www.google.com.


School
Network
Bob
TCP
00:16:D3:23:68:8A
10.5.5.5
NAT
68.85.2.1
00:22:6B:45:1F:1B
DNS Server
68.87.71.226
Comcast
Network
Google
Network
TCP
Web Server
64.233.169.105

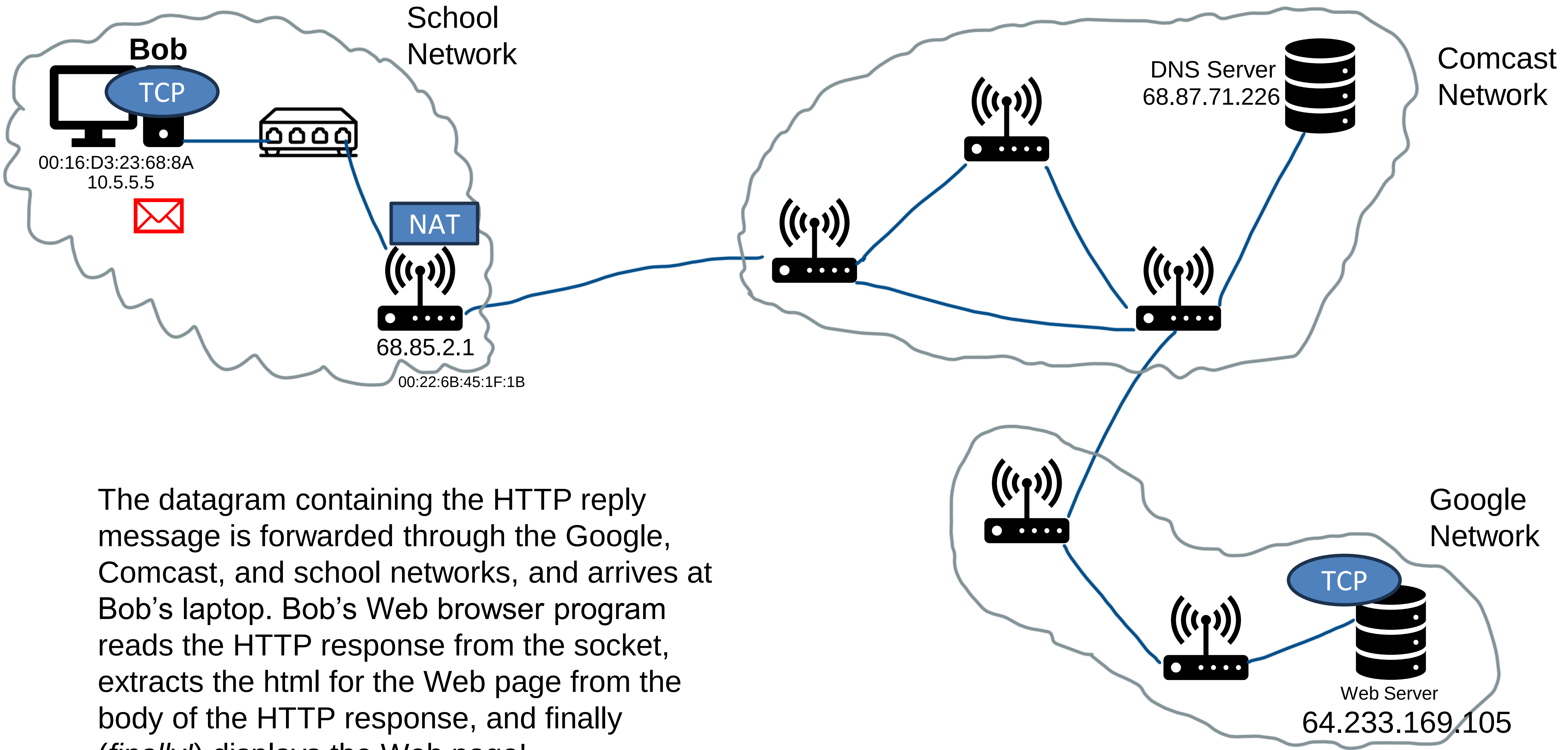

Bob
TCP
00:16:D3:23:68:8A
10.5.5.5
School Network
NAT
68.85.2.1
00:22:6B:45:1F:1B
DNS Server
68.87.71.226
Comcast Network
Google Network
TCP
Web Server
64.233.169.105



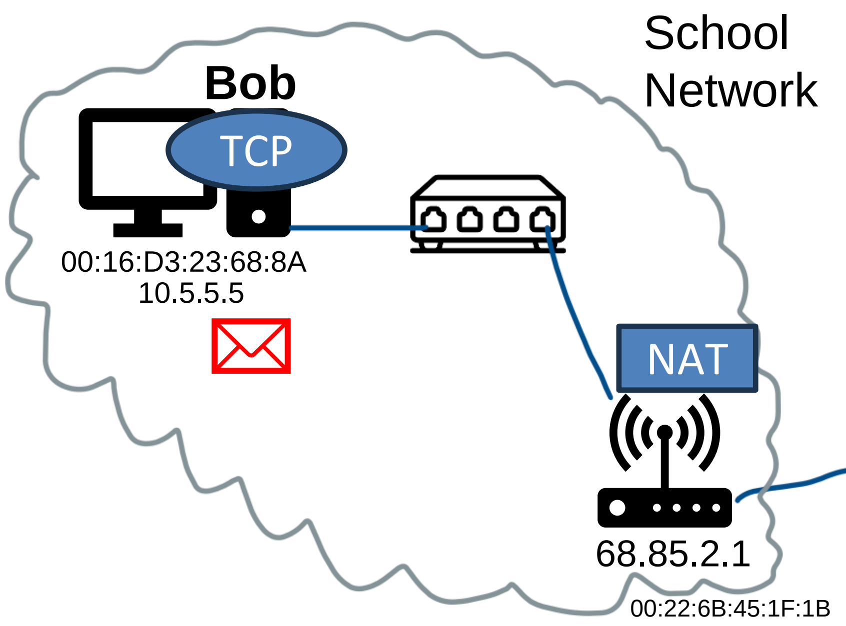


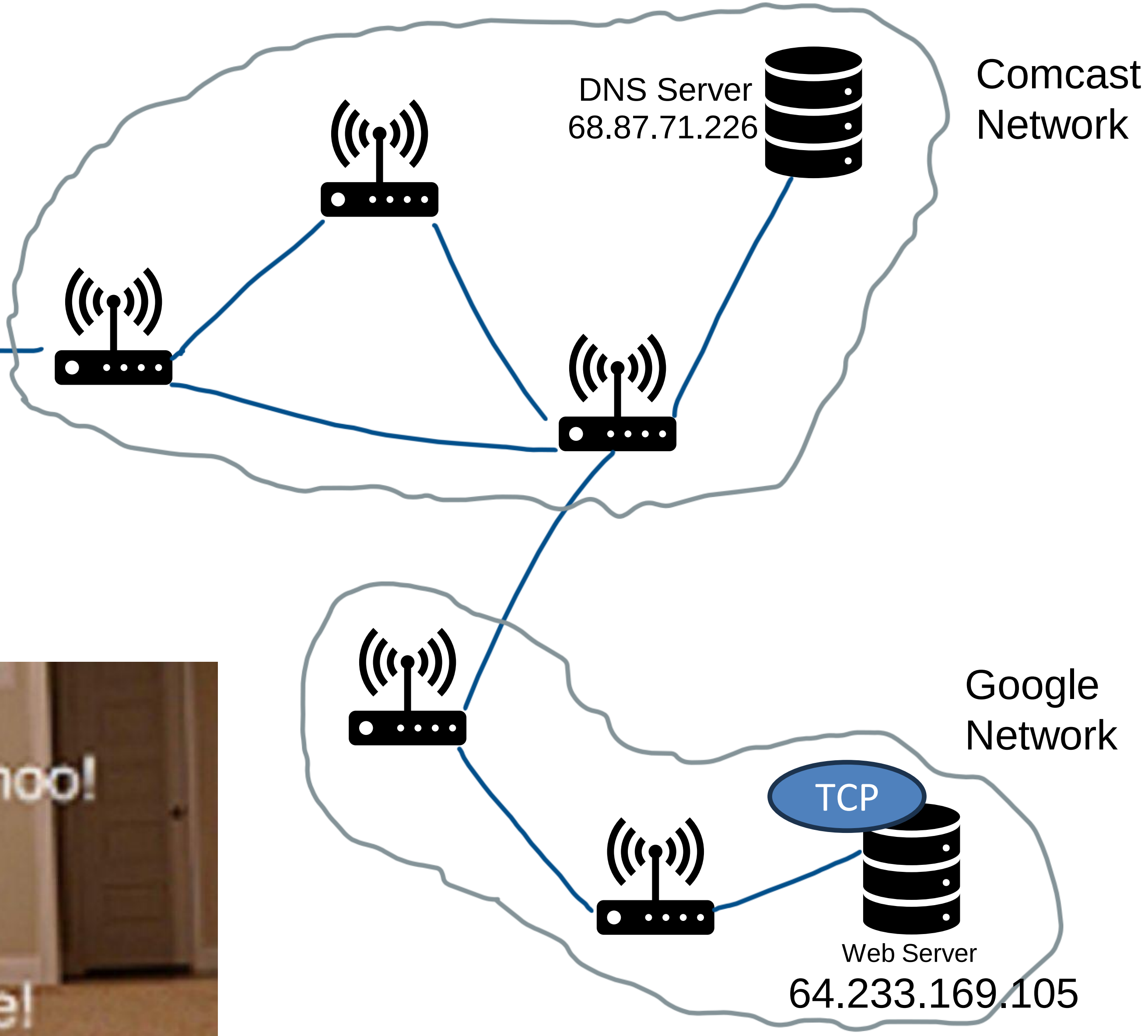
The HTTP server at www.google.com reads the HTTP GET message from the TCP socket, creates an **HTTP response** message places the requested Web page content in the body of the HTTP response message, and sends the message into the TCP socket

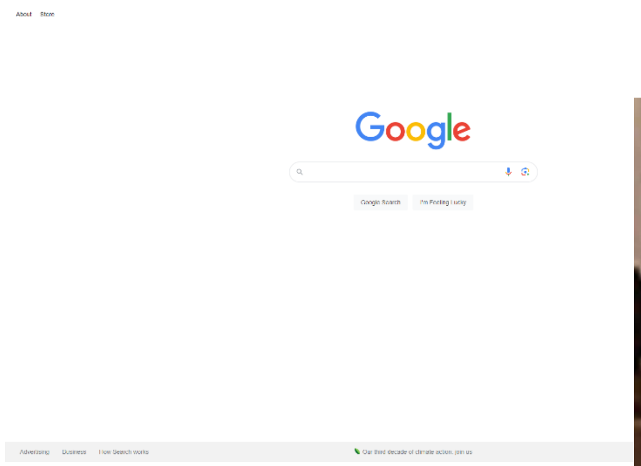
This is only one web object…



The datagram containing the HTTP reply message is forwarded through the Google, Comcast, and school networks, and arrives at Bob's laptop. Bob's Web browser program reads the HTTP response from the socket, extracts the html for the Web page from the body of the HTTP response, and finally (*finally!*) displays the Web page!


School
Network
Bob
TCP
00:16:D3:23:68:8A
10.5.5.5
NAT
68.85.2.1
00:22:6B:45:1F:1B


DNS Server
68.87.71.226
Comcast
Network
Google
Network
TCP
Web Server
64.233.169.105


Google


..yippie! yahoo!
whoopee!