

CSE3213 Computer Network I

Chapter 2

Application Layer Protocols & IP Utilities

Course page:

<http://www.cse.yorku.ca/course/3213>

Slides modified from Alberto Leon-Garcia and Indra Widjaja

CSE 3213 Summer 2008

Protocols, Services & Layering

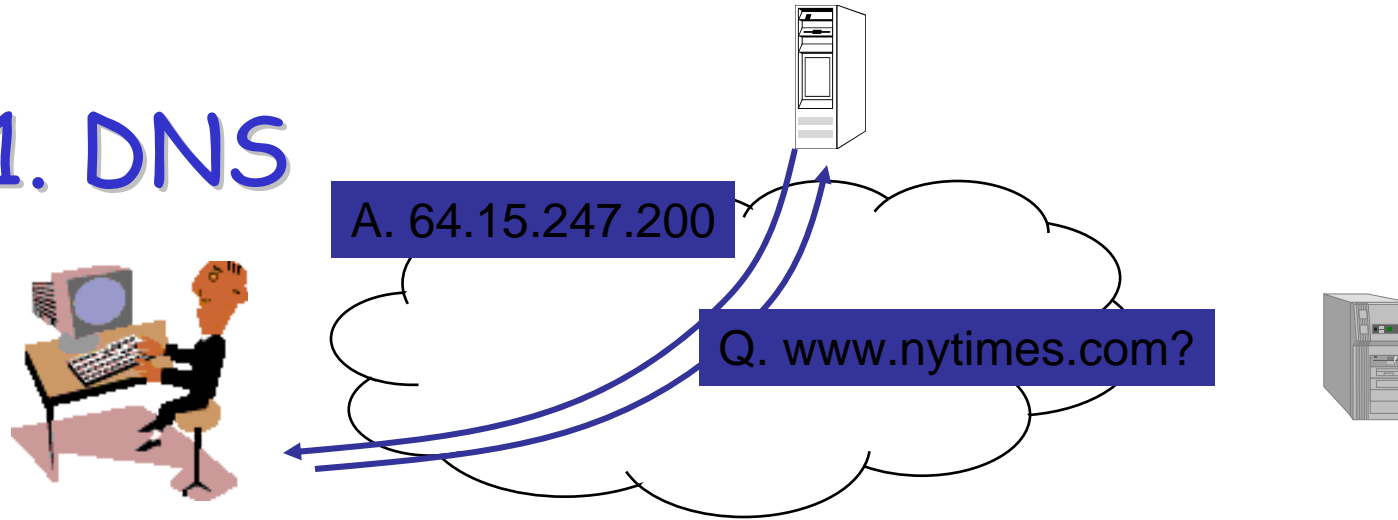
Layers, Services & Protocols

- The overall communications process between two or more machines connected across one or more networks is very complex
- *Layering* partitions related communications functions into groups that are manageable
- Each layer provides a *service* to the layer above
- Each layer operates according to a *protocol*
- Let's use examples to show what we mean

Web Browsing Application

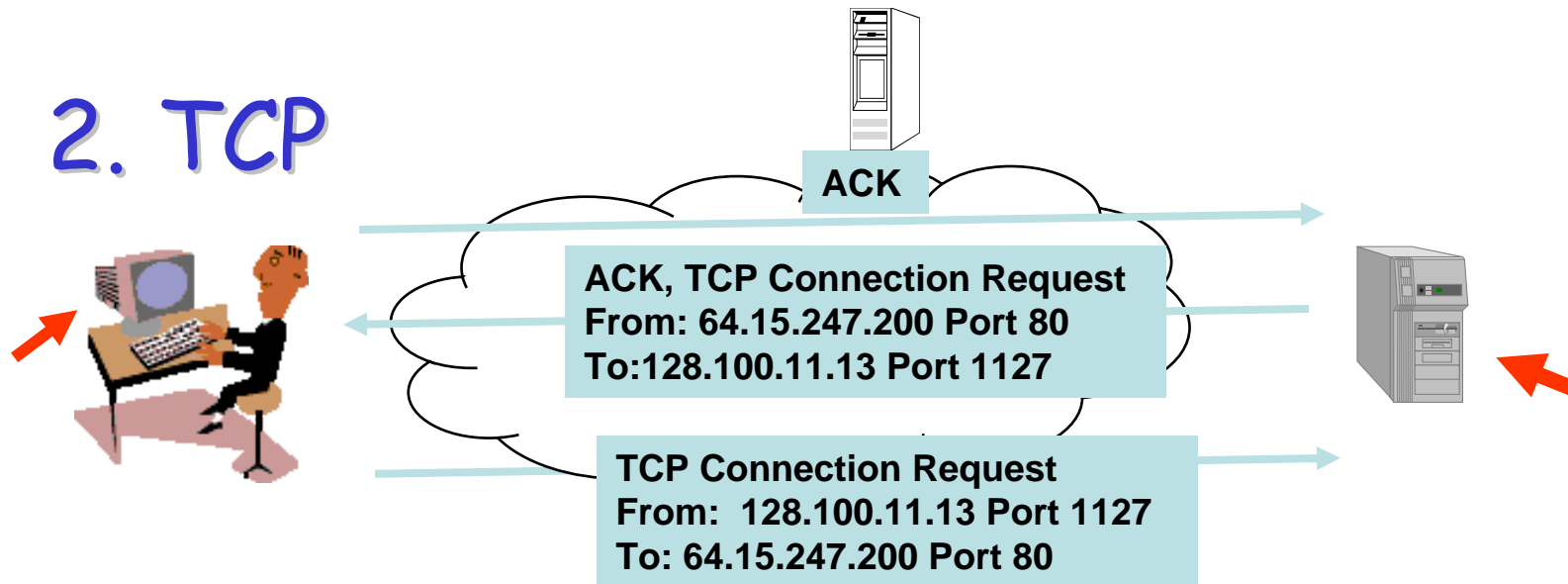
- World Wide Web allows users to access resources (i.e. documents) located in computers connected to the Internet
- Documents are prepared using HyperText Markup Language (HTML)
- A browser application program is used to access the web
- The browser displays HTML documents that include *links* to other documents
- Each link references a *Uniform Resource Locator* (URL) that gives the name of the machine and the location of the given document
- Let's see what happens when a user clicks on a link

1. DNS



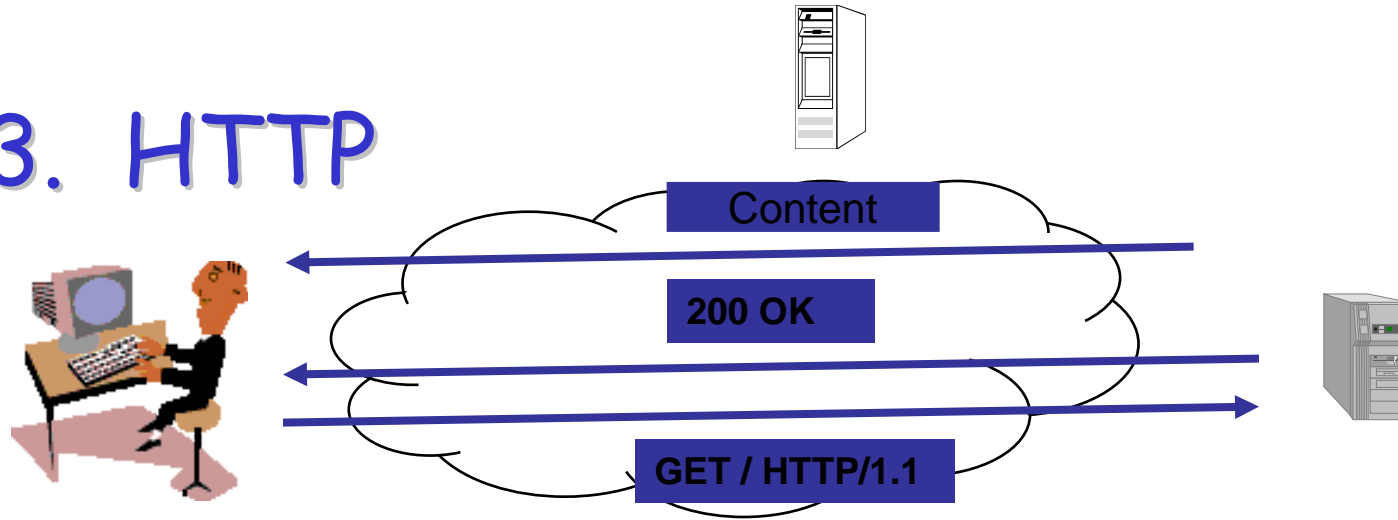
- User clicks on <http://www.nytimes.com/>
- URL contains Internet name of machine (www.nytimes.com), but not Internet address
- Internet needs Internet address to send information to a machine
- Browser software uses Domain Name System (DNS) protocol to send query for Internet address
- DNS system responds with Internet address

2. TCP



- Browser software uses HyperText Transfer Protocol (HTTP) to send request for document
- HTTP server waits for requests by listening to a well-known port number (80 for HTTP)
- HTTP client sends request messages through an "ephemeral port number," e.g. 1127
- HTTP needs a Transmission Control Protocol (TCP) connection between the HTTP client and the HTTP server to transfer messages reliably

3. HTTP



- HTTP client sends its request message: "GET ..."
- HTTP server sends a status response: "200 OK"
- HTTP server sends requested file
- Browser displays document
- Clicking a link sets off a chain of events across the Internet!
- Let's see how protocols & layers come into play...

Protocols

- A *protocol* is a set of rules that governs how two or more communicating entities in a layer are to interact
- *Messages* that can be sent and received
- *Actions* that are to be taken when a certain event occurs, e.g. sending or receiving messages, expiry of timers
- **The purpose of a protocol is to provide a service to the layer above**

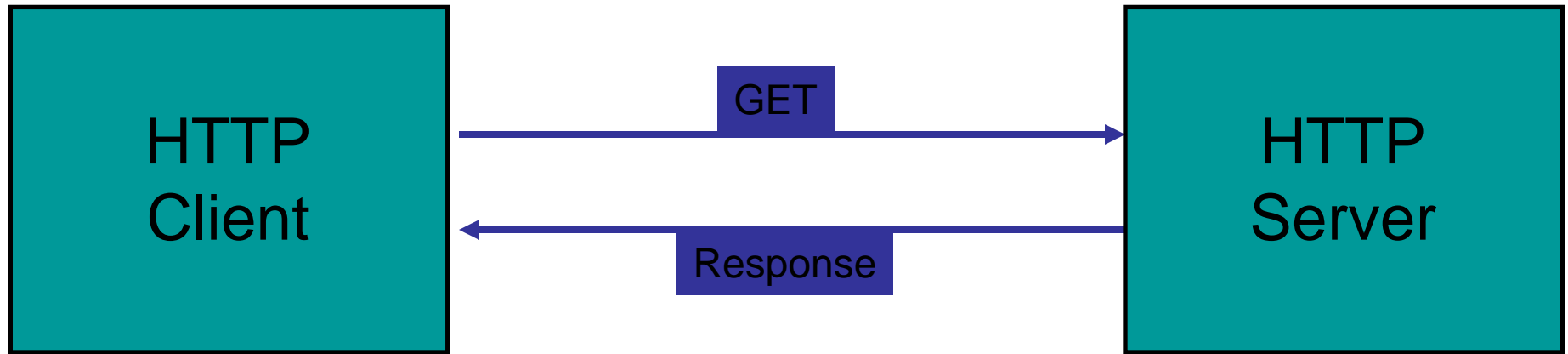
Layers

- A set of related communication functions that can be managed and grouped together
- Application Layer: communications functions that are used by application programs
 - HTTP, DNS, SMTP (email)
- Transport Layer: end-to-end communications between two processes in two machines
 - TCP, User Datagram Protocol (UDP)
- Network Layer: node-to-node communications between two machines
 - Internet Protocol (IP)

Example: HTTP

- HTTP is an application layer protocol
- Retrieves documents on behalf of a browser application program
- HTTP specifies fields in request messages and response messages
 - Request types; Response codes
 - Content type, options, cookies, ...
- HTTP specifies actions to be taken upon receipt of certain messages

HTTP Protocol

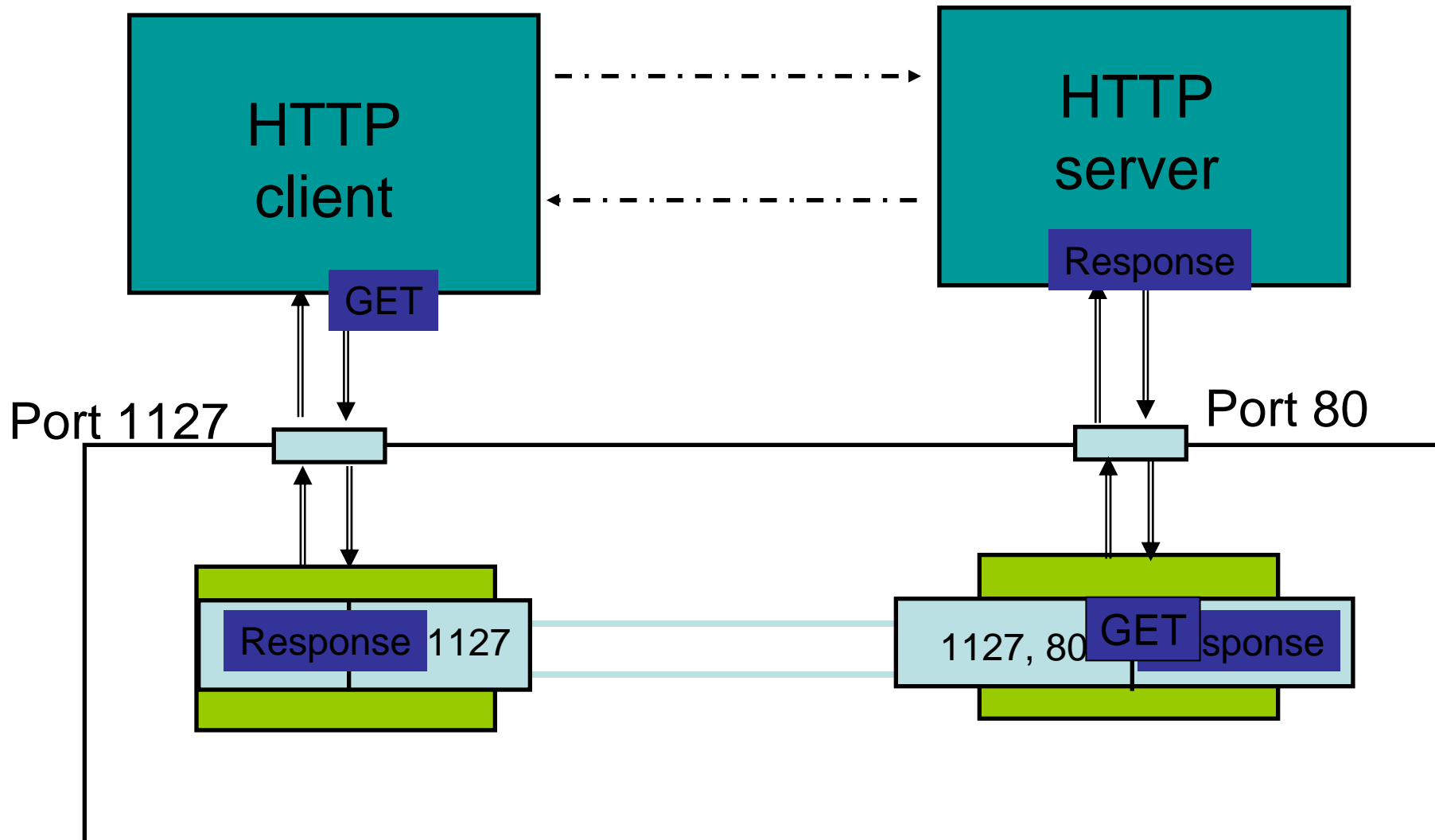


- HTTP assumes messages can be exchanged directly between HTTP client and HTTP server
- In fact, HTTP client and server are processes running in two different machines across the Internet
- HTTP uses the reliable stream transfer service provided by TCP

Example: TCP

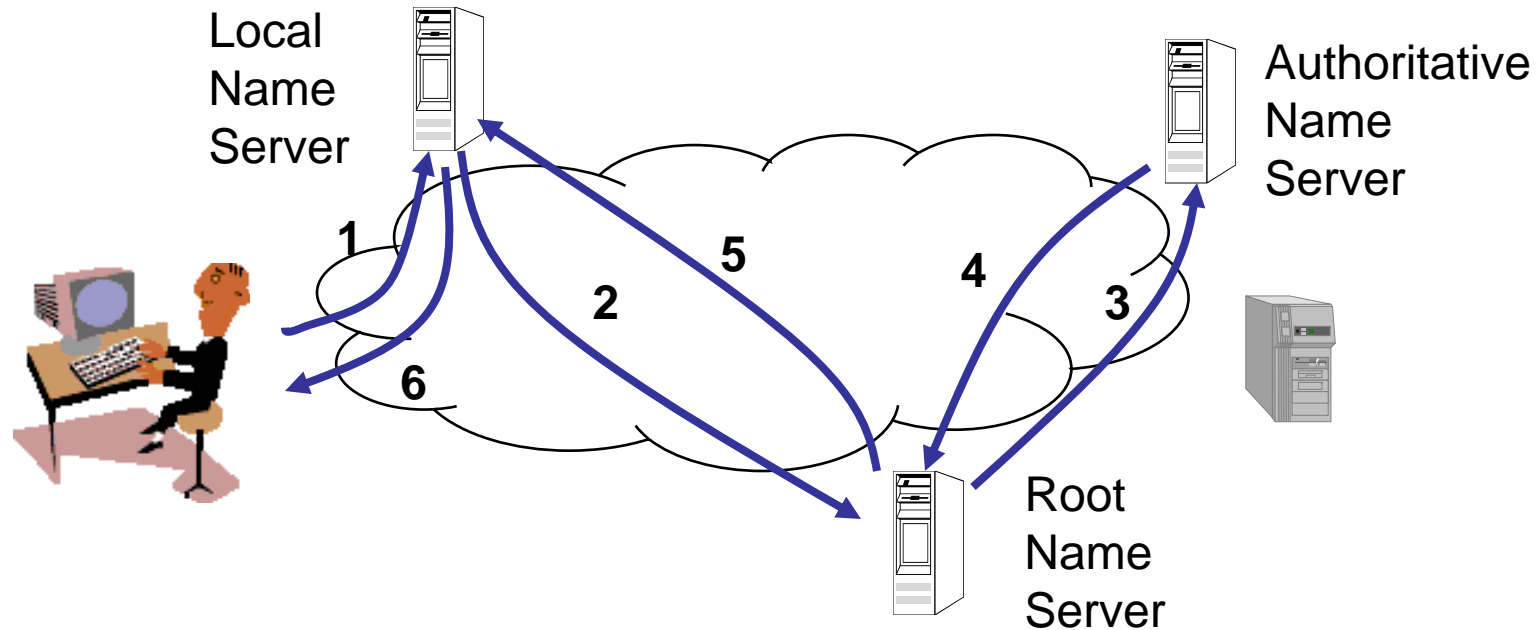
- TCP is a transport layer protocol
- Provides *reliable byte stream service* between two processes in two computers across the Internet
- Sequence numbers keep track of the bytes that have been transmitted and received
- Error detection and retransmission used to recover from transmission errors and losses
- TCP is *connection-oriented*: the sender and receiver must first establish an association and set initial sequence numbers before data is transferred
- Connection ID is specified uniquely by
(*send port #, send IP address, receive port #, receiver IP address*)

HTTP uses service of TCP



Example: DNS Protocol

- DNS protocol is an application layer protocol
- DNS is a distributed database that resides in multiple machines in the Internet
- DNS protocol allows queries of different types
 - Name-to-address or Address-to-name
 - Mail exchange
- DNS usually involves short messages and so uses service provided by UDP
- Well-known port 53



- Local Name Server: resolve frequently-used names
 - University department, ISP
 - Contacts Root Name server if it cannot resolve query
- Root Name Servers: 13 globally
 - Resolves query or refers query to Authoritative Name Server
- Authoritative Name Server: last resort
 - Every machine must register its address with at least two authoritative name servers

Example: UDP

- UDP is a transport layer protocol
- Provides *best-effort datagram service* between two processes in two computers across the Internet
- Port numbers distinguish various processes in the same machine
- UDP is *connectionless*
- Datagram is sent immediately
- Quick, simple, but not reliable

Summary

- Layers: related communications functions
 - Application Layer: HTTP, DNS
 - Transport Layer: TCP, UDP
 - Network Layer: IP
- Services: a protocol provides a communications service to the layer above
 - TCP provides connection-oriented reliable byte transfer service
 - UDP provides best-effort datagram service
- Each layer builds on services of lower layers
 - HTTP builds on top of TCP
 - DNS builds on top of UDP
 - TCP and UDP build on top of IP

OSI Reference Model

Why Layering?

- Layering simplifies design, implementation, and testing by partitioning overall communications process into parts
- Protocol in each layer can be designed separately from those in other layers
- Protocol makes “calls” for services from layer below
- Layering provides flexibility for modifying and evolving protocols and services without having to change layers below
- Monolithic non-layered architectures are costly, inflexible, and soon obsolete

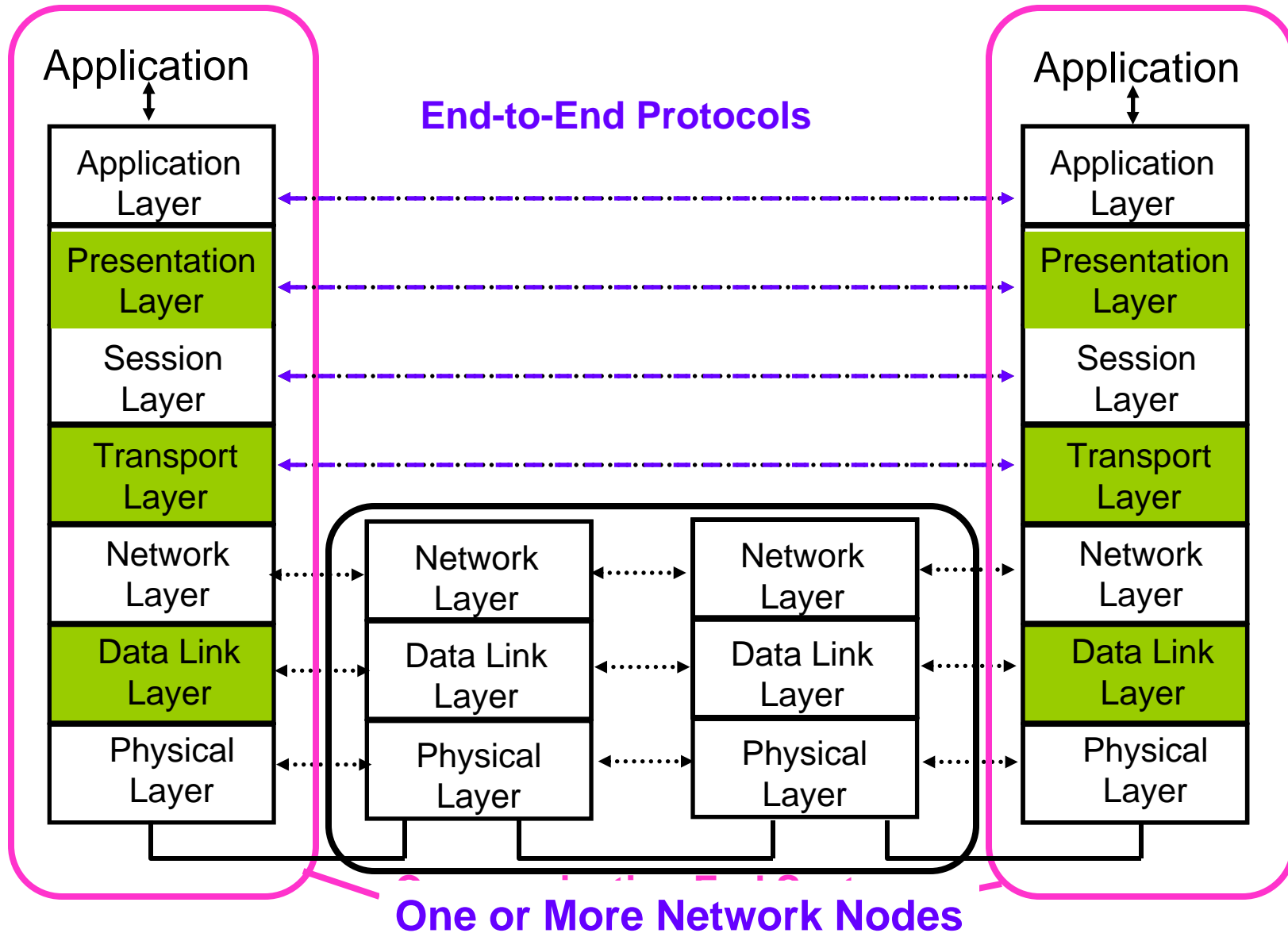
Open Systems Interconnection

- Network architecture:
 - Definition of all the layers
 - Design of protocols for every layer
- By the 1970s every computer vendor had developed its own proprietary layered network architecture
- Problem: computers from different vendors could not be networked together
- Open Systems Interconnection (OSI) was an international effort by the International Organization for Standardization (ISO) to enable multivendor computer interconnection

OSI Reference Model

- Describes a seven-layer abstract reference model for a network architecture
- Purpose of the reference model was to provide a framework for the development of protocols
- OSI also provided a unified view of layers, protocols, and services which is still in use in the development of new protocols
- Detailed standards were developed for each layer, but most of these are not in use
- TCP/IP protocols preempted deployment of OSI protocols

7-Layer OSI Reference Model



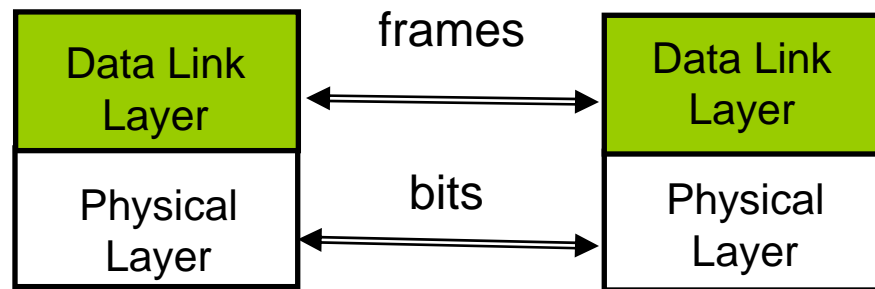
Physical Layer



- Transfers bits across link
- Definition & specification of the physical aspects of a communications link
 - Mechanical: cable, plugs, pins...
 - Electrical/optical: modulation, signal strength, voltage levels, bit times, ...
 - functional/procedural: how to activate, maintain, and deactivate physical links...
- Ethernet, DSL, cable modem, telephone modems...
- Twisted-pair cable, coaxial cable optical fiber, radio, infrared, ...

Data Link Layer

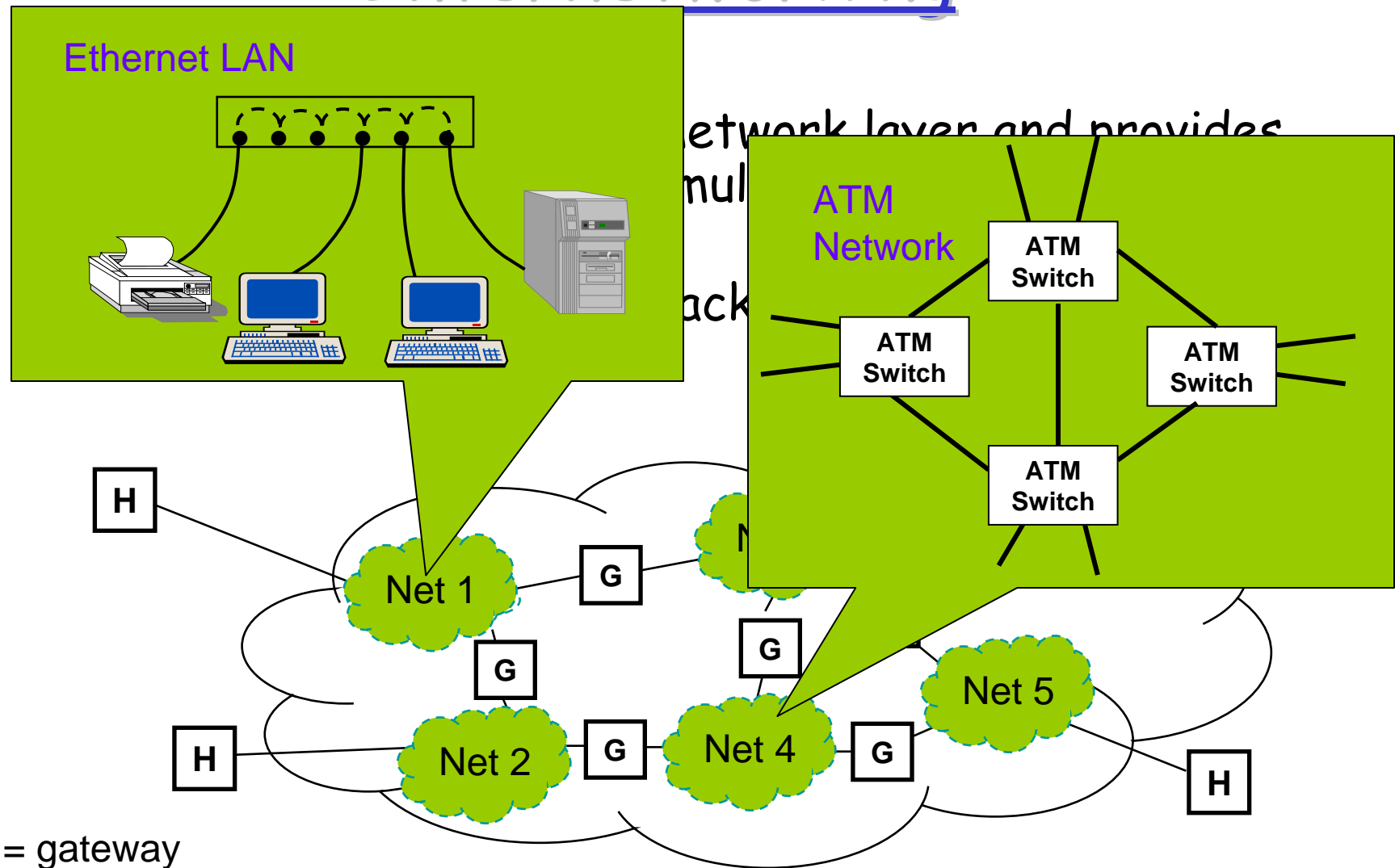
- Transfers *frames* across *direct* connections
- Groups bits into frames
- Detection of bit errors; Retransmission of frames
- Activation, maintenance, & deactivation of data link connections
- Medium access control for local area networks
- Flow control



Network Layer

- Transfers *packets* across multiple links and/or multiple networks
- Addressing must scale to large networks
- Nodes *jointly* execute routing algorithm to determine paths across the network
- Forwarding transfers packet across a node
- Congestion control to deal with traffic surges
- Connection setup, maintenance, and teardown when connection-based

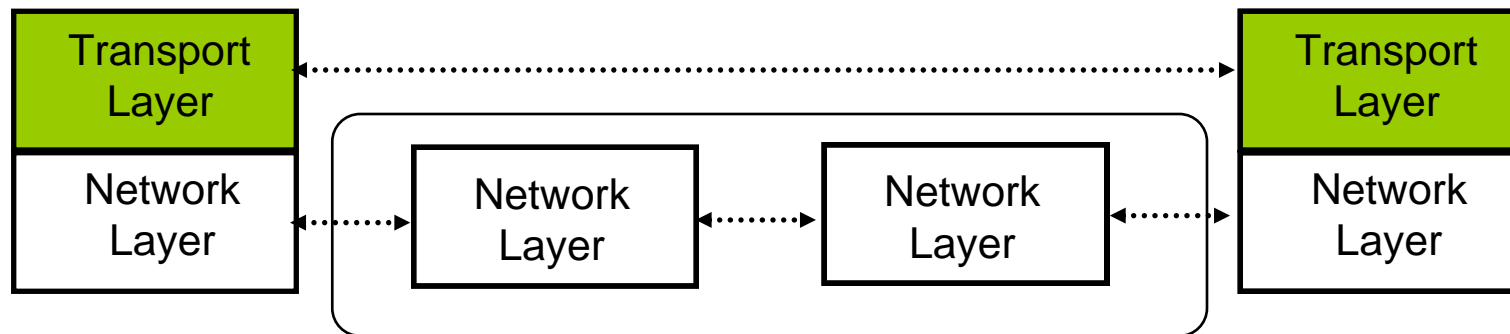
Internetworking



G = gateway
H = host

Transport Layer

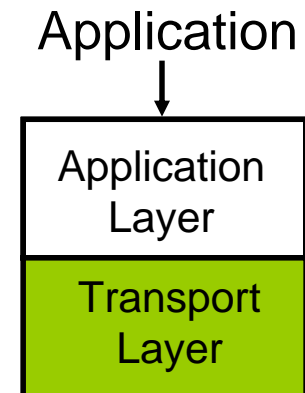
- Transfers data end-to-end from process in a machine to process in another machine
- Reliable stream transfer or quick-and-simple single-block transfer
- Port numbers enable multiplexing
- Message segmentation and reassembly
- Connection setup, maintenance, and release



Application & Upper Layers

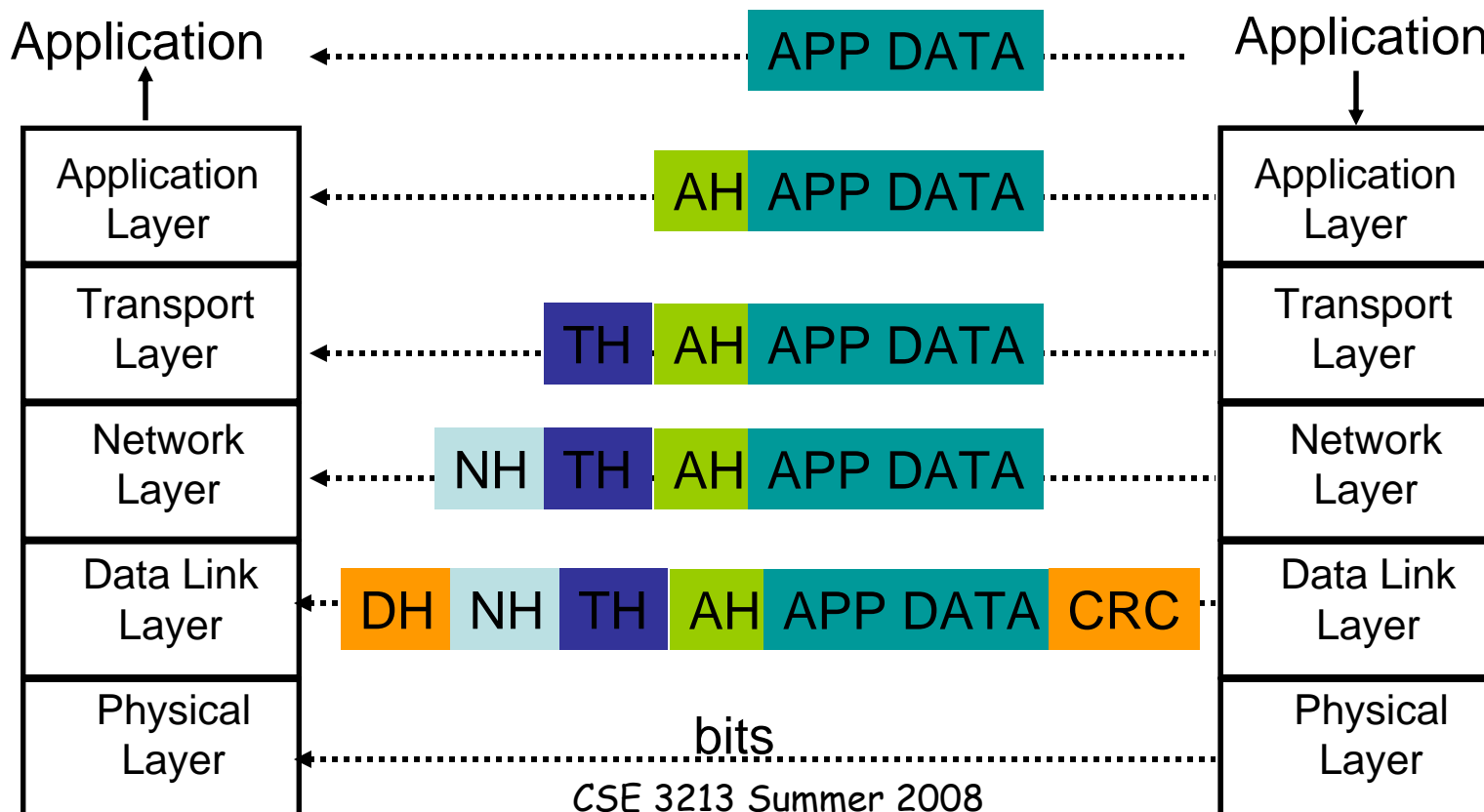
- Application Layer: Provides services that are frequently required by applications: DNS, web access, file transfer, email...
- ~~• Presentation Layer: machine-independent representation of data...~~
- ~~• Session Layer: dialog management, recovery from errors, ...~~

**Incorporated into
Application Layer**



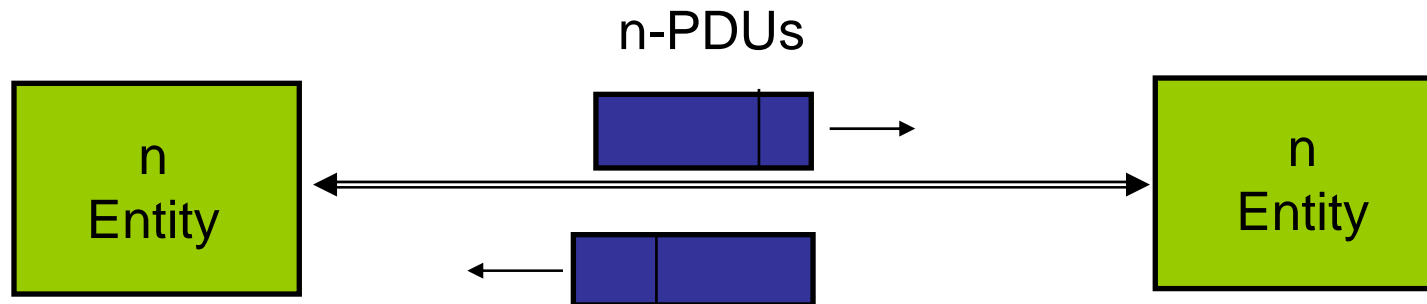
Headers & Trailers

- Each protocol uses a header that carries addresses, sequence numbers, flag bits, length indicators, etc...
- CRC check bits may be appended for error detection



OSI Unified View: Protocols

- Layer n in one machine interacts with layer n in another machine to provide a service to layer $n + 1$
- The entities comprising the corresponding layers on different machines are called *peer processes*.
- The machines use a set of rules and conventions called the *layer- n protocol*.
- Layer- n peer processes communicate by exchanging *Protocol Data Units (PDUs)*



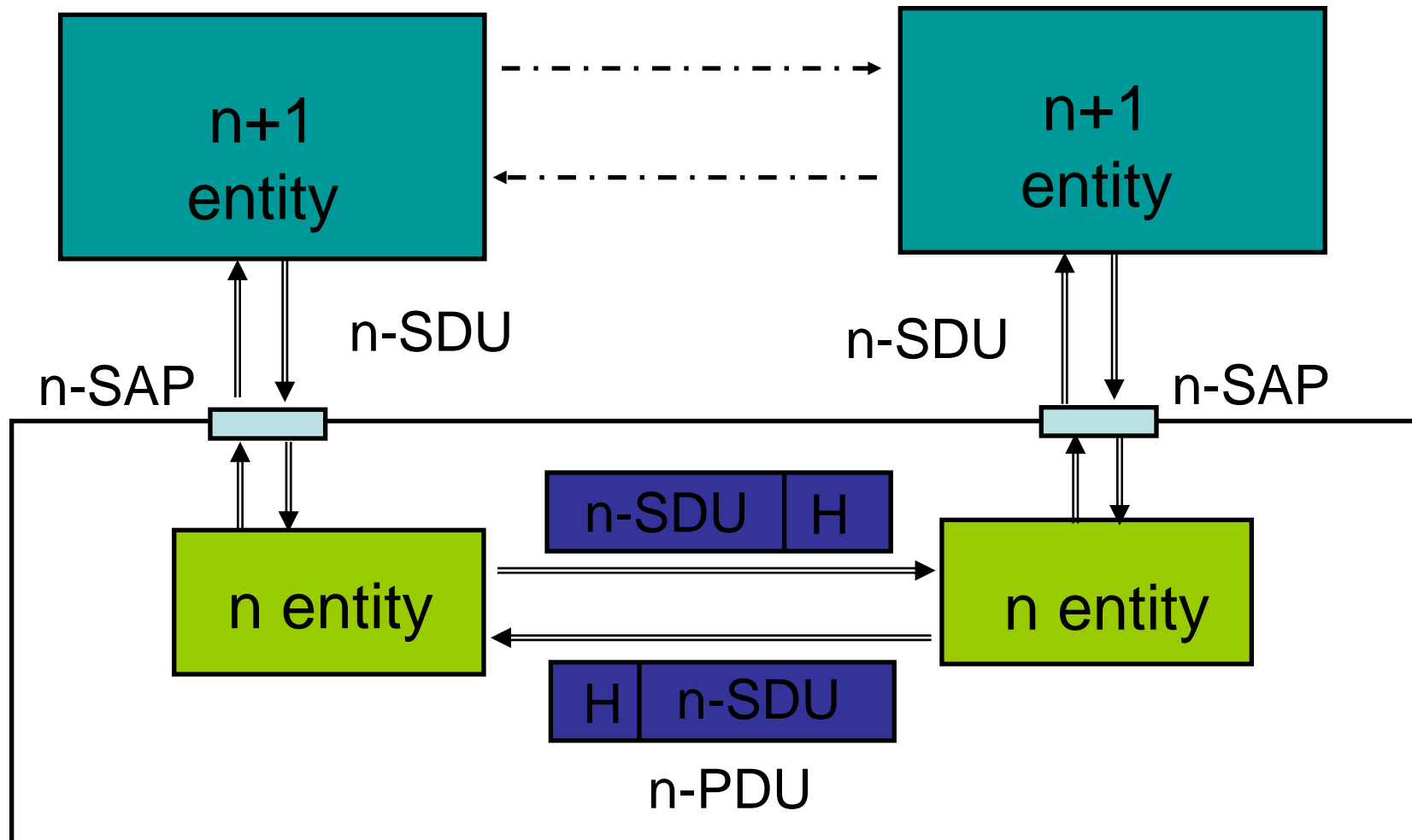
Layer n peer protocol

CSE 3213 Summer 2008

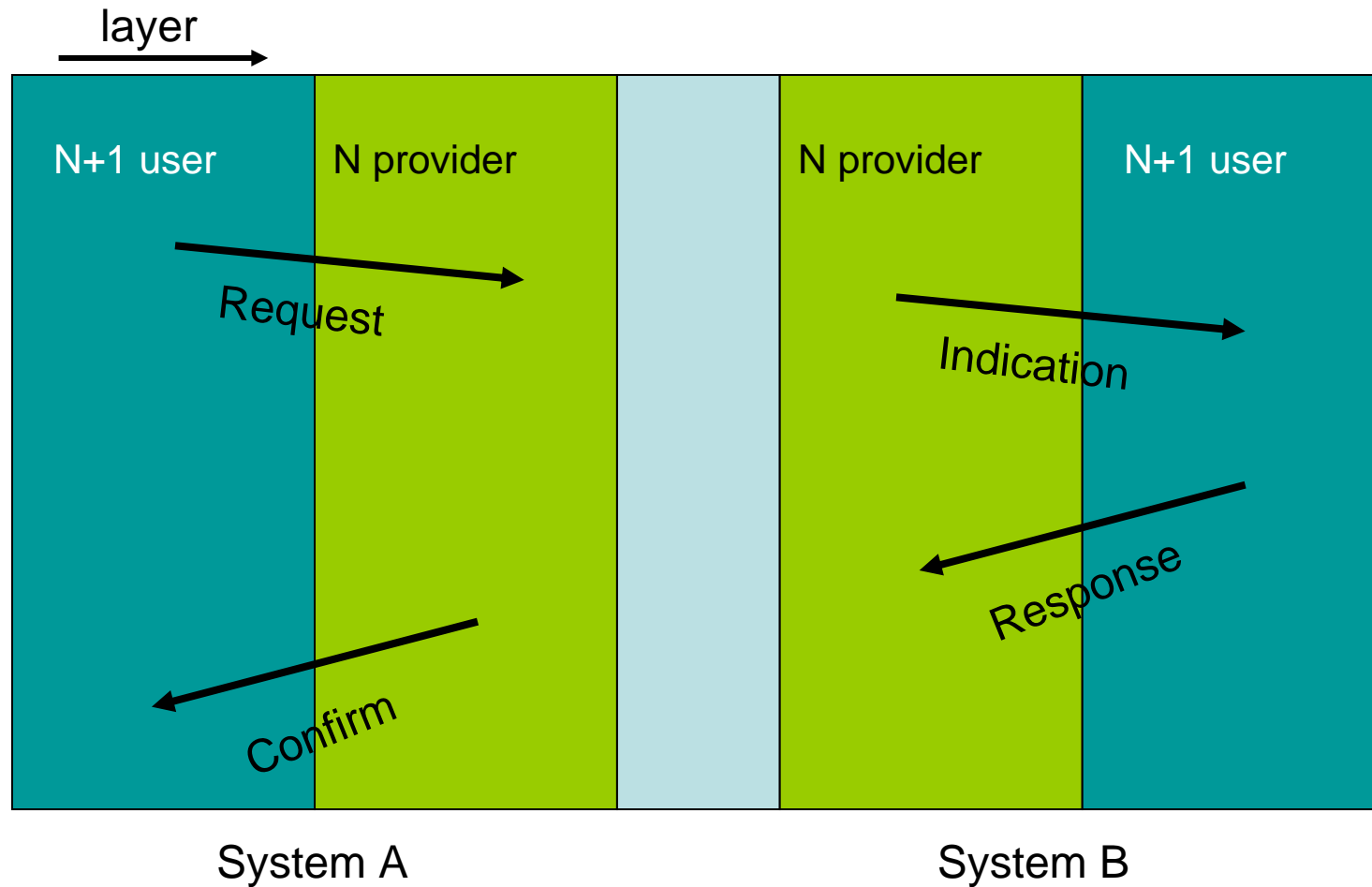
OSI Unified View: Services

- Communication between peer processes is virtual and actually indirect
- Layer $n+1$ transfers information by invoking the services provided by layer n
- Services are available at *Service Access Points (SAP's)*
- Each layer passes data & control information to the layer below it until the physical layer is reached and transfer occurs
- The data passed to the layer below is called a *Service Data Unit (SDU)*
- SDU's are *encapsulated* in PDU's

Layers, Services & Protocols



Interlayer Interaction

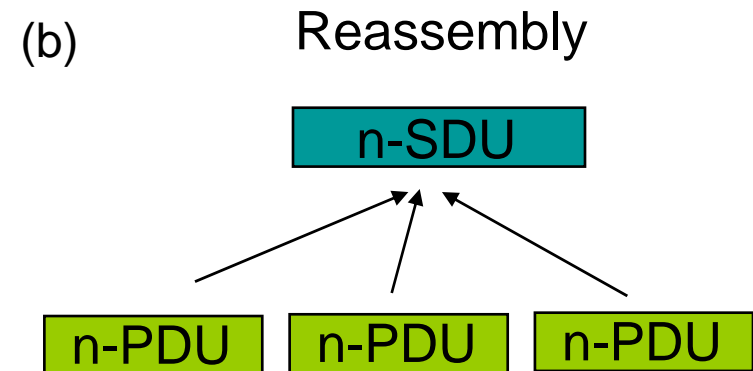
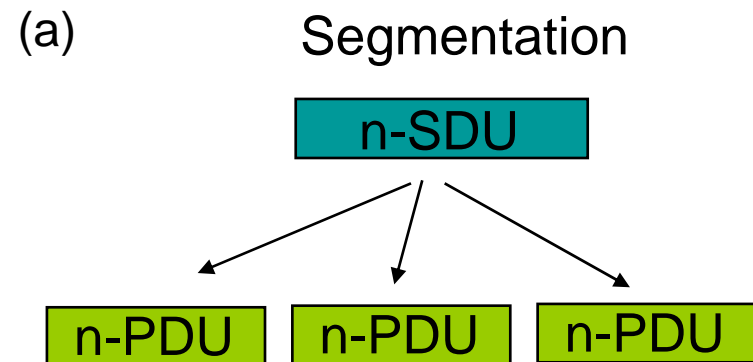


Connectionless & Connection-Oriented Services

- Connection-Oriented
 - Three-phases:
 1. Connection setup between two SAPs to initialize state information
 2. SDU transfer
 3. Connection release
 - E.g. TCP, ATM
- Connectionless
 - Immediate SDU transfer
 - No connection setup
 - E.g. UDP, IP
- Layered services need not be of same type
 - TCP operates over IP
 - IP operates over ATM

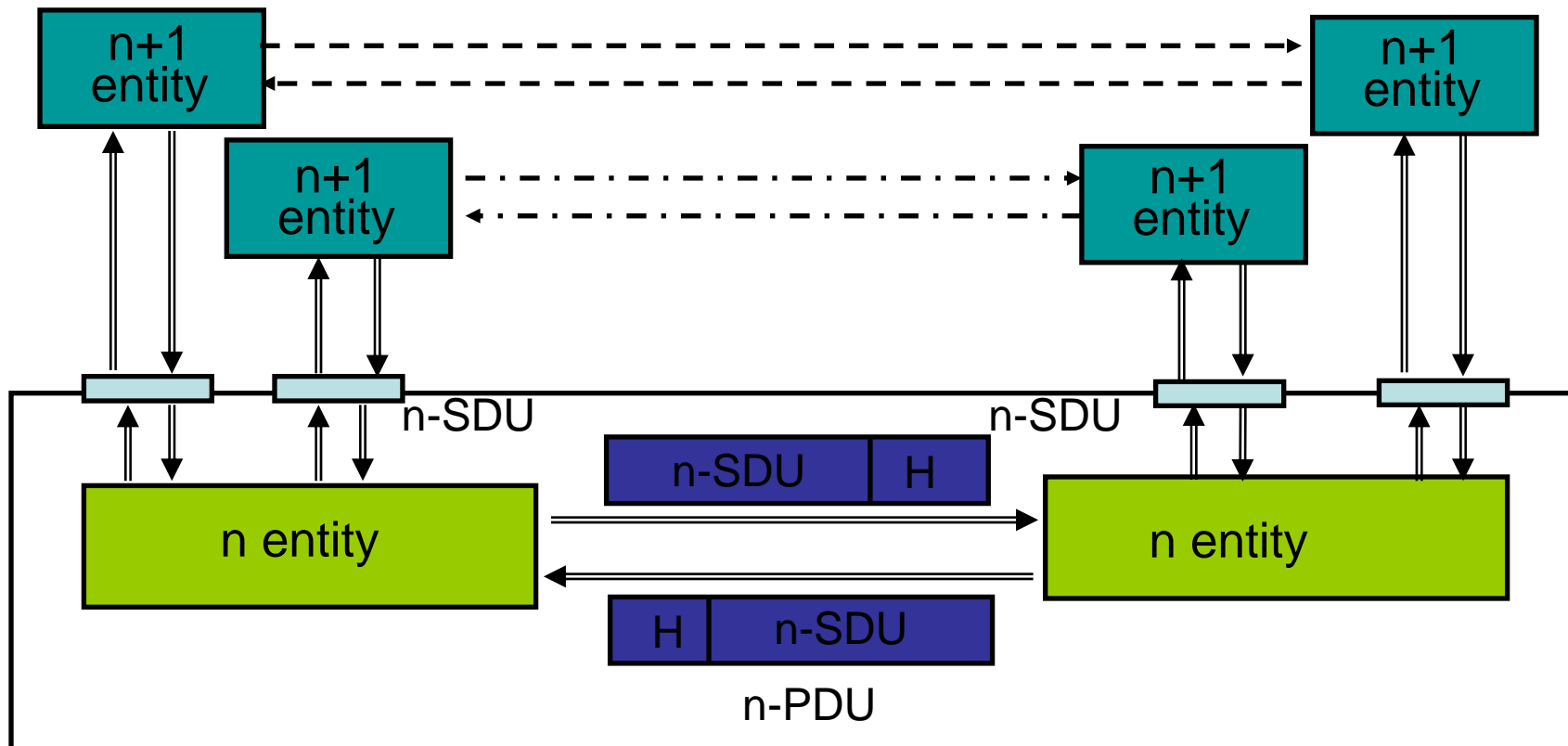
Segmentation & Reassembly

- A layer may impose a limit on the size of a data block that it can transfer for implementation or other reasons
- Thus a layer-n SDU may be too large to be handled as a single unit by layer-(n-1)
- Sender side: SDU is segmented into multiple PDUs
- Receiver side: SDU is reassembled from sequence of PDUs



Multiplexing

- Sharing of layer n service by *multiple* layer $n+1$ users
- Multiplexing tag or ID required in each PDU to determine which users an SDU belongs to



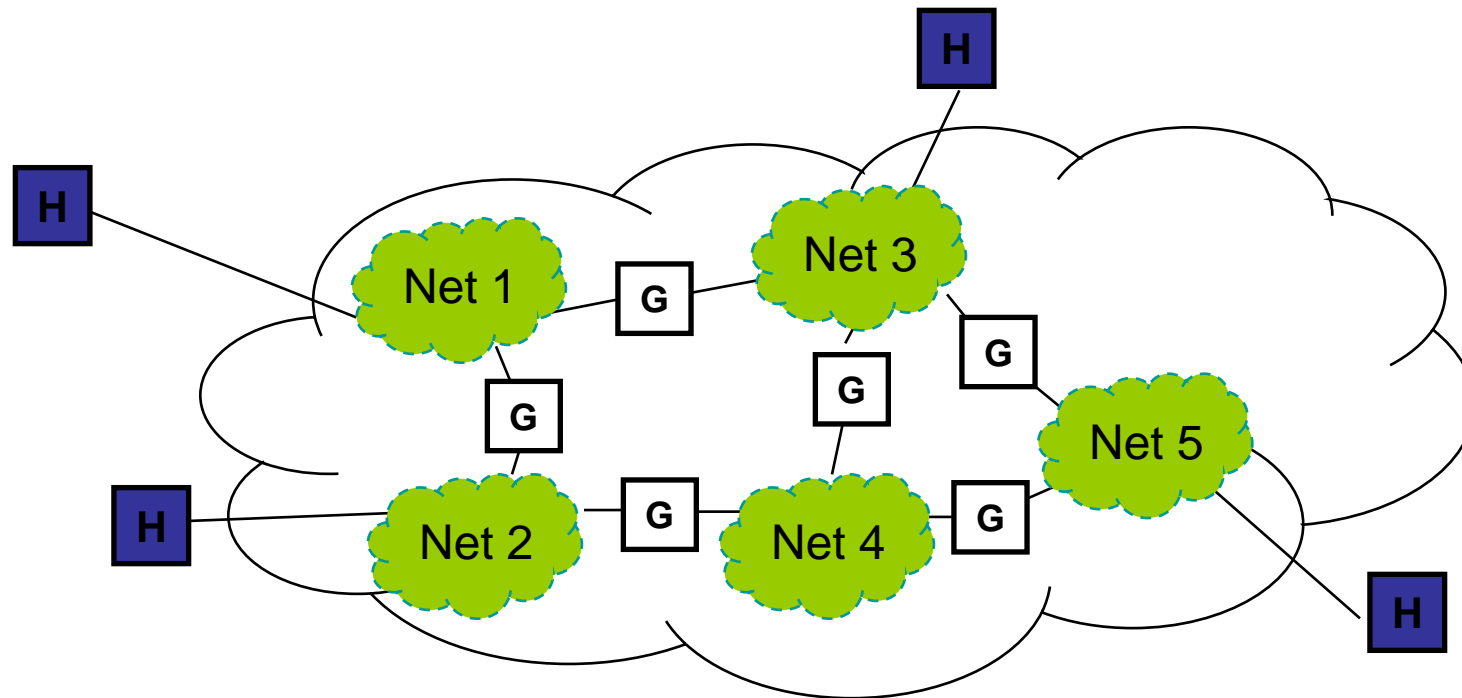
Summary

- Layers: related communications functions
 - Application Layer: HTTP, DNS
 - Transport Layer: TCP, UDP
 - Network Layer: IP
- Services: a protocol provides a communications service to the layer above
 - TCP provides connection-oriented reliable byte transfer service
 - UDP provides best-effort datagram service
- Each layer builds on services of lower layers
 - HTTP builds on top of TCP
 - DNS builds on top of UDP
 - TCP and UDP build on top of IP

TCP/IP Architecture
How the Layers Work Together

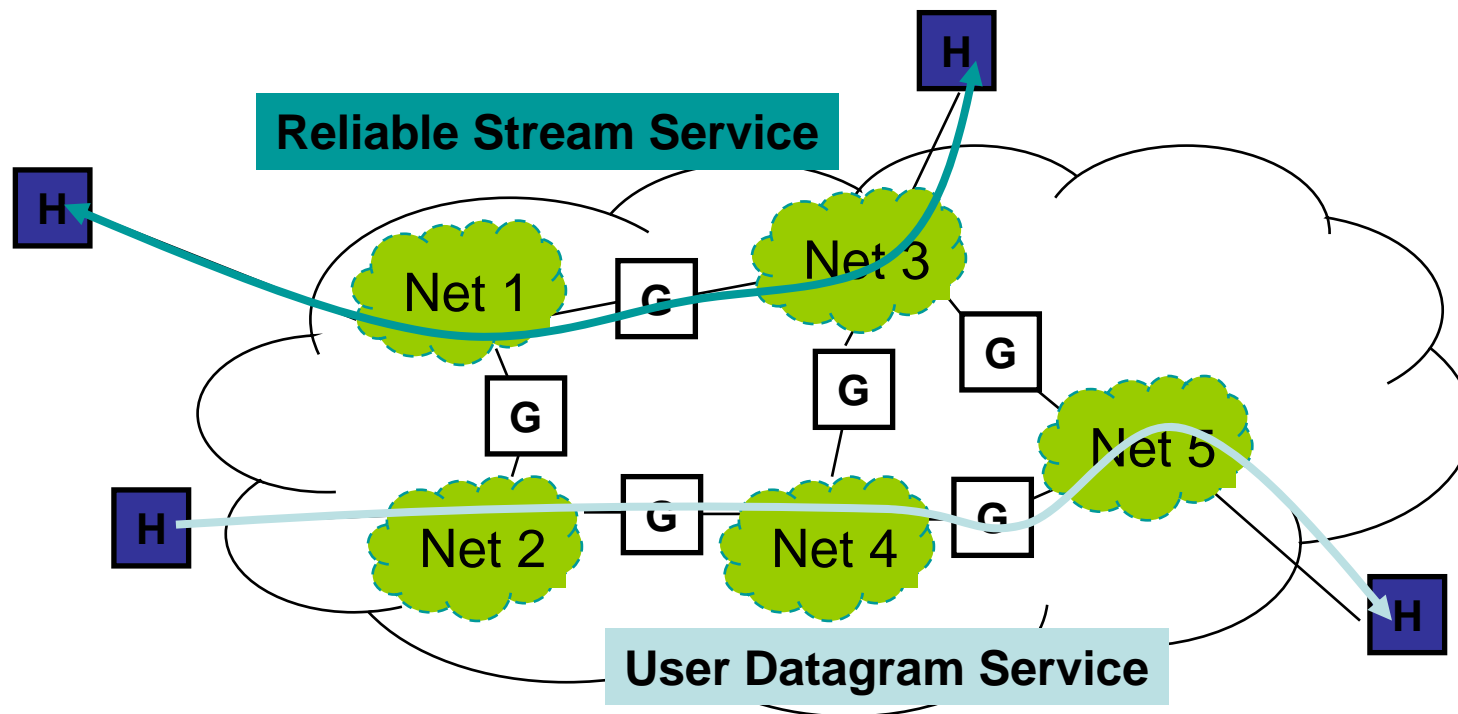
Why Internetworking?

- To build a “network of networks” or internet
 - operating over multiple, coexisting, different network technologies
 - providing ubiquitous connectivity through IP packet transfer
 - achieving huge economies of scale



Why Internetworking?

- To provide *universal communication services*
 - independent of underlying network technologies
 - providing common interface to user applications

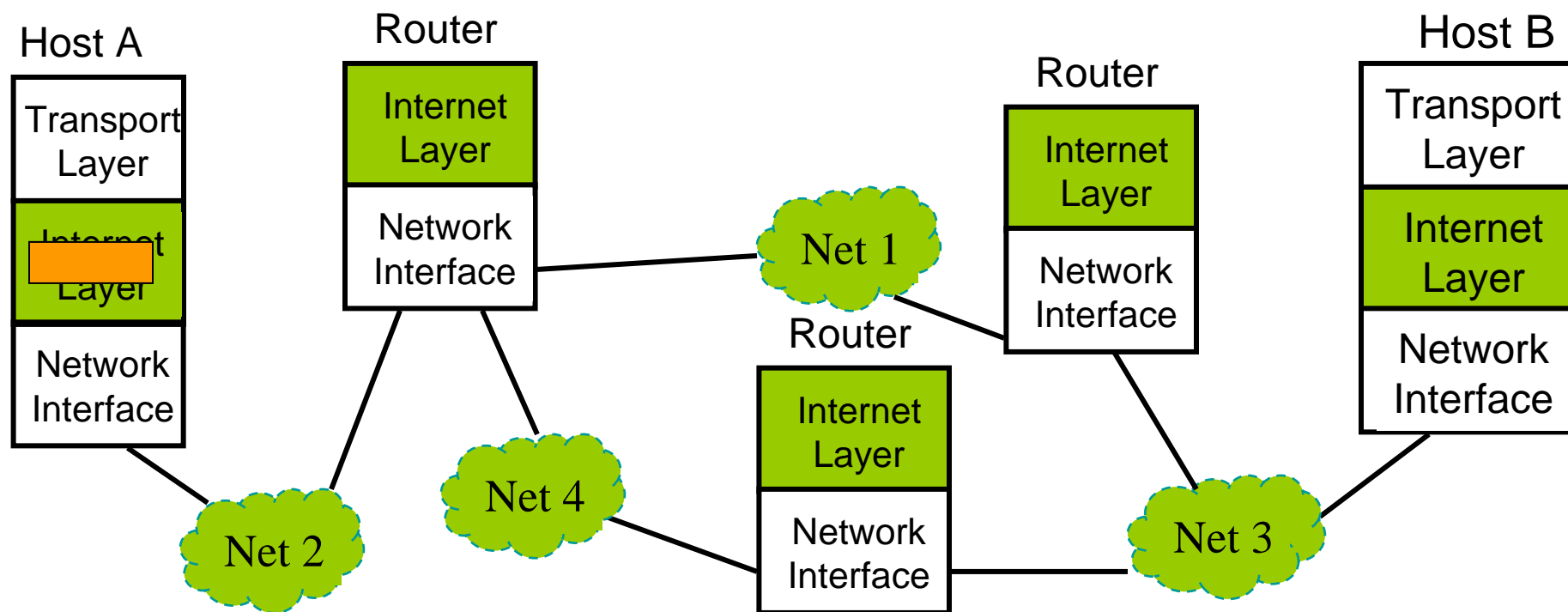


Why Internetworking?

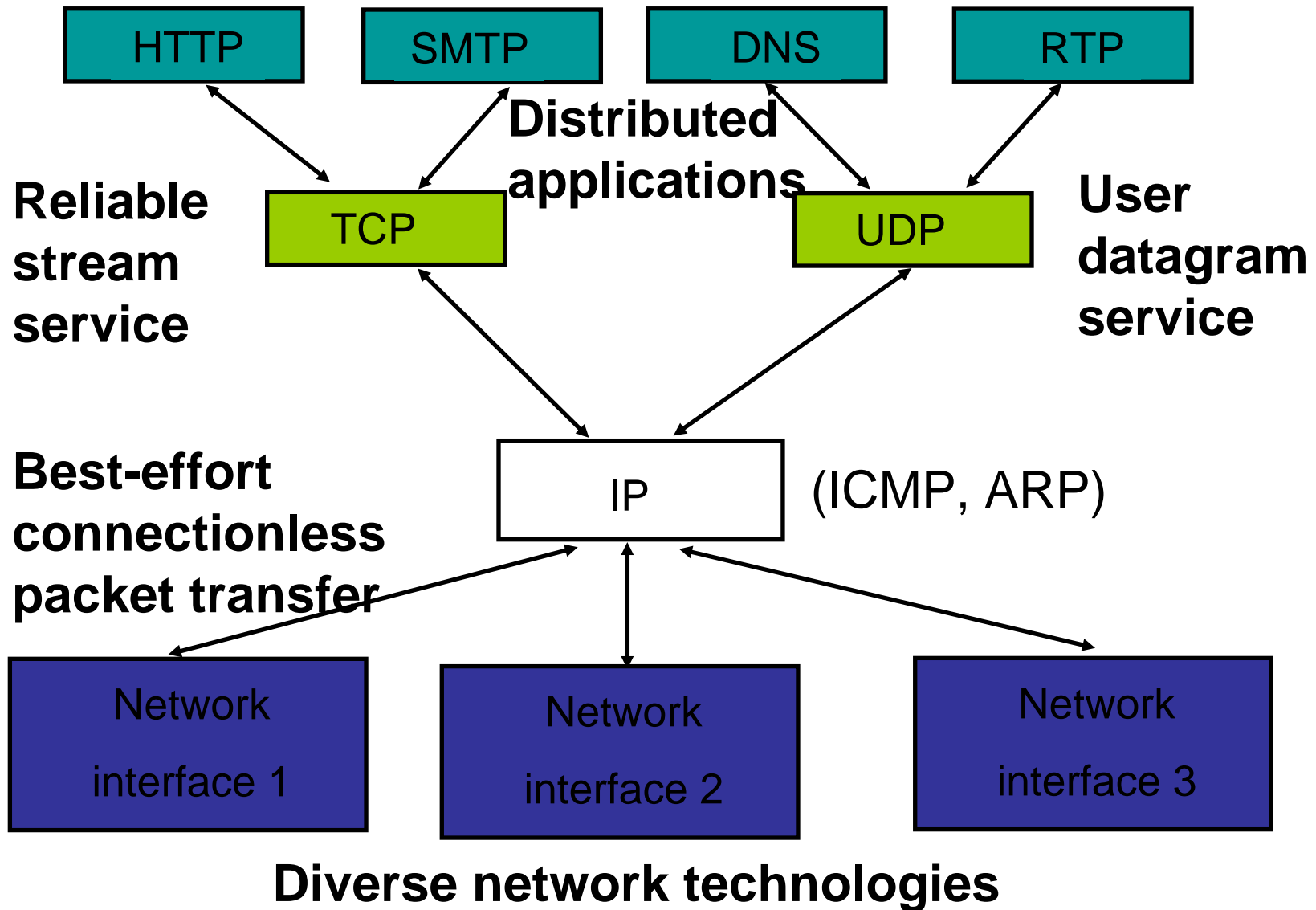
- To provide *distributed applications*
 - Any application designed to operate based on Internet communication services immediately operates across the entire Internet
 - Rapid deployment of new applications
 - Email, WWW, Peer-to-peer
 - Applications independent of network technology
 - New networks can be introduced below
 - Old network technologies can be retired

Internet Protocol Approach

- IP packets transfer information across Internet
Host A IP → router → router... → router → Host B IP
- IP layer in each router determines next hop (router)
- Network interfaces transfer IP packets across networks



TCP/IP Protocol Suite



Internet Names & Addresses

Internet Names

- Each host has a unique name
 - Independent of physical location
 - Facilitate memorization by humans
 - Domain Name
 - Organization under single administrative unit
- Host Name
 - Name given to host computer
- User Name
 - Name assigned to user

leongarcia@comm.utoronto.ca

Internet Addresses

- Each host has globally unique *logical* 32 bit IP address
- Separate address for each physical connection to a network
- Routing decision is done based on destination IP address
- IP address has two parts:
 - *netid* and *hostid*
 - *netid* unique
 - *netid* facilitates routing
- Dotted Decimal Notation:
int1.int2.int3.int4
(intj = jth octet)
128.100.10.13

DNS resolves IP name to IP address

CSE 3213 Summer 2008

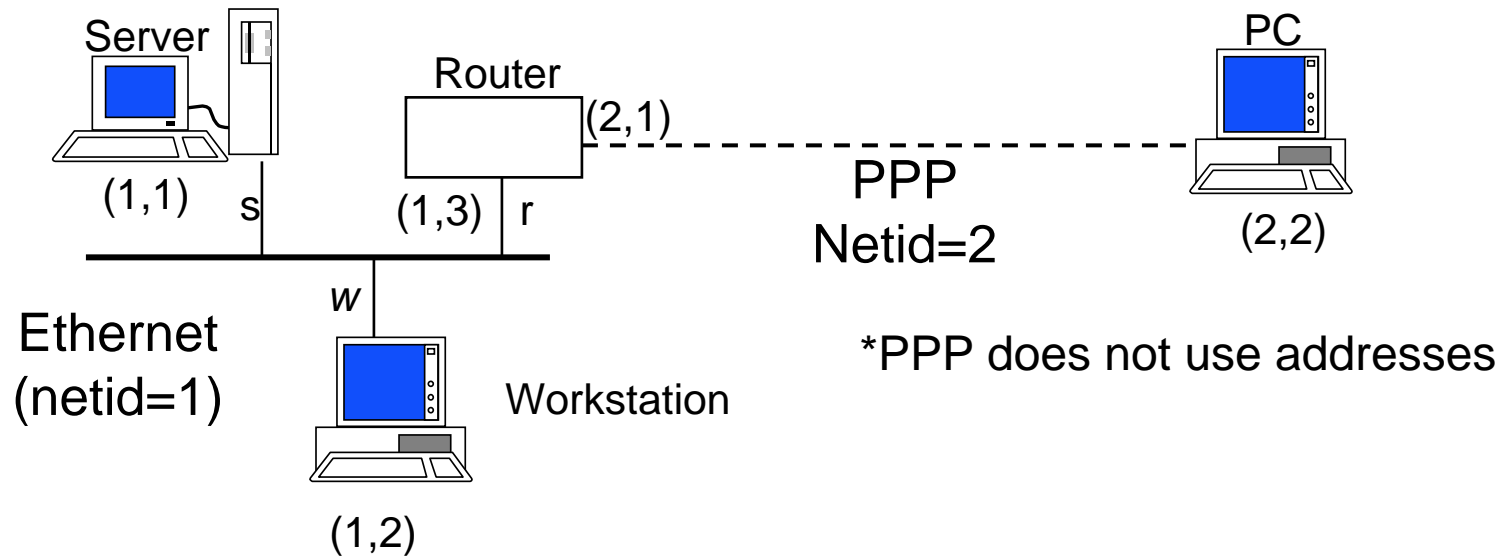
Physical Addresses

- LANs (and other networks) assign physical addresses to the physical attachment to the network
- The network uses its own address to transfer packets or frames to the appropriate destination
- IP address needs to be resolved to physical address at each IP network interface
- Example: Ethernet uses 48-bit addresses
 - Each Ethernet network interface card (NIC) has globally unique Medium Access Control (MAC) or physical address
 - First 24 bits identify NIC manufacturer; second 24 bits are serial number
 - 00:90:27:96:68:07 12 hex numbers



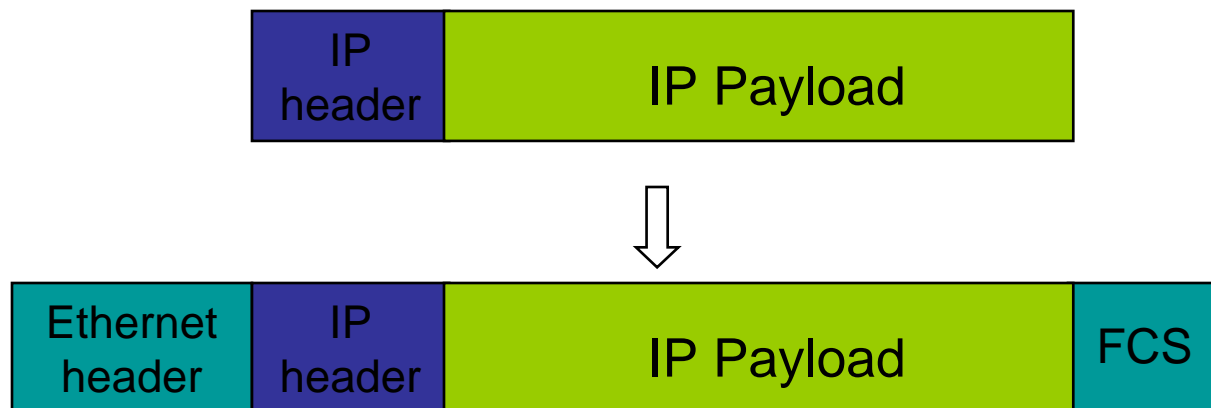
Intel

Example internet



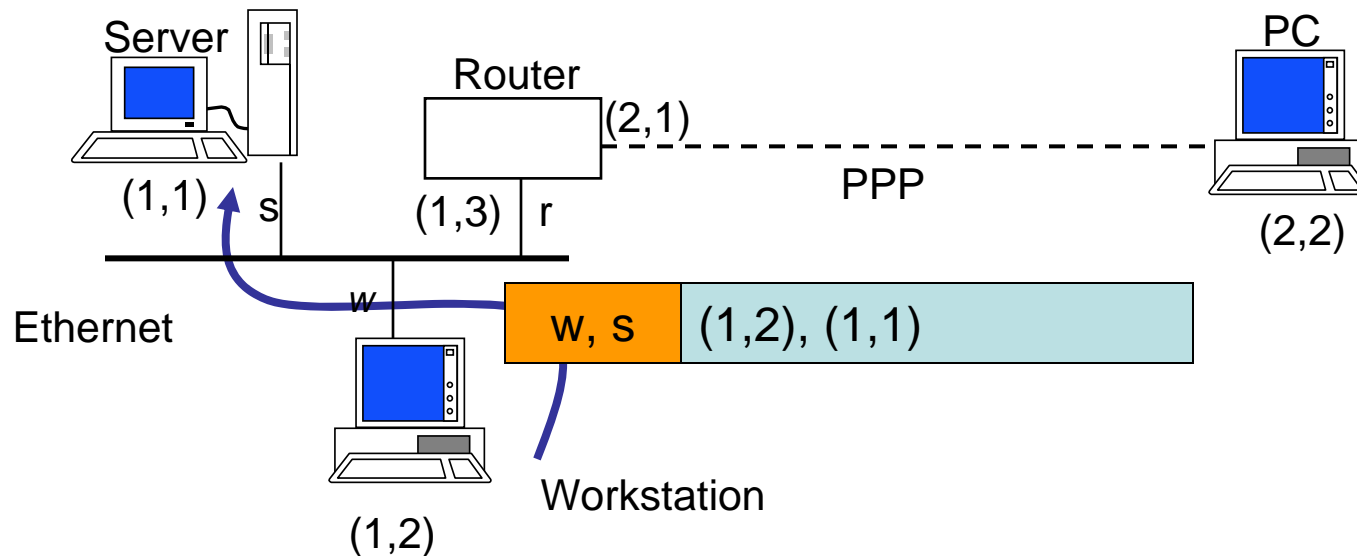
	netid	hostid	Physical address
server	1	1	s
workstation	1	2	w
router	1	3	r
router	2	1	-
PC	2	2	-

Encapsulation



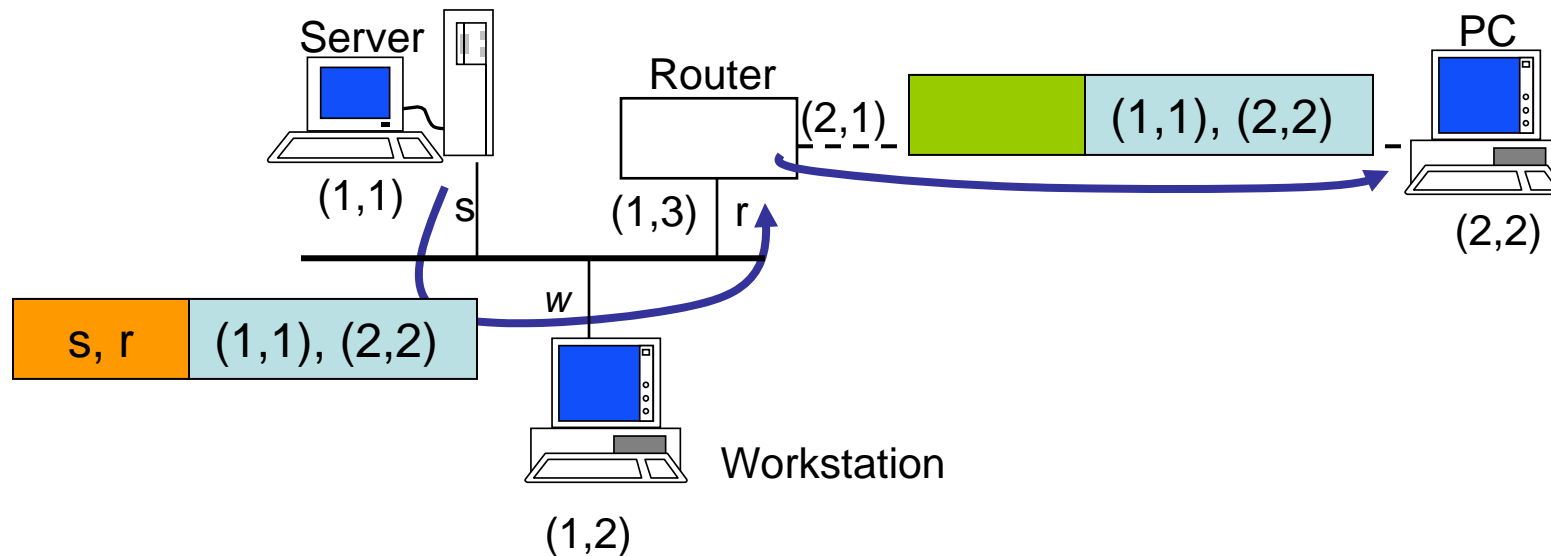
- Ethernet header contains:
 - source and destination physical addresses
 - network protocol type (e.g. IP)

IP packet from workstation to server



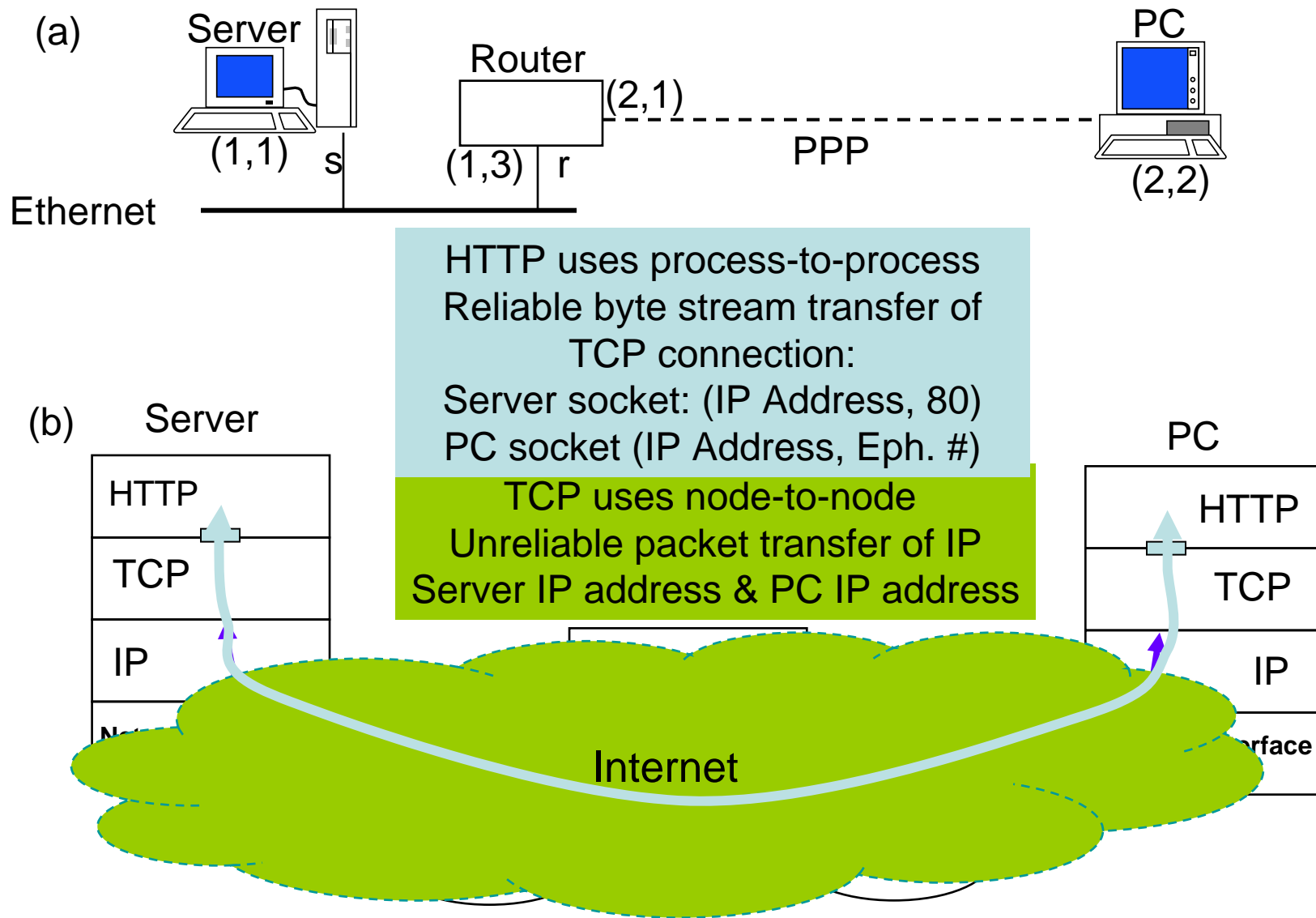
1. IP packet has (1,2) IP address for source and (1,1) IP address for destination
2. IP table at workstation indicates (1,1) connected to same network, so IP packet is encapsulated in Ethernet frame with addresses *w* and *s*
3. Ethernet frame is broadcast by workstation NIC and captured by server NIC
4. NIC examines protocol type field and then delivers packet to its IP layer

IP packet from server to PC



1. IP packet has (1,1) and (2,2) as IP source and destination addresses
2. IP table at server indicates packet should be sent to router, so IP packet is encapsulated in Ethernet frame with addresses s and r
3. Ethernet frame is broadcast by server NIC and captured by router NIC
4. NIC examines protocol type field and then delivers packet to its IP layer
5. IP layer examines IP packet destination address and determines IP packet should be routed to (2,2)
6. Router's table indicates (2,2) is directly connected via PPP link
7. IP packet is encapsulated in PPP frame and delivered to PC
8. PPP at PC examines protocol type field and delivers packet to PC IP layer

How the layers work together

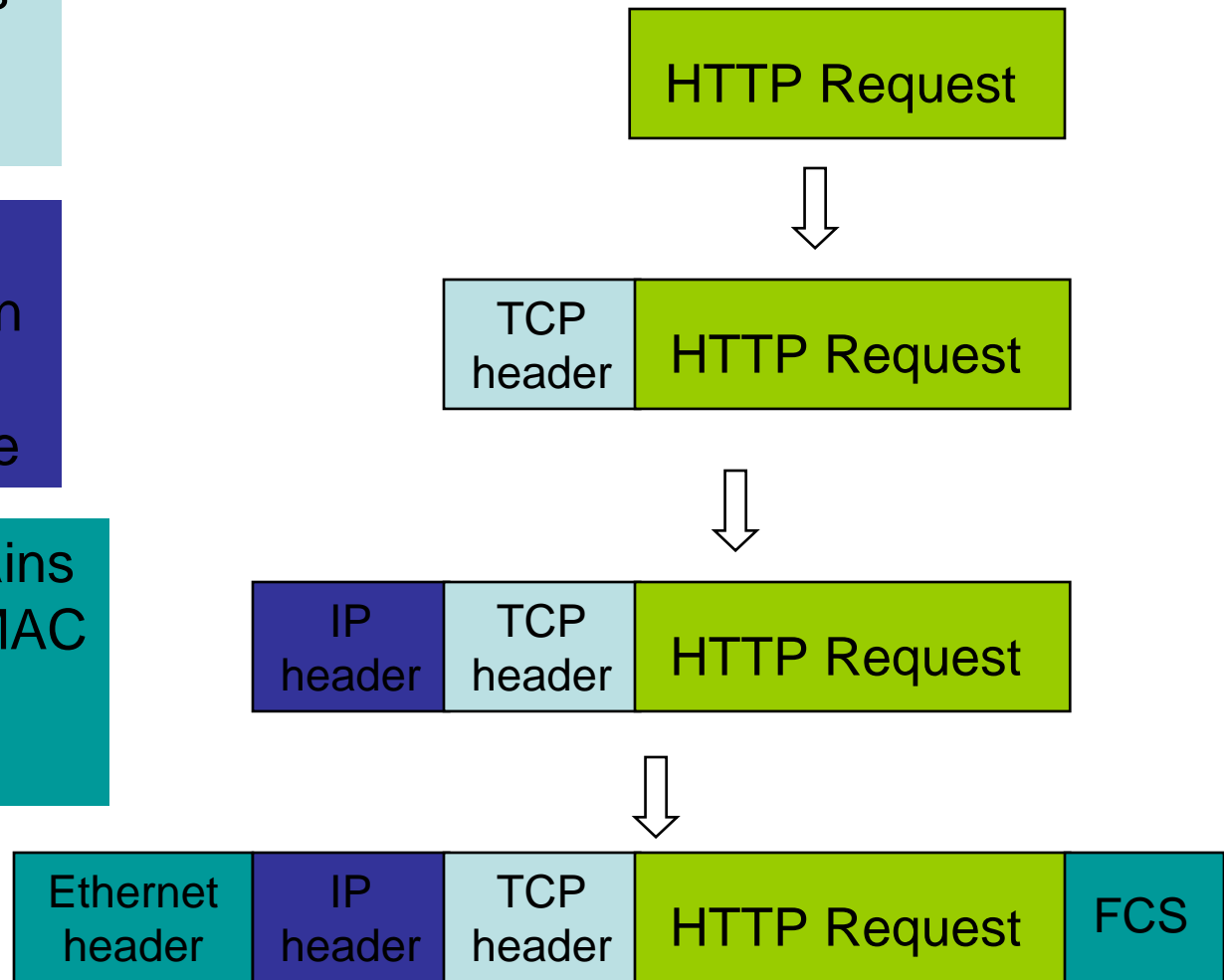


Encapsulation

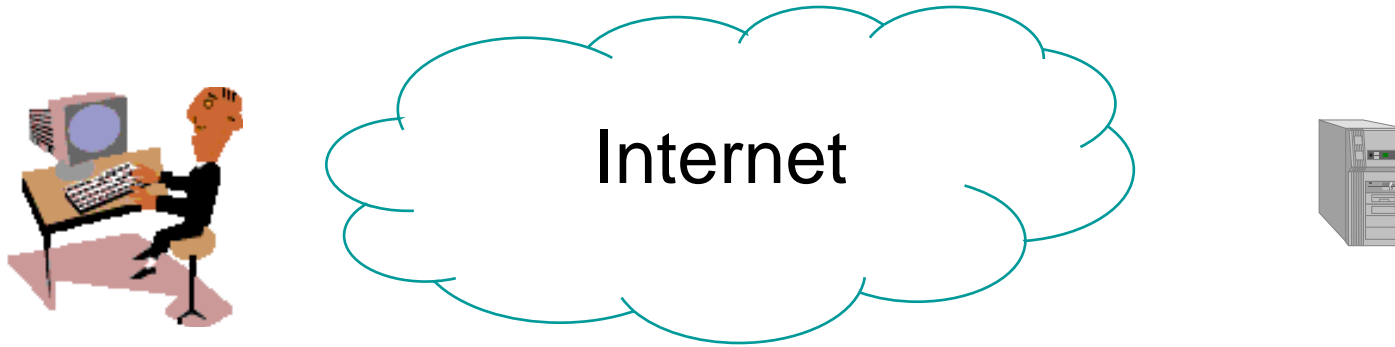
TCP Header contains
source & destination
port numbers

IP Header contains
source and destination
IP addresses;
transport protocol type

Ethernet Header contains
source & destination MAC
addresses;
network protocol type



How the layers work together: Network Analyzer Example



- User clicks on <http://www.nytimes.com/>
- *Ethereal* network analyzer captures all frames observed by its Ethernet NIC
- Sequence of frames and contents of frame can be examined in detail down to individual bytes

ereal window

Top Pane
shows
frame/packet
sequence

Middle Pane
shows
encapsulation for
a given frame

The image shows a Wireshark capture window titled "nytimespac". The top pane displays a list of captured packets. The middle pane shows the details of the selected packet (Frame 1), and the bottom pane shows the raw hex and ASCII data.

No.	Time	Source	Destination	Protocol	Info
1	0.000000	128.100.11.13	128.100.100.128	DNS	Standard query A www.nytimes.com
2	0.129976	128.100.100.128	128.100.11.13	DNS	Standard query response A 64.15.247.200 A 64.15.247.24
3	0.131524	128.100.11.13	64.15.247.200	TCP	1127 > http [SYN] Seq=3638689752 Ack=0 win=16384 Len=0
4	0.168286	64.15.247.200	128.100.11.13	TCP	http > 1127 [SYN] Seq=1396200325 Ack=3638689753 win=17
5	0.168320	128.100.11.13	64.15.247.200	TCP	1127 > http [ACK] Seq=3638689753 Ack=1396200326 win=17
6	0.168688	128.100.11.13	64.15.247.200	HTTP	GET / HTTP/1.1
7	0.205439	64.15.247.200	128.100.11.13	TCP	http > 1127 [ACK] Seq=1396200326 Ack=3638690402 win=32
8	0.236676	64.15.247.200	128.100.11.13	HTTP	HTTP/1.1 200 OK

Frame 1 (75 bytes on wire, 75 bytes captured)
Ethernet II, Src: 00:90:27:96:b8:07, Dst: 00:e0:52:ea:b5:00
Internet Protocol, Src Addr: 128.100.11.13 (128.100.11.13), Dst Addr: 128.100.100.128 (128.100.100.128)
User Datagram Protocol, Src Port: 1126 (1126), Dst Port: domain (53)
Domain Name System (query)

```
0000  00 e0 52 ea b5 00 00 90 27 96 b8 07 08 00 45 00  ..R.... '.....E.
0010  00 3d 54 41 00 00 80 11 76 19 80 64 0b 0d 80 64  .=TA.... v..d...d
0020  64 80 04 66 00 35 00 29 49 83 00 a5 01 00 00 01  d..f.5.) I.....
0030  00 00 00 00 00 00 03 77 77 77 07 6e 79 74 69 6d  ....w ww.nytim
0040  65 73 03 63 6f 6d 00 00 01 00 01                es.com.. ...
```

Bottom Pane shows hex & text

Top pane: frame sequence

The image shows a Wireshark packet capture window titled "nytimespac". The top pane displays a list of captured packets. Three specific frames are highlighted with red callouts:

- DNS Query:** Points to packet 1, a DNS standard query A for www.nytimes.com.
- TCP Connection Setup:** Points to packet 4, a TCP SYN packet from 1127 to 1127.
- HTTP Request & Response:** Points to packet 6, an HTTP GET request for / HTTP/1.1.

The packet list table is as follows:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	128.100.11.13	128.100.100.128	DNS	75	Standard query A www.nytimes.com
2	0.129976	128.100.100.128	128.100.11.13	DNS	75	Standard query response
3	0.131324	128.100.11.13	64.15.247.200	TCP	60	1127 > 1127 [SYN] Seq=3638690402
4	0.168286	64.15.247.200	128.100.11.13	TCP	60	1127 > 1127 [SYN, ACK] Seq=1396200326
5	0.168320	128.100.11.13	64.15.247.200	TCP	60	1127 > 1127 [ACK] Seq=3638690402
6	0.168688	128.100.11.13	64.15.247.200	HTTP	100	GET / HTTP/1.1
7	0.205439	64.15.247.200	128.100.11.13	TCP	60	1127 > 1127 [ACK] Seq=1396200326
8	0.236676	64.15.247.200	128.100.11.13	HTTP	100	HTTP/1.1 200 OK

The bottom pane shows the details of the selected frame (Frame 1):

- Frame 1 (75 bytes on wire, 75 bytes captured)
- Ethernet II, Src: 00:90:27:96:b8:07, Dst: 00:e0:52:ea:b5:00
- Internet Protocol, Src Addr: 128.100.11.13 (128.100.11.13), Dst Addr: 128.100.100.128 (128.100.100.128)
- User Datagram Protocol, Src Port: 1126 (1126), Dst Port: domain (53)
- Domain Name System (query)

The packet bytes pane shows the raw data in hexadecimal and ASCII:

```
0000  00 e0 52 ea b5 00 00 90 27 96 b8 07 08 00 45 00  ..R.....E.
0010  00 3d 54 41 00 00 80 11 76 19 80 64 0b 0d 80 64  .=TA...v..d...
0020  64 80 04 66 00 35 00 29 49 83 00 a5 01 00 00 01  d..f.5.)I.....
0030  00 00 00 00 00 00 03 77 77 77 07 6e 79 74 69 6d  .....w ww.nytim
0040  65 73 03 63 6f 6d 00 00 01 00 01                es.com.. ...
```

Middle pane: Encapsulation

The screenshot shows the Wireshark interface with the title bar "nytimespackets - Ethereal". The menu bar includes File, Edit, Capture, Display, Tools, and Help. The packet list pane shows a single packet (No. 6) at time 0.168688, from source 128.100.11.13 to destination 64.15.247.200, with protocol HTTP and method GET. The packet details pane shows the following structure:

- Ethernet II** (702 bytes captured on interface 0):
 - Destination: 00:e0:52:ea:b5:00 (Foundry_ea:b5:00)
 - Source: 00:90:27:96:b8:07 (Intel_96:b8:07)
 - Type: IP (0x0800)
- Internet Protocol Version 4** (60 bytes):
 - Version: 4
 - Header length: 20 bytes
 - Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN 0)
 - Total length: 128
 - Identification: 0
 - Flags: 0x00
 - Fragment offset: 0
 - Time to live: 128
 - Protocol: TCP (0x06)
 - Header checksum: 0xe0b8 (correct)
 - Source: 128.100.11.13 (128.100.11.13)
 - Destination: 64.15.247.200 (64.15.247.200)
- Transmission Control Protocol** (108 bytes):
 - Src Port: 1127 (1127)
 - Dst Port: http (80)
 - Seq: 3638689753
 - Ack: 139620032
- Hypertext Transfer Protocol**

The packet bytes pane shows the raw data in hexadecimal and ASCII. The first 14 bytes (0000 to 000f) represent the Ethernet II header, which is highlighted by a red box labeled "Ethernet Frame". The next 20 bytes (0010 to 001f) represent the IP header, which is highlighted by a red box labeled "Protocol Type". The final 74 bytes (0020 to 004f) represent the TCP and HTTP data, which is highlighted by a red box labeled "Ethernet Destination and Source Addresses".

Filter: [] [Reset] [Apply] File: nytimespackets

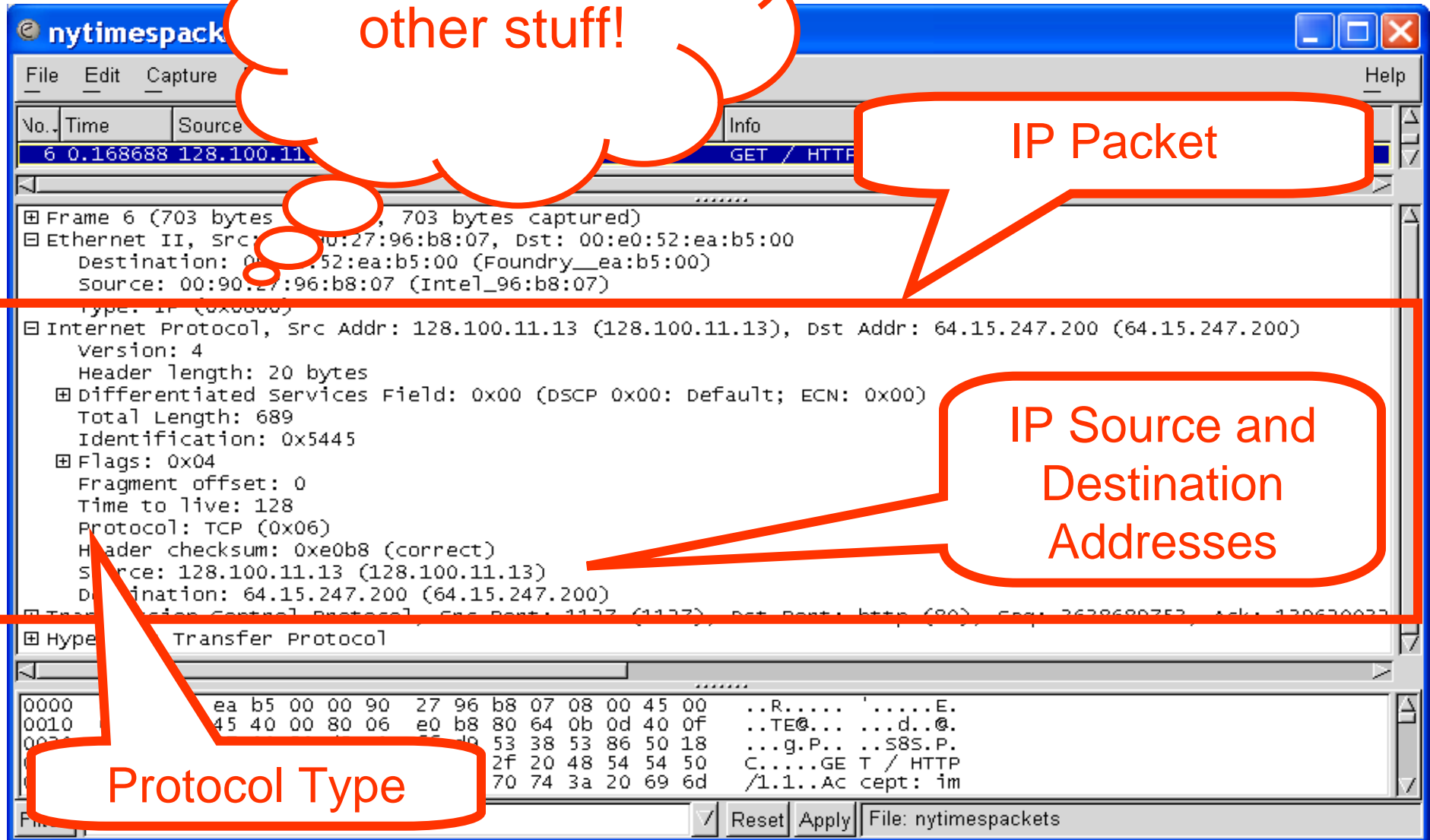
Encapsulation

And a lot of
other stuff!

IP Packet

IP Source and
Destination
Addresses

Protocol Type



Middle pane: Encapsulation

The screenshot shows the 'nytimespackets - Ethereal' window. The packet list at the top shows packet 6 at time 0.168688, from source 128.100.11.13 to destination 64.15.247.200, protocol HTTP, with info GET / HTTP. The packet details pane shows the following structure:

- Frame 6 (703 bytes on wire, 703 bytes captured)
- Ethernet II, Src: 00:90:27:96:b8:07, Dst: 00:e0:52:ea:b5:00
- Transmission Control Protocol, Src Port: 1127 (1127), Dst Port: http (80), Seq: 3638689753, Ack: 1396200326
 - Source port: 1127 (1127)
 - Destination port: http (80)
 - Sequence number: 3638689753
 - Next sequence number: 3638690402
 - Acknowledgement number: 1396200326
 - Header length: 20 bytes
 - Flags: 0x0018 (PSH, ACK)
 - Window size: 17316
- Hypertext Transfer Protocol
 - GET / HTTP/1.1\r\n
 - Accept: image/gif, image/x-xbitmap, image/ined, image/pjpeg, application/vnd.ms-powerpoint, application,
 - Accept-Language: en-us\r\n
 - Accept-Encoding: gzip, deflate\r\n
 - User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; windows NT 5.0)\r\n
 - Host: www.nytimes.com\r\n
 - Connection: keep-alive\r\n
 - Cookie: RMID=80e7478f5a393db9fc19f2c4; NYT-S=1002xv091grjagxb2AZ9Oxq4lqDEe/...

At the bottom, the packet bytes are displayed in hexadecimal and ASCII format.

Red callouts highlight the following sections:

- TCP Segment**: Points to the Transmission Control Protocol section.
- Source and Destination Port Numbers**: Points to the Source port and Destination port fields.
- GET**: Points to the GET / HTTP/1.1\r\n line in the Hypertext Transfer Protocol section.
- HTTP Request**: Points to the entire Hypertext Transfer Protocol section.

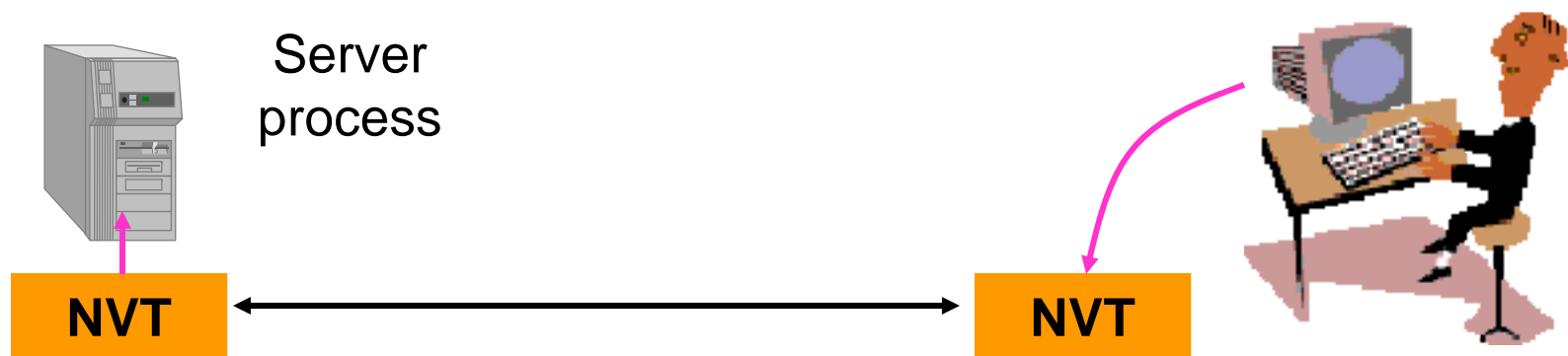
Summary

- Encapsulation is key to layering
- IP provides for transfer of packets across diverse networks
- TCP and UDP provide universal communications services across the Internet
- Distributed applications that use TCP and UDP can operate over the entire Internet
- Internet names, IP addresses, port numbers, sockets, connections, physical addresses

Application Layer Protocols & IP Utilities

Telnet (RFC 854)

- Provides general bi-directional byte-oriented TCP-based communications facility (Network Virtual Terminal)
- Initiating machine treated as local to the remote host
- Used to connect to port # of other servers and to interact with them using command line



Network Virtual Terminal

- *Network Virtual Terminal*
- Lowest common denominator terminal
- Each machine maps characteristics to NVT
- Negotiate options for changes to the NVT
- Data input sent to server & echoed back
- Server control functions : interrupt, abort output, are-you-there, erase character, erase line
- Default requires login & password

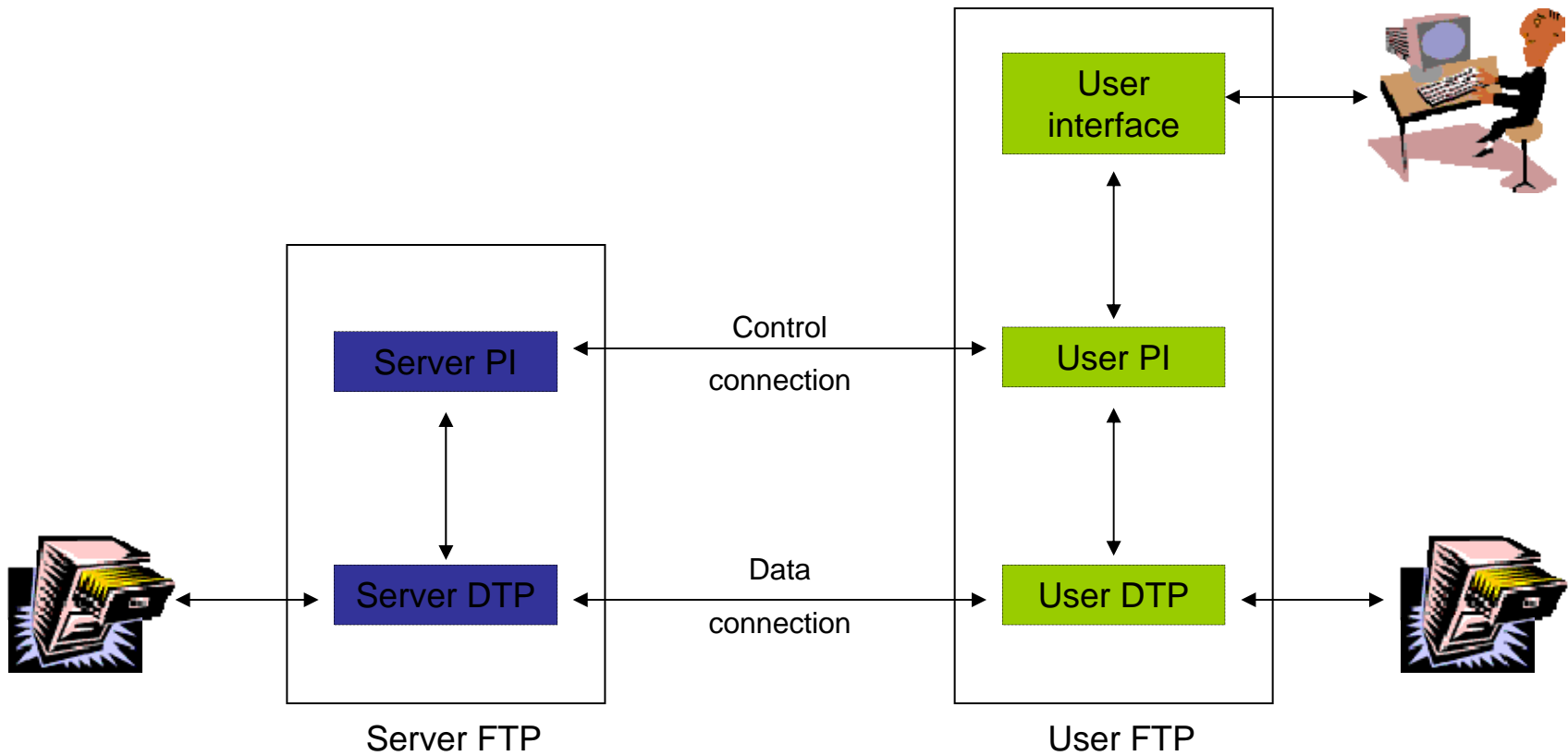
telnet

- A program that uses the Telnet protocol
- Establishes TCP socket
- Sends typed characters to server
- Prints whatever characters arrive
- Try it to retrieve a web page (HTTP) or to send an email (SMTP)

File Transfer Protocol (RFC 959)

- Provides for transfer of file from one machine to another machine
- Designed to hide variations in file storage
- FTP parameter commands specify file info
 - File Type: *ASCII*, *EBCDIC*, *image*, *local*.
 - Data Structure: *file*, *record*, or *page*
 - Transmission Mode: *stream*, *block*, *compressed*
- Other FTP commands
 - Access Control: *USER*, *PASS*, *CWD*, *QUIT*, ...
 - Service: *RETR*, *STOR*, *PWD*, *LIST*, ...

FTP File Transfer



PI = Protocol interface
DTP = Data transfer process

Two TCP Connections

Control connection

- Set up using Telnet protocol on well-known port 21
- FTP commands & replies between protocol interpreters
- PIs control the data transfer process
- User requests close of control connection; server performs the close

Data connection

- To perform file transfer, obtain lists of files, directories
- Each transfer requires new data connection
- Passive open by user PI with ephemeral port #
- Port # sent over control connection
- Active open by server using port 20

FTP Replies

Reply	Meaning
1yz	Positive preliminary reply (action has begun, but wait for another reply before sending a new command).
2yz	Positive completion reply (action completed successfully; new command may be sent).
3yz	Positive intermediary reply (command accepted, but action cannot be performed without additional information; user should send a command with the necessary information).
4yz	Transient negative completion reply (action currently cannot be performed; resend command later).
5zy	Permanent negative completion reply (action cannot be performed; do not resend it).
x0z	Syntax errors.
x1z	Information (replies to requests for status or help).
x2z	Connections (replies referring to the control and data connections).
x3z	Authentication and accounting (replies for the login process and accounting procedures).
x4z	Unspecified.
x5z	File system status.

FTP Client (192.168.1.132: 1421) establishes Control Connection to FTP Server (128.100.132.23: 21)

The image shows a Wireshark capture of an FTP control connection. The packet list at the top shows several packets, with packets 13, 14, and 15 highlighted in orange. Packet 13 is a TCP SYN packet from 192.168.1.132 to 128.100.132.23. Packet 14 is a TCP ACK packet from 128.100.132.23 to 192.168.1.132. Packet 15 is a TCP ACK packet from 192.168.1.132 to 128.100.132.23. The packet details pane shows the selected packet (13) expanded, showing the Ethernet II, Internet Protocol, and Transmission Control Protocol layers. The IP layer shows the source address 192.168.1.132 and destination address 128.100.132.23. The TCP layer shows the source port 1421 and destination port ftp (21). The packet bytes pane shows the raw data of the packet.

No.	Time	Source	Destination	Protocol	Info
13	1.196932	192.168.1.132	128.100.132.23	TCP	1421 > ftp [SYN] Seq=1319718353 Ack=0 win=64240 Len=0 MSS=
14	1.381146	128.100.132.23	192.168.1.132	TCP	ftp > 1421 [SYN, ACK] Seq=718506651 Ack=1319718354 win=14
15	1.381212	192.168.1.132	128.100.132.23	TCP	1421 > ftp [ACK] Seq=1319718354 Ack=718506652 win=64240 L
16	1.892608	128.100.132.23	192.168.1.132	FTP	response: 220 pweb.ns.utoronto.ca FTP server ready.
17	2.065063	192.168.1.132	128.100.132.23	TCP	1421 > ftp [ACK] Seq=1319718354 Ack=718506695 win=64197 L
22	5.661757	192.168.1.132	128.100.132.23	FTP	Request: USER sirikang
23	5.868685	128.100.132.23	192.168.1.132	TCP	ftp > 1421 [ACK] Seq=718506695 Ack=1319718369 win=24820 L
24	5.870992	128.100.132.23	192.168.1.132	FTP	Response: 331 Password required for sirikang.

Frame 13 (62 bytes on wire, 62 bytes captured)
Ethernet II, Src: 00:00:00:00:00:00, Dst: 00:00:00:00:00:00
Internet Protocol, Src Addr: 192.168.1.132 (192.168.1.132), Dst Addr: 128.100.132.23 (128.100.132.23)
Version: 4
Header length: 20 bytes
Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
Total Length: 48
Identification: 0x8c02
Flags: 0x04
Fragment offset: 0
Time to live: 128
Protocol: TCP (0x06)
Header checksum: 0xa81d (correct)
Source: 192.168.1.132 (192.168.1.132)
Destination: 128.100.132.23 (128.100.132.23)
Transmission Control Protocol, Src Port: 1421 (1421), Dst Port: ftp (21), Seq: 1319718353, Ack: 0, Len: 0
Source port: 1421 (1421)
Destination port: ftp (21)
Sequence number: 1319718353
Header length: 28 bytes
Flags: 0x0002 (SYN)
window size: 64240
Checksum: 0x1f6a (correct)
options: (8 bytes)

0000 00 06 25 65 5b 08 00 00 39 ff 62 d6 08 00 45 00 ..%e[... 9.b...E.
0010 00 30 8c 02 40 00 80 06 a8 1d c0 a8 01 84 80 64 .0..@... ..d
0020 84 17 05 8d 00 15 4e a9 4d d1 00 00 00 00 70 02 ...N. M....p.
0030 fa f0 1f 6a 00 00 02 04 05 b4 01 01 04 02 ...j....

Filter: tcp.port != 1900 / Reset Apply Internet Protocol (ip), 20 bytes

User types /s to list files in directory (frame 31 on control)
FTP Server (128.100.132.23: 20) establishes Data Connection to FTP
Client (192.168.1.132: 1422)

The image shows a Wireshark packet capture of an FTP session. The main packet list shows frames 29 through 45. Frame 31 is highlighted with an orange box and a red arrow, showing an FTP 'NLST' request. Frame 32 is also highlighted with an orange box, showing a TCP SYN packet from the server to the client, establishing a data connection. The packet details pane for frame 32 is expanded, showing the Ethernet II, Internet Protocol, and Transmission Control Protocol layers. The TCP layer shows a SYN flag and sequence number 724151515. The packet bytes pane at the bottom shows the raw data in hexadecimal and ASCII.

No.	Time	Source	Destination	Protocol	Info
29	11.08	192.100.132.23	192.168.1.132	FTP	Request: PORT 192.100.132.23, 1422
30	11.81	128.100.132.23	192.168.1.132	FTP	Response: 200 PORT command successful.
31	11.81	192.168.1.132	128.100.132.23	FTP	Request: NLST
32	11.93	128.100.132.23	192.168.1.132	TCP	ftp-data > 1422 [SYN] Seq=724151515 Ack=0 win=24820 Len=0 MSS=1460
33	11.93	192.168.1.132	128.100.132.23	TCP	1422 > ftp-data [SYN, ACK] Seq=1322456863 Ack=724151516 win=64240 L
34	12.02	128.100.132.23	192.168.1.132	TCP	ftp > 1421 [ACK] Seq=718506820 Ack=1319718416 win=24820 Len=0
35	12.05	128.100.132.23	192.168.1.132	TCP	ftp-data > 1422 [ACK] Seq=724151516 Ack=1322456864 win=1460 Len=0
36	12.06	128.100.132.23	192.168.1.132	FTP	Response: 150 Opening ASCII mode data connection for file list.
37	12.06	128.100.132.23	192.168.1.132	FTP-DATA	FTP Data: 12 bytes
38	12.06	128.100.132.23	192.168.1.132	TCP	ftp-data > 1422 [FIN, ACK] Seq=724151528 Ack=1322456864 win=24820 L
39	12.06	192.168.1.132	128.100.132.23	TCP	1422 > ftp-data [ACK] Seq=1322456864 Ack=724151529 win=64228 Len=0
40	12.06	192.168.1.132	128.100.132.23	TCP	1422 > ftp-data [FIN, ACK] Seq=1322456864 Ack=724151529 win=64228 L
41	12.17	192.168.1.132	128.100.132.23	TCP	1421 > ftp [ACK] Seq=1319718416 Ack=718506875 win=64017 Len=0
42	12.19	128.100.132.23	192.168.1.132	TCP	ftp-data > 1422 [ACK] Seq=724151529 Ack=1322456865 win=24820 Len=0
43	12.29	128.100.132.23	192.168.1.132	FTP	Response: 226 Transfer complete.
44	12.48	192.168.1.132	128.100.132.23	TCP	1421 > ftp [ACK] Seq=1319718416 Ack=718506899 win=63993 Len=0
45	24.75	192.168.1.132	128.100.132.23	FTP	Request: PORT 192.168.1.132, 5143

Frame 32 (62 bytes on wire (62 bytes captured))

Ethernet II, Src: 00:06:25:65:5b:08, Dst: 00:00:20:ff:67:d6

Internet Protocol, Src Addr: 128.100.132.23 (128.100.132.23), Dst Addr: 192.168.1.132 (192.168.1.132)

Transmission Control Protocol, Src Port: ftp-data (20), Dst Port: 1422 (1422), Seq: 724151515, Ack: 0, Len: 0

Source port: ftp-data (20)

Destination port: 1422 (1422)

Sequence number: 724151515

Header length: 28 bytes

Flags: 0x0002 (SYN)

- 0... .. = Congestion window Reduced (CWR): Not set
- .0.. = ECN-Echo: Not set
- ..0. = Urgent: Not set
- ...0 = Acknowledgment: Not set
- 0... = Push: Not set
-0.. = Reset: Not set
-1. = Syn: Set

0000 00 00 39 ff 62 d6 00 06 25 65 5b 08 08 00 45 08 ..9.b... %e[...E.

0010 00 30 64 a5 40 00 36 06 19 73 80 64 84 17 c0 a8 .0d.@.6. .s.d....

0020 01 84 00 14 05 8e 2b 29 ac db 00 00 00 00 70 02+)p.

0030 60 f4 7d dc 00 00 01 01 04 02 02 04 05 b4 .}......

Filter: tcp.port != 1900

User types *get index.html* to request file transfer in control connection (frame 47 request): File transfer on new data connection (port 1423, fr. 48, 49, 51)

FTP - Ethereal

No.	Time	Source	Destination	Protocol	Info
46	24.91	128.100.132.23	192.168.1.132	FTP	Response: 200 PORT command successful.
47	24.91	192.168.1.132	128.100.132.23	FTP	Request: RETR index.html
48	25.04	128.100.132.23	192.168.1.132	TCP	ftp-data > 1423 [SYN] Seq=729455232 Ack=0 win=24820 Len=0 MSS=1460
49	25.04	192.168.1.132	128.100.132.23	TCP	1423 > ftp-data [SYN, ACK] Seq=1325791977 Ack=729455233 win=64240 L
50	25.13	128.100.132.23	192.168.1.132	TCP	ftp > 1421 [ACK] Seq=718506929 Ack=1319718459 win=24820 Len=0
51	25.16	128.100.132.23	192.168.1.132	TCP	ftp-data > 1423 [ACK] Seq=729455233 Ack=1325791978 win=1460 Len=0
52	25.16	128.100.132.23	192.168.1.132	FTP	Response: 150 Opening ASCII mode data connection for index.html (11
53	25.16	128.100.132.23	192.168.1.132	FTP-DATA	FTP Data: 125 bytes
54	25.16	128.100.132.23	192.168.1.132	TCP	ftp-data > 1423 [FIN, ACK] Seq=729455358 Ack=1325791978 win=24820 L
55	25.16	192.168.1.132	128.100.132.23	TCP	1423 > ftp-data [ACK] Seq=1325791978 Ack=729455359 win=64115 Len=0
56	25.16	192.168.1.132	128.100.132.23	TCP	1423 > ftp-data [FIN, ACK] Seq=1325791978 Ack=729455359 win=64115 L
57	25.29	128.100.132.23	192.168.1.132	TCP	ftp-data > 1423 [ACK] Seq=729455359 Ack=1325791979 win=24820 Len=0
58	25.29	192.168.1.132	128.100.132.23	TCP	1421 > ftp [ACK] Seq=1319718459 Ack=718506997 win=63895 Len=0

Frame 47 (71 bytes on wire, 71 bytes captured)

- Ethernet II, Src: 00:00:39:ff:62:d6, Dst: 00:06:25:65:5b:08
- Internet Protocol, Src Addr: 192.168.1.132 (192.168.1.132), Dst Addr: 128.100.132.23 (128.100.132.23)
- Transmission Control Protocol, Src Port: 1421 (1421), Dst Port: ftp (21), Seq: 1319718442, Ack: 718506929, Len: 17
 - Source port: 1421 (1421)
 - Destination port: ftp (21)
 - Sequence number: 1319718442
 - Next sequence number: 1319718459
 - Acknowledgement number: 718506929
 - Header length: 20 bytes
 - Flags: 0x0018 (PSH, ACK)
 - 0... = Congestion Window Reduced (CWR): Not set
 - .0.. = ECN-Echo: Not set
 - ..0. = Urgent: Not set
 - ...1 = Acknowledgment: set
 - 1... = Push: Set
 -0.. = Reset: Not set
 -0. = Syn: Not set
 -0 = Fin: Not set

Packet Bytes:

```

0000  00 06 25 65 5b 08 00 00 39 ff 62 d6 08 00 45 00  ..%e[... 9.b...E.
0010  00 39 8c 19 40 00 80 06 a7 fd c0 a8 01 84 80 64  .9..@... .....d
0020  84 17 05 8d 00 15 4e a9 4e 2a 2a d3 8b b1 50 18  ....N. N**...P.
0030  f9 db e2 7c 00 00 52 45 54 52 20 69 6e 64 65 78  ...|..RE TR index
0040  2e 68 74 6d 6c 0d 0a                .html..
  
```

Filter: / Reset Apply File: FTP

Hypertext Transfer Protocol

- RFC 1945 (HTTP 1.0), RFC 2616 (HTTP 1.1)
- HTTP provides communications between web browsers & web servers
- Web: framework for accessing documents & resources through the Internet
- Hypertext documents: text, graphics, images, hyperlinks
- Documents prepared using Hypertext Markup Language (HTML)

HTTP Protocol

- HTTP servers use well-known port 80
- Client request / Server reply
- Stateless: server does not keep any information about client
- HTTP 1.0 new TCP connection per request/reply (non-persistent)
- HTTP 1.1 persistent operation is default

HTTP Typical Exchange

nytimespackets - Ethereal

File Edit Capture Display Tools Help

No.	Time	Source	Destination	Protocol	Info
1	0.000000	128.100.11.13	128.100.100.128	DNS	Standard query A www.nytimes.com
2	0.129976	128.100.100.128	128.100.11.13	DNS	Standard query response A 64.15.247.200 A 64
3	0.131524	128.100.11.13	64.15.247.200	TCP	1127 > http [SYN] Seq=3638689752 Ack=0 win=1
4	0.168286	64.15.247.200	128.100.11.13	TCP	http > 1127 [SYN, ACK] Seq=1396200325 Ack=36
5	0.168320	128.100.11.13	64.15.247.200	TCP	1127 > http [ACK] Seq=3638689753 Ack=1396200
6	0.168688	128.100.11.13	64.15.247.200	HTTP	GET / HTTP/1.1
7	0.205439	64.15.247.200	128.100.11.13	TCP	http > 1127 [ACK] Seq=1396200326 Ack=3638690
8	0.236676	64.15.247.200	128.100.11.13	HTTP	HTTP/1.1 200 OK

Frame 6 (703 bytes on wire, 703 bytes captured)

Ethernet II, Src: 00:90:27:96:b8:07, Dst: 00:e0:52:ea:b5:00

Internet Protocol, Src Addr: 128.100.11.13 (128.100.11.13), Dst Addr: 64.15.247.200 (64.15.247.200)

Transmission Control Protocol, Src Port: 1127 (1127), Dst Port: http (80), Seq: 3638689753, Ack: 139620032

Hypertext Transfer Protocol

GET / HTTP/1.1\r\n

Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-powerpoint, application,

Accept-Language: en-us\r\n

Accept-Encoding: gzip, deflate\r\n

User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; windows NT 5.0)\r\n

Host: www.nytimes.com\r\n

Connection: keep-alive\r\n

Cookie: RMID=80e7478f5a393db9fc19f2c4; NYT-S=1002xv091grjagxb2AZ9Oxq4lqdEe/irdKSU3XunLr287eqe2QOMe5m08Re\r\n

0000 00 e0 52 ea b5 00 00 90 27 96 b8 07 08 00 45 00 ..R.....E.

0010 02 b1 54 45 40 00 80 06 e0 b8 80 64 0b 0d 40 0f ..TE@...d..@.

0020 f7 c8 04 67 00 50 d8 e1 ff d9 53 38 53 86 50 18 ...g.P...S8S.P.

0030 43 a4 87 81 00 00 47 45 54 20 2f 20 48 54 54 50 C....GE T / HTTP

0040 2f 31 2e 31 0d 0a 41 63 63 65 70 74 3a 20 69 6d /1.1..Ac cept: im

0050 61 67 65 2f 67 69 66 2c 20 69 6d 61 67 65 2f 78 age/gif, image/x

0060 2d 78 63 60 74 6d 61 70 7c 70 60 6d 61 67 65 7f x-bitmap image/

Filter: / Reset Apply File: nytimespackets

HTTP Message Formats

- HTTP messages written in ASCII text
- Request Message Format
 - *Request Line (Each line ends with carriage return)*
 - *Method URL HTTP-Version \r\n*
 - Method specifies action to apply to object
 - URL specifies object
 - *Header Lines (Ea. line ends with carriage return)*
 - *Attribute Name: Attribute Value*
 - E.g. type of client, content, identity of requester, ...
 - Last header line has extra carriage return)
 - *Entity Body (Content)*
 - Additional information to server

HTTP Request Methods

Request method	Meaning
GET	Retrieve information (object) identified by the URL.
HEAD	Retrieve meta-information about the object, but do not transfer the object; Can be used to find out if a document has changed.
POST	Send information to a URL (using the entity body) and retrieve result; used when a user fills out a form in a browser.
PUT	Store information in location named by URL
DELETE	Remove object identified by URL
TRACE	Trace HTTP forwarding through proxies, tunnels, etc.
OPTIONS	Used to determine the capabilities of the server, or characteristics of a named resource.

Universal Resource Locator

- Absolute URL
 - scheme://hostname[:port]/path
 - <http://www.nytimes.com/>
- Relative URL
 - /path
 - /

HTTP Request Message

The screenshot displays the Wireshark interface with the title bar "nytimespackets - Ethereal". The menu bar includes File, Edit, Capture, Display, Tools, and Help. The packet list pane shows eight captured packets. Packet 6, at time 0.168688, is selected and highlighted in blue. It is an HTTP GET request from 128.100.11.13 to 64.15.247.200. The packet details pane for packet 6 is expanded, showing the following information:

- Frame 6 (703 bytes on wire, 703 bytes captured)
- Ethernet II, Src: 00:90:27:96:b8:07, Dst: 00:e0:52:ea:b5:00
- Internet Protocol, Src Addr: 128.100.11.13 (128.100.11.13), Dst Addr: 64.15.247.200 (64.15.247.200)
- Transmission Control Protocol, Src Port: 1127 (1127), Dst Port: http (80), Seq: 3638689753, Ack: 139620032
- HyperText Transfer Protocol
 - GET / HTTP/1.1\r\n
 - Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-powerpoint, application,
 - Accept-Language: en-us\r\n
 - Accept-Encoding: gzip, deflate\r\n
 - User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; windows NT 5.0)\r\n
 - Host: www.nytimes.com\r\n
 - Connection: Keep-Alive\r\n
 - Cookie: RMID=80e7478f5a393db9fc19f2c4; NYT-S=1002xv091grjagxb2AZ9Oxq4lqdEe/irdKSU3XunLr287eqe2QOMe5m08Re\r\n

The packet bytes pane at the bottom shows the raw data in hexadecimal and ASCII. The ASCII column shows the beginning of the HTTP request: "GET / HTTP/1.1\r\nAccept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-powerpoint, application,".

HTTP Response Message

- Response Message Format
 - *Status Line*
 - *HTTP-Version Status-Code Message*
 - Status Code: 3-digit code indicating result
 - E.g. HTTP/1.0 200 OK
 - *Headers Section*
 - Information about object transferred to client
 - E.g. server type, content length, content type, ...
 - *Content*
 - Object (document)

HTTP Response Message

nytimespackets - Ethereal

File Edit Capture Display Tools Help

No.	Time	Source	Destination	Protocol	Info
1	0.000000	128.100.11.13	128.100.100.128	DNS	standard query A www.nytimes.com
2	0.129976	128.100.100.128	128.100.11.13	DNS	standard query response A 64.15.247.200 A 64
3	0.131524	128.100.11.13	64.15.247.200	TCP	1127 > http [SYN] Seq=3638689752 Ack=0 win=1
4	0.168286	64.15.247.200	128.100.11.13	TCP	http > 1127 [SYN, ACK] Seq=1396200325 Ack=36
5	0.168320	128.100.11.13	64.15.247.200	TCP	1127 > http [ACK] Seq=3638689753 Ack=1396200
6	0.168688	128.100.11.13	64.15.247.200	HTTP	GET / HTTP/1.1
7	0.205439	64.15.247.200	128.100.11.13	TCP	http > 1127 [ACK] Seq=1396200326 Ack=3638690
8	0.236676	64.15.247.200	128.100.11.13	HTTP	HTTP/1.1 200 OK

Frame 8 (284 bytes on wire, 284 bytes captured)

Ethernet II, Src: 00:e0:52:ea:b5:00, Dst: 00:90:27:96:b8:07

Internet Protocol, Src Addr: 64.15.247.200 (64.15.247.200), Dst Addr: 128.100.11.13 (128.100.11.13)

Transmission Control Protocol, Src Port: http (80), Dst Port: 1127 (1127), Seq: 1396200326, Ack: 363869040

Hypertext Transfer Protocol

HTTP/1.1 200 OK\r\n

Server: Netscape-Enterprise/4.1\r\n

Date: Sat, 02 Nov 2002 02:53:48 GMT\r\n

Set-cookie: spopunder=1; path=/; domain=.nytimes.com\r\n

Cache-control: no-cache\r\n

Pragma: no-cache\r\n

Content-type: text/html\r\n

Connection: close\r\n

\r\n

0000 00 90 27 96 b8 07 00 e0 52 ea b5 00 08 00 45 00 ..'.....R.....E.
0010 01 0e b3 93 40 00 ed 06 16 0d 40 0f f7 c8 80 64@... ..@....d
0020 0b 0d 00 50 04 67 53 38 53 86 d8 e2 02 62 50 18 ...P.gS8 S....bP.
0030 7f ff 8a f6 00 00 48 54 54 50 2f 31 2e 31 20 32 0.....HT TP/1.1 2
0040 30 30 20 4f 4b 0d 0a 53 65 72 76 65 72 3a 20 4e 00 OK..S erver: N
0050 65 74 73 63 61 70 65 2d 45 6e 74 65 72 70 72 69 etscape- Enterpri
0060 72 65 7f 34 7e 31 0d 0e 44 61 74 65 72 70 72 69 22/4.1 Date: Sa

Filter: / Reset Apply File: nytimespackets

HTTP Proxy Server & Caching

- Web users generate large traffic volumes
- Traffic causes congestion & delay
- Can improve delay performance and reduce traffic in Internet by moving content to servers closer to the user
- Web proxy servers cache web information
 - Deployed by ISPs
 - Customer browsers configured to first access ISPs proxy servers
 - Proxy replies immediately when it has requested object or retrieves the object if it does not

Cookies and Web Sessions

- Cookies are data exchanged by clients & servers as header lines
- Since HTTP stateless, cookies can provide context for HTTP interaction
- *Set cookie* header line in reply message from server + unique ID number for client
- If client accepts cookie, cookie added to client's cookie file (must include expiration date)
- Henceforth client requests include ID
- Server site can track client interactions, store these in a separate database, and access database to prepare appropriate responses

Cookie Header Line; ID is 24 hexadecimal numeral

nytimespackets - Ethereal

File Edit Capture Display Tools Help

No.	Time	Source	Destination	Protocol	Info
1	0.000000	128.100.11.13	128.100.100.128	DNS	Standard query A www.nytimes.com
2	0.129976	128.100.100.128	128.100.11.13	DNS	standard query response A 64.15.247.200 A 64
3	0.131524	128.100.11.13	64.15.247.200	TCP	1127 > http [SYN] Seq=3638689752 Ack=0 win=1
4	0.168286	64.15.247.200	128.100.11.13	TCP	http > 1127 [SYN, ACK] Seq=1396200325 Ack=36
5	0.168320	128.100.11.13	64.15.247.200	TCP	1127 > http [ACK] Seq=3638689753 Ack=1396200
6	0.168688	128.100.11.13	64.15.247.200	HTTP	GET / HTTP/1.1
7	0.205439	64.15.247.200	128.100.11.13	TCP	http > 1127 [ACK] Seq=1396200326 Ack=3638690
8	0.236676	64.15.247.200	128.100.11.13	HTTP	HTTP/1.1 200 OK

Frame 6 (703 bytes on wire, 703 bytes captured)

Ethernet II, Src: 00:90:27:96:b8:07, Dst: 00:e0:52:ea:b5:00

Internet Protocol, Src Addr: 128.100.11.13 (128.100.11.13), Dst Addr: 64.15.247.200 (64.15.247.200)

Transmission Control Protocol, Src Port: 1127 (1127), Dst Port: http (80), Seq: 3638689753, Ack: 139620032

Hypertext Transfer Protocol

GET / HTTP/1.1\r\n

Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-powerpoint, application,

Accept-Language: en-us\r\n

Accept-Encoding: gzip, deflate\r\n

User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; windows NT 5.0)\r\n

Host: www.nytimes.com\r\n

Connection: keep-alive\r\n

Cookie: RMID=80e7478f5a393db9fc19f2c4; NYT-S=1002xv091grjagxb2AZ9Oxq4lqdEe/irdKSU3XunLr287eqe2QOMe5m08Re\r\n

0000 00 e0 52 ea b5 00 00 90 27 96 b8 07 08 00 45 00 ..R.....E.

0010 02 b1 54 45 40 00 80 06 e0 b8 80 64 0b 0d 40 0f ..TE@...d..@.

0020 f7 c8 04 67 00 50 d8 e1 ff d9 53 38 53 86 50 18 ...g.P...S8S.P.

0030 43 a4 87 81 00 00 47 45 54 20 2f 20 48 54 54 50 C....GE T / HTTP

0040 2f 31 2e 31 0d 0a 41 63 63 65 70 74 3a 20 69 6d /1.1..Ac cept: im

0050 61 67 65 2f 67 69 66 2c 20 69 6d 61 67 65 2f 78 age/gif, image/x

0060 2d 78 63 60 74 6d 61 70 7c 70 60 6d 61 67 65 7f x-bitmap image/

Filter: / Reset Apply File: nytimespackets

PING

- Application to determine if host is reachable
- Based on Internet Control Message Protocol
 - ICMP informs source host about errors encountered in IP packet processing by routers or by destination host
 - ICMP Echo message requests reply from destination host
- PING sends echo message & sequence #
- Determines reachability & round-trip delay
- Sometimes disabled for security reasons

PING from NAL host

```
Microsoft(R) Windows DOS  
(c)Copyright Microsoft Corp 1990-2001.
```

```
C:\DOCUME~1\1>ping nal.toronto.edu
```

```
Pinging nal.toronto.edu [128.100.244.3] with 32 bytes of data:
```

```
Reply from 128.100.244.3: bytes=32 time=84ms TTL=240  
Reply from 128.100.244.3: bytes=32 time=110ms TTL=240  
Reply from 128.100.244.3: bytes=32 time=81ms TTL=240  
Reply from 128.100.244.3: bytes=32 time=79ms TTL=240
```

```
Ping statistics for 128.100.244.3:
```

```
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),  
Approximate round trip times in milli-seconds:  
    Minimum = 79ms, Maximum = 110ms, Average = 88ms
```

```
C:\DOCUME~1\1>
```

Traceroute

- Find route from local host to a remote host
- Time-to-Live (TTL)
 - IP packets have TTL field that specifies maximum # hops traversed before packet discarded
 - Each router decrements TTL by 1
 - When TTL reaches 0 packet is discarded
- Traceroute
 - Send UDP to remote host with TTL=1
 - First router will reply ICMP Time Exceeded Msg
 - Send UDP to remote host with TTL=2, ...
 - Each step reveals next router in path to remote host

Traceroute from home PC to university host

Tracing route to www.comm.utoronto.ca [128.100.11.60]
over a maximum of 30 hops:

1	1 ms	<10 ms	<10 ms	192.168.2.1	Home Network
2	3 ms	3 ms	3 ms	10.202.128.1	
3	4 ms	3 ms	3 ms	gw04.ym.phub.net.cable.rogers.com [66.185.83.142]	
4	*	*	*	Request timed out.	
5	47 ms	59 ms	66 ms	gw01.bloor.phub.net.cable.rogers.com [66.185.80.230]	
6	3 ms	3 ms	38 ms	gw02.bloor.phub.net.cable.rogers.com [66.185.80.242]	
7	8 ms	3 ms	5 ms	gw01.wlfdle.phub.net.cable.rogers.com [66.185.80.2]	Rogers Cable ISP
8	8 ms	7 ms	7 ms	gw02.wlfdle.phub.net.cable.rogers.com [66.185.80.142]	
9	4 ms	10 ms	4 ms	gw01.front.phub.net.cable.rogers.com [66.185.81.18]	
10	6 ms	4 ms	5 ms	ralsh-ge3-4.mt.bigpipeinc.com [66.244.223.237]	Shaw Net
11	16 ms	17 ms	13 ms	rx0sh-hydro-one-telecom.mt.bigpipeinc.com [66.244.223.246]	Hydro One
12	7 ms	14 ms	8 ms	142.46.4.2	
13	10 ms	7 ms	6 ms	utorgw.onet.on.ca [206.248.221.6]	Ontario Net
14	7 ms	6 ms	11 ms	mcl-gateway.gw.utoronto.ca [128.100.96.101]	
15	7 ms	5 ms	8 ms	sf-gpb.gw.utoronto.ca [128.100.96.17]	University of Toronto
16	7 ms	7 ms	10 ms	bi15000.ece.utoronto.ca [128.100.96.236]	
17	7 ms	9 ms	9 ms	www.comm.utoronto.ca [128.100.11.60]	

Trace complete.

ipconfig

- Utility in Microsoft® Windows to display TCP/IP information about a host
- Many options
 - Simplest: IP address, subnet mask, default gateway for the host
 - Information about each IP interface of a host
 - DNS hostname, IP addresses of DNS servers, physical address of network card, IP address, ...
 - Renew IP address from DHCP server

netstat

- Queries a host about TCP/IP network status
- Status of network drivers & their interface cards
 - #packets in, #packets out, errored packets, ...
- State of routing table in host
- TCP/IP active server processes
- TCP active connections

netstat protocol statistics

IPv4 Statistics

Packets Received	= 71271
Received Header Errors	= 0
Received Address Errors	= 9
Datagrams Forwarded	= 0
Unknown Protocols Received	= 0
Received Packets Discarded	= 0
Received Packets Delivered	= 71271
Output Requests	= 70138
Routing Discards	= 0
Discarded Output Packets	= 0
Output Packet No Route	= 0
Reassembly Required	= 0
Reassembly Successful	= 0
Reassembly Failures	= 0
Datagrams Successfully Fragmented	= 0
Datagrams Failing Fragmentation	= 0
Fragments Created	= 0

UDP Statistics for IPv4

Datagrams Received	= 6810
No Ports	= 15
Receive Errors	= 0
Datagrams Sent	= 6309

ICMPv4 Statistics

	Received	Sent
Messages	10	6
Errors	0	0
Destination Unreachable	8	1
Time Exceeded	0	0
Parameter Problems	0	0
Source Quenches	0	0
Redirects	0	0
Echos	0	2
Echo Replies	2	0
Timestamps	0	0
Timestamp Replies	0	0
Address Masks	0	0
Address Mask Replies	0	0

TCP Statistics for IPv4

Active Opens	= 798
Passive Opens	= 17
Failed Connection Attempts	= 13
Reset Connections	= 467
Current Connections	= 0
Segments Received	= 64443
Segments Sent	= 63724
Segments Retransmitted	= 80

tcpdump and Network Protocol Analyzers

- `tcpdump` program captures IP packets on a network interface (usually Ethernet NIC)
- Filtering used to select packets of interest
- Packets & higher-layer messages can be displayed and analyzed
- `tcpdump` basis for many network protocol analyzers for troubleshooting networks
- We use the open source Ethereal analyzer to generate examples
 - www.ethereal.com