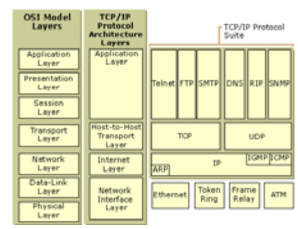
Review for mid-term exam

- Contents from Chapter 4.3 through Chapter 6
- Chapter 4: Network layer
- Chapter 5: Data link layer
- Chapter 6: Wireless and mobile networks
- See the list of topics from the review sheet.

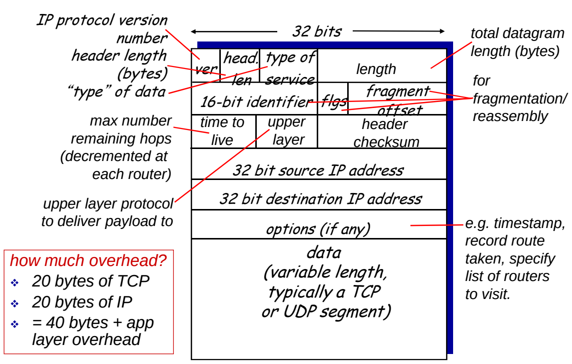
Internet Protocol (IP)

- Remember the context:

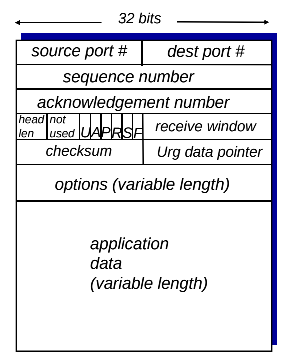


http://softpanorama.net/Net/Images/tcp_ip_layers.gif

IP datagram format



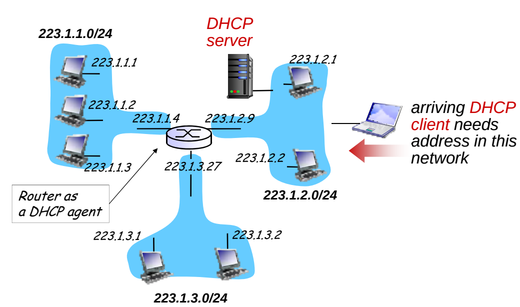
Review of TCP segment structure



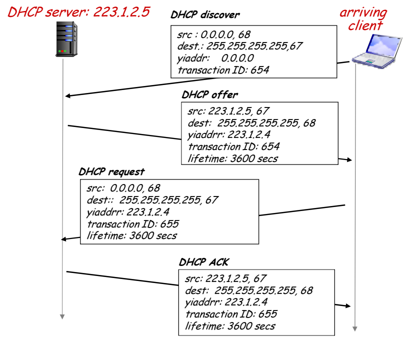
IP layer functions

- Fragmentation, resemble data
- Addressing (IP addresses)
  - Subnet
    - Class-full address: Class A, Class B, Class C
    - CIDR: Classless InterDomain Routing
  - Gateway
  - DHCP: dynamic host configuration protocol
  - NAT: network address translation protocol
- ICMP: Internet Control Message Protocol
- IPv4 and IPv6, Tunneling
- Routing (link-state routing and distance vector routing)
- Multi-casting

DHCP client-server scenario



DHCP client-server scenario

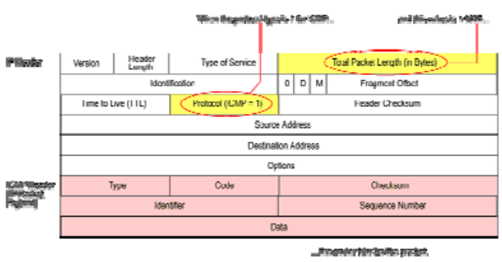


ICMP: internet control message protocol

- used by hosts & routers to communicate network-level information
  - error reporting: unreachable host, network, port, protocol
  - echo request/reply (used by ping)
- network-layer "above" IP:
  - ICMP msgs carried in IP datagrams
- ICMP message: type, code plus first 8 bytes of IP datagram causing error

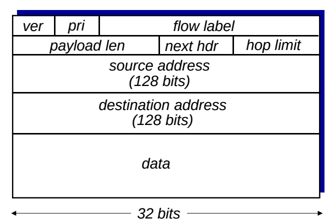
Type	*Code*	*description*
0	*0*	*echo reply (ping)*
3	*0*	*dest. network unreachable*
3	*1*	*dest host unreachable*
3	*2*	*dest protocol unreachable*
3	*3*	*dest port unreachable*
3	*6*	*dest network unknown*
3	*7*	*dest host unknown*
4	*0*	*source quench (congestion control - not used)*
8	*0*	*echo request (ping)*
9	*0*	*route advertisement*
10	*0*	*router discovery*
11	*0*	*TTL expired*
12	*0*	*bad IP header*

ICMP Packet Format



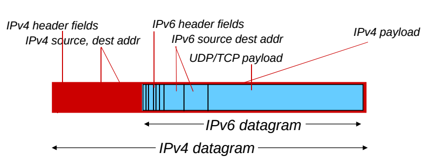
IPv6 datagram format

ver (4 bit): version number (6)
Priority (8 bit): identify priority among datagrams in flow
flow Label (20 bit): identify datagrams in same "flow." (concept of "flow" not well defined).
next header: identify upper layer protocol for data (same as in IPv4)

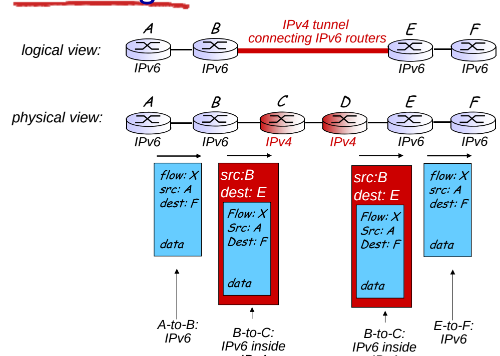


Transition from IPv4 to IPv6

- not all routers can be upgraded simultaneously
  - no "flag days"
  - how will network operate with mixed IPv4 and IPv6 routers?
- *tunneling*: IPv6 datagram carried as *payload* in IPv4 datagram among IPv4 routers



Tunneling



A Link-State Routing Algorithm

Dijkstra's algorithm

- net topology, link costs known to all nodes
  - accomplished via "link state broadcast"
  - all nodes have same info
- computes least cost paths from one node ('source") to all other nodes
  - gives *forwarding table* for that node
- iterative: after k iterations, know least cost path to k dest.'s

notation:

- c(x,y): link cost from node x to y; = ∞ if not direct neighbors
- D(v): current value of cost of path from source to dest. v
- p(v): predecessor node along path from source to v
- N': set of nodes whose least cost path definitively known

Dijsktra's Algorithm

```
Initialization:
  N' = {u}
  for all nodes v
    if v adjacent to u
      then D(v) = c(u,v)
    else D(v) = ∞

Loop
  find w not in N' such that D(w) is a minimum
  add w to N'
  update D(v) for all v adjacent to w and not in N' :
     D(v) = min( D(v), D(w) + c(w,v) )
  /* new cost to v is either old cost to v or known
   shortest path cost to w plus cost from w to v */
until all nodes in N'
```


Distance vector algorithm

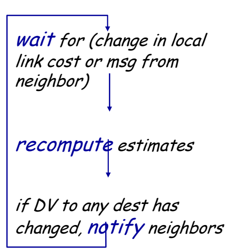
iterative, asynchronous: each local iteration caused by:

- local link cost change
- DV update message from neighbor

distributed:

- each node notifies neighbors *only* when its DV changes
  - neighbors then notify their neighbors if necessary

each node:



Distance vector algorithm

key idea:

- from time-to-time, each node sends its own distance vector estimate to neighbors
- when x receives new DV estimate from neighbor, it updates its own DV using B-F equation:

$$D_x(y) \leftarrow \min_v\{c(x,v) + D_v(y)\} \text{ for each node } y \in N$$

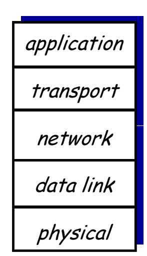
- *under minor, natural conditions, the estimate $D_x(y)$ converge to the actual least cost $d_x(y)$*

Routing on the Internet

- Autonomous Systems (AS): Internet is divided into a number of AS'es, each of which is more or less independent
- Intra-AS routing: (within AS), a.k.a. Interior Gateway Protocols (IGP)
  - RIP
  - OSPF
- Inter-AS routing: (between AS)
  - BGP (Border Gateway Protocols)

Data link layer protocols

- Putting things in perspective:
- *application:* supporting network applications
  - FTP, SMTP, HTTP
- *transport:* process-process data transfer
  - TCP, UDP
- *network:* routing of datagrams from source to destination
  - IP, routing protocols
- *link:* data transfer between neighboring network elements
  - Ethernet, 802.111 (WiFi), PPP
- *physical:* bits "on the wire"



MAC protocols: taxonomy

Three broad classes in Medium Access Control (MAC) protocols:

- *channel partitioning*
  - divide channel into smaller "pieces" (time slots, frequency, code)
  - allocate piece to node for exclusive use
- *random access*
  - channel not divided, allow collisions
  - "recover" from collisions
- *"taking turns"*
  - nodes take turns, but nodes with more to send can take longer turns

Channel partition

- TDMA: time division multiple access
- FDMA: frequency division multiple access
- CDMA: code division multiple access
- Or any combinations of the above

"Taking turns" MAC protocols

channel partitioning MAC protocols:

- share channel *efficiently* and *fairly* at high load
- inefficient at low load: delay in channel access, 1/N bandwidth allocated even if only 1 active node!

random access MAC protocols

- efficient at low load: single node can fully utilize channel
- high load: collision overhead

examples include token ring and token passing

"taking turns" protocols

look for best of both worlds!

Random access protocols

- when node has packet to send
  - transmit at full channel data rate R.
  - no *a priori* coordination among nodes
- two or more transmitting nodes ➜ "collision",
- random access MAC protocol specifies:
  - how to detect collisions
  - how to recover from collisions (e.g., via delayed retransmissions)
- examples of random access MAC protocols:
  - slotted ALOHA
  - ALOHA
  - CSMA, CSMA/CD, CSMA/CA

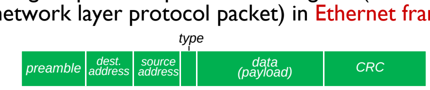
Examples of random access protocols

- Slotted ALOHA: nodes transmit at the beginning of a time slot; if collision, try again. Efficiency: ~0.36
- Pure ALOHA: nodes transmit at any time; if collision, try again. Efficiency: ~0.18
- CSMA: nodes listen before transmitting. Only if the medium is idle then nodes send data. If collision, try again.
- CSMA/CD: same as CSMA with collision detection (CD) so that when a collision is detected, the data transmission stops. Efficiency: 1/(1+5a) where a = tau/T

Ethernet specifics:

- Ethernet implements CSMA/CD using the binary backoff algorithm
- ARP: address resolution protocol is used to find the host which has the needed IP address

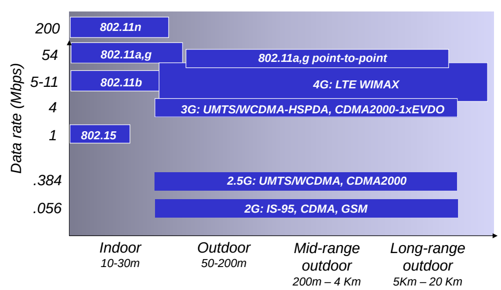
sending adapter encapsulates IP datagram (or other network layer protocol packet) in Ethernet frame



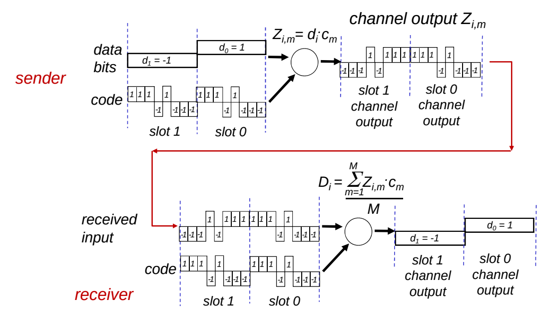
preamble:

- 7 bytes with pattern 10101010 followed by one byte with pattern 10101011
- used to synchronize receiver, sender clock rates

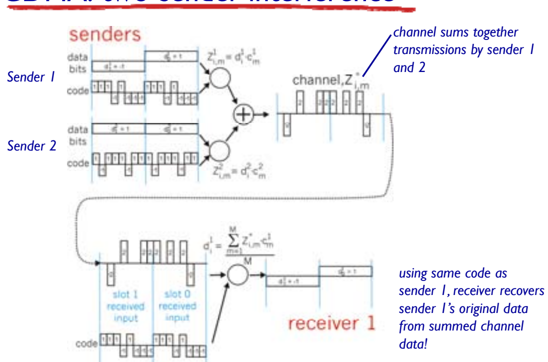
Characteristics of selected wireless links



CDMA encode/decode



CDMA: two-sender interference



The complete sender protocol

- When a station has a frame to transmit:
  1. If initially the state senses the channel idle, it transmits its frame after a short period of time known as *Distributed Inter-frame Space* (DIFS).
  2. Otherwise (sensing other transmission is on-going) the station chooses a random backoff value using binary exponential backoff and counts down this value when the channel is sensed idle. While the channel is sensed busy, the counter value remains frozen.
  3. When the counter reaches zero, the station transmits the entire frame and then waits for an acknowledgment.
  4. If an ack is received, the transmitting station knows that its frame has been received correctly. Continue Step 2 if more frames to send. If no ack is received, Continue Step 2 to resend the previous frame.

Collision avoidance mechanism

idea: allow sender to "reserve" channel rather than random access of data frames: avoid collisions of long data frames

- sender first transmits *small* request-to-send (RTS) packets to base station (BS) using CSMA
  - RTSs may still collide with each other (but they're short)
- BS broadcasts clear-to-send CTS in response to RTS
- CTS heard by all nodes
  - sender transmits data frame
  - other stations defer transmissions

avoid data frame collisions completely using small reservation packets!

802.11 frame: addressing

2	2	6	6	6	2	6	0 - 2312	4
frame control	duration	address 1	address 2	address 3	seq control	address 4	payload	CRC

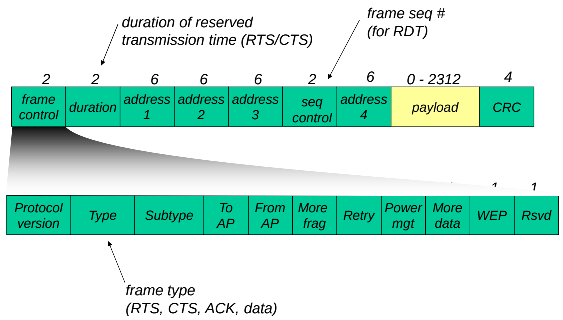
Address 1: MAC address of wireless host or AP to receive this frame

Address 2: MAC address of wireless host or AP transmitting this frame

Address 3: MAC address of router interface to which AP is attached

Address 4: used only in ad hoc mode

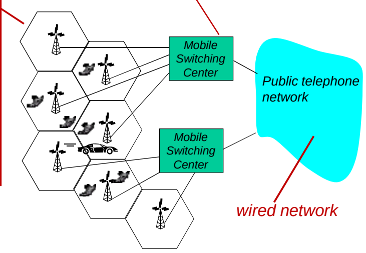
802.11 frame: more



Components of cellular network architecture

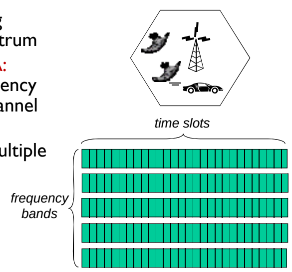
MSC
- connects cells to wired tel. net.
- manages call setup (more later!)
- handles mobility (more later!)

cell (BSS)
- covers geographical region
- *base station* (BS) analogous to 802.11 AP
- *mobile users* attach to network through BS
- *air-interface:* physical and link layer protocol between mobile and BS



Cellular networks: the first hop

Two techniques for sharing mobile-to-BS radio spectrum
- combined FDMA/TDMA: divide spectrum in frequency channels, divide each channel into time slots
- CDMA: code division multiple access



IEEE 802.11, 15, 16 compared

Parameters	EEE802.16d (802.16-2004 Fixed WiMAX)	IEEE802.16e (802.16-2005 Mobile WiMAX)	802.11 (WLAN, aka WiFi)	802.15.1 (Bluetooth)
Frequency Band:	2-66 GHz	2-11 GHz	2.4-5.8 GHz	2.4GHz
Range:	~31 miles	~31 miles	~100 meters	~10 meters
Maximum Data rate:	~134 Mbps	~15 Mbps	~55 Mbps	~3Mbps
Number of users:	Thousands	Thousands	Dozens	Dozens

http://www.javvin.com/protocolWiMAX.html

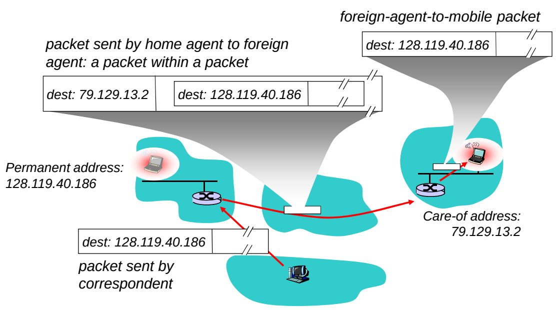
A brief comparison of different G's

http://techtectology.blogspot.com/2011/11/4g-vs-3g-vs-25g-vs-2g-vs-1g.html

Mobile IP

- RFC 3344
- has many features we've seen:
  - home agents, foreign agents, foreign-agent registration, care-of-addresses, encapsulation (packet-within-a-packet)
- three components to standard:
  - indirect routing of datagrams
  - agent discovery
  - registration with home agent

Mobile IP: indirect routing



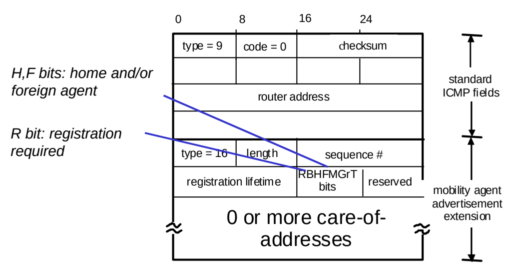
ICMP: Internet Control Message Protocol review

	Bit 0 - 7	Bit 8 - 15	Bit 16 - 23	Bit 24 - 31
IP Header (160 bits OR 20 Bytes)	Version/IHL	Type of service	Length	
	Identification		flags and offset	
	Time To Live(TTL)	Protocol	Checksum	
	Source IP address			
	Destination IP address			
ICMP Payload (64+ bits OR 8+ Bytes)	Type of message	Code	Checksum	
	Quench			
	Data (optional)			

A combination of "type of message" and "code" specifies the meaning of this ICMP packet. Among others
-- Type 9 is for "route advertising"
-- See a complete list from Wikipedia at
http://en.wikipedia.org/wiki/Internet_Control_Message_Protocol#Header

Mobile IP: agent discovery

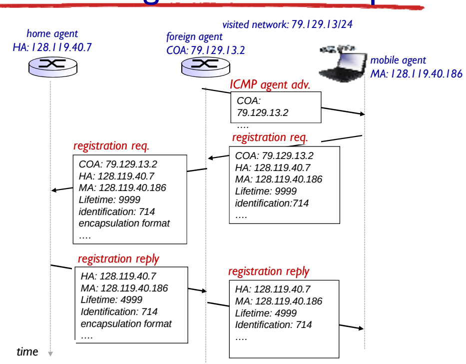
- *agent advertisement:* foreign/home agents advertise service by broadcasting ICMP messages (typefield = 9)



Flags in ICMP mobile extension

- H: home agent bit
- F: foreign agent bit
- R: registration required bit
- M,G: encapsulation bits (minimal or GRE encapsulation)
- B: busy
- r: reserved
- T: reverse tunneling

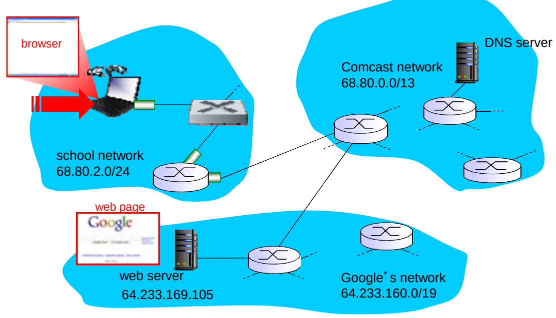
Mobile IP: registration example



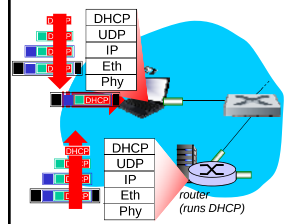
Synthesis: a day in the life of a web request

- journey down protocol stack complete!
  - application, transport, network, link
- putting-it-all-together: synthesis!
  - *goal:* identify, review, understand protocols (at all layers) involved in seemingly simple scenario: requesting www page
  - *scenario:* student attaches laptop to campus network, requests/receives www.google.com

A day in the life: scenario

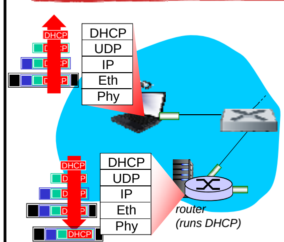


A day in the life… connecting to the Internet



- connecting laptop needs to get its own IP address, addr of first-hop router, addr of DNS server: use *DHCP*
- DHCP request *encapsulated* in *UDP*, encapsulated in *IP*, encapsulated in *802.3* Ethernet
- Ethernet frame *broadcast* (dest: FFFFFFFFFFFF) on LAN, received at router running *DHCP* server
- Ethernet *demuxed* to IP demuxed, UDP demuxed to DHCP

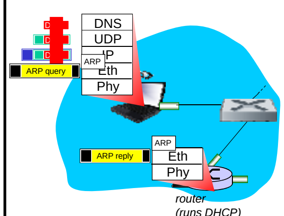
A day in the life… connecting to the Internet



- DHCP server formulates *DHCP ACK* containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server
- encapsulation at DHCP server, frame forwarded (*switch learning*) through LAN, demultiplexing at client
- DHCP client receives DHCP ACK reply

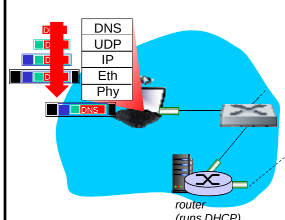
Client now has IP address, knows name & addr of DNS server, IP address of its first-hop router

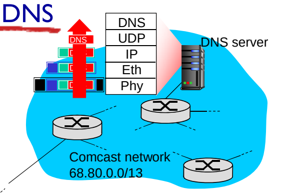
A day in the life… ARP (before DNS, before HTTP)



- before sending *HTTP* request, need IP address of www.google.com: *DNS*
- DNS query created, encapsulated in UDP, encapsulated in IP, encapsulated in Eth. To send frame to router, need MAC address of router interface: *ARP*
- *ARP query* broadcast, received by router, which replies with *ARP reply* giving MAC address of router interface
- client now knows MAC address of first hop router, so can now send frame containing DNS query

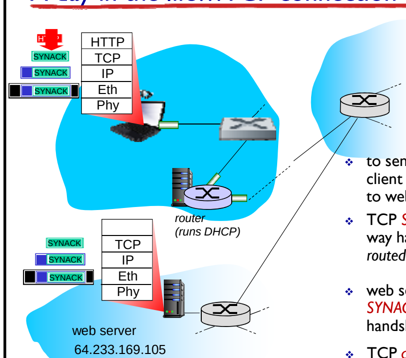
A day in the life… using DNS



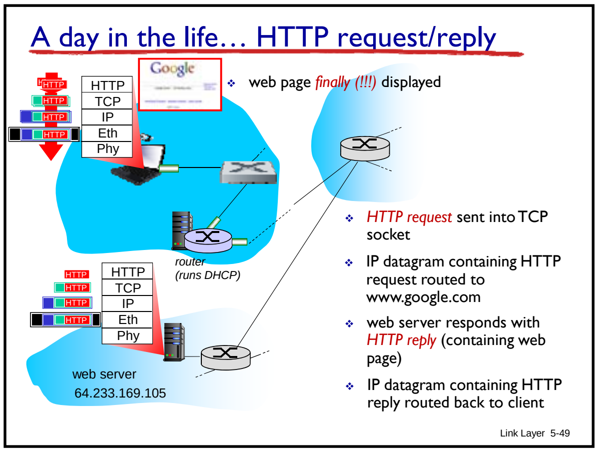


- IP datagram containing DNS query forwarded via LAN switch from client to 1st hop router
- IP datagram forwarded from campus network into comcast network, routed (tables created by *RIP, OSPF, IS-IS* and/or *BGP* routing protocols) to DNS server
- demux'ed to DNS server
- DNS server replies to client with IP address of www.google.com

A day in the life…TCP connection carrying HTTP



- to send HTTP request, client first opens *TCP socket* to web server
- TCP *SYN segment* (step 1 in 3-way handshake) *inter-domain routed* to web server
- web server responds with *TCP SYNACK* (step 2 in 3-way handshake)
- TCP *connection established!*


A day in the life… HTTP request/reply
web page *finally (!!!)* displayed
HTTP
TCP
IP
Eth
Phy
router
(runs DHCP)
web server
64.233.169.105
HTTP request sent into TCP socket
IP datagram containing HTTP request routed to www.google.com
web server responds with *HTTP reply* (containing web page)
IP datagram containing HTTP reply routed back to client
Link Layer 5-49