

I. Internet Control Message Protocol (ICMP)

Week 10

- described in RFC 792
- helper protocol for IP, but more like a 3 ½ layer protocol (like ARP as a 2 ½ layer protocol)
- since a helper protocol for IP, uses IP datagrams to deliver ICMP packets
- any IP device can send/receive ICMP packets (i.e. nodes & routers)
- provides a mechanism for IP devices to exchange information about network problems
- purpose of ICMP control messages is to provide feedback about problems in the communication environment, not to make IP reliable
- even though IP is an ? (unreliable) protocol, it is valuable to know about **semi-permanent** errors

General ICMP types:

1. Network errors

- network unreachable/unknown
- host unreachable/unknown
- protocol unreachable/unknown
- port unreachable/unknown

2. Network congestion

- *source quench* – sender sending too fast, message to slow down

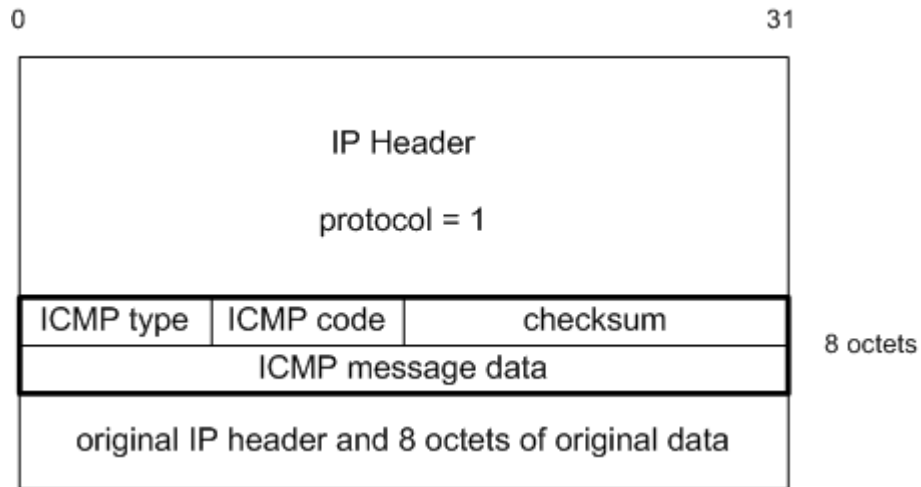
3. Time exceeded

- TTL becomes zero (see TraceRoute)

4. Network queries

- echo request/reply (see ping)

General ICMP PDU format:



- why original header and 8 octets? (see pg. 197 & 204)
- some (not all) sample type values (see pg 196)

ICMP Type	Description	Family
0	echo reply	query
3	destination unreachable	error
4	source quench	error
8	echo request	query
9	router advertisement	query
11	time exceeded	error

- some (not all) sample code values for type 3 (see pg 206)

ICMP Code	Description
0	network unreachable
1	host unreachable
2	protocol unreachable
3	port unreachable
4	frag required but DF bit set
5	source route failed

- note that all ICMP type packets do not have all parts following the checksum, or some parts may be unused, or some parts may be subdivided, e.g.
 - echo request/reply packets **do not have** original IP header data
 - the echo request/reply message field is subdivided into an ID field and a sequence # field (ID field does not change but the seq # does for a single ping session)

II. Introduction to the Transport Layer

Week 11

In general the transport layer is:

- layer 4 protocol (in TCP/IP model)
- connection oriented & reliable
- a provider of error detection / correction
- recall PDU called segments (aka TPDU)
- transport layer provides logical connection between end nodes application processes (where layer 3 provided connections between nodes)
- application identification based upon **ports**, numeric IDs which identify the sending/receiving application

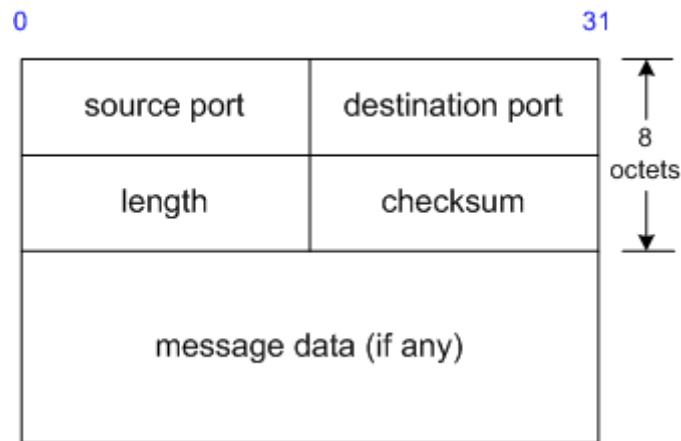
Name	Range	Description
well-known*	0-1023	basic core services
registered*	1024-49151	registered industry applications
dynamic or ephemeral	49152-65535	temporary ports

*well known or registered ports usually on server side, e.g. DNS:53, WWW:80, IM: 5190
– see **/etc/services**

- 2 main protocols
 - TCP – transport control protocol (IP protocol = 6)
 - UDP – user datagram protocol (IP protocol = 17)

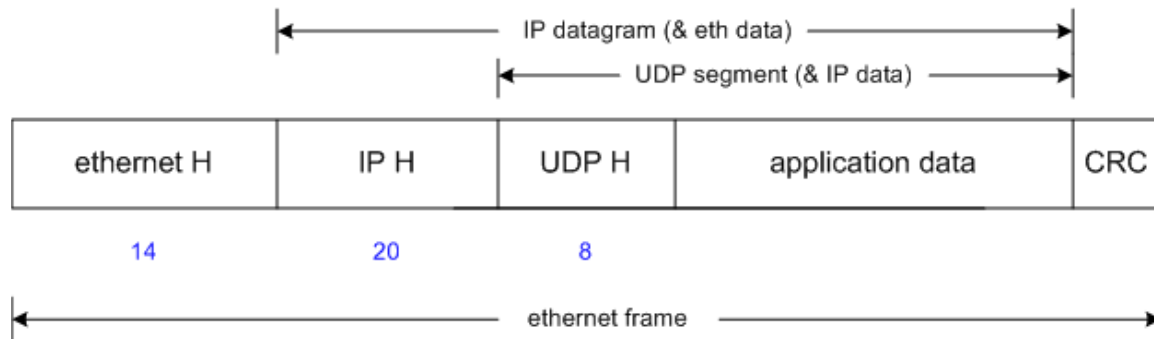
III. User Datagram Protocol (UDP)

- defined in RFC 768
- UDP provides none of the reliability (or connection) associated with the transport layer – so why use it?
 - offers low overhead and high performance
 - many applications cannot use TCP, e.g. streaming audio & video where error correction is a liability
 - most networking technologies today are fairly reliable, so an unreliable protocol is still fairly reliable
 - if errors do occur, these are most likely caught at the application layer
- UDP PDU – 8 octets (typically), pg 258



- **source port:** identifies points at which upper-layer source processes send the UDP data
- **destination port:** identifies points at which upper-layer destination processes receive the UDP data
- **length:** length (in bytes) of the entire UDP message, including header & data
- **checksum:** error check of entire message, including the header & data (recall IP only checksums its header)

- checksum also includes a “pseudo-header” which includes source & destination IP addresses, protocol id (UDP = 17) and the size of the UDP PDU
- optional for UDP (why?), required for TCP



- see /etc/services (on linux)
- search for etc on Win XP
- see <http://www.iana.org/assignments/port-numbers>

IV. Transmission Control Protocol (TCP)

- RFC 793
- recall TCP is a reliable, connection based protocol
- **services** provided by TCP
 1. virtual circuits
 - provide guaranteed connection
 - data exchange between VC is full duplex
 2. application I/O management
 - internal addressing (port assignment)
 - connection setup/teardown
 - data transfer
 3. network I/O management
 - efficient segment sizing
 - MTU/MRU/buffers/header sizes

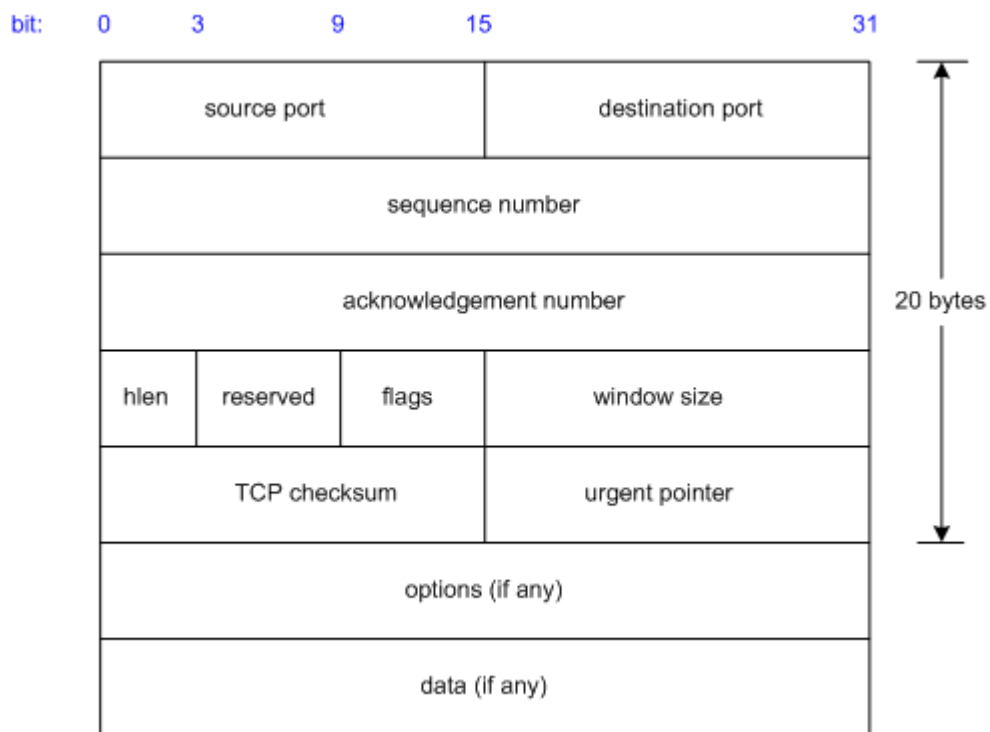
4. flow control

- adjustment of send/receive rates

5. reliability

- error detection
- error correction

- TCP PDU (see pg. 327)



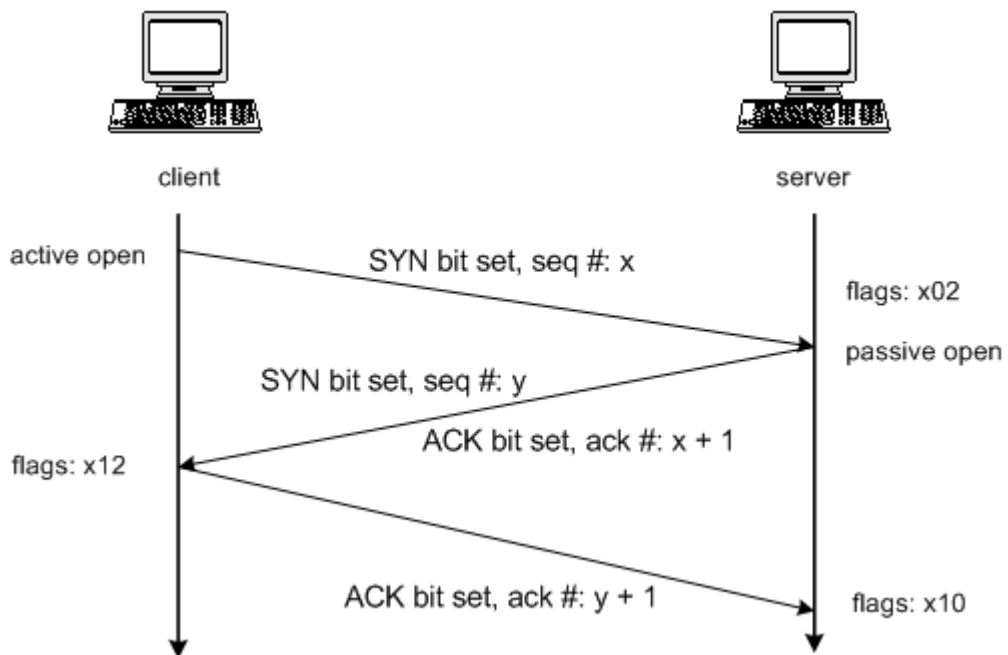
- **source port** (16 bits): identifies points at which upper-layer source that created the data
- **destination port** (16 bits): identifies points at which upper-layer destination should receive the data

- **sequence #** (32 bits): identifies the first byte of data in the stream from the sender to the receiver
 - allows the destination to sort data in proper order
 - ISN: initial sequence number when VC is established
- **acknowledgement #** (32 bits): identifies the next sequence number the destination expects to receive
 - identifies all data up to, but not including, this number has been received
 - simplified, $\text{sequence \# in} + \text{bytes data received} = \text{ack \# out}$
- **header length** (4 bits): size of header in 32 bit multiples, only size of header, not the size of data (unlike UDP)
- **reserved** (6 bits): currently unused, set to zero
- **control flags** (6 bits): provide VC management services
 - URG (urgent)
 - ACK – every segment sent will set this (except for 1st one and reset)
 - PSH
 - RST
 - SYN – VC endpoints use to sync their sequence numbers (ISN's)
 - FIN
- **window**: flow control mechanism
- **checksum**: checksum of entire segment (header & data)
 - mandatory (unlike UDP), why?
 - if checksum bad, segment is dropped
- **urgent**: indicates any urgent segments
- **options**: see pg 346

V. TCP Connection Establishment Sequence

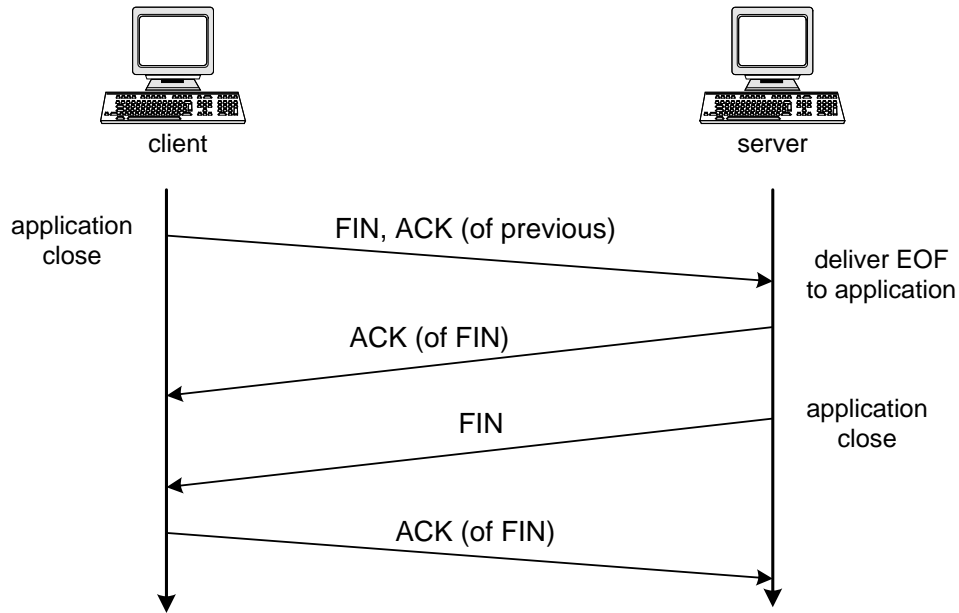
- referred to as the **3-way handshake**
- see pg 271 - 281
- recall ACKs set for every segment except the 1st (and RST)
- flag values:

32	16	8	4	2	1
U	A	P	R	S	F
R	C	S	S	Y	I
G	K	H	T	N	N



VI. TCP Connection Teardown

- requires four segments to terminate a connection
- since TCP connection is full-duplex, each direction must be terminated independently
- when a side receives a FIN, it will send an ACK of the incoming sequence number + 1



Note 2nd and 3rd segments usually combined

from Stevens, pg 234

VII. Incomplete Termination Types

1. half close

- one end of the VC sends a FIN, other continues sending data
- see handout

2. half open

- one end of the VC closes or aborts without knowledge of the other end
- e.g. client turned off