- TCP
  - is an end-to-end protocol
  - creates a virtual "circuit" between hosts (e.g. like establishing a telephone connection)
  - uses handshaking (via ACK messages) and sequence numbers to ensure <u>all data</u> gets to the receiving host <u>in order</u>
  - data is treated as a continuous stream of bytes delivered in segments of at most 64 kiB (often 1460 B to fit in an Ethernet frame with IP header (20 B) and TCP header (20 B)
  - the ends of the virtual circuit are called <u>sockets</u> which are specified by IP addresses and port number
  - some common port numbers:

20	FTP	file transfer protocol
22	SSH	secure shell
80	HTTP	hypertext transfer protocol
143	IMAP	internet message access protocol
443	HTTPS	http secure
587	SMTP	simple mail transfer protocol

- TCP header

\# of bits

source port 16	destination port 16
sequence # 32	
ack # 32	
flags 16	receiver capacity 16
checksum 16	urgent pointer 16

$\}$ 20 bytes

urgent pointer: "interrupt" mechansim

- UDP: User Datagram Protocol
  - a <u>connectionless</u> alternative to ~~[illegible]~~ TCP
  - faster than TCP (smaller header, no acknowlegement) but not reliable

- UDP header

source port 16	destination port 16
message length 16	optional checksum 16

$\}$ 8 bytes

optional checksum: $0 \Rightarrow$ no checksum

Fault Tolerance

1) Concepts [13.1]

Fault – cause of a problem (e.g. environmental faults, mechanical faults, design fault)

Error – manifestation of a fault within the system

Failure – system deviates from its specification as the result of an error

- approaches to dealing with faults

① avoidance
  - prevent them from happening (shielding, breakers, reliable prog. tech.)

② tolerance
  - continue to execute within spec.
  - graceful degradation (operate with degraded service)
  - failsafe (halt in a safe manor)

- case study: Voyager 2
  - launched August 20, 1977
  - explored Jupiter and Saturn then flew by Uranus and Neptune and headed to interstellar space (~~heliospa~~ heliopause)
  - in May 2010 its transmissions became corrupted
  - it was 13.8 billion km from earth ⇒ transmission delay was ~13 hours
  - examined the memory of the Flight Data System (FDS) computer and found a flipped bit that affected a command
  - it's hypothesized that cosmic radiation flipped the bit
  - they corrected the bit and correct transmission resumed

fault: insufficient radiation hardening + solar radiation

error: flipped bit

failure: garbled ~~transm~~ communication

2) Clock Synchronization [13.4]

- independent devices can have their own clocks
  - they are usually quartz crystal oscillators
- clocks are imperfect devices (affected by temperature, supply voltage) and ~~as~~ so "drift" over time
  - 0.5 s drift per day is not uncommon
- clock C_i gives time $C_i(t)$ at real time t
- options for clocks in a system to keep a common time:
  1. have a standard clock C_s against which the others synchronize
  2. clocks synchronize with each other

2.1) Standard Clock

-requirements for synchronizing against C_s

1. correctness: $|C_i(t) - C_s(t)| < \varepsilon$
   e.g. an absolute error bound
2. bounded drifted: $\left| \frac{dC_i(t)}{dt} - 1 \right| < \rho$
   e.g. clock i runs at approximately the correct rate
3. monotinicity: $C_i(t_1) \geq C_i(t_0)$ where $t_1 > t_0$
   e.g. time goes forward, not backward
4. chronoscopicity: if $t_2 - t_1 = t_4 - t_3$, then $C_i(t_2) - C_i(t_1) \approx C_i(t_4) - C_i(t_3)$
   e.g. measurement of two equal intervals should be approximately equal

- clock correction

e.g. a 32768 Hz clock is determined to be 100ms fast and this needs to be corrected

- a sudden correction (e.g. subtract 100ms) would satisfy Requirement 1 but would violate Requirements 3 and 4

- ∴ a gradual correction is preferred e.g. count 32768 + 1 ticks as 1 s

  - slows the clock by $\frac{1}{32768 \text{ Hz}} = 3.05 \times 10^{-5}$ s per second

  - the correction will take $\frac{0.1 \text{ s}}{3.05 \times 10^{-5} \text{ s/s}} =$

    $3276.8 \text{ s} \approx 55 \text{ min}$.

- see further A4, Q7

- how often should we synchronize?
  - assume clock i is ~~synch~~ synchronized at t_0

$$|C_i(t_0) - t_0| < \delta$$

synchronization error

- after drifting

when to sync. next

$$|C_i(t) - t| < \delta + \rho(t - t_0)$$

- we must ensure that $|C_i(t) - t| < \varepsilon$

absolute error bound

$$\delta + \rho(t - t_0) \leq \varepsilon$$

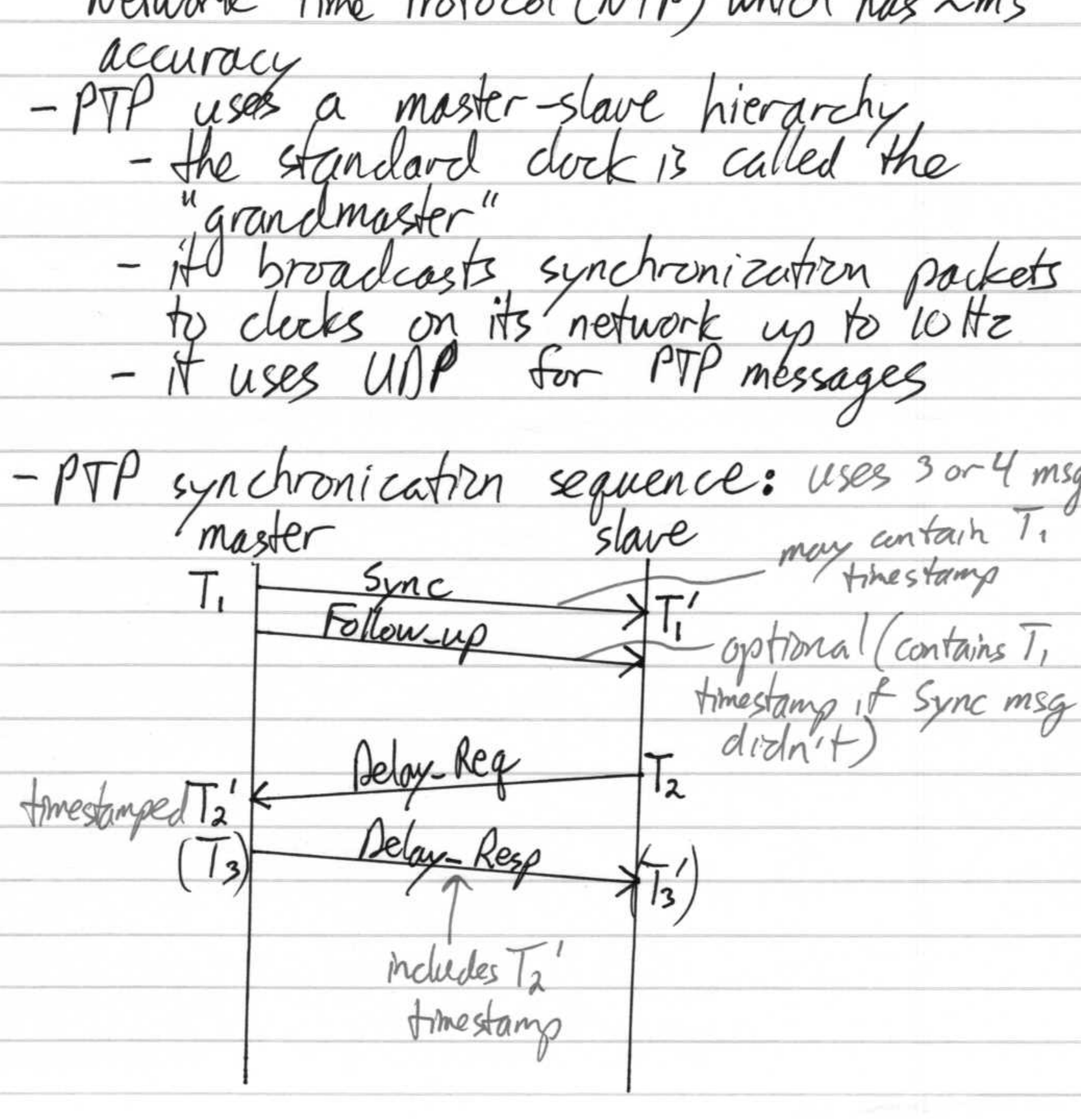
$$\underbrace{t - t_0}_{\text{maximum synchronization period}} \leq \frac{\varepsilon - \delta}{\rho}$$

- see example A4 Q8

2.2) Precision Time Protocol (PTP)

- IEEE standard 1588-2008
- achieves up to nanosecond synchronization accuracy using hardware timestamping
- PTP is a more precise alternative to the Network Time Protocol (NTP) which has ~ms accuracy
- PTP uses a master-slave hierarchy
  - the standard clock is called the "grandmaster"
  - it broadcasts synchronization packets to clocks on its network up to 10 Hz
  - it uses UDP for PTP messages

- PTP synchronization sequence: uses 3 or 4 msg.



slave sync. error

network latency

$$T_1' = T_1 + \delta + \ell$$

known

$$T_2' = T_2 - \delta + \ell$$

$$\delta = T_1' - T_1 - \ell \quad , \quad \delta = T_2 - T_2' + \ell$$

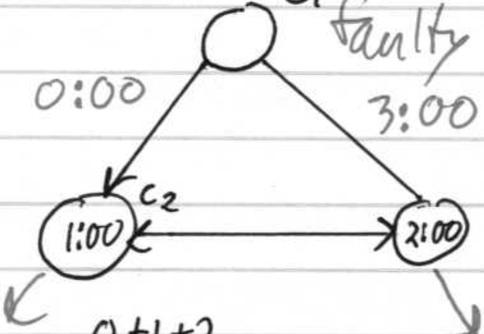
$$2\delta = T_1' - T_1 - \ell + T_2 - T_2' + \ell$$

$$\delta = \frac{T_1' - T_1 + T_2 - T_2'}{2}$$

correction factor for slave clock

2.3) Distributed Clock Synchronization [13.4, 4.2]

- there are n clocks in the system that use averaging to synchronize
- faulty clocks or faulty communication can interfere with synchronization



average $= \frac{0+1+2}{3} = 1$ average $= \frac{1+2+3}{3} = 2$

- averaging doesn't always work
- CNV (Convergence) algorithm:
  - at time t all clocks report their time C_j
  - each clock C_k records the time from each clock as

$$\tilde{C}_j = \begin{cases} C_j & \text{if } |C_j - C_k| < \varepsilon \\ C_k & \text{otherwise} \end{cases}$$

  - update each clock C_k

$$C_k \leftarrow \frac{1}{n} \sum_{j=1}^{n} \tilde{C}_j$$

e.g. 4 clocks, $\varepsilon = 1$

- at $t = 1000$:

	read	updated time
C_1	999.6	$(999.6 + 999.8 + 999.9 + 1000.3)/4 = 999.9$
C_2	999.8	"
C_3	999.9	"
C_4	1000.3	"

- at $t = 2000$: C_1 is faulty (slow)

	read	updated time
C_1	1998.6	$(1998.6 + 1999.5 + 1998.6 + 1998.6)/4 = 1998.825$
C_2	1999.5	$(1998.6 + 1999.5 + 2000.0 + 2000.2)/4 = 1999.575$
C_3	2000.0	$(2000.0 + 1999.5 + 2000.0 + 2000.2)/4 = 1999.925$
C_4	2000.2	$(2000.2 + 1999.5 + 2000.0 + 2000.2)/4 = 1999.975$

- average of the working clocks $(C_2, C_3, C_4) = 1999.825$
  - excluding ~~the~~ C_1 instead of replacing it with C_3, C_4 would give an average $= 1999.792$ (worse

- this will keep the majority of working clocks in sync if $n \geq 3d + 1$

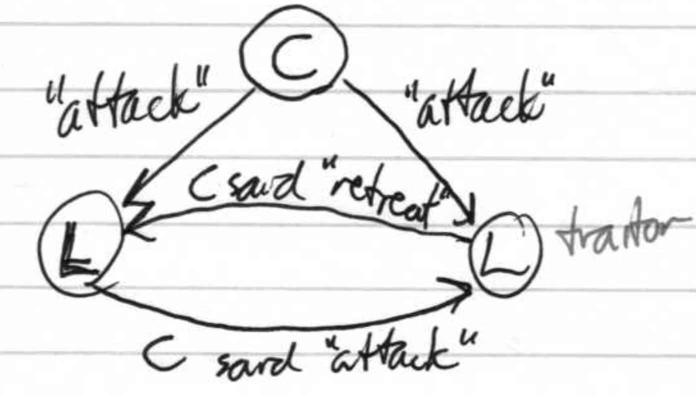
  \# of faulty clocks

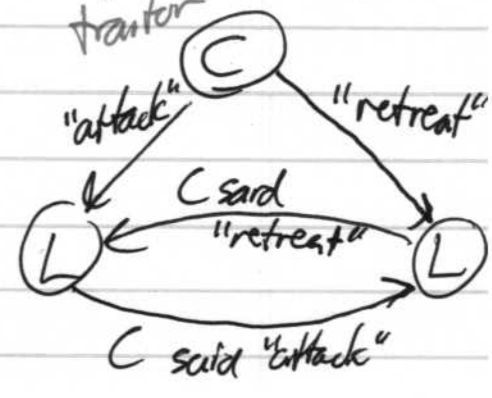
3) Byzantine Generals Problem [13.5]

- this problem describes when one or more processors in a distributed system ~~mail~~ may fail and send inconsistent messages to the other processors e.g. present different symptoms to different observers
- this failure is hard to detect in a fault tolerant system

- the story goes that the Byzantine generals have surrounded a city and must decide together whether to all attack or all retreat
- partial attack/retreat will result in failure
- they only communicate by messages
- some generals may be traitors
- it gets rephrased as 1 commanding general sending an order to n-1 lieutenant generals
- the commander or lieutenants may be disloyal

e.g.





- what should the loyal participants do?
- we need a default action e.g. "retreat"