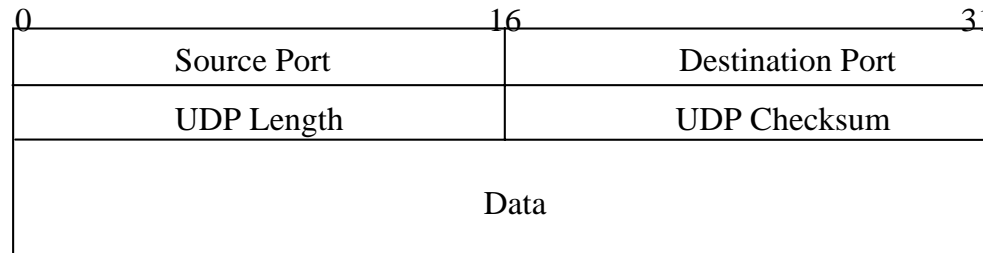


UDP and TCP

- User Datagram Protocol

- unreliable, connectionless transport layer protocol
- services beyond IP : demultiplexing and error checking



- » port numbers added in the header

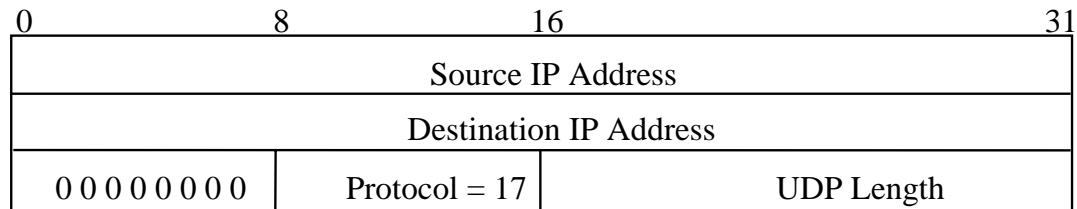
- to identify the particular application in the given host

- » optional checksum on the whole datagram

- 0 checksum value means no checksum wanted
- a checksum calculated to be 0 has to be included as all 1s

- IP checksum algorithm uses 1's complement arithmetic; so all 1's = 0

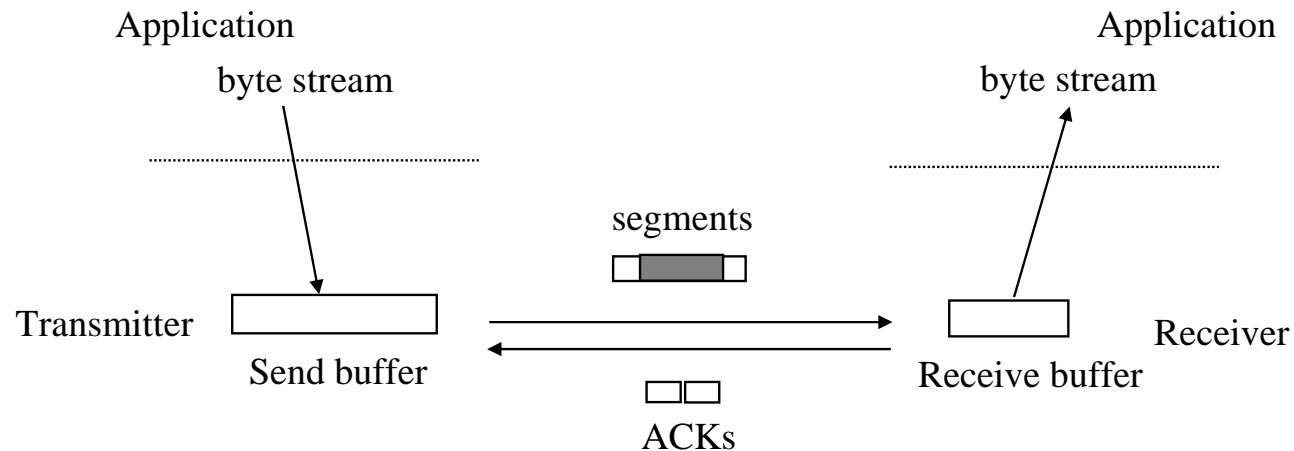
- » the datagram padded out to a multiple of 16 bits for checksum calculation
 - but the padding not transmitted
- » a pseudo-header also included in the checksum calculation
 - and not transmitted with the datagram



- » permits a check that the datagram has reached the correct destination
- » corrupted datagrams are discarded
 - no error message returned to source

- Transmission Control Protocol

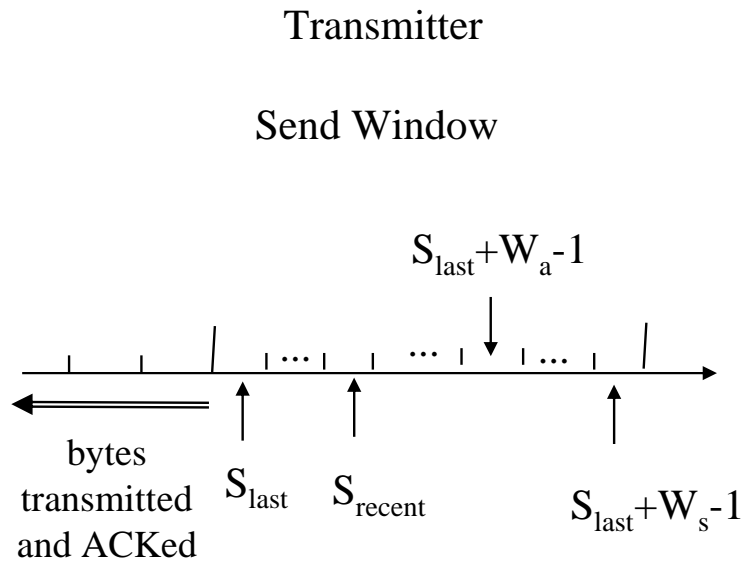
- reliable in-sequence connection-oriented stream service over IP
- provides a full duplex bidirectional logical connection between two application processes
- uses a version of the selective repeat ARQ protocol incorporating flow control and congestion control
- data received from source application treated as a stream of bytes
 - » TCP transmits in blocks convenient to itself
- data taken by the destination at its own convenience



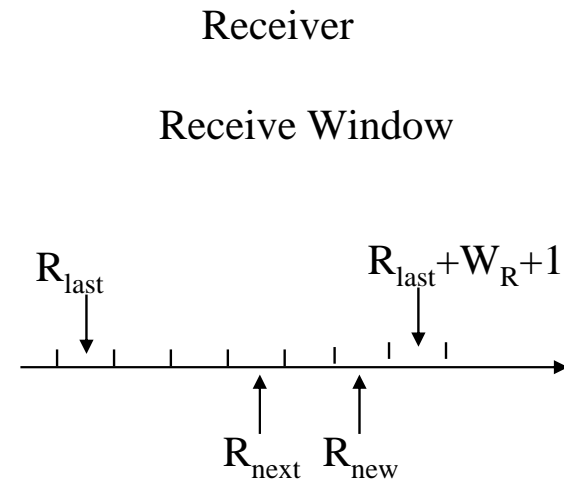
- IP packets are transported in a connectionless fashion
 - » can take different routes through the network
 - » some may be delayed and can therefore arrive out of sequence

- sequence numbers used in the ARQ protocol must be unambiguous
 - » packets from old TCP connections may arrive at the receiver during the next TCP connection
 - » problem avoided by using 32-bit sequence numbers
 - from some randomly selected starting number onwards
 - selected during the connection phase
 - » at any time, sequence numbers are being accepted from a very much smaller window
 - so likelihood of accepting an old packet very low
 - » TCP also enforces a time-out period at the end of each connection
 - allows out-of-date packets to be discarded

– the TCP sliding window mechanism :



S_{last} oldest unacknowledged byte
 S_{recent} highest-numbered transmitted byte
 $S_{last} + W_a - 1$ highest-numbered byte that can be transmitted
 $S_{last} + W_s - 1$ highest-numbered byte that can be accepted from the application



R_{last} highest-numbered byte not yet read by the application
 R_{next} next expected byte
 R_{new} highest numbered byte received correctly
 $R_{last} + W_R - 1$ highest-numbered byte that can be accommodated in receive buffer

- R_{new} can be greater than R_{next}
 - » since the receiver can accept out-of-sequence error-free packets of bytes
- receiver can buffer at most W_r bytes at any one time
 - » so $R_{\text{last}} + W_r - 1$ is the maximum numbered byte that the receiver is prepared to accept
 - » since the destination application may not take bytes as soon as they are received
- transmitter puts a string of bytes to be transmitted into a *segment*
- segment has a header containing source and destination *ports*
 - » the IP packet header contains the source and destination IP addresses
- header also contains the *sequence number*
 - » which corresponds to the *first byte* in the string of bytes being transmitted
- transmitter decides to transmit a segment when the number of bytes ready to be transmitted exceeds some threshold
 - » or when a time-out expires after the last segment transmitted
- source application can use a *push* command to *force* the transmitter to send a segment

- when a segment arrives, receiver performs an error check
- if segment is error free, the bytes are inserted into the appropriate locations in the receive buffer
- if the received bytes correspond to R_{next} onwards, R_{next} is advanced to the location of the next byte that has not yet been received
- an acknowledgment with the sequence number R_{next} is piggybacked in a segment being sent back to the transmitter
- this acknowledges correct receipt of all the bytes up to $R_{next}-1$
 - » the transmitter can then update its S_{last} up to R_{next}
- the receiver will accept out-of-sequence but error-free segments of bytes
 - » since same order of arrival is not guaranteed
 - » R_{next} could therefore be advanced further on than just the contents of this last segment received
- a NAK system à la basic Selective Repeat is not appropriate
 - » since an out-of-sequence segment does not imply a lost segment
 - » a NAK system has been proposed, however
 - then duplicate segments may have to be discarded

- flow control :
- implemented by means of an *advertised window* field in the segment header
- segments travelling back to the transmitter contain the advertised window size
 - » which informs the transmitter of the buffering available at the receive end
- the advertised window size is given by :

$$W_a = W_r - (R_{\text{new}} - R_{\text{last}})$$

- the transmitter is obliged to keep the number of outstanding bytes below this advertised window size :

- » i.e. $S_{\text{recent}} - S_{\text{last}} \leq W_a$

- » if destination application stops taking bytes from the receive buffer

- » R_{last} will remain fixed but R_{new} will increase as bytes are received

- » eventually the receive window will be full when $R_{\text{new}} - R_{\text{last}} = W_r$

- » i.e. $W_a = 0$, which causes the transmitter to stop sending

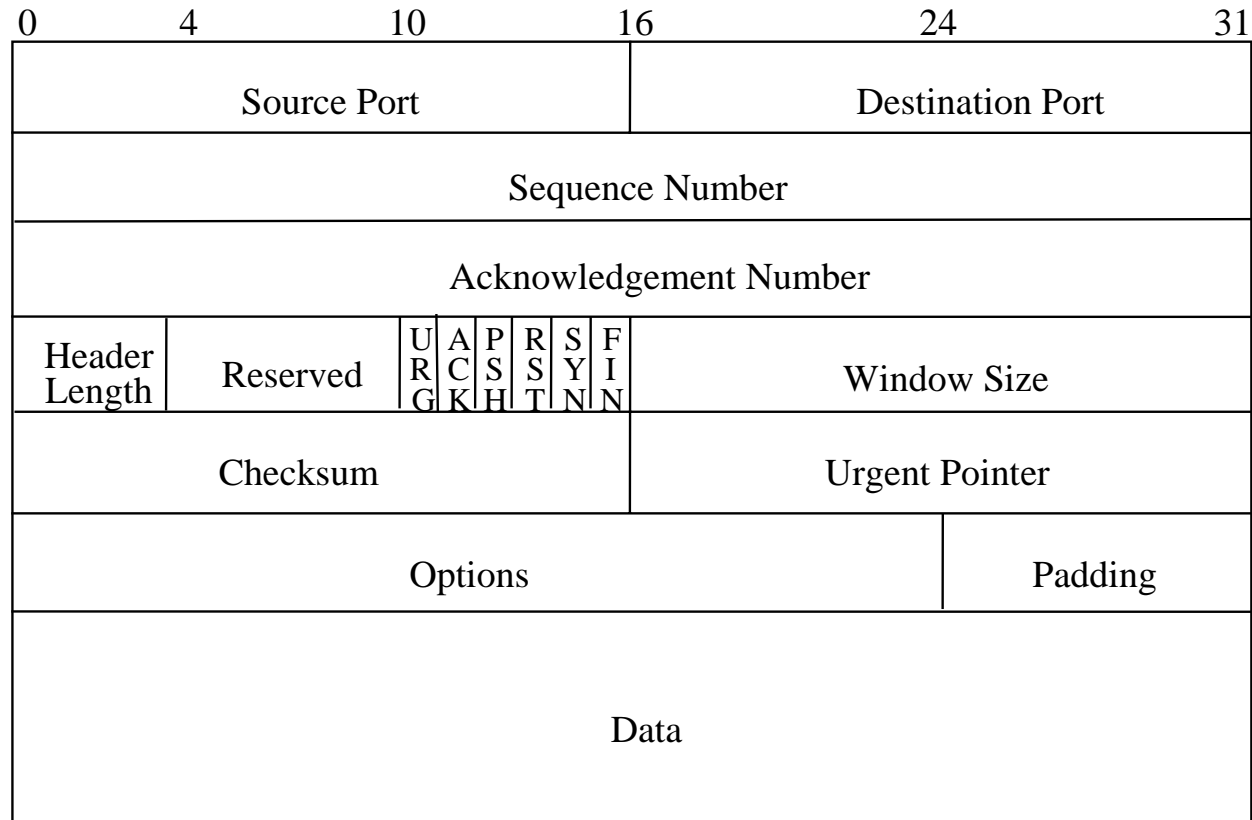
- » transmitter can continue accepting bytes from its source application until its buffer contains W_s bytes

- recall that to also deal with congestion, the transmitter can only transmit the minimum of the *advertised window* and the *congestion window*

- retransmission procedure:
 - » transmitter starts a timer for each segment sent
 - » if this expires before any of the bytes are acknowledged, the segment is retransmitted
- segment transmission times can vary widely under varying network load
 - » so TCP uses an adaptive technique for setting the time-out value
 - » using a continuously updated round-trip time (RTT)
 - » and an estimate of its variance/standard deviation (DEV)
 - » typically $RTT + 4 \times DEV$

- TCP Segment

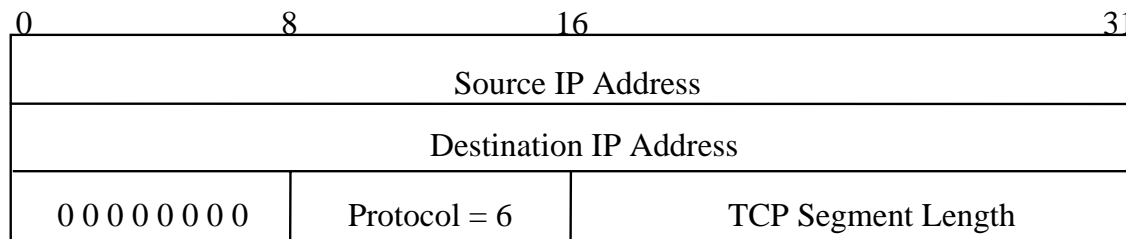
- a 20-byte fixed part plus a variable size options field:



» source and destination ports of sending and receiving applications

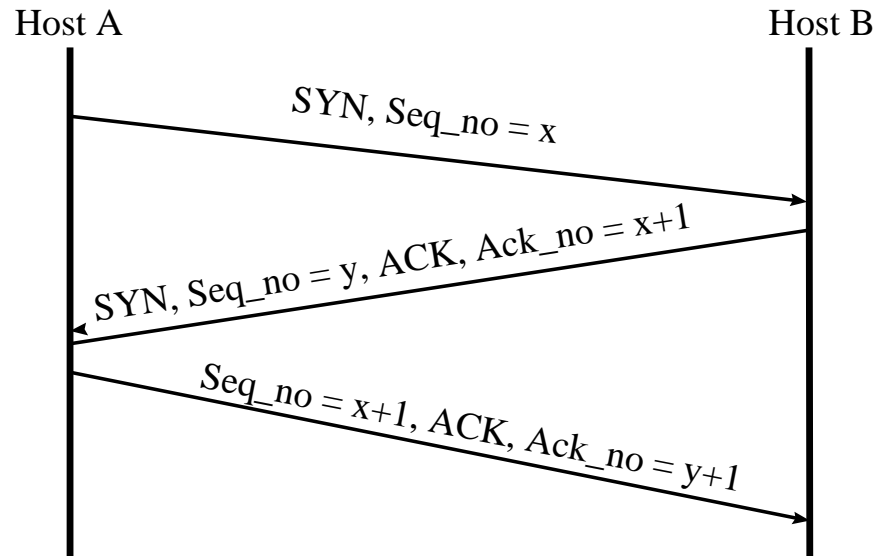
- » sequence number : of first byte in this segment
 - wraps around at $2^{32}-1$
 - if SYN bit set to 1, sequence number is the *initial sequence number*
 - a TCP connection is full duplex so each end point maintains its own independent sequence numbers
- » acknowledgment number : if ACK bit set to 1, the piggybacked R_{next} value
 - unused if ACK not set
- » header length : in 32-bit words
 - necessary since headers can be variable length
- » window size : number of bytes the sender is willing to accept
 - flow control and congestion control
- » urgent pointer : when URG set to 1, value indicates the last byte of *urgent* data that needs immediate delivery
 - first byte of urgent data not specified, so all up to last is considered urgent
- » PSH : tells receiver to pass the data to its application immediately
 - rather than waiting for a buffer load of some threshold in size
- » RST : tells receiver to abort the connection

- » SYN : used to request a connection
- » FIN : tells receiver sender has no more data to send
 - sender can still receive data in the reverse direction
- » Options :
 - maximum segment size : specified during connection set up
 - window scale : allows use of larger advertised windows
 - scaling up to a factor of 2^{14} : allows size up to 2^{30}
 - timestamp : used for more accurate RTT calculation
 - and for very high speed connections, when sequence numbers may wrap around during lifetime of the connection
- » Checksum : calculation includes
 - segment padded with zeroes to a multiple of 16 bits
 - a pseudo header added to the beginning of the header



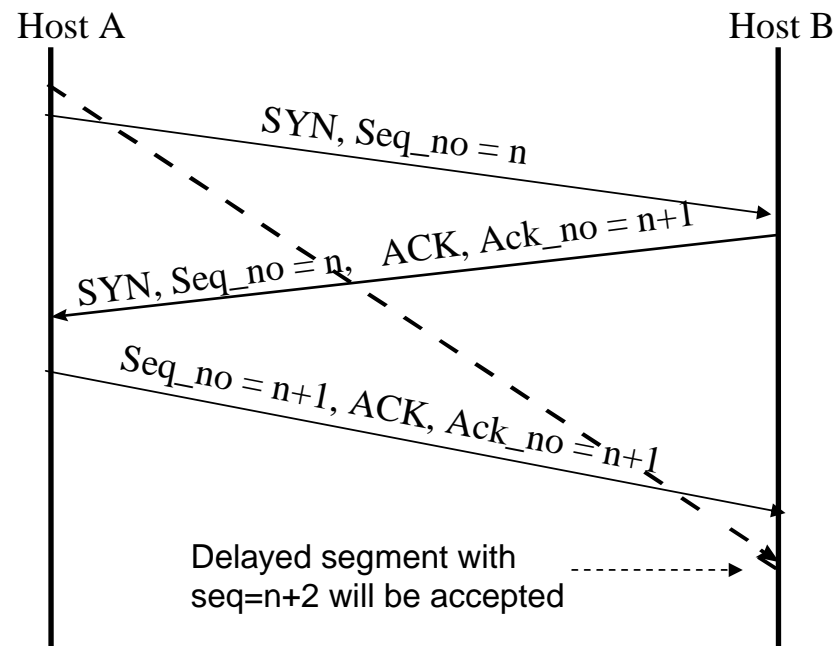
- but neither transmitted
- permits checking that segment reached the intended destination

- Connection establishment:



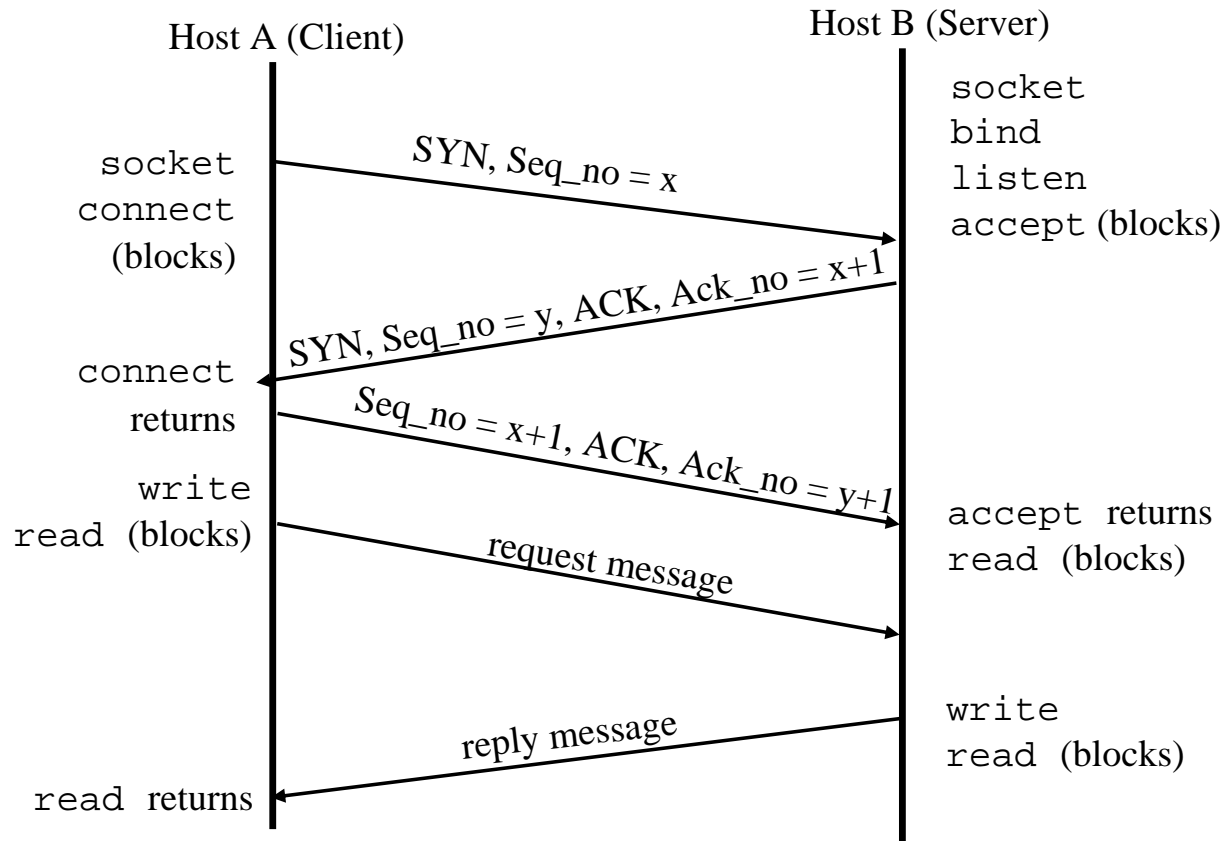
- » A sends connection request to B
 - SYN=1; (initial) sequence number = x
- » B acknowledges connection request
 - ACK=1; SYN=1; (initial) sequence number = y ; next data byte expected= $x+1$
 - (A's SYN consumes the first seq. no.)
- » A acknowledges the request from B
 - ACK=1; sequence number = $x+1$; next data byte expected= $y+1$
 - on receipt at B, connection is established

- B can refuse the request by replying with RST set
- SYN messages can specify options e.g. maximum segment size, scaling etc.
- initial sequence number has to be different for every connection request
 - » three-way handshake procedure ensures both agree on their initial values
- if the same initial sequence number was used for each connection, a delayed segment could be accepted in error:



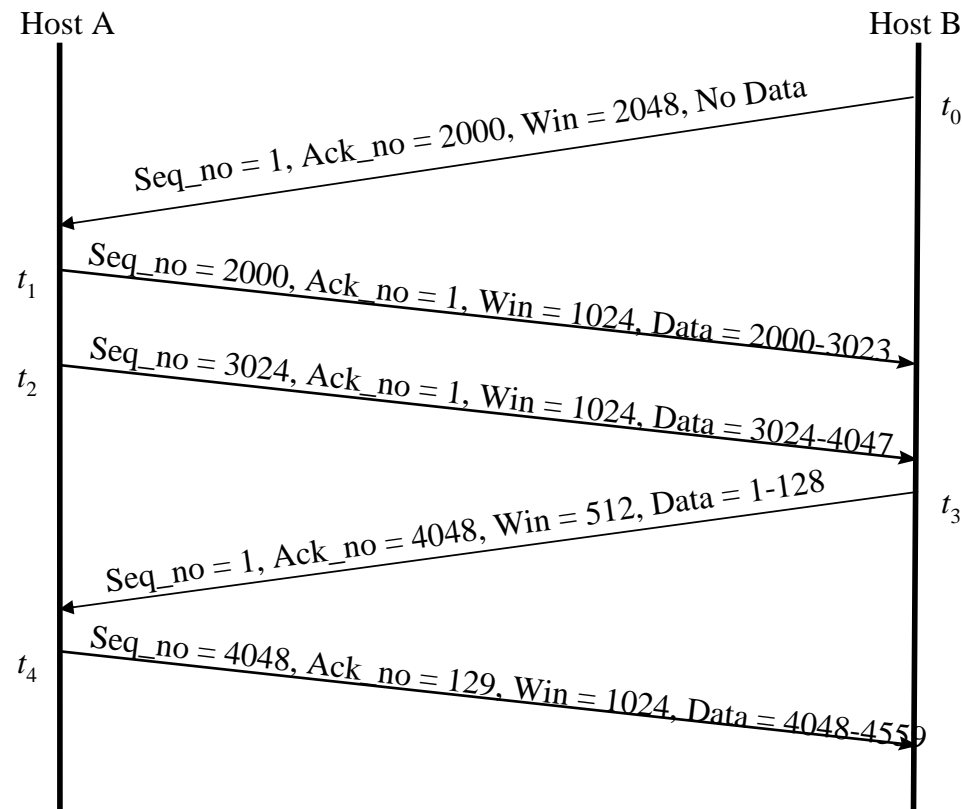
- if no response to a SYN after a time-out period, connection attempt abandoned
 - » default 75 seconds

– a client-server application using Berkeley Sockets:



- » server B carries out a *passive open* (*socket, bind, listen, accept*)
 - tells local TCP that it is ready to accept connections
- » client A carries out an *active open* (*socket, connect*)
 - tells local TCP to initiate a connection
- » followed by blocking (normally) *reads* and *writes*

– Data transfer example:



» at time t₀ :

- B passes next data byte expected sequence number 1 to A
- advertises a window size of 2048 bytes
- acknowledges all bytes less than sequence number 2000
- no data

- » at time t_1 :
 - A only has 1024 bytes to transmit - sequence numbers 2000 to 3023
 - B chooses to delay sending an ACK
 - perhaps waiting for a return segment
- » at time t_2 :
 - A sends next 1024 bytes - sequence numbers 3024 to 4047
 - A's sending window is now closed
- » at time t_3 :
 - B has 128 bytes to send from sequence number 1 onwards
 - piggybacks an acknowledgment for the whole 2048 bytes that A sent
 - and advertises a reduced receive window size of 512 bytes
- » at time t_4 :
 - A sends bytes 4048 to 4559 to fill B's receive window
 - acknowledges B's 128 bytes with a next byte expected sequence number 129
 - and advertises its window to be 1024 bytes

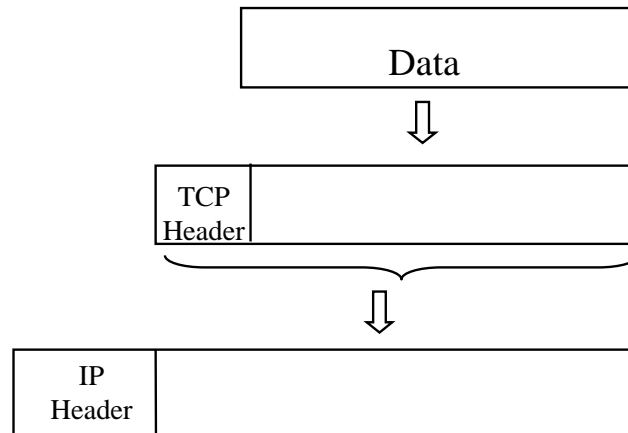
– window size advertising dynamically controls the flow of data

- Efficiency :
 - e.g. possible remote interactive character input scenario:
 - » user types one character at a time at the keyboard
 - » TCP module sends a segment containing just one byte of data
 - » destination acknowledges receipt of the character in a segment
 - » application then echoes the character in a segment
 - » user acknowledges echoed character in a segment
 - four minimal segments for each character typed!
 - » wasting bandwidth with segment header overheads
 - solution (Nagle's algorithm) :
 - » user TCP transmits first character in a segment
 - » waits until it receives the ACK before sending any more characters
 - meanwhile buffers any more characters typed in
 - » when ACK arrives, transmits *all* the buffered characters in a *single* segment
 - on a (typically) underused LAN, ACK comes back *immediately*, so *no* delay
 - on a busy connection, ACK may be delayed somewhat, so next transmission sends several characters and *saves* extra segments being sent
 - *self-adjusting* to network load

- e.g. a sender with a large volume of data to transmit
 - » but receiver application can only deal with a few bytes at a time
 - » when the receive buffer gets full, receiver only able to remove a few bytes
 - » receiver's TCP sends an acknowledgment with a *small advertised window*
 - » transmitter sends *small amount* of data to fill the small receive window
 - » receiver takes next small number of bytes of data
 - » and advertises a small window again
 - » etc. etc.
- a succession of small data segments is sent
 - » wasting bandwidth with segment overhead
- this is the *silly window syndrome*
- avoided by having the receiver not advertise the window until it is a reasonable size
 - » e.g. *at least half* the receive buffer size, or *maximum* segment size
- sender can also cooperate by *not sending* small segments

– overheads versus maximum segment size

» data to be transmitted is encapsulated in a TCP segment and then in an IP packet:

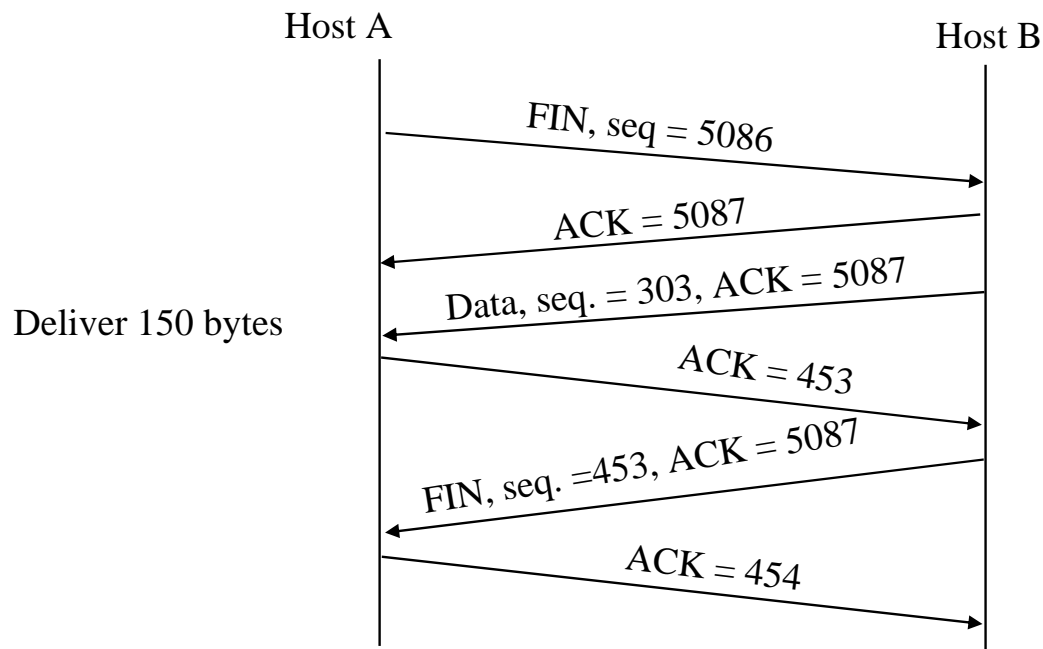


- » 20 bytes TCP header + 20 bytes IP header
- » *default* maximum segment size in TCP is 536 bytes
 - i.e. an IP packet of 576 bytes = 7% overhead
- » maximum TCP segment length is 65535 bytes (16 bits)
 - data length of 65495 = 0.06% overhead
- » but physical links impose smaller limits anyway
 - e.g. ethernet 1500 byte packets = 1460 byte TCP data

- Sequence number wraparound and timestamps
 - for 32-bit sequence numbers, maximum allowable window size is 2^{31}
 - » for a T-1 line at 1.5Mbps, wraparound time = 6 hours
 - » for a T-3 line at 45Mbps, wraparound time = 12 minutes
 - » for a SONET OC-48 at 2.5Gbps, wraparound time = 14 seconds!
 - a 32-bit timestamp can optionally be inserted in a TCP header
 - » receiving TCP echoes the timestamp in its acknowledgment
 - combination of timestamp and sequence number effectively gives 64 bits of sequence number
 - to be effective, timestamp should increment at least once every 2^{31} bytes sent
 - » before wraparound happens
 - » this puts a lower bound on the clock frequency
 - clock should not be too fast
 - » must not wraparound in less than a maximum sized segment time
 - frequencies in the range 1ms to 1 second satisfy these bound requirements
 - » 1ms works up to 8Tbps and timestamp wraps around in 25 days!

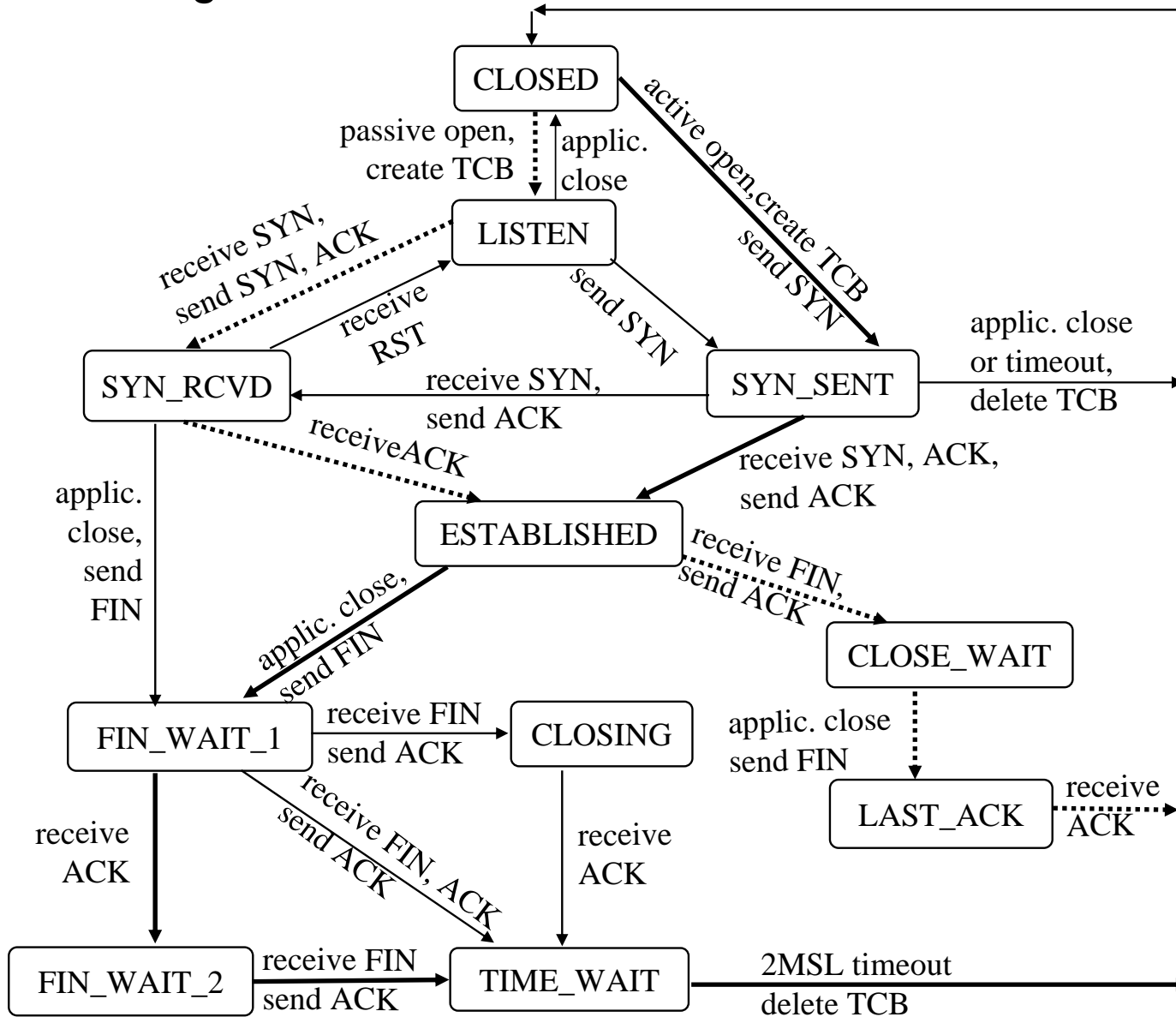
- Connection termination :
 - each end of a TCP connection terminates independently
 - » when an application has no more data to send
 - the transmitter completes the transmission of its data
 - when it receives the ACK for the last data, it sends a segment with FIN set
 - upon receiving the FIN segment, the receiving TCP entity informs its application that the other entity has terminated its connection
 - » this receiver can still transmit in the other direction until it has finished
 - then it also sends a FIN segment and waits for the ACK
 - when this ACK has been sent, the sender goes into a *wait state*
 - » and starts a TIME-WAIT timer set to $2 \times \text{maximum segment lifetime}$
 - first MSL accounts for time a segment in one direction can remain in the network
 - second MSL accounts for time a reply in other direction can be in the network
 - maximum segment lifetime (MSL) = 2 minutes by default
 - the only valid segment that can arrive in this interval is a retransmission of the FIN segment
 - » e.g. if the ACK was lost

- if a FIN retransmission is detected, the ACK is retransmitted
 - » and the timer restarted
- when the timer expires, the connection is closed
- this protects future incarnations of this connection from getting delayed segments
- example:



- Aborting connections
 - an application can terminate a connection by issuing an abort
 - » which causes its TCP entity to send a segment with RST set
 - RST segments also sent when inappropriate segments are received
 - » e.g. a segment received when the application on that port not listening
 - » e.g. a half-open connection after some error has occurred
 - the application in one half of the connection might have crashed
 - but the transmitter may not realise this
 - so when the receiving TCP entity receives a segment, it returns an RST segment to close the connection
- Keep Alive
 - TCP connections in principle can stay established indefinitely
 - » in practice, a time-out of 2 hours is the default
 - to keep a connection alive, a *probe* segment can be sent
 - » or just to check that the connection is still intended
 - » a dummy segment with an old sequence number
 - not an official part of the TCP spec but widely implemented
 - » must be set by default to *off*

- the TCP State Diagram:



Dynamic Host Configuration Protocol (DHCP)

- provides a method for dynamically assigning temporary IP addresses
 - » also allows clients to acquire IP configuration parameters they need
 - e.g. subnet mask, broadcast IP address, default router etc.
- an ISP allocates IP addresses from its assigned set on a per-use basis
 - » IP addresses *leased* for a time from a limited pool
 - » address can be re-used for another client at a later time
- when a host wants to obtain an IP address, it broadcasts a *DHCP Discover* message on its physical network
- any server on the network willing to do so may offer an IP address and other configuration with a *DCHP Offer* message
- requesting host chooses one of the replying servers and broadcasts a *DHCP Request* message that includes the IP address of the chosen server
- selected server allocates the offered IP address and sends a *DHCP ACK* to the host for some *lease period, t*
 - » *t* chosen at the ISP's convenience

- the host may decline the offer if it wishes with a *DHCP Decline* message
 - » e.g. it doesn't like the configuration parameters offered
- ACK message also holds two other times :
 - » t_1 : period after which the host should try to extend the lease period if it wishes
 - $t_1 = 0.5 \times t$ usually
 - by sending another DHCP Request to the same server as before
 - » if no ACK has been received by time t_2 , host broadcasts a DHCP Request to all the servers
 - $t_2 = 0.875 t$ usually
 - » if no ACK by time t , host must relinquish the IP address and make a new request from scratch
- a host can relinquish the IP address before its lease runs out by issuing a *DHCP Release* message to the server

Internet Routing Protocols

- the Internet is a collection of *Autonomous Systems* (AS)
 - sets of routers and networks each administered by a separate organisation
 - » e.g. corporate networks, ISPs, campus networks
 - AS's form the top level of hierarchy of the Internet
 - AS's are further sub-divided into *areas*
 - a hierarchy of routing protocols are used for the various levels:
 - » an *Exterior Gateway Protocol* (EGP) between AS's
 - » *Interior Gateway Protocols* (IGP) inside AS's
 - an EGP is used for routers to communicate paths between and through AS's
 - » the *Border Gateway Protocol* , BGP-4, is the *de facto* standard
 - » AS's use 16-bit AS numbers (ASNs) as addresses
 - separate from IP addresses
 - more than 10000 in use already – will probably run out soon
 - assigned by the same authorities as issue IP addresses

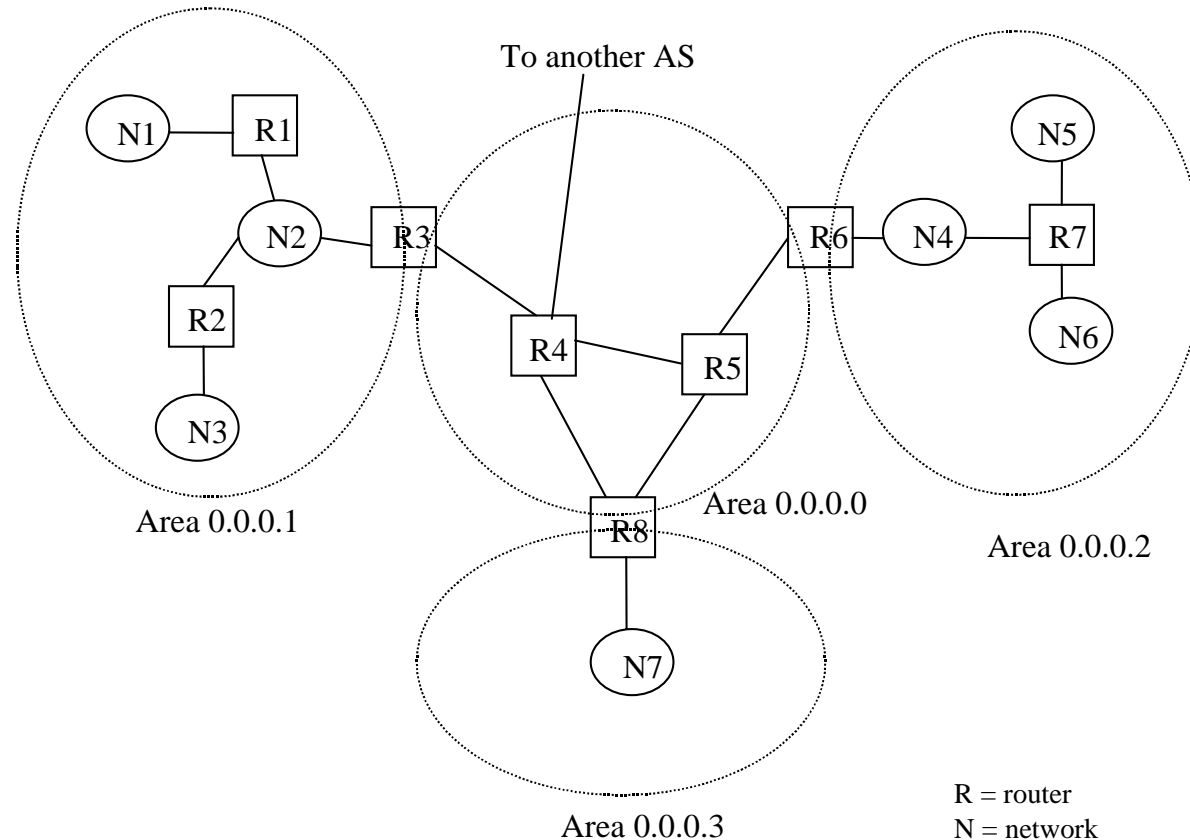
- a Stub AS : only connected to one other AS
 - » no through traffic
 - » can be treated as an extension of the parent AS it is connected to
 - » does not normally need its own ASN
- a Transit AS : has connections to more than one other AS
 - » allows itself to be used for transit traffic
 - » and needs its own ASN
 - » most large ISP's are transit AS's
- a Multihomed AS : has multiple AS connections but does not allow through traffic
- AS's are further subdivided into smaller *Areas*
 - » hierarchical division eases the problem of finding optimal routes and routing table size
 - » shortest path algorithms are applied within areas
 - » route information back-propagated through inter-area gateways as necessary
 - and routes aggregated with variable-length subnet masks when possible
 - » different routing protocols can be used in different areas and between areas
 - e.g. the Informatics network area uses OSPF internally and RIP at its borders

- the Routing Information Protocol (RIP)
 - an IGP based on the distance-vector *Bellman-Ford* algorithm
 - runs on top of UDP – port number 520
 - distance metric is the *number of hops*
 - » maximum number of hops limited to 15
 - » i.e. the diameter of the network must be no greater than 15
 - » 16 is regarded as *infinity*
 - » works fine in small network areas
 - a RIP router sends an update message to its neighbours every 30 seconds
 - » the neighbours can then recalculate their shortest paths
 - a router expects to receive an update message from each of its neighbours at least once every 180 seconds
 - » more than 30 seconds since UDP is unreliable
 - update messages may get lost
 - this deals with changes in network topology
 - » such as a transmission link failure

- if a router does not receive an update within the time limit, it assumes the link to its neighbour has failed
 - » and sets the corresponding cost to 16 (infinity)
 - » this can be replaced with a correct value if the neighbour eventually responds
- RIP uses *split horizon with poisoned reverse* to minimise routing loops
- convergence of shortest paths can be slow
 - » *triggered updates* can improve this
 - » if a router gets a new link cost it can update its neighbours immediately
 - » or when an update request is received
- RIP is not flexible in choice of metric
 - » e.g. cannot take account of network load conditions
- RIP-2 offers improvements:
 - » packets carry a variable-length subnet mask – allows CIDR to work
 - » and an authentication procedure
 - to authenticate neighbouring routers
- RIP a proven protocol and still very widely used

- Open Shortest Path First (OSPF)
 - an IGP based on link-state methods, using Dijkstra's algorithm
 - » link-state information is exchanged between routers
 - » to allow them each to build up the complete network topology
 - » then they can apply a shortest path algorithm locally to find optimal routes
 - » link state update messages, *link state advertisements* (LSAs), can be flooded into the network when the topology changes
 - designed to be better and much more flexible than RIP
 - » converges to a stable state much more quickly
 - » allows different optimal routes to be calculated for different types of service
 - metric can be based on any criterion e.g. network load
 - » variable-length subnet masks
 - » allows load balancing over paths of equal cost
 - » incorporates concept of a designated router to improve efficiency
 - one router from a fully interconnected set of routers can be designated to gather all the link state information rather than each of them doing so
 - ⌘ reduces the number of OSPF message flying around

- can be used in areas and among areas on a variety of router types:
 - » internal routers : all links connected to networks in the same area (R1, R2, R7)
 - » area border routers : links to more than one area (R3, R6, R8)
 - » backbone routers : links to a backbone of routers connecting areas (R3,4,5,6,8)
 - » autonomous system boundary routers : links to another AS (R4)

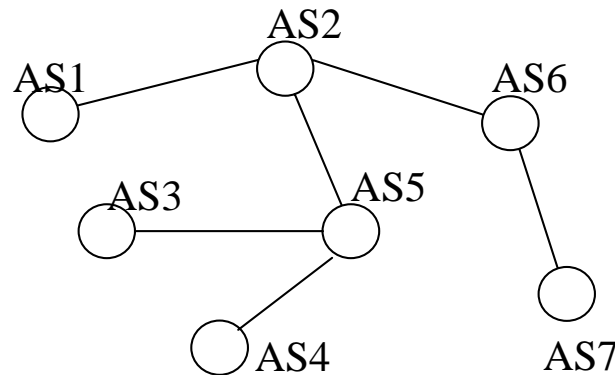


- the aim of OSPF is to build up and maintain a *link state database* of the network
 - » an extremely complex system in detail
- phase 1 : neighbours are discovered using the *Hello* protocol
 - » each OSPF periodically broadcasts a hello packet onto each of its interfaces
 - every 10 seconds typically
 - » when a router receives a hello packet, it replies with a hello packet containing the ID of each neighbouring router it knows about
 - » when the originating router sees a hello packet with its own ID in it, it knows there is a working *bidirectional* link between it and the sending router
 - designated routers and backup designated routers are elected at this stage
- phase 2 : neighbours exchange database description packets
 - » in order to synchronise their databases
 - » containing multiple LSA *headers* for LSAs that are in their own databases
 - » only headers are exchanged
 - a neighbour can then request the full LSA database entry if it does not have it or it is out-of-date

- phase 3 : maintenance and updating the database
 - » a router wanting to update its link state database can send a *link state request* to its neighbour
 - » the neighbour will reply with the new information
 - » when a router has a change to propagate, it can update its neighbours with a *link state update* packet containing LSAs
 - » when a router receives an update packet
 - it sends an acknowledgment
 - updates its database
 - and forwards the LSAs to all its neighbours except the one it came from
 - a *flooding* technique which will propagate the new LSA to all the network routers
 - » OSPF routers are obliged to refresh their LSAs every 30 minutes

- the Border Gateway Protocol

- allows two different AS's to exchange routing information
 - » so that IP traffic can flow across the border
- each BGP router establishes TCP connections with other BGP routers
 - » and pass messages detailing new and out-of-date routes
- BGP messages pass reachability information
 - » containing a sequence of AS's that packets must traverse to reach a destination network
 - or group of networks with a common prefix
- BGP routers use the information exchanged to build up a graph of AS's:



- BGP routers can enforce routing policies
 - » e.g. about what type of information is allowed to cross certain boundaries