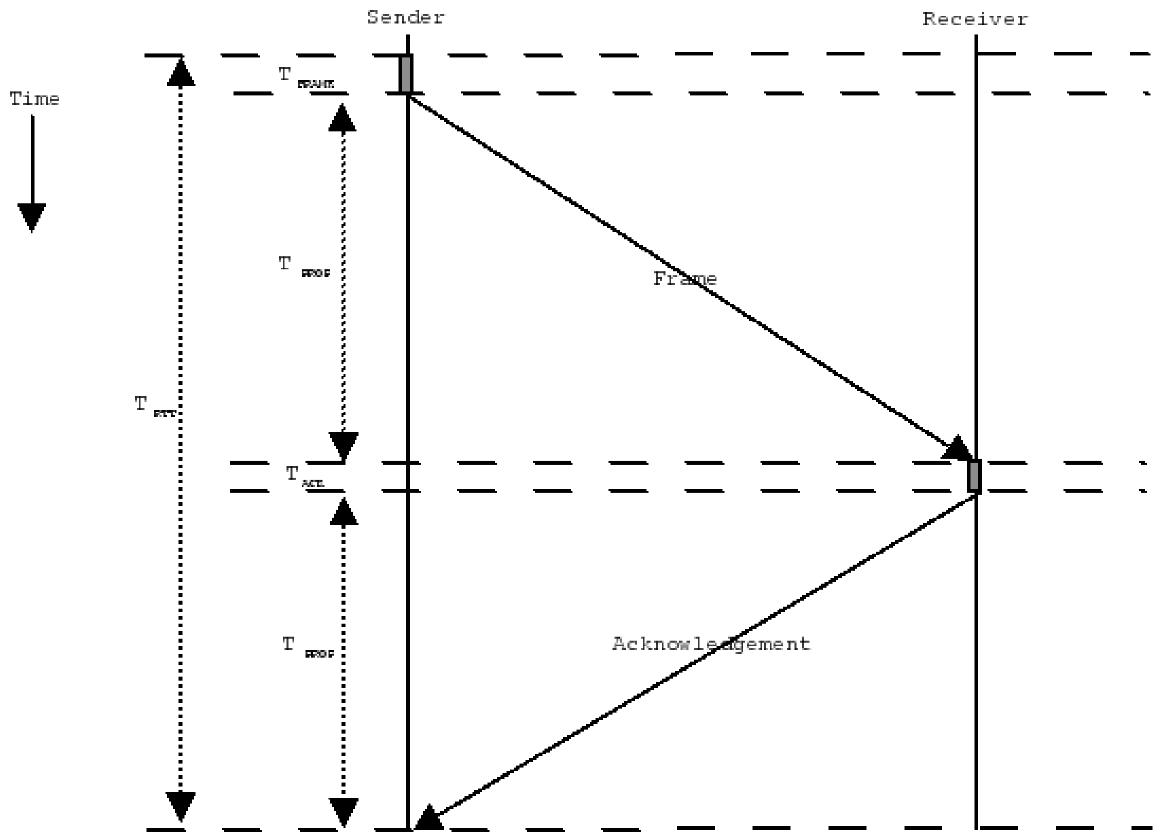
COMP3721 Week Eight Lab Synopsis

ARQ Protocols

- assuming we have an error detection mechanism, we need a protocol that allows for retransmission
- the general strategy used is for
  - the transmitter to number the frames, and
  - the receiver to number acknowledgments that let the transmitter know the frame was received
    - the common mechanism for numbering acknowledgments is to send ACK_{n+1} to state that $Frame_n$ was received and $Frame_{n+1}$ is the next frame expected
- the transmitter keeps unacknowledged frames in a buffer in case they need to be retransmitted
  - then, the more frames that can be transmitted before waiting for an acknowledgment, the more buffer space required and the greater the complexity of implementing the protocol
- the transmitter will retransmit in two cases
  - no acknowledgment is received. This is called a timeout. The transmitter expects a response in a certain period of time and sets a timer when transmitting the packet.
  - a negative acknowledgment is received. In some cases, the receiver upon detecting an error will send a negative acknowledgment to indicate a problem has occurred. Alternately, depending on the specific protocol, the receiver may rely on the transmitter to timeout and retransmit.

Stop and Wait

- probably the simplest possible ARQ protocol
  - the transmitter sends one frame and then waits for an ACK
  - the receiver is only ever expecting one specific frame (the next one in sequence)
  - simple to implement, can be efficient given the right scenario
- The key to understanding more complex ARQ protocols lies in first understanding stop-and-wait.



- Concentrate on understand what happens in one round-trip – that is what happens from starting to send a packet until an acknowledgment is fully received
  - The pattern established in one round-trip will repeat over and over – if we understand one round-trip, then we understand the overall behaviour of the protocol
- We need to know and/or calculate the following:

1. T_{frame} – the time to place a frame into the channel.

$$T_{frame} = \frac{PacketSize(bits)}{DTR(bps)}$$

2. T_{prop} – the time for a bit to propagate across the channel

$$T_{prp} = \frac{Distance(m)}{PropagationRate(m/s)}$$

3. T_{ack} – the time for the receiver to place the ACK into the channel. There are two common cases:

- ACKs are negligible – that is the number of bits involved in placing the acknowledgment into the channel is so small that it can be ignored.

$$T_{ack}=0$$

- ACKs are piggybacked – that is the acknowledgment is carried as part of the header for a packet traveling from receiver back to the transmitter. It is assumed that the frame in the opposite direction is the same size as the frame in the Tx->Rx direction, therefore:

$$T_{ack}=T_{frame}$$

4. T_{rtt} – the round-trip time, which is just the following sum:

$$T_{rtt}=T_{frame}+T_{prp}+T_{ack}+T_{prp}=T_{frame}+T_{ack}+2\mathrm{T}_{prp}$$

- With these pieces in place, we can calculate an important measure – efficiency.
  - assume we have a channel with a DTR of *k* bps, (i.e. *k* = 56 kbps).
  - we will generally not achieve *k* kbps because we have to stop transmitting at times to wait for an acknowledgment
  - we can measure efficiency as the percentage of time we actual spend transmitting information from the transmitter to the receiver

$$Efficiency=\frac{TimeTransmitting}{TotalTime}$$

  - from our work above, we know that the time transmitting is just T_{frame} and total time is T_{rtt}. Then,

$$Efficiency=\frac{T_{frame}}{T_{rtt}}$$

  - Example: if T_{frame} = 1ms and T_{rtt} = 5 ms, then

$$Efficiency=\frac{1\mathrm{ms}}{5\mathrm{ms}}=0.20$$

  The system only transmits 20% of the time.

- Stop and wait can be reasonably efficient. See practice problem 1.

Sliding Windows

- A generalization of stop-and-wait – instead of stopping after 1 frame, the transmitter must stop (and wait for an ACK) after *X* frames.
- X above is the sending window size (SWS). That is the transmitter has a window of SWS frames that it may send before must wait.
- Now the time spent transmitting (during one roundtrip) is SWS * T_{frame}; then efficiency is

$$Efficiency = \frac{SWS * T_{frame}}{T_{rtt}}$$

  - Note 1: we can use this same equation for stop-and-wait. In that case SWS = 1
  - Note 2: efficiency obviously cannot exceed 100%. This should be understandable if you clearly understand the model underlying the above calculation. Perhaps a more correct statement of the above though would be:

$$Efficiency = min(100\ percent, \frac{SWS * T_{frame}}{T_{rtt}})$$

- The above efficiency equation can be rearranged to solve for an appropriate SWS if necessary

$$SWS = Efficiency * \frac{T_{rtt}}{T_{frame}}$$

- In the common case of trying to achieve 100% efficiency, the above simply becomes:

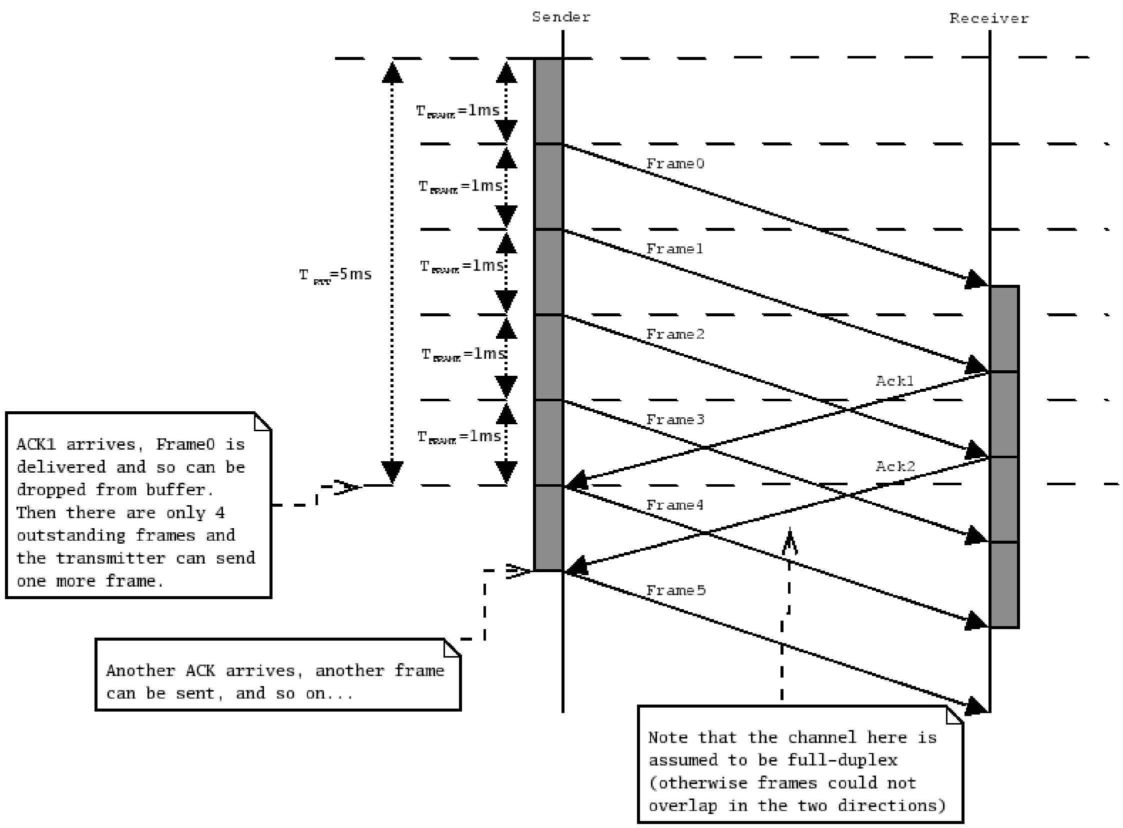
$$SWS = \frac{T_{rtt}}{T_{frame}}$$

- Example: if T_{frame} = 1ms and T_{rtt} = 5ms, suggest a SWS

$$SWS = \frac{5\text{ms}}{1\text{ms}} = 5\ frames$$

  - This should make sense – in the stop-and-wait scenario, the system essential sits still waiting for 4ms
  - Instead, here we allow the transmitter to continue sending during those 4ms
  - So in total, the transmitter sends for 5ms out of the 5ms it takes before an acknowledgment arrives

- when that acknowledgment arrives, the first frame is no longer outstanding, so the transmitter can then send another frame
- ACKs for the various packets should be arriving in constant succession allowing the send to constantly continue transmitting



Retransmission

- up until now, the assumption is that no frames are lost or in error
- when errors do occur, there are two re-transmission strategies: go-back-n and selective repeat

Go-Back-N

- the receiving window size (RWS) = 1.
  - that is, the receiver is only ever willing to accept the next frame in order
  - so out-of-order frames are not buffered (this can happen in a packet-switched channel or if a frame is totally destroyed)
  - in the case of an error, the transmitter must retransmit from the frame in error forward (as any successive frames, if received,

would have been discarded by the receiver).

Selective Repeat

- the receiving window size (RWS) = SWS.
  - that is, the receiver is willing to accept any frame the transmitter can validly transmit
  - out-of-order frames are buffered (complex to implement properly)
  - the transmitter need only retransmit the frame in error in the case of corruption or a lost frame

Sequence Space

- the frame and acknowledgment numbers are always represented by a fixed number of bits in the frame header
- we call that number *n* – it is the *sequence space*, or number of sequence bits
  - then there are only 2^n unique numbers
  - so frame numbers and acknowledgment numbers will be re-used at some point in time
- where errors can occur, it is absolutely necessary that the sender and receiver are clear about the frame in error
  - if the sequence space is too small, different frames with the same sequence number (because numbers will be re-used) may be mistaken for one another
  - so then a relationship exists between the SWS and the sequence space (n)
    - in general the relationship (to guarantee correct behaviour) is

      $SWS + RWS \leq 2^n$

    - since the RWS is known for go-back-n and selective repeat, we can derive specific instance:
      - with Go-Back-N:

        $SWS \leq 2^n - 1$, or alternately
        $n \geq \log_2(SWS + 1)$

      - with Selective Repeat:

        $SWS \leq 2^{n-1}$, or alternately

$n \geq \log_2(SWS) + 1$

Working in Bits

- we have worked using time as the primary unit all the way through the above case
- we could alternately work in bits; for example, with stop-and-wait:

$$Efficiency = \frac{BitsTransmitted}{PotentialBits} = \frac{T_{frame} * DTR}{T_{rtt} * DTR} = \frac{FrameSize}{BandwidthDelayProduct}$$

  - The above calculation is obviously equivalent to our earlier calculation, we've just multiplied by DTR/DTR.
  - Note that we get something we understand in the numerator, the frame size
  - The denominator however is something new – the Bandwidth-Delay product.
    - this is just the number of bits that will fit into the channel in a period of time (where we're interested specifically in the RTT period here)

Practice Problems

1. Describe a scenario where stop-and-wait would be efficient.
2. Frames of 1000 bits are sent over a 1 Mbps satellite channel. The one-way end-to-end delay associated with the satellite link is 270ms. Acknowledgments are always piggy-backed onto data frames. The headers are very short. Three-bit sequence numbers are used. With is the maximum achievable channel utilization for
    1. stop and wait
    2. sliding windows with Go-Back-N ARQ
    3. sliding windows with Selective Repeat
3. A 100km long cable runs at the T1 data rate. The propagation speed of the cable is 2/3 the speed of light. How many bits fit in the cable?
4. Calculate the efficiency for a stop-and-wait system on a 5km cable. Data is transferred at 10Mbps, frames are 1500 bytes in size and acknowledgments are negligible in size.
5. Consider a 10Mbps sliding window system using fixed-sized frames and a SWS of 10. The bandwidth-delay product is 1Mb. Determine the minimum frame size that would yield 100% channel utilization.