Data Link Control

- So far we have explored the transmission of signals with encoded data over a basic *transmission link.*
- For effective communications much more is needed to control and manage the exchange of ***Data***.
- A layer of *control logic* is required above the physical layer in each Source/Destination device:
  - This logic is referred to as ***Data Link Control***,
  - When used it transforms a basic *Transmission Link* into a fully functioning ***Data Communications Channel***.

Data Link Control

- This *Logic* is implemented as a set of *Rules* which are well defined:
  - The definition of the Logic/Rules are known as a ***<u>Protocol</u>***.
- Protocols are *synonymous* (closely linked to) with Data Communications:
  - They are often referred to in texts on the subject,
  - The first protocol to be explored is known as the ***<u>Data Link Control Protocol</u>***.

Data Link Control Requirements

- There are six basic requirements for effective Data Link Control as follows:
  - ***Frame Synchronisation.*** Frames must be recognizable by the Receiver. Already covered.
  - ***Flow Control.*** The Sender must not overload the Receiver,
  - ***Error Control.*** Errors should be detectable by the Receiver,
  - ***Addressing.*** For a multi-point configuration each station must be uniquely identifiable,

Data Link Control Requirements

- ***Control and Data on same link****.* The Receiver must not have to wait for control information to arrive before it can process a frame,
- ***Link Management****.* Procedures for establishing and relinquishing the link must be adhered to.

Flow Control

- Receiver and Sender stations each allocate a data *buffer* of a fixed size:
  - Data remains in the buffer until it has been processed by the Receiving station,
  - Consequently the buffers can fill to capacity.
- *Flow control* enables the Receiving station to indicate to the Sending station that its buffers are full.

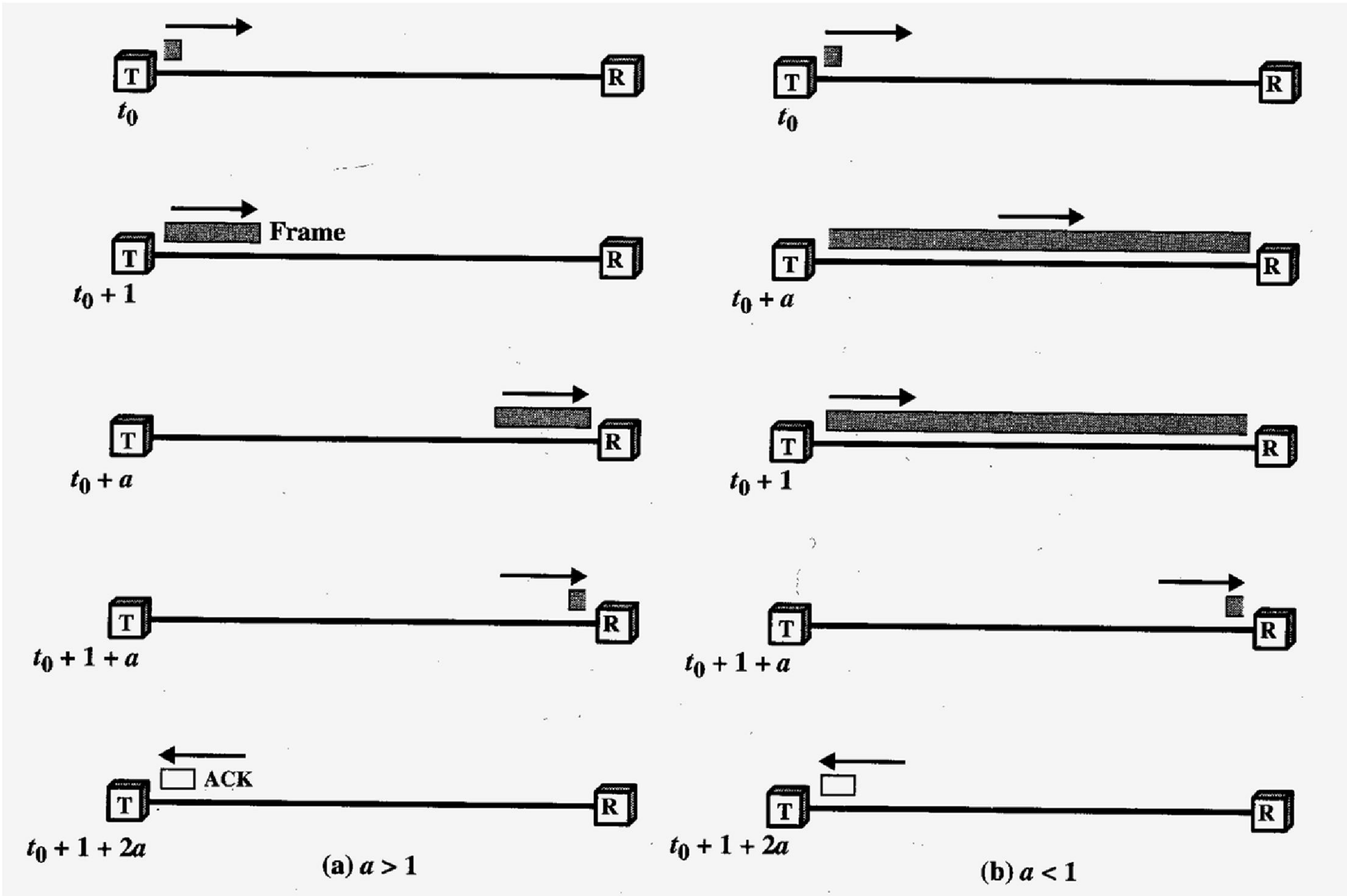
Flow Control

- There are <u>two</u> *flow control* techniques to consider:
  - *Stop-and-Wait,*
  - *Sliding Windows.*

Stop-and-Wait Flow Control

- Here the Sending station transmits a frame:
  - The Sending station must get an ***acknowledgement*** (an ***ACK***) from the Receiving station before transmitting the next frame,
  - The Receiving station can control the flow of data by *withholding* **ACKs**.
- This technique only allows for the transmission of a single frame in either direction i.e. ***Half Duplex*** communications.

Stop and Wait Flow Control



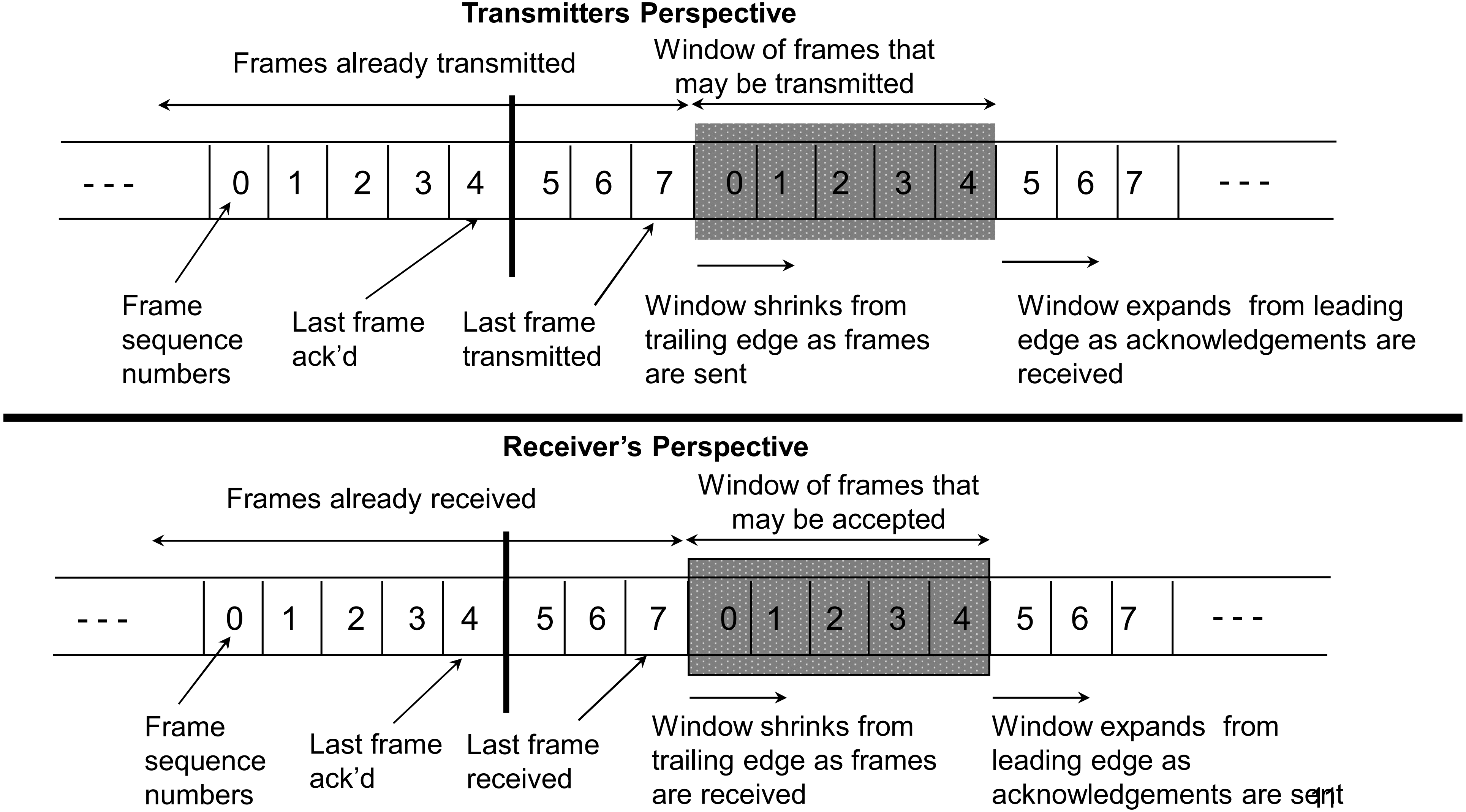
Sliding Window Flow Control

- Here both stations use an <u>extended</u> buffer size to hold multiple frames.
- The Sending and Receiving stations maintain a list of frames already sent/received.
- This technique allows for <u>more</u> *efficient link utilization:*
  - The transmission link is effectively treated as a *pipeline* that can be *filled* with many frames in transit *simultaneously.*

Sliding Window Flow Control

- This technique allows *multiple* frames to be in transit simultaneously in either direction i.e. Full Duplex communications.
- The next slide shows an example of the technique.

Sliding Window Flow Control



Sliding Window Flow Control

- In the example Stations A and B each allocate buffer space for *W* frames:
  - i.e. Station B can *accept W* frames and Station A can *send W* frames without any acknowledgement being sent or received.
- Each frame contains a *sequence number*.
- Station B sends *ACKs* that include the sequence number of the *next* frame expected:
  - i.e. Station B is prepared to receive the frame starting at the *sequence number* indicated e.g. RR5.

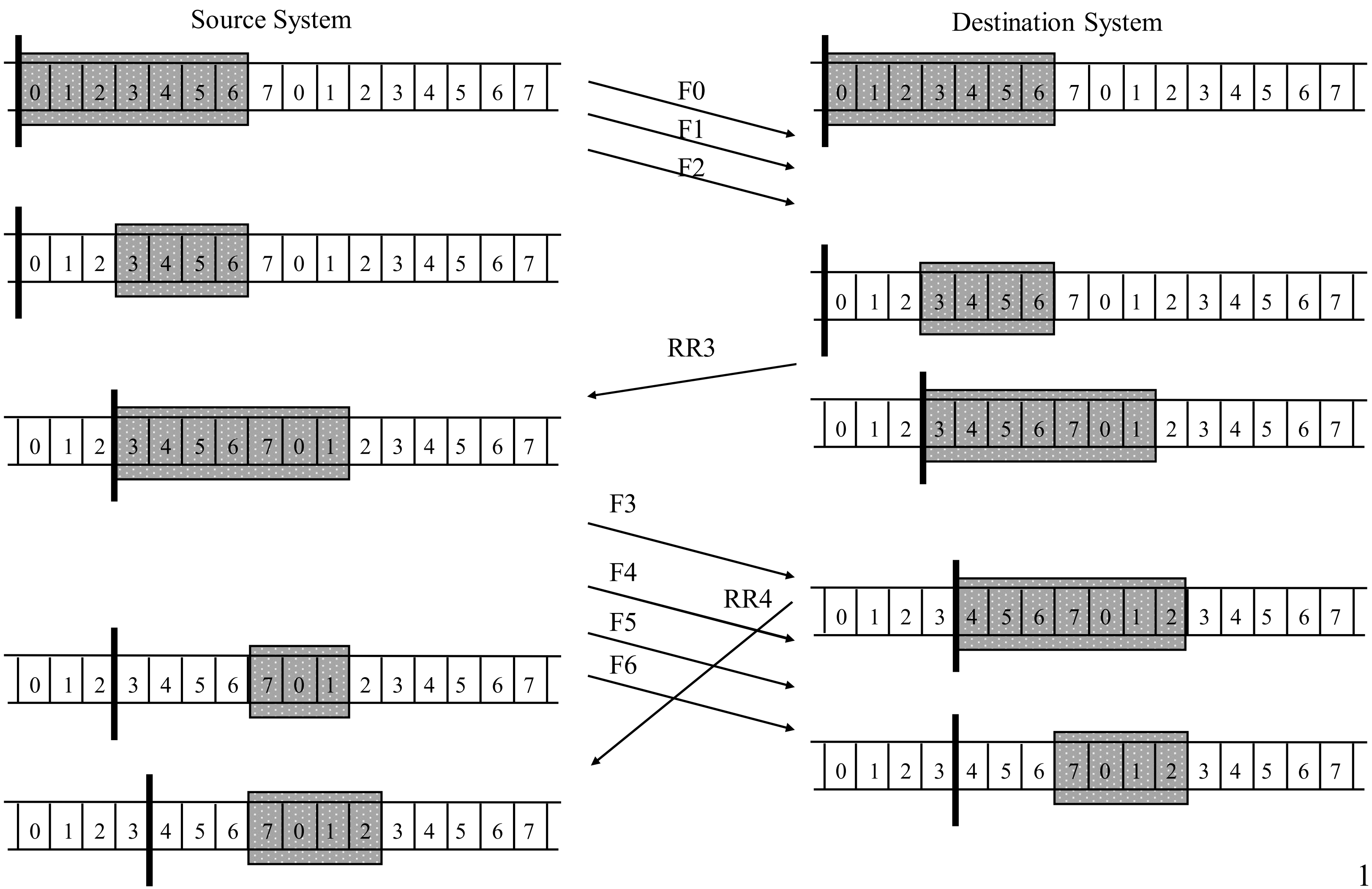
Sliding Window Flow Control

- *Multiple* frames can be *ACK'd* using a single **control message** (*implicit acknowledgement*)
  - e.g. Receipt of ACK for frame **2** (RR3) followed later by ACK for frame **5** (RR6) *implies* acknowledgement of frames **3** and **4.**
- Station A maintains a list of frame numbers it is allowed to *send.*
- Station B maintains a list of frame numbers it is prepared to receive.
- These lists can be considered as *windows.*

Sliding Window Flow Control

- To impose flow control the Receiver can send a different control message:
  - *Receiver Not Ready* (RNR),
  - This acknowledges previous frames but stems the flow from the Sender e.g. RNR5 acknowledges all frames up to frame 4.
- To resume flow the Receiver sends a *Receiver Ready* (RR) message
  - e.g. RR5 – recommence at frame 5

Example Sliding Window



Sliding Window Flow Control

- In the previous example the operation of Sliding Windows can be clearly seen:
  - Frames 0-2 are transmitted,
  - Copies of these frames are held in memory on both the Sender (Tx) and Receiver (Rx) station until an ACK is returned,
  - When the Rx sends the ACK it removes the frames from memory and the Sliding Window expands,
  - When the Tx receives the ACK it removes the frames from memory and the Sliding Window expands.

Sliding Window Flow Control

- Note on ACKs:
  - Here an ACK is shown as a special message, (RR3, or Receiver Ready, 3),
  - Both RR and RNR messages are **positive** ACKs. They are acknowledging that a successful transmission has occurred,
  - Only one ACK is required for three frames. RR3 acknowledges all frames up to and including frame 2 i.e. all three frames: 0, 1 and 2,
  - This is one of the efficiencies of the technique. Recall that *S-and-W* required one ACK per transmission.

Sliding Window Flow Control

- This approach to sending one ACK for multiple frames allows for **Full Duplex** communications:
  - ACKs can be returned anytime i.e. after receipt of a single frame or multiple frames.
- This Full Duplex mode can be seen in the later transmission of Frames 3-6:
  - Frame 3 has arrived and an ACK (RR4) is immediately returned. The ACK can clearly be seen in transit at the <u>same time</u> as Frames 4, 5 and 6 are arriving,

Sliding Window Flow Control

- The consequence of this is that Frame 3 is held in memory very briefly and then it is removed. This is too quick for it to be seen stored in memory:
- Frames 4, 5 and 6 arrive soon after Frame 3; these frames can be clearly be seen stored in memory,

• Question: Assuming Frames 4, 5 and 6 arrive without incident, what would the next **positive** ACK be?

Sliding Window Flow Control

- Answer: The next <u>**positive**</u> ACK could potentially be one of two messages depending on the status of the Rx buffers:
  - If, upon receipt of Frame 6, the Rx buffers become <u>**full**</u> then it could be **RNR7**. This message would acknowledge receipt of all frames up to and including Frame 6 but it would also instruct the Tx to <u>stop</u> sending frames,
  - If, upon receipt of Frame 6, there is still space in the Rx buffers (which is the case here), then it would be **RR7**. Once again this message would acknowledge receipt of all frames up to and including Frame 6 but it would also permit the Tx to <u>continue</u> sending frames.

Sliding Window Flow Control

- With the introduction of Full Duplex mode each station can be a Tx'er and a Rx'er:
  - Full Duplex mode allows for frames to be in transit between the stations simultaneously and in opposite directions. This includes **Data** frames,
- With the capability to transmit Data frames in opposite directions it is possible to introduce another efficiency:
  - ACKs don't have to be sent as separate frames.

Sliding Window Flow Control

- Instead, ACKs can be included in outgoing **<u>Data</u>** frames:
  - i.e. Data frames moving in one direction can carry ACKs in their Control fields for Data frames moving in the opposite direction,
  - This technique is known as *piggybacking* i.e. a <u>single</u> frame can carry both *Data* <u>and</u> *ACK*s.
- Consequently, in *Full Duplex* mode each station must maintain <u>two</u> sliding windows:
  - One for *Transmitted* frames <u>and</u> one for *Received* frames.

Sliding Window Versus *Stop-and-Wait* Flow Control

- *Sliding Windows* is potentially more efficient than *Stop-and-Wait in relation to **Link Utilization***:
  - With Stop-and-Wait only one frame can be in transit at any time i.e. Half Duplex communications,
  - With *Sliding Window* the link is treated like a *pipeline* that can be filled with *multiple* frames,
  - This allows for simultaneous two-way communications i.e. Full Duplex communications.

Sliding Window Versus *Stop-and-Wait* Flow Control

- *Sliding Windows* uses a special message, RNR, to impose Flow Control:
  - This RNR message is a <u>positive</u> ACK which allows the Tx to clear frames from its outgoing buffer,
  - Recall that *Stop-and-Wait* simply with-held the ACK. The Tx must hold onto the frame until the ACK arrives.