CSE and EE 461
Lecture 9

Spring Quarter 1998

Goals of a Data Link Protocol

- **Flow Control** - Sender and Receiver coordinate so that neither sends more data than can be handled by the other
- **Error Contol** - Sender and Receiver coordinate so that if frames are lost then frames are resent properly. We want no duplicate data, out of order data, or missing data passed on to the network layer.
- **Good Utilization of Bandwidth** - As the Sender and Receiver coordinate they should manage to utilize the bandwidth to a maximum.

Basic Simplex Protocol

- Sender - repeat forever
  - get data from network layer
  - create frame
  - send frame to physical layer
- Receiver - repeat forever
  - wait for event {framearrival}
  - remove data from frame
  - put data to network layer

Problem:

Receiver may be overwhelmed.

(Assume an errorless physical layer)

Stop-and-Wait Protocol

- Sender - repeat forever
  - get data from network layer
  - create frame
  - send frame to physical layer
  - wait for event {ackarrival}
- Receiver - repeat forever
  - wait for event {framearrival}
  - remove data from frame
  - put data to network layer
  - send ack frame to physical layer

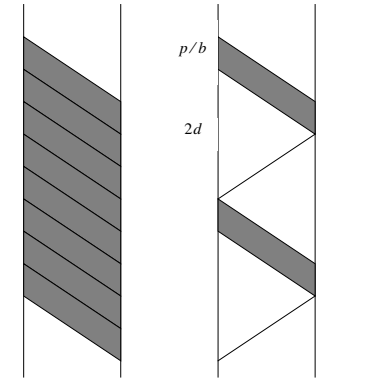
Problem solved:

Receiver nor Sender can be overwhelmed.

New Problem:

physical layer may be under-utilized

Under-Utilization



Simlex Protocol Stop-and Wait Protocol

$$U = \frac{p/b}{p/b + 2d} = \frac{1}{1 + \frac{2db}{p}}$$

Example:
d = .001 sec.
b = 1 Mb/sec
p = 1,000 bits

U = 1/3

What about Errors

- If Simplex is modified to throw away bad frames then there will be gaps in data put to the network layer.
- If Stop-and-Wait is modified to throw away bad packets then either

  1. It hangs if no ack is sent

  2. There are gaps in the data put to the network layer if acks are always sent.

Retransmission

- Sender - repeat forever

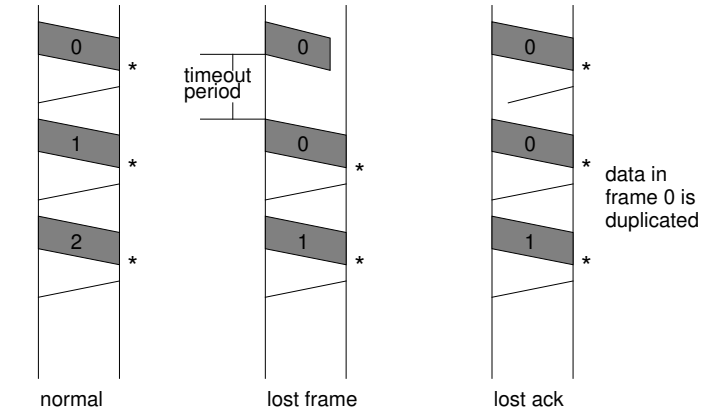
```
get data from network layer
create frame using error detection
done := false
repeat
   send frame to physical layer
   set timer
   wait for event {ackarrival, frameerror, timeout}
      if event = ackarrival then done := true  endif
until done
endrepeat
```


Receiver

- Receiver - repeat forever

```
wait for event {framearrival, frameerror}
   if event = framearrival then
      remove data from frame and put to network layer
      send ack to physical layer
   endif
endrepeat
```

Retransmission Executions



Positive Ack with Retransmission (PAR)

- Sender

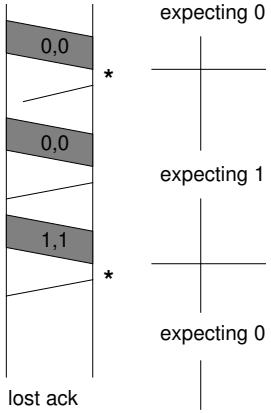
```
seq_no := 0
repeat forever
  get data from network layer
  create frame with seq_no field
  done := false
  repeat
    send frame to physical layer
    set timer
    wait for event {ackarrival, frameerror, timeout}
    if event = ackarrival then done := true  endif
  until done
  seq_no := 1 - seq_no
endrepeat
```


Receiver

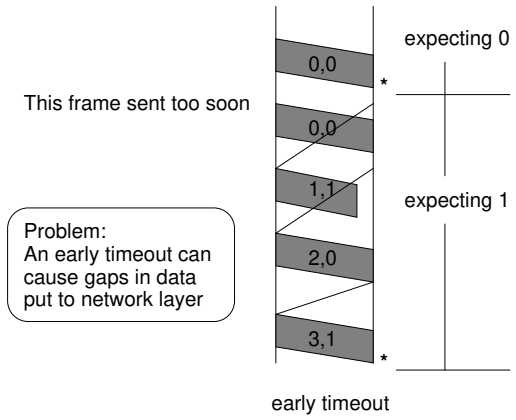
- Receiver

```
expected_seq_no = 0
repeat forever
  wait for event {framearrival, frameerror}
    if event = framearrival then
      if expected_seq_no = seq_no in frame  then
        remove data from frame and put to network layer
        expected_seq_no := 1 - expected_seq_no
      endif
    send ack to physical layer
    endif
endrepeat
```


PAR Execution 1

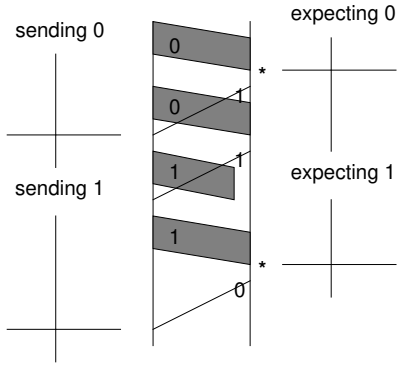


PAR Fails on a Duplex Channel



Fixing PAR

- Put seq_no into the acknowledgements. Then the Sender would know exactly what is being acknowledged.
- This is the alternating bit protocol (ABP)



Alternating Bit Protocol (ABP)

(This is protocol 3 in the book)

- Sender

```
seq_no := 0
repeat forever
   get data from network layer
   create frame with seq_no field
   done := false
   repeat
      send frame to physical layer
      set timer
      wait for event {ackarrival, frameerror, timeout}
      if event = ackarrival and ack's seq_no = 1-seq_no
            then done := true  endif
   until done
   seq_no := 1 - seq_no
endrepeat
```

Note: It is often best to think of the ack's seq_no as a request for data with that sequence number.

APB Receiver

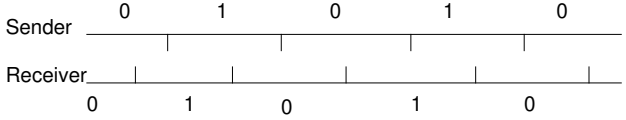
- Receiver

```
expected_seq_no = 0
repeat forever
  wait for event {framearrival, frameerror}
    if event = framearrival then
      if expected_seq_no = seq_no in frame  then
          remove data from frame and put to network layer
          expected_seq_no := 1 - expected_seq_no
       endif
    send ack with expected_seq_no included to physical layer
    endif
endrepeat
```

Note: the ack with expected_seq_no can be thought of as a request for data with that sequence number.

Properties of ABP

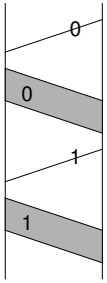
- Stages of ABP are interlocked like a zipper



- ABP can fail if the inititial sequence number are set improperly. For example, Sender starts out with sequence number = 0 and Receiver starts out with expected sequence number = 1.
- If ABP times out too early too often then unnecessary frames are sent.
- ABP may have a utilization problem - we'll fix this with sliding windows.
- ABP as written is "sender driven", that is the sender is in control and the receiver just reacts.
- ABP is a simplex protocol in that data flows only in one direction - to make a duplex protocol use "piggybacking."

Receiver Driven Protocol

- The Receiver instead of the Sender has timeout.
- The Receiver sends "request" and the Sender responds. So acknowledgements become requests.
- Receiver driven ABP



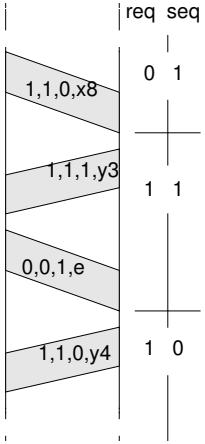
Duplex Protocol - Using Piggybacking

req number	data bit	seq number	data	CRC, etc

Typical Frame

- req number tells other party what data is expected next by me.
- data bit tells other party if there is data from me on board.
- seq. number tells other party the sequence number of the data on board
- data, if any, is data for the other party from me.
- CRC, etc. is other stuff.
- The protocol can be symmetric with both side employing timeouts.

Duplex Communication Example



(req no , data bit, seq no, data)

- The duplex protocol executes two simplex protocols simultaneously.
- The state of the sender1 and sender2 can be thought of as constisting to two bits.

  - request number for data being sent to me

  - sequence number of data being sent from me.