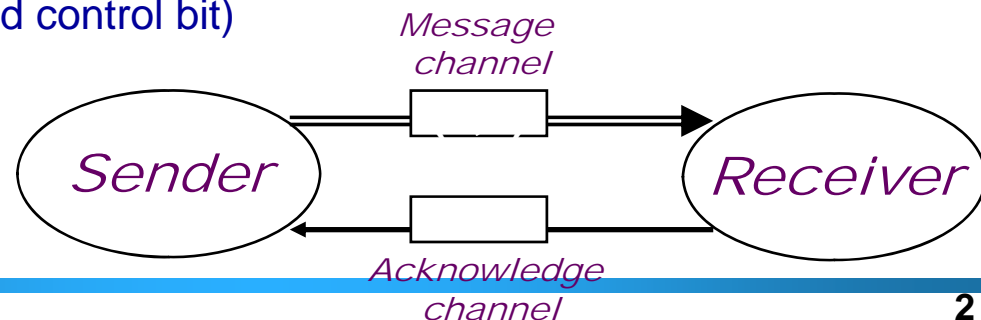

Temporal Logic

-Alternating Bit Protocol



The alternating bit protocol (ABP)

- ABP is a protocol for transmitting messages along a 'lossy line', i.e., a line which may **lose** or **duplicate** messages, but not corrupt messages
 - ✦ this lossy characteristic is common to data link and physical link layers
- ABP has four entities
 - ✦ the sender **S**, the receiver **R**, the message channel, and the acknowledgement channel
- ABP works as follows
 - ✦ **S** transmits the first part of the message together with the 'control' bit **b**.
 - ✦ If **R** receives a message with the control bit **b**, it sends **b** along the acknowledgement channel.
 - If not, R ignores the message.
 - ✦ If **S** receives acknowledge **b** from **R**, **S** sends next message with $\neg b$.
 - If not, S resends the message again with **b**
 - ✦ By **alternating the control bit**, both R and S can guard against losing messages (they ignore messages with unexpected control bit)



The ABP sender

- message1: current bit of the message being sent

- ✚ it is non-deterministic
 - assuming that it is received from higher protocol layer (i.e., environment)

- message2: the control bit

- ✚ note that message2 alternates bit

- **Req. property** says that we can always succeed in sending the current message

- ✚ For eliminating uninteresting violation of this property, we add FAIRNESS running

- ✚ Note that we use CTL formula, with an universal path quantifier 'A'

```
MODULE sender(ack)
VAR
  st      : {sending, sent};
  message1 : boolean;
  message2 : boolean;
ASSIGN
  init(st) := sending;
  next(st) := case
    ack = message2 & !(st=sent) : sent;
    1                          : sending;
  esac;
  next(message1) :=
    case
      st = sent : {0,1};
      1         : message1;
    esac;
  next(message2) :=
    case
      st = sent : !message2;
      1         : message2;
    esac;
FAIRNESS running
SPEC AG AF st=sent
```



The ABP receiver

```
MODULE receiver(message1,message2)
VAR
  st      : {receiving,received};
  ack     : boolean;
  expected : boolean;
ASSIGN
```

The ABP channels

- Lossy characteristics is modeled using forget
 - ✦ the value of input should be transmitted to output unless forget is true
- Fairness assumption enforces that they infinitely often transmit the message correctly.
 - ✦ Note that FAIRNESS !forget is not enough.

Why?

```
MODULE one-bit-chan(input)
VAR
  output: boolean;
  forget : boolean;
ASSIGN
  next(output) := case
    forget : output;
    1 : input;
  esac;
```

```
FAIRNESS running
FAIRNESS input & !forget
FAIRNESS !input & !forget
```

```
MODULE two-bit-chan(input1,input2)
VAR
  output1: boolean;
  output2: boolean;
  forget : boolean;
ASSIGN
  next(output1) := case
    forget : output1;
    1 : input1;
  esac;
  next(output2) := case
    forget : output2;
    1 : input2;
  esac;
```

```
FAIRNESS running
FAIRNESS input1 & !forget
FAIRNESS !input1 & !forget
FAIRNESS input2 & !forget
FAIRNESS !input2 & !forget
```



The overall ABP

- Integrate S,R, message channel and acknowledge channel
- Initially, the first control bit is 0.
- This ABP satisfies the following specification
 - ✦ Safety: if the message bit 1 has been sent and the correct acknowledgement has been returned, then a 1 was indeed received by the receiver
 - ✦ Liveness: Messages get through eventually.
 - For any state, there is inevitably a future state in which the current message has got

```
MODULE main
VAR
  S : process sender(ack_chan.output);
  R : process receiver(msg_chan.output1,msg_chan.output2);
  msg_chan : process two-bit-chan(S.message1,S.message2);
  ack_chan : process one-bit-chan(R.ack);
ASSIGN
  init(S.message2) := 0;
  init(R.expected) := 0;
  init(R.ack)      := 1;
  init(msg_chan.output2) := 1;
  init(ack_chan.output) := 1;
```