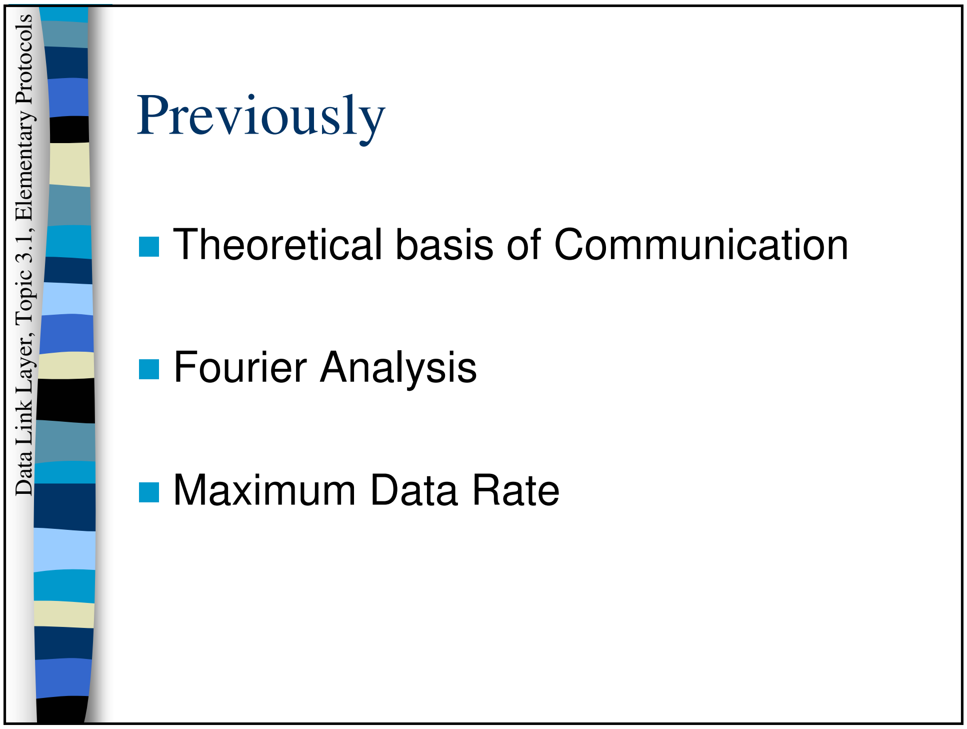
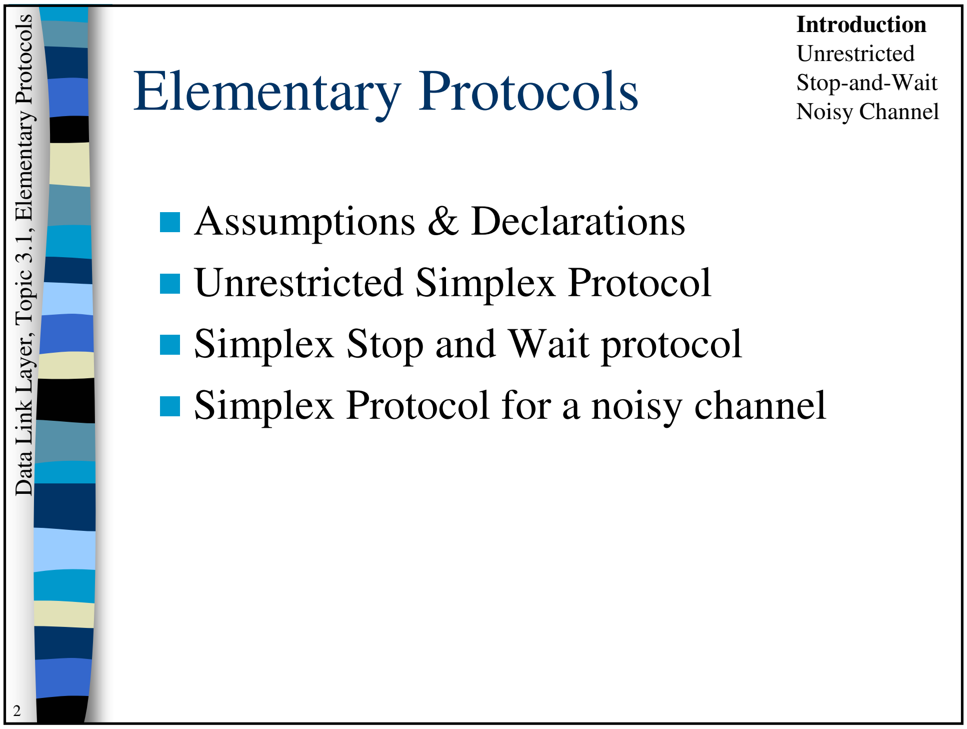


Previously

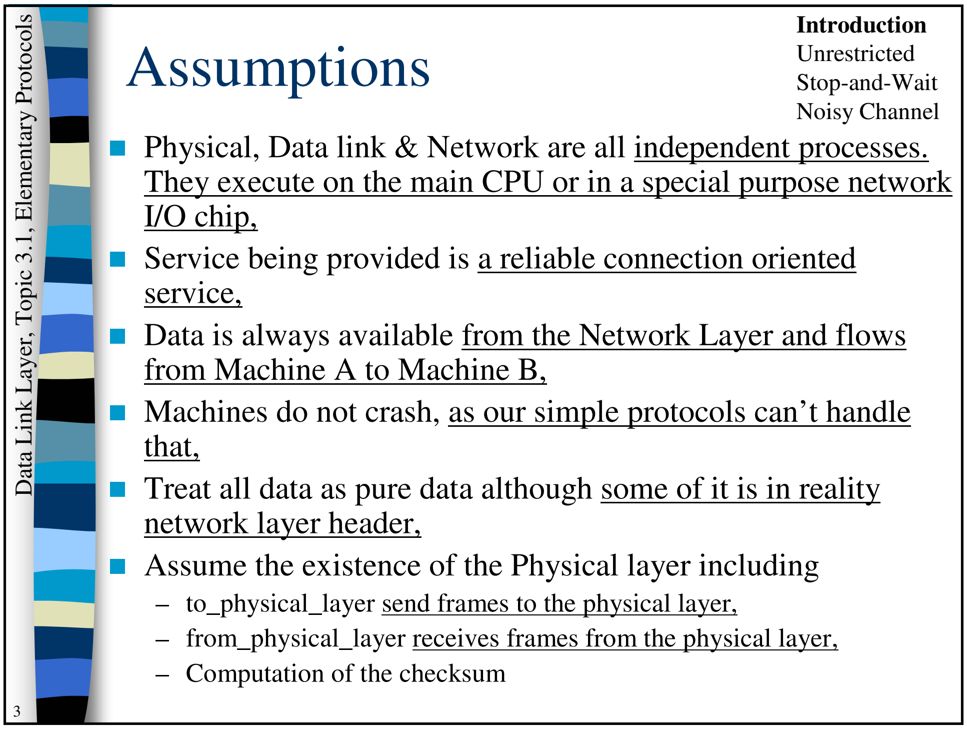
- Theoretical basis of Communication
- Fourier Analysis
- Maximum Data Rate



Introduction
Unrestricted
Stop-and-Wait
Noisy Channel

Elementary Protocols

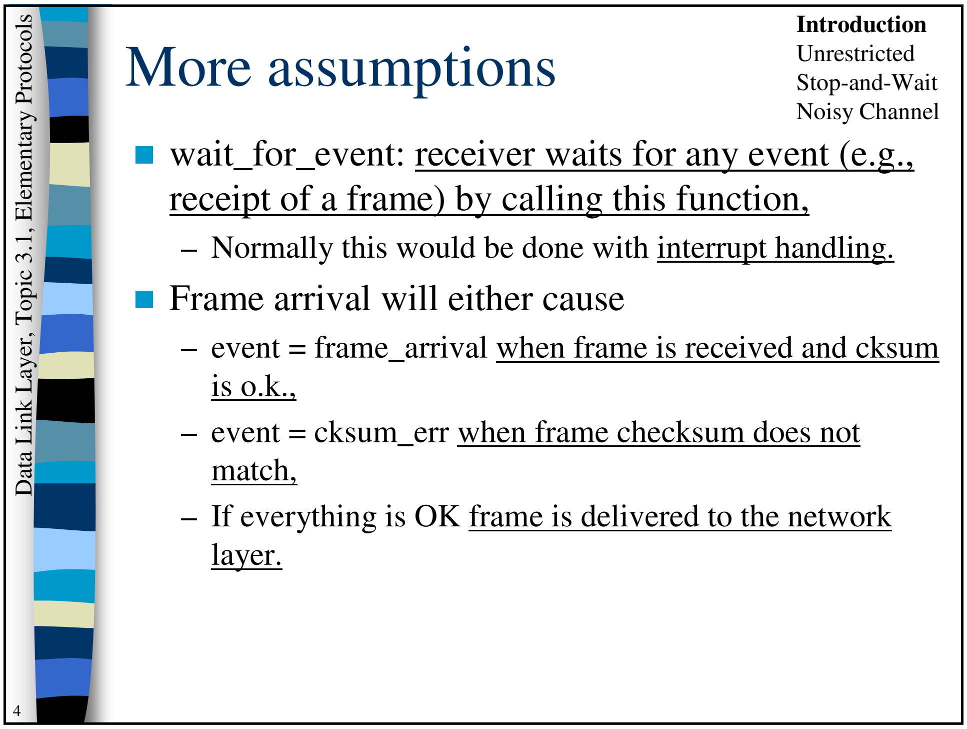
- Assumptions & Declarations
- Unrestricted Simplex Protocol
- Simplex Stop and Wait protocol
- Simplex Protocol for a noisy channel



Introduction
Unrestricted
Stop-and-Wait
Noisy Channel

Assumptions

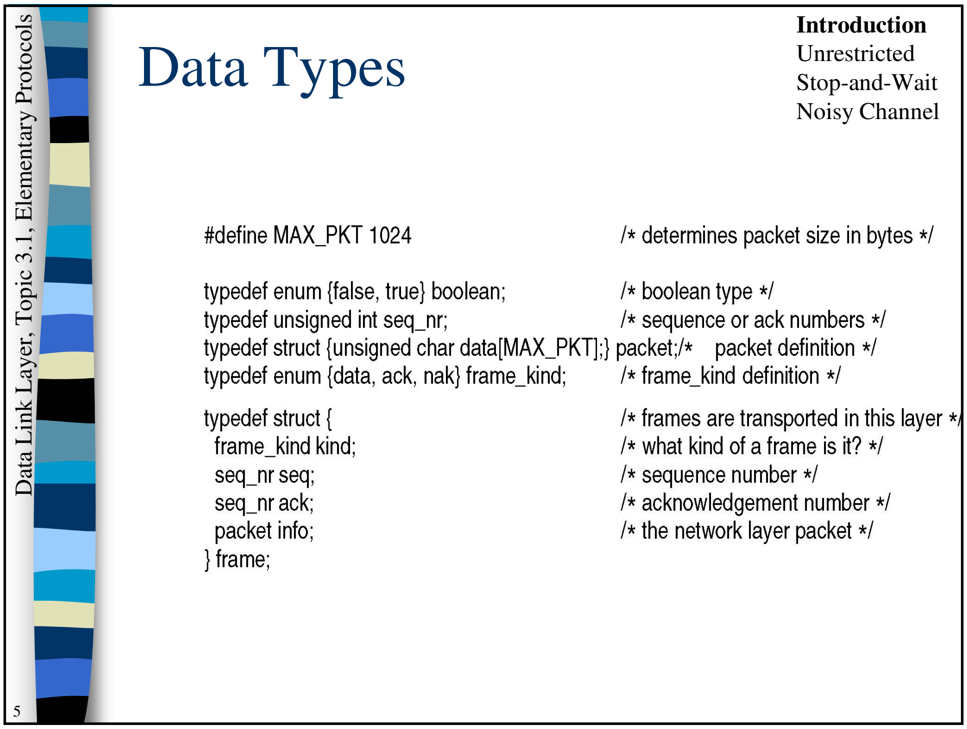
- Physical, Data link & Network are all independent processes. They execute on the main CPU or in a special purpose network I/O chip,
- Service being provided is a reliable connection oriented service,
- Data is always available from the Network Layer and flows from Machine A to Machine B,
- Machines do not crash, as our simple protocols can't handle that,
- Treat all data as pure data although some of it is in reality network layer header,
- Assume the existence of the Physical layer including
  - to_physical_layer send frames to the physical layer,
  - from_physical_layer receives frames from the physical layer,
  - Computation of the checksum



Introduction
Unrestricted
Stop-and-Wait
Noisy Channel

More assumptions

- wait_for_event: receiver waits for any event (e.g., receipt of a frame) by calling this function,
  - Normally this would be done with interrupt handling.
- Frame arrival will either cause
  - event = frame_arrival when frame is received and cksum is o.k.,
  - event = cksum_err when frame checksum does not match,
  - If everything is OK frame is delivered to the network layer.



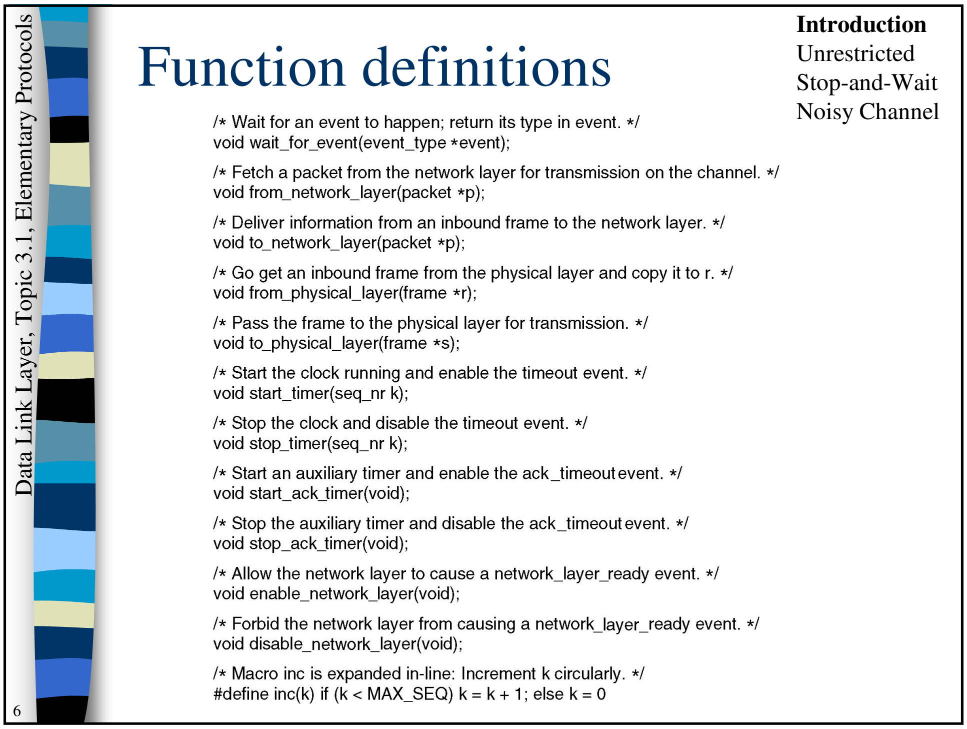
Data Link Layer, Topic 3.1, Elementary Protocols

Data Types

Introduction
Unrestricted
Stop-and-Wait
Noisy Channel

```
#define MAX_PKT 1024                         /* determines packet size in bytes */

typedef enum {false, true} boolean;          /* boolean type */
typedef unsigned int seq_nr;                 /* sequence or ack numbers */
typedef struct {unsigned char data[MAX_PKT];} packet;/*    packet definition */
typedef enum {data, ack, nak} frame_kind;    /* frame_kind definition */

typedef struct {                             /* frames are transported in this layer */
  frame_kind kind;                           /* what kind of a frame is it? */
  seq_nr seq;                                /* sequence number */
  seq_nr ack;                                /* acknowledgement number */
  packet info;                               /* the network layer packet */
} frame;
```

5



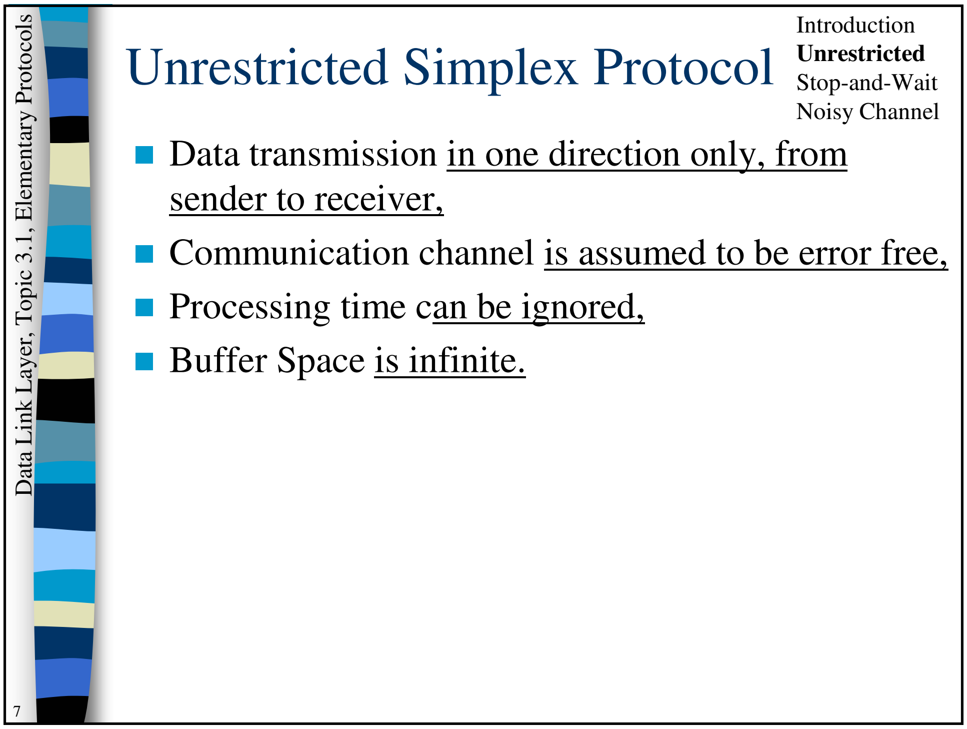
Data Link Layer, Topic 3.1, Elementary Protocols

Function definitions

Introduction
Unrestricted
Stop-and-Wait
Noisy Channel

```
/* Wait for an event to happen; return its type in event. */
void wait_for_event(event_type *event);

/* Fetch a packet from the network layer for transmission on the channel. */
void from_network_layer(packet *p);

/* Deliver information from an inbound frame to the network layer. */
void to_network_layer(packet *p);

/* Go get an inbound frame from the physical layer and copy it to r. */
void from_physical_layer(frame *r);

/* Pass the frame to the physical layer for transmission. */
void to_physical_layer(frame *s);

/* Start the clock running and enable the timeout event. */
void start_timer(seq_nr k);

/* Stop the clock and disable the timeout event. */
void stop_timer(seq_nr k);

/* Start an auxiliary timer and enable the ack_timeout event. */
void start_ack_timer(void);

/* Stop the auxiliary timer and disable the ack_timeout event. */
void stop_ack_timer(void);

/* Allow the network layer to cause a network_layer_ready event. */
void enable_network_layer(void);

/* Forbid the network layer from causing a network_layer_ready event. */
void disable_network_layer(void);

/* Macro inc is expanded in-line: Increment k circularly. */
#define inc(k) if (k < MAX_SEQ) k = k + 1; else k = 0
```

6



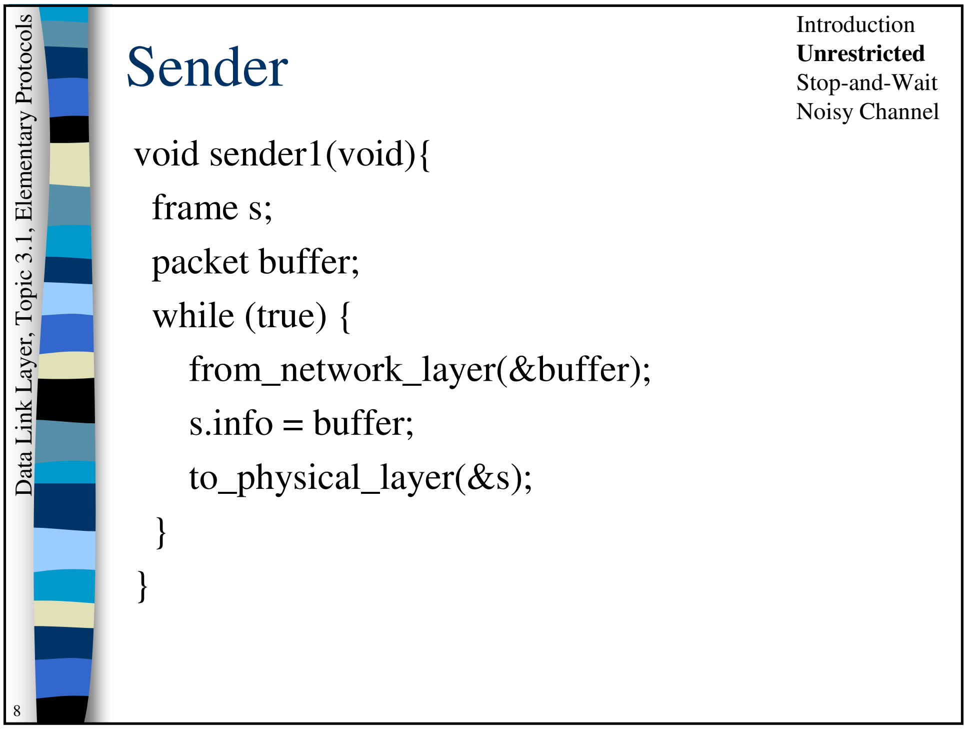
Data Link Layer, Topic 3.1, Elementary Protocols

Introduction
Unrestricted
Stop-and-Wait
Noisy Channel

Unrestricted Simplex Protocol

- Data transmission in one direction only, from sender to receiver,
- Communication channel is assumed to be error free,
- Processing time can be ignored,
- Buffer Space is infinite.

7



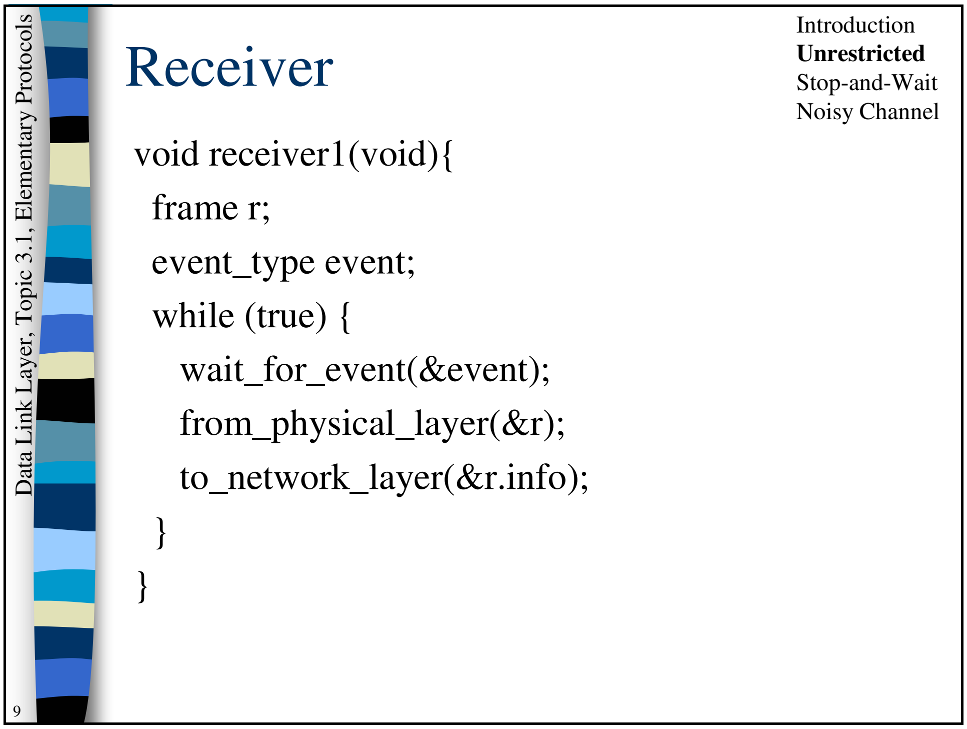
Data Link Layer, Topic 3.1, Elementary Protocols

Introduction
Unrestricted
Stop-and-Wait
Noisy Channel

Sender

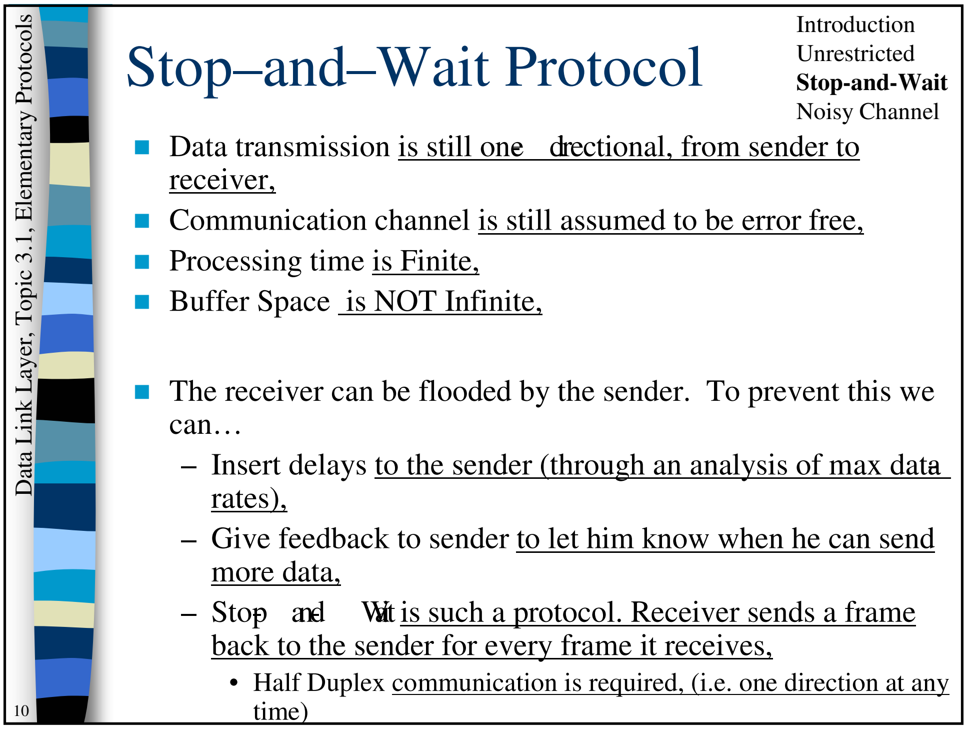
```
void sender1(void){
  frame s;
  packet buffer;
  while (true) {
      from_network_layer(&buffer);
      s.info = buffer;
      to_physical_layer(&s);
  }
}
```

8



Receiver

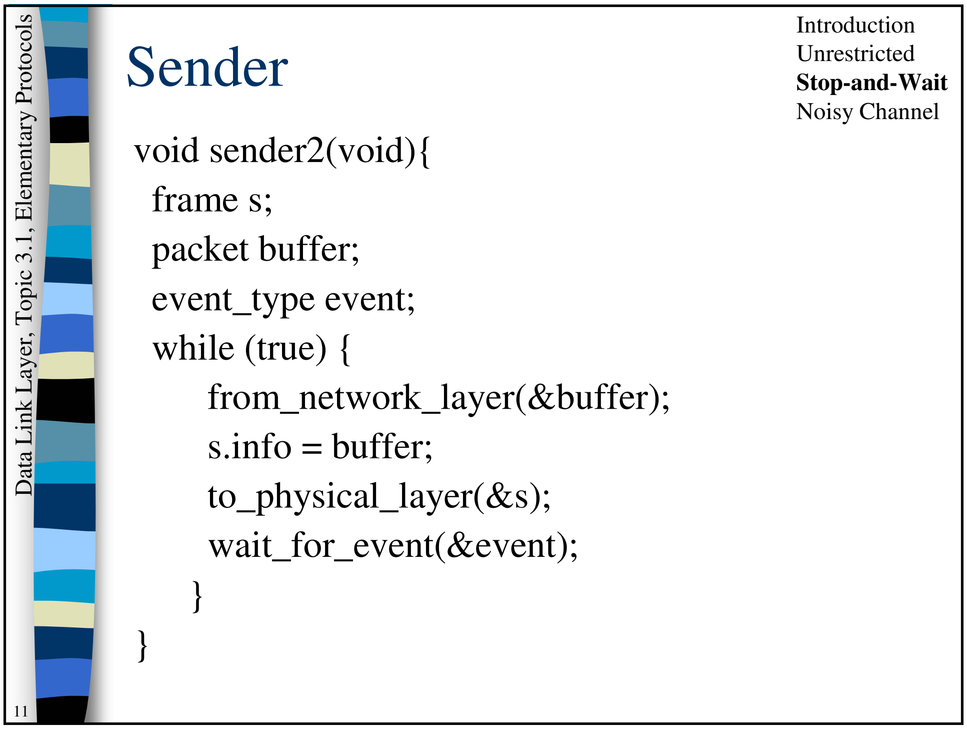
```
void receiver1(void){
  frame r;
  event_type event;
  while (true) {
     wait_for_event(&event);
     from_physical_layer(&r);
     to_network_layer(&r.info);
  }
}
```



Stop–and–Wait Protocol

- Data transmission is still one drectional, from sender to receiver,
- Communication channel is still assumed to be error free,
- Processing time is Finite,
- Buffer Space is NOT Infinite,

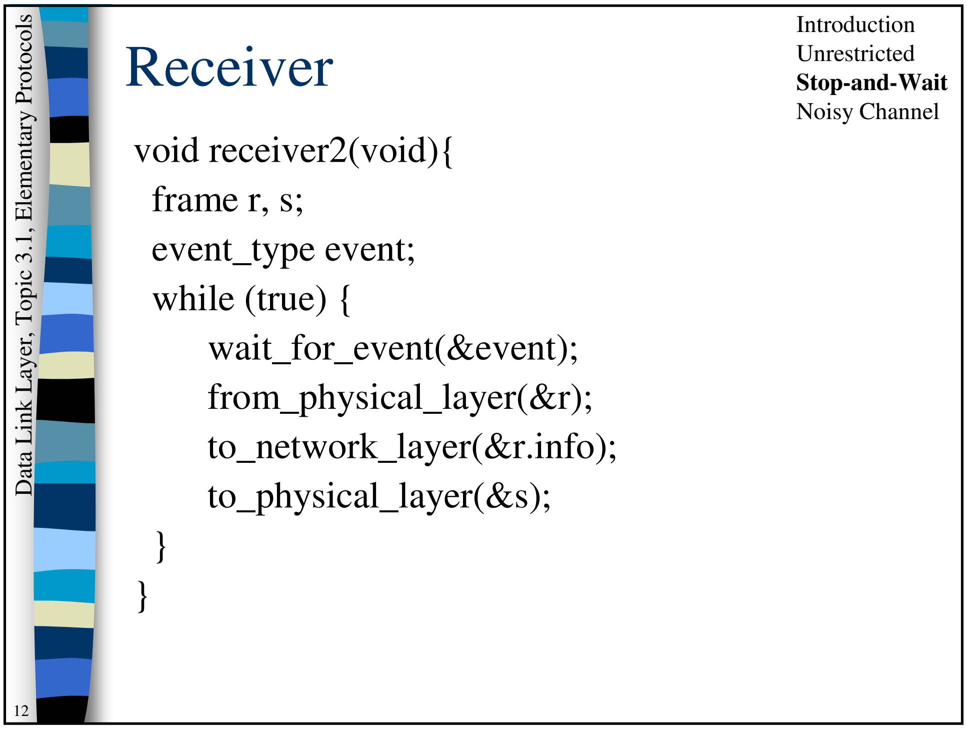
- The receiver can be flooded by the sender. To prevent this we can…
  - Insert delays to the sender (through an analysis of max data rates),
  - Give feedback to sender to let him know when he can send more data,
  - Stop-and-Wait is such a protocol. Receiver sends a frame back to the sender for every frame it receives,
    - Half Duplex communication is required, (i.e. one direction at any time)



Sender

Introduction
Unrestricted
Stop-and-Wait
Noisy Channel

```
void sender2(void){
  frame s;
  packet buffer;
  event_type event;
  while (true) {
        from_network_layer(&buffer);
        s.info = buffer;
        to_physical_layer(&s);
        wait_for_event(&event);
      }
}
```



Receiver

Introduction
Unrestricted
Stop-and-Wait
Noisy Channel

```
void receiver2(void){
  frame r, s;
  event_type event;
  while (true) {
        wait_for_event(&event);
        from_physical_layer(&r);
        to_network_layer(&r.info);
        to_physical_layer(&s);
  }
}
```

Introduction
Unrestricted
Stop-and-Wait
Noisy Channel

Results

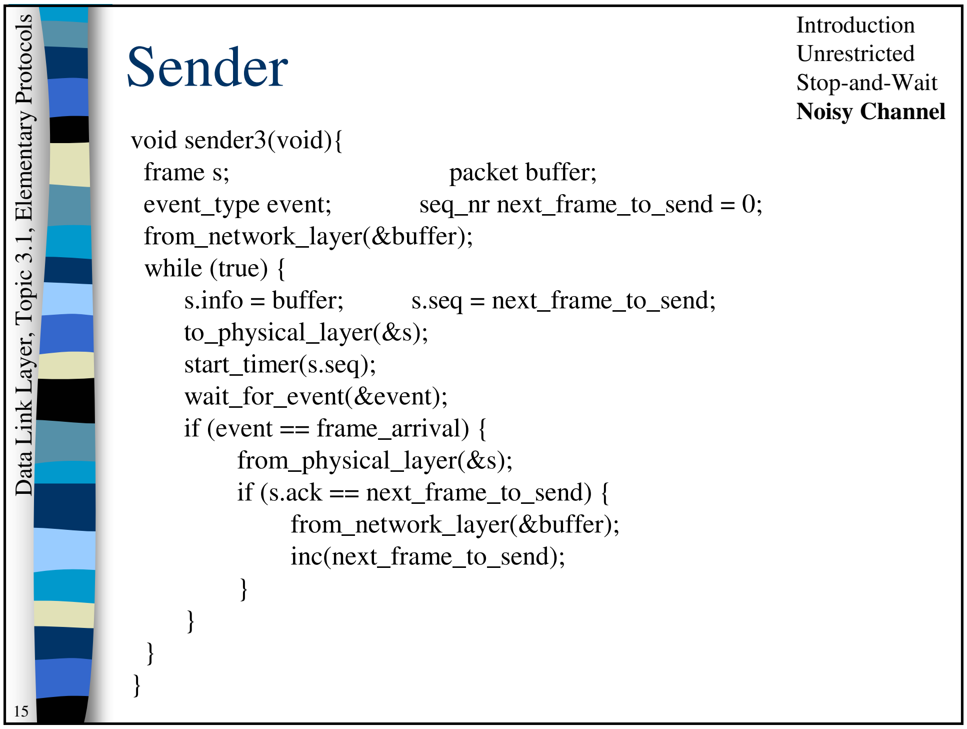
Simulating Protocol 2	
Events	**100000**
Timoute	**30**
pkt_loss	**0**
pkt_cksum	**0**

	Process 0	**Process 1**
Total data frames sent	12580	0
Data frames lost	0	0
Data frames not lost	12580	0
Frames retransmitted	0	0
Good ack frames rec'd	12579	0
Bad ack frames rec'd	0	0
Good data frames rec'd	0	12580
Bad data frames rec'd	0	0
Payloads accepted	0	12580
Total ack frames sent	0	12580
Ack frames lost	0	0
Ack frames not lost	0	12580
Timeouts	0	0
Ack timeouts	0	0

Introduction
Unrestricted
Stop-and-Wait
Noisy Channel

Protocol for a Noisy Channel

- Data transmission is still one drectional, from sender to receiver,
- Communication channel now allows errors,
  - Corrupt Frames: checksum does not match,
  - Missing Frames: frames do not arrive at all
- Solution
  - Use a timer: That way the sender does not wait for the receiver to tell it that something is wrong,
  - Problem: If the ack of a frame is lost then the sender would resend the frame – causing the receiver to get the frame twice,
  - Use Sequence Numbers: Identify each frame. If one frame can be outstanding, then a single bit seq. no. is enough.
  - PAR/ARQ: **P**ositive **A**cknowledgement with **R**etransmission / **A**utomatic **R**epeat re**Q**uest



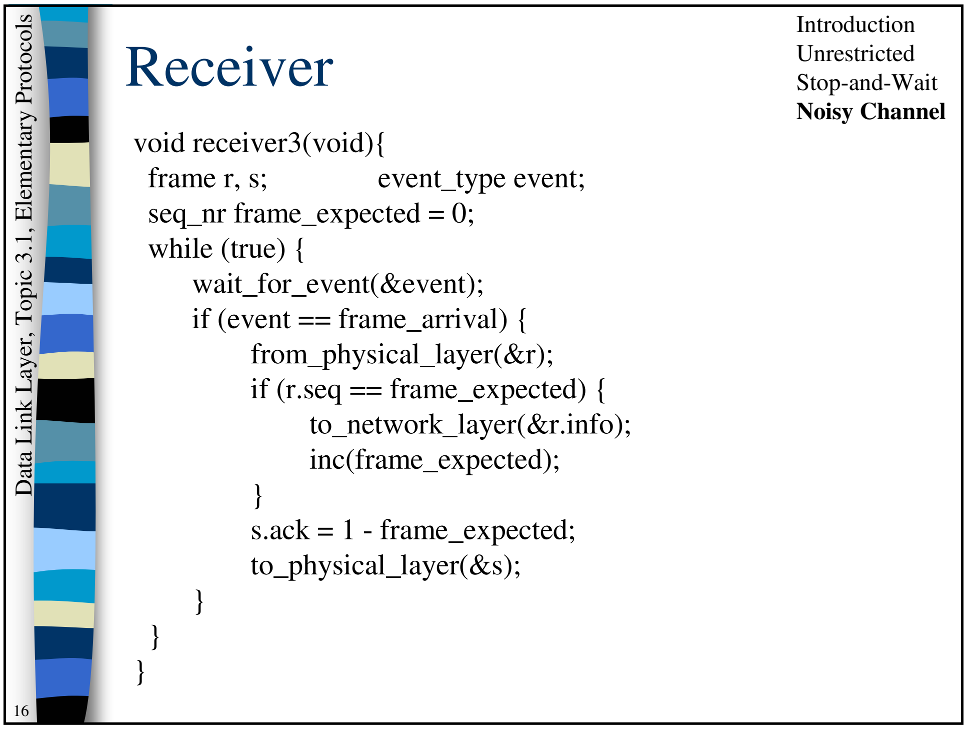
Data Link Layer, Topic 3.1, Elementary Protocols

Sender

Introduction
Unrestricted
Stop-and-Wait
Noisy Channel

```
void sender3(void){
  frame s;                        packet buffer;
  event_type event;               seq_nr next_frame_to_send = 0;
  from_network_layer(&buffer);
  while (true) {
        s.info = buffer;          s.seq = next_frame_to_send;
        to_physical_layer(&s);
        start_timer(s.seq);
        wait_for_event(&event);
        if (event == frame_arrival) {
                from_physical_layer(&s);
                if (s.ack == next_frame_to_send) {
                        from_network_layer(&buffer);
                        inc(next_frame_to_send);
                }
        }
  }
}
```

15



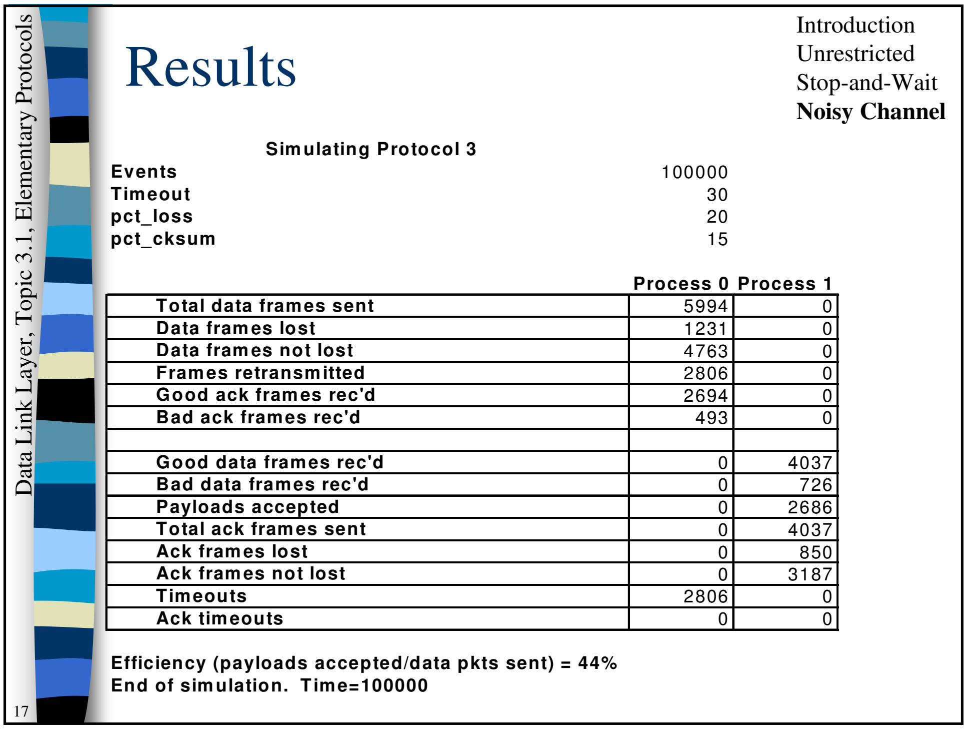
Data Link Layer, Topic 3.1, Elementary Protocols

Receiver

Introduction
Unrestricted
Stop-and-Wait
Noisy Channel

```
void receiver3(void){
  frame r, s;           event_type event;
  seq_nr frame_expected = 0;
  while (true) {
        wait_for_event(&event);
        if (event == frame_arrival) {
                from_physical_layer(&r);
                if (r.seq == frame_expected) {
                        to_network_layer(&r.info);
                        inc(frame_expected);
                }
                s.ack = 1 - frame_expected;
                to_physical_layer(&s);
        }
  }
}
```

16



Results

Introduction
Unrestricted
Stop-and-Wait
Noisy Channel

Simulating Protocol 3

Events	100000
Timeout	30
pct_loss	20
pct_cksum	15

	Process 0	**Process 1**
Total data frames sent	5994	0
Data frames lost	1231	0
Data frames not lost	4763	0
Frames retransmitted	2806	0
Good ack frames rec'd	2694	0
Bad ack frames rec'd	493	0
Good data frames rec'd	0	4037
Bad data frames rec'd	0	726
Payloads accepted	0	2686
Total ack frames sent	0	4037
Ack frames lost	0	850
Ack frames not lost	0	3187
Timeouts	2806	0
Ack timeouts	0	0

Efficiency (payloads accepted/data pkts sent) = 44%
End of simulation. Time=100000