INF3190 – First Home Exam

UNIVERSITY OF OSLO

INF3190 – First Home Exam

HE-1 stands for **20%** of the final grade

Individual work

The **goal** is to implement the link layer communication between multiple computers (bridging)

⇒ providing *reliable full-duplex* communication service to higher layers

Requirement: PHY later (L1) that
1) *can lose frames*, but
2) *free of bit error* and
3) doesn't change the *sequence* of frames

Simple Assumption: no loop on the link layer => no need for a spanning-tree protocol

INF3190 – First Home Exam

The flow control mechanism to be used: **Sliding Window** with moving window of *up to* 10 frames with 100-bytes frames

Two approaches:

1) *Without* use of flow control in pre-code ☺

2) *Using Go-Back-N flow control* in the pre-code (full mark not given ☹)

INF3190 – HE-1, Approach #1

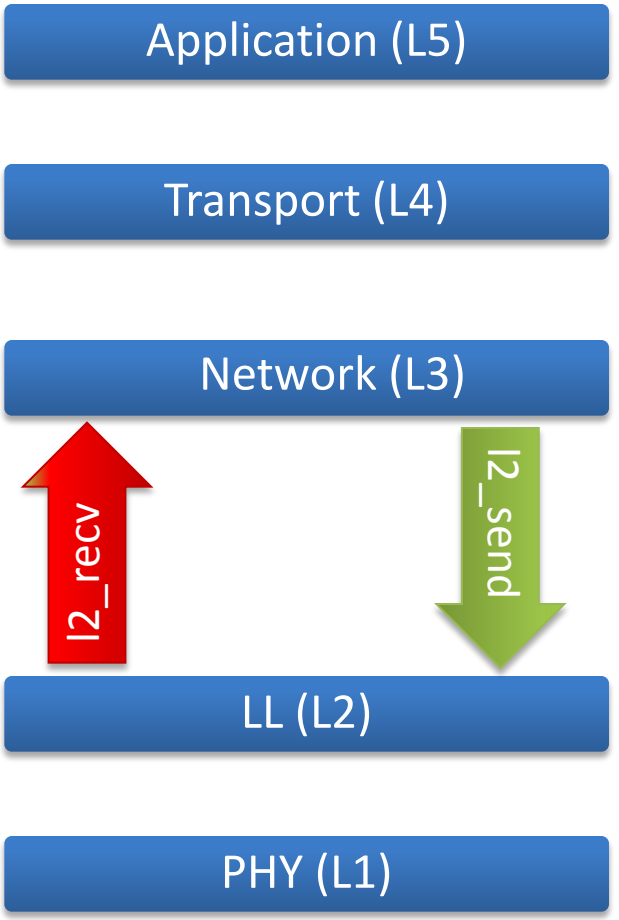
1) Data is sent to *PHY (L1)* through

```
int l2_send( int dest_mac_address, const char* buf, int length );
```

2) Data arrives to *l2_recv()*

```
void l2_recv( int device, const char* buf, int length );
```

Here you will add the flow control and bridging!



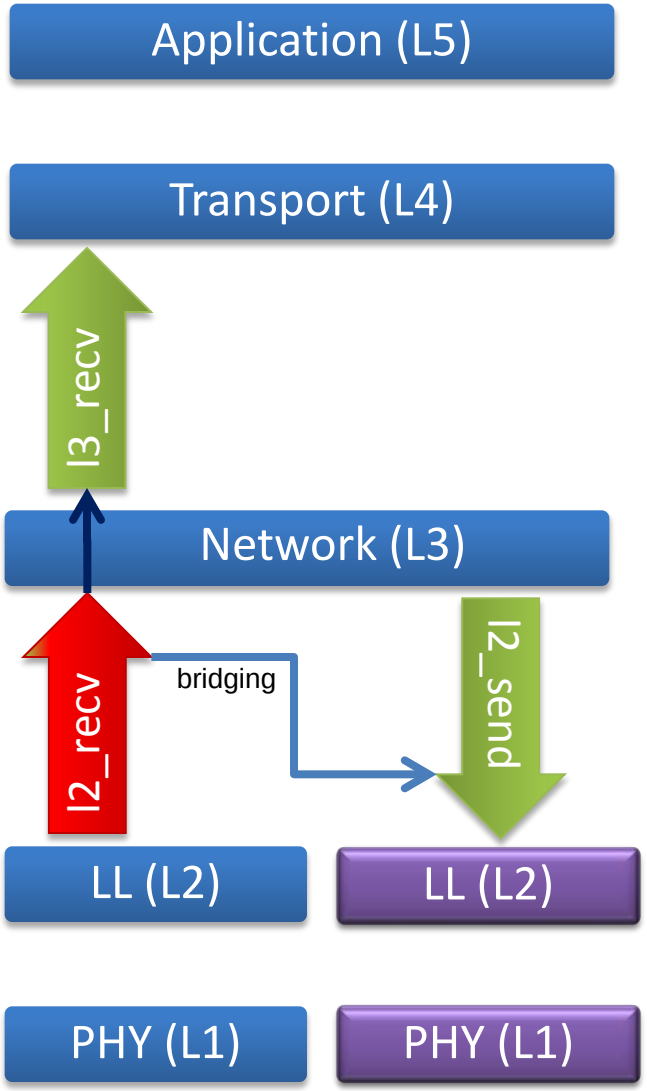
INF3190 – HE-1, Approach #1

3) If data belongs to the local machine *l2_recv()* calls *l3_recv()*

```
int l3_recv( int mac_address, const char* buf, int length )
```

4) If it belongs to another machine call *l2_send()*

```
int l2_send( int mac_address, const char* buf, int length )
```



INF3190 – HE-1, Approach #2

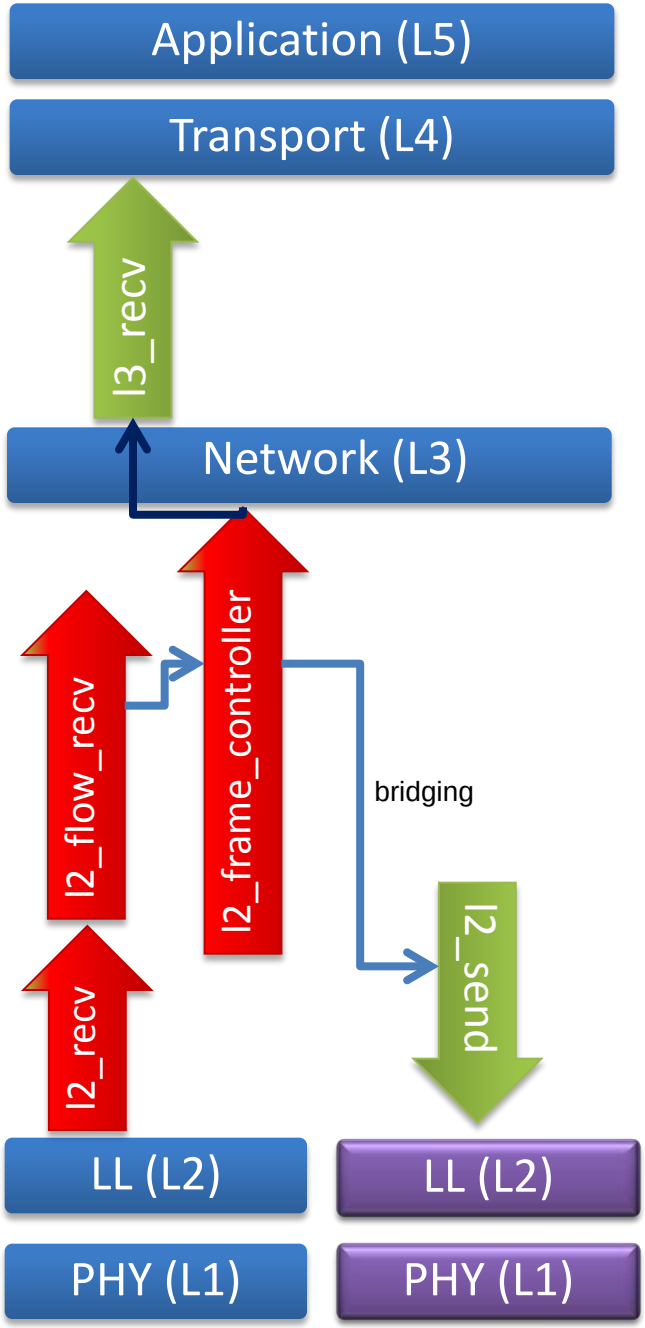
1) Data is sent to *PHY (L1)* through

```
int l2_send( int dest_mac_address, const char* buf, int length );
```

2) When data arrives to *l2_recv()*, it is delivered to the flow control in *l2_flow_recv()* (available in *l2_flow.o file*)

```
void l2_flow_recv( int device, const char* buf, int length );
```

It takes care of flow control and ACK processing!



INF3190 – HE-1, Approach #2

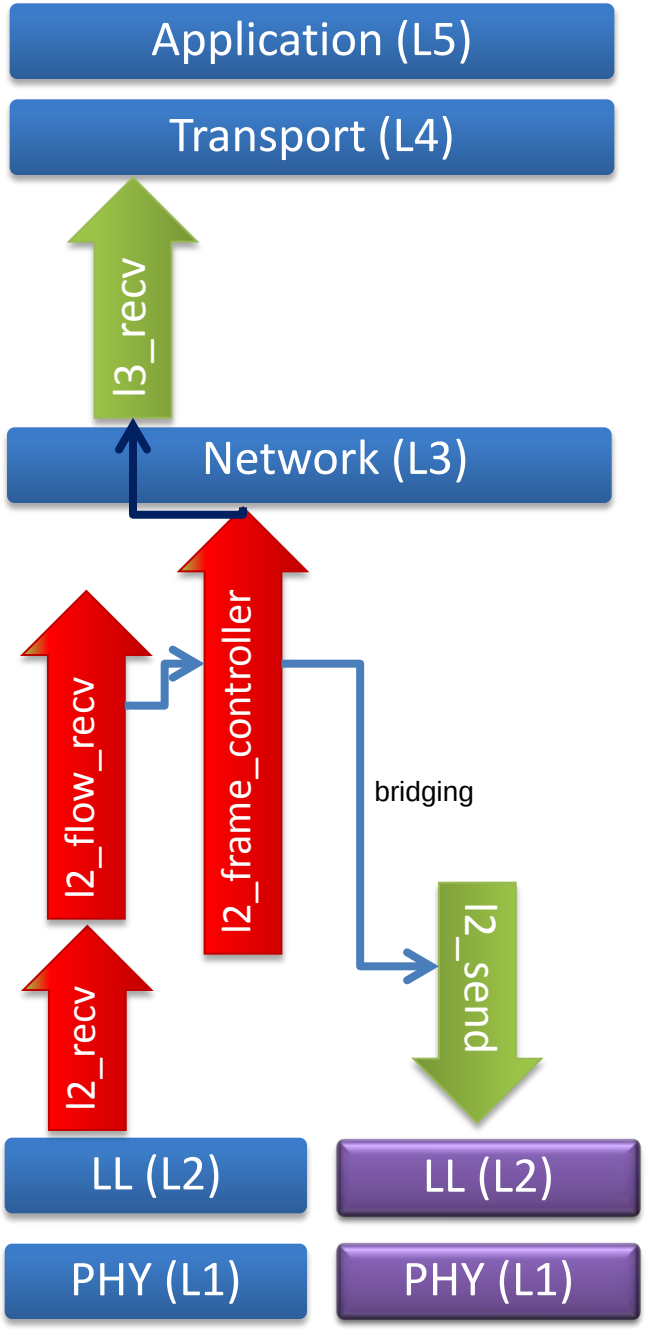
3) *l2_flow_recv()* tries to push the limits set in the receiver buffer to *l2_frame_controller()* => implement this!

```
int l2_frame_controller(int device, const char* buf, int length);
```

4) *l2_frame_controller()* will deliver frames to *l3_recv()* if it belongs to the local machine

5) Otherwise the frame is forwarded to *l2_send()* for bridging

Note: make sure to use the *other* links when calling *l2_send()*



Program's Structure (UI)

1. Create PHY links between *two machines* using UDP packet exchanges, while being able to accept *new* PHY connections (needs more parameters e.g. port number and MAC addresses)

2. MAC addresses must be sent to the link layer via *l2_init()* => support connectivity to and from multiple computers simultaneously but no need for PHY fault-tolerance

> CONNECT <hostname> <port_number>

> SEND <mac_address> <filename>

> Quit

Program's Structure (UI)

Issues to remember:

1) *Separate* physical link and flow control per PHY for a connection

2) The file must be stored on the receiver!

3) If a forwarding frame on a bridge meets a *full sliding window* on the next link:

   A. You can decide to drop the frame relying on the retransmission from the sender

   B. or don't take the frame out of the moving window on the receiving end

   C. or make a separated queue for the bridge

4) Use the *slow_receiver* to write data to the file

5) To send data to a socket, use the *delayed_sendto* / *delayed_dropping_sendto* instead of *Sendto*

Grading

1) Maximum grade is *not given* for **Approach #2**

2) Maximum grade is given for *a proper Go-Back-N implementation*, and few extra points for *Selective Repeat*

if (isdigit(CandidateNum)) { Devilry=Delivery; }

When to deliver? **Before Friday 30 March 2012 23:59**

Only use your candidate number when delivering!

Q&A?