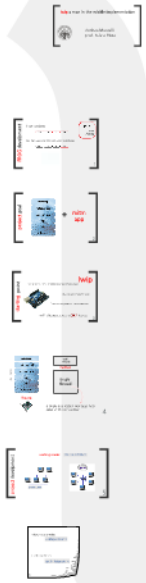
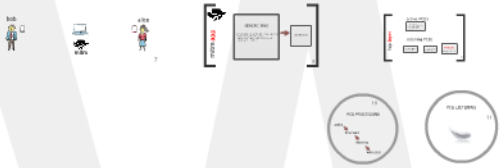


how does it work?

phase 1



phase 2

phase 3

lwip a man in the middle implementation

Andrea Marcelli
prof. Fulvio Risso

FROG develpment

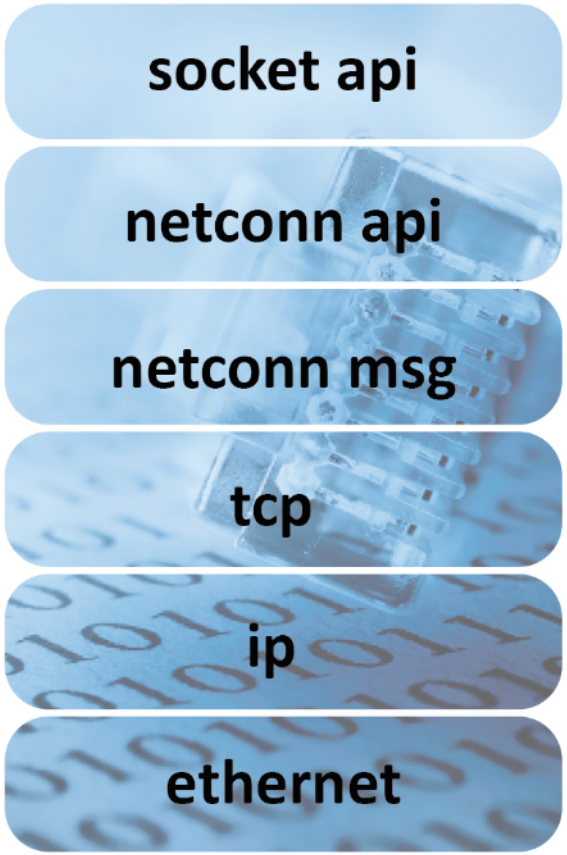
from packets

PEX

Private
Execution
Environment

to the awareness of a connection.

project goal



+ mitm app

starting point

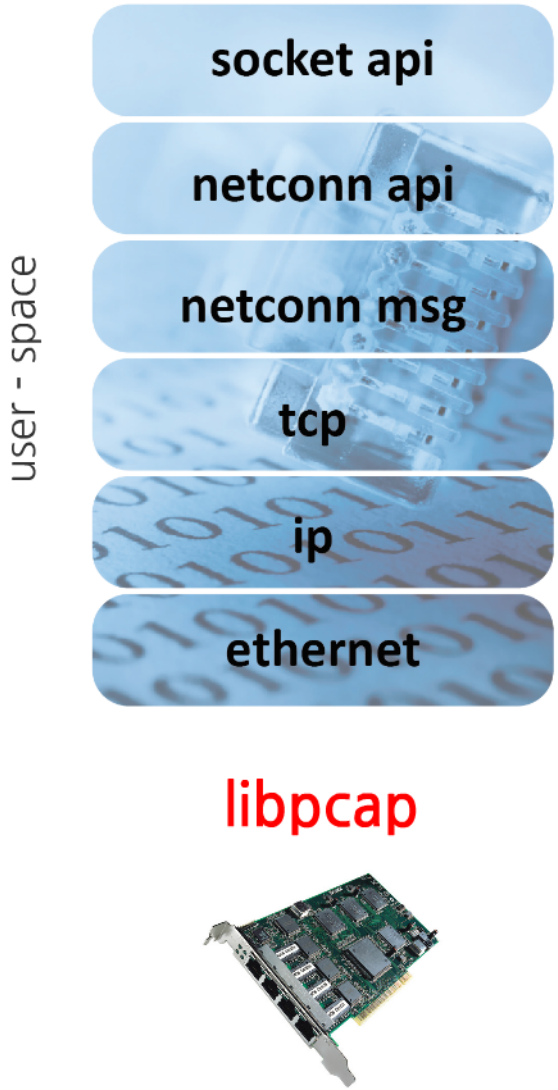
lwip

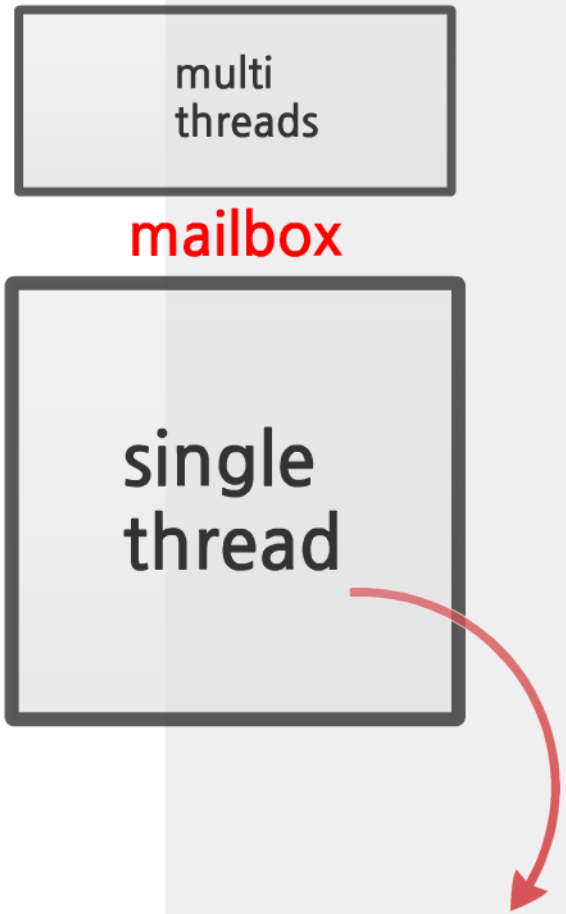
lwIP is used by many manufacturers of embedded systems.

GOAL: to reduce resource usage.

"lwIP has been ported to multiple platforms."

lwIP is licensed under a BSD license.



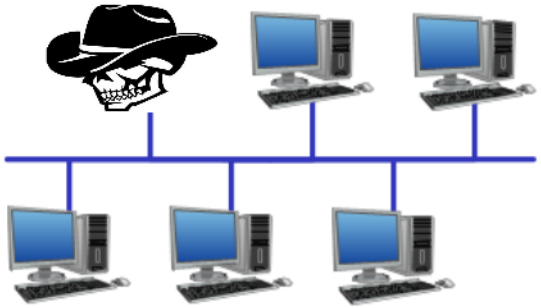


a single application manages both sides of the connection

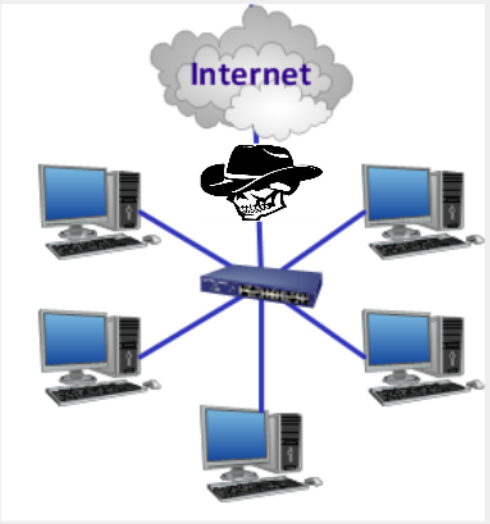
project develpment

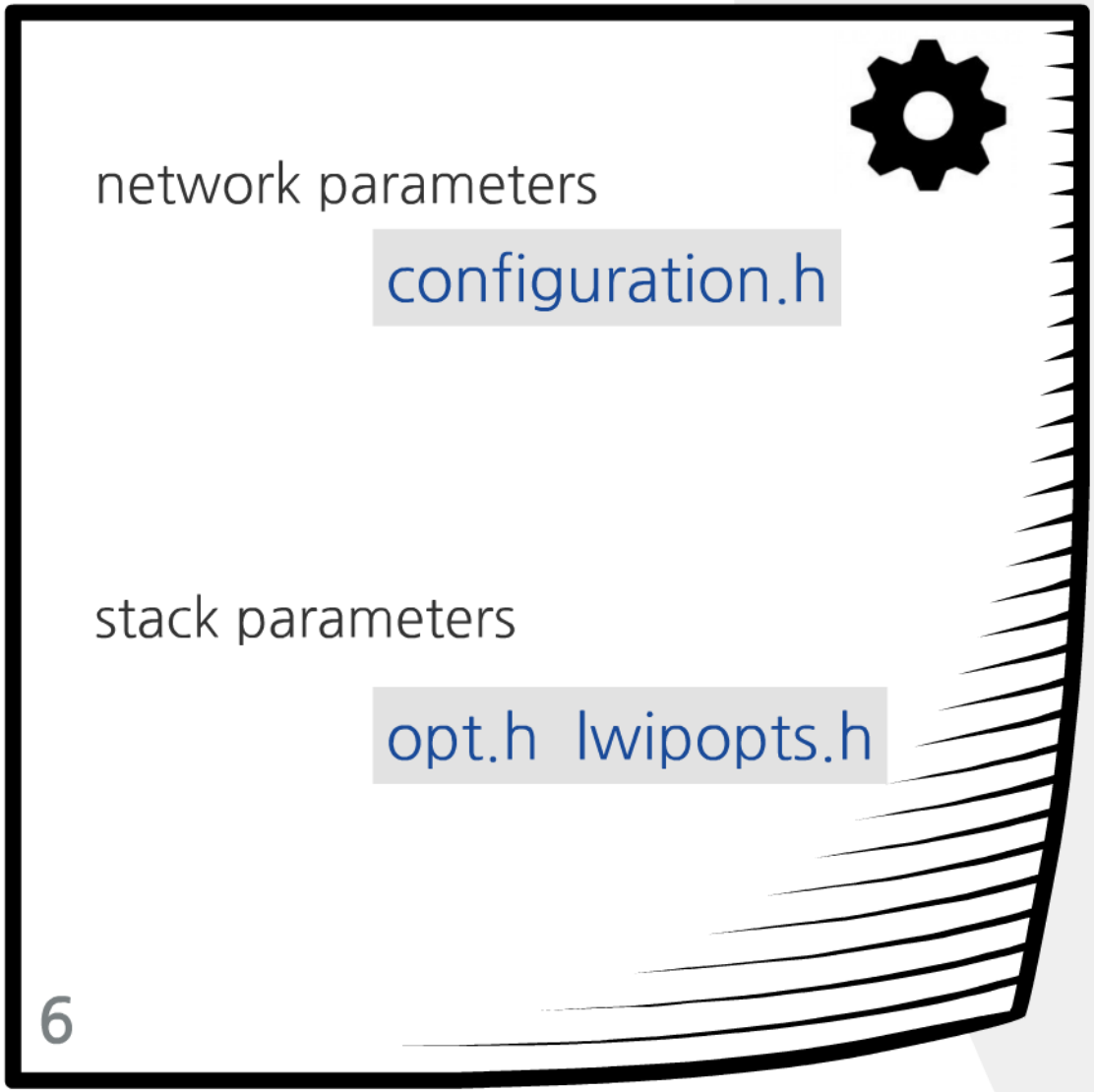
working mode:

#SECOND_INTERFACE



#STATIC_ARP




network parameters
configuration.h
stack parameters
opt.h lwipopts.h
6

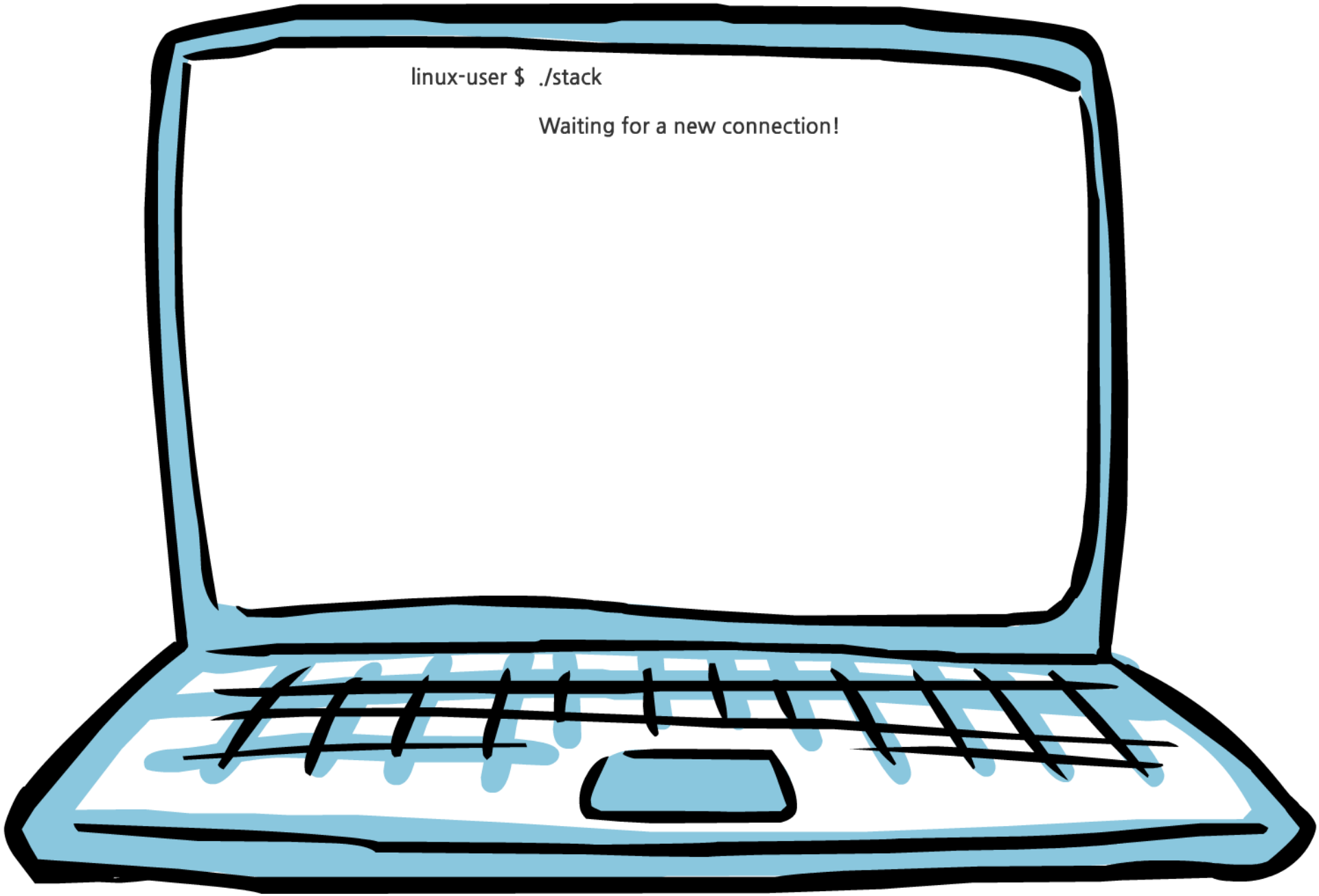
how does it work 

phase 1

bob

mitm

alice


linux-user $ ./stack
Waiting for a new connection!

mitm app

GENERIC BIND

```
memset(&listening, 0, sizeof(struct sockaddr_in));
    listening.sin_family = AF_INET;
    listening.sin_addr.s_addr = IPADDR_ANY;
    listening.sin_port = 0;
```

LISTENING MODE

tcp layer

active PCBs

local addr: 120.192.2.13
remote addr: 23.54.45.44
local port: 8080
remote port: 54353

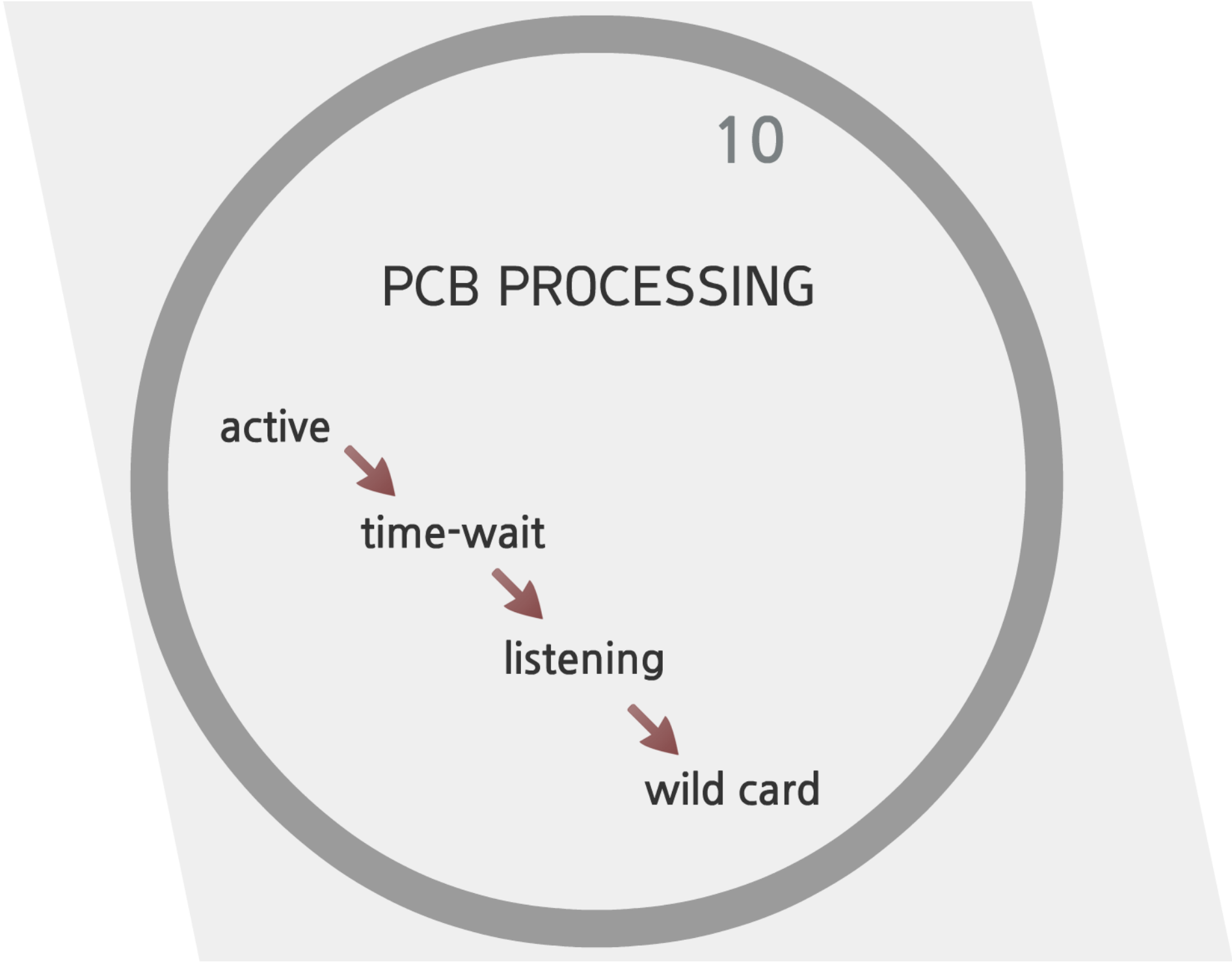
listening PCBs

local address: IPADDR_ANY
local port: 8080

local address: IPADDR_ANY
local port: 80

wild card

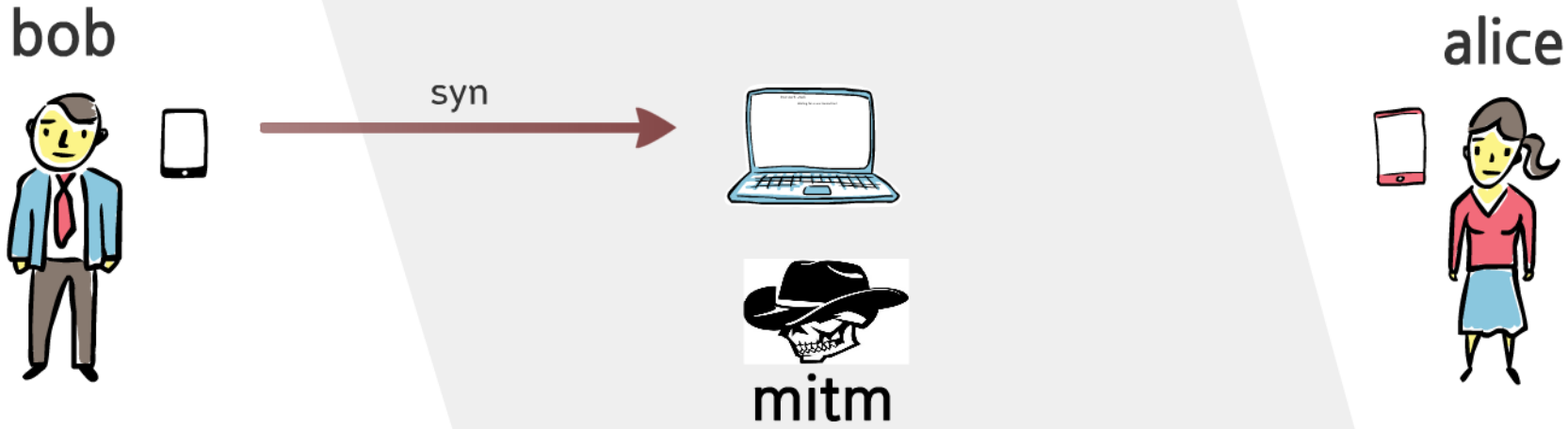
local address: IPADDR_ANY
local port: 0


10
PCB PROCESSING
active
time-wait
listening
wild card

PCB LISTENING

11

phase 2


bob
syn
alice
mitm

tcp layer

active PCBs

local addr: 120.192.2.13
remote addr: 23.54.45.44
local port: 8080
remote port: 54353

listening PCBs

local address: IPADDR_ANY
local port: 8080

local address: IPADDR_ANY
local port: 80

wild card

local address: IPADDR_ANY
local port: 0

tcp layer

active PCBs

local addr: 120.192.2.13
remote addr: 23.54.45.44
local port: 8080
remote port: 54353

bob side

local addr: alice_addr
remote addr: bob_addr

local port: alice_port
remote port: bob_port

listening PCBs

local address: IPADDR_ANY
local port: 8080

local address: IPADDR_ANY
local port: 80

wild card

local address: IPADDR_ANY
local port: 0

mitm app

ACCEPTED NEW CONNECTIONS

```
sys_thread_new("Manage connection", manage_connection, &info1, 0, 0);
```

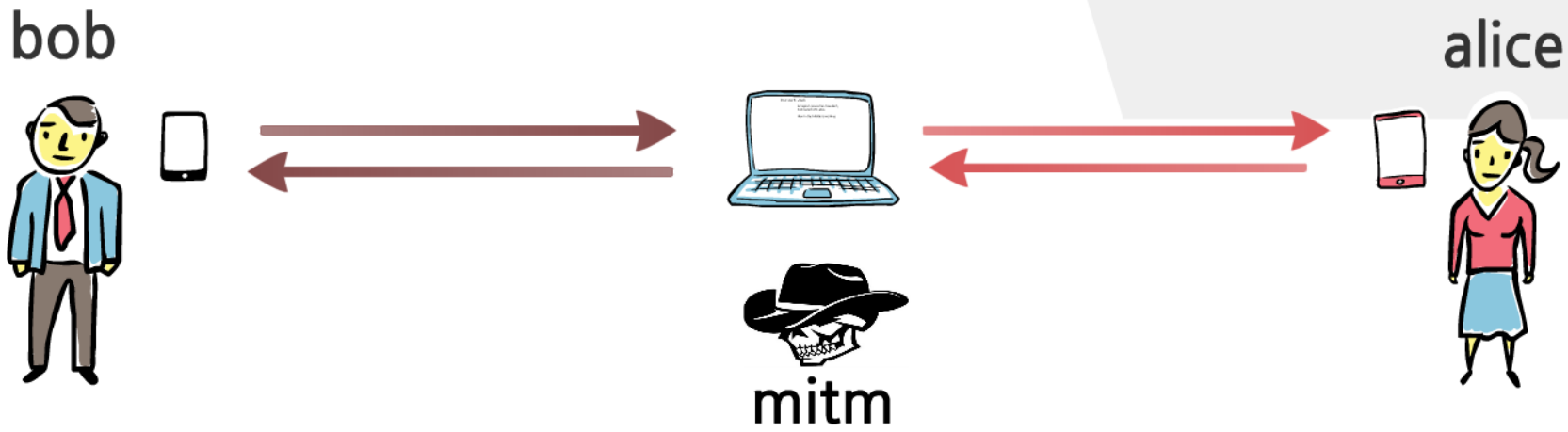
bob side

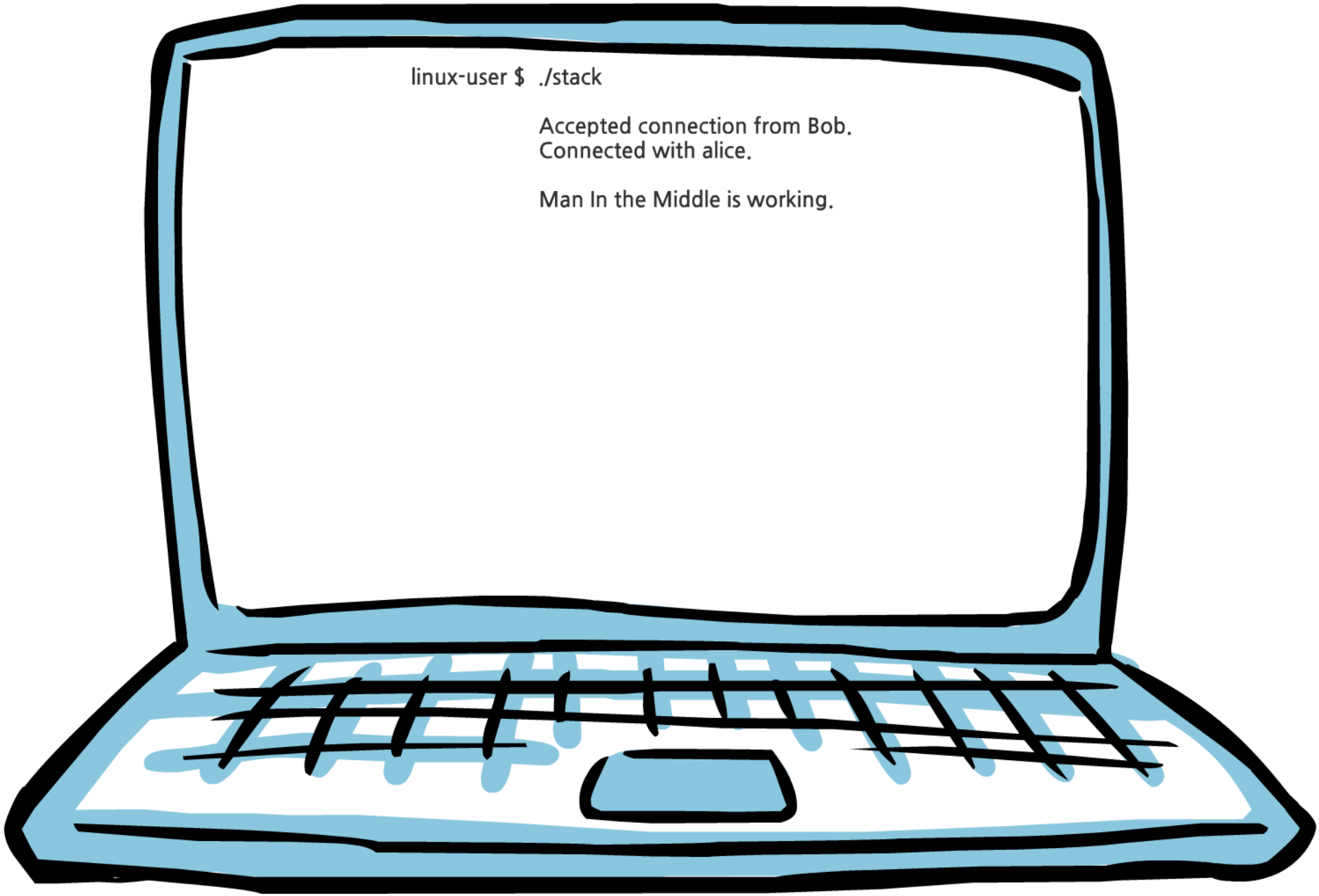
MANAGE_LEFT

```
static void manage_left(void *arg)
```

new thread

phase 3


bob
alice
mitm


linux-user $ ./stack
Accepted connection from Bob.
Connected with alice.
Man In the Middle is working.

tcp layer

active PCBs

#PURE_MIM

local addr: 120.192.2.13
remote addr: 23.54.45.44
local port: 8080
remote port: 54353

bob side

local addr: alice_addr
remote addr: bob_addr

local port: alice_port
remote port: bob_port

alice side

local addr: bob_addr
remote addr: alice_addr

local port: mitm_port
remote port: alice_port

listening PCBs

local address: IPADDR_ANY
local port: 8080

local address: IPADDR_ANY
local port: 80

wild card

local address: IPADDR_ANY
local port: 0

#PURE_MIM

err_t lwip_connect_from_source(...)
err_t netconn_connect_from_source(...)
void do_connect_from_source(...)

tcp layer

active PCBs

local addr: 120.192.2.13
remote addr: 23.54.45.44
local port: 8080
remote port: 54353

bob side

local addr: alice_addr
remote addr: bob_addr

local port: alice_port
remote port: bob_port

alice side

local addr: bob_addr
remote addr: alice_addr

local port: mitm_port
remote port: alice_port

listening PCBs

local address: IPADDR_ANY
local port: 8080

local address: IPADDR_ANY
local port: 80

wild card

local address: IPADDR_ANY
local port: 0

mitm app

ACCEPT NEW CONNECTIONS

sys_thread_new("Manage connection", manage_connection, &info1, 0, 0);

alice side

MANAGE_RIGHT

static void manage_right(void *arg)

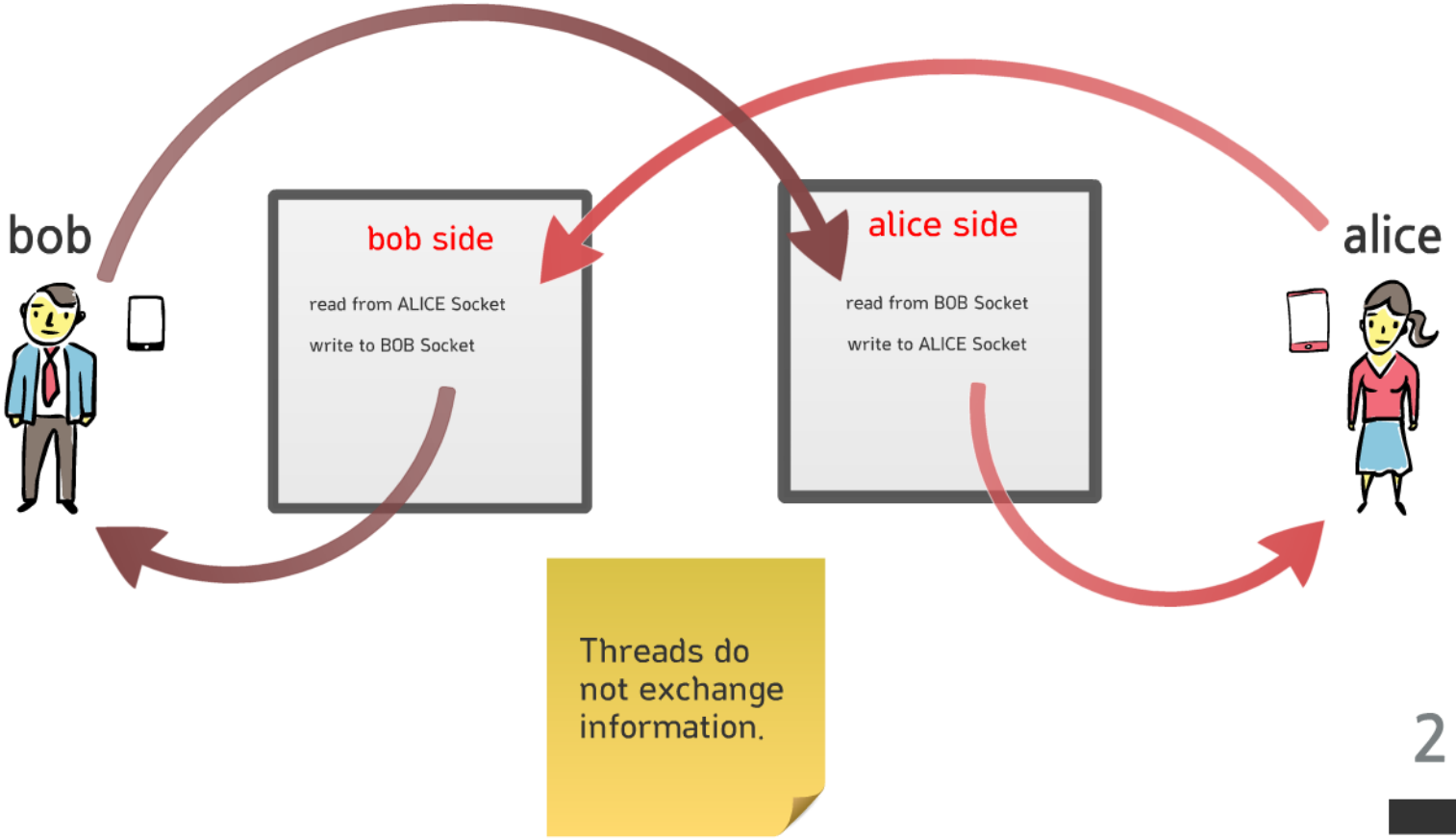
bob side

MANAGE_LEFT

static void manage_left(void *arg)

One thread for each side.

mitm app


bob
bob side
read from ALICE Socket
write to BOB Socket
alice side
read from BOB Socket
write to ALICE Socket
alice
Threads do not exchange information.

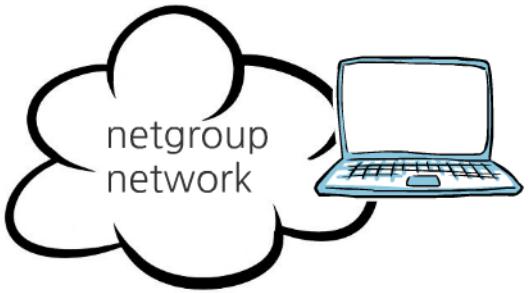
mitm test result

download of a recorded lesson

from Politecnico web server

filesize: 134MB


polito's
network

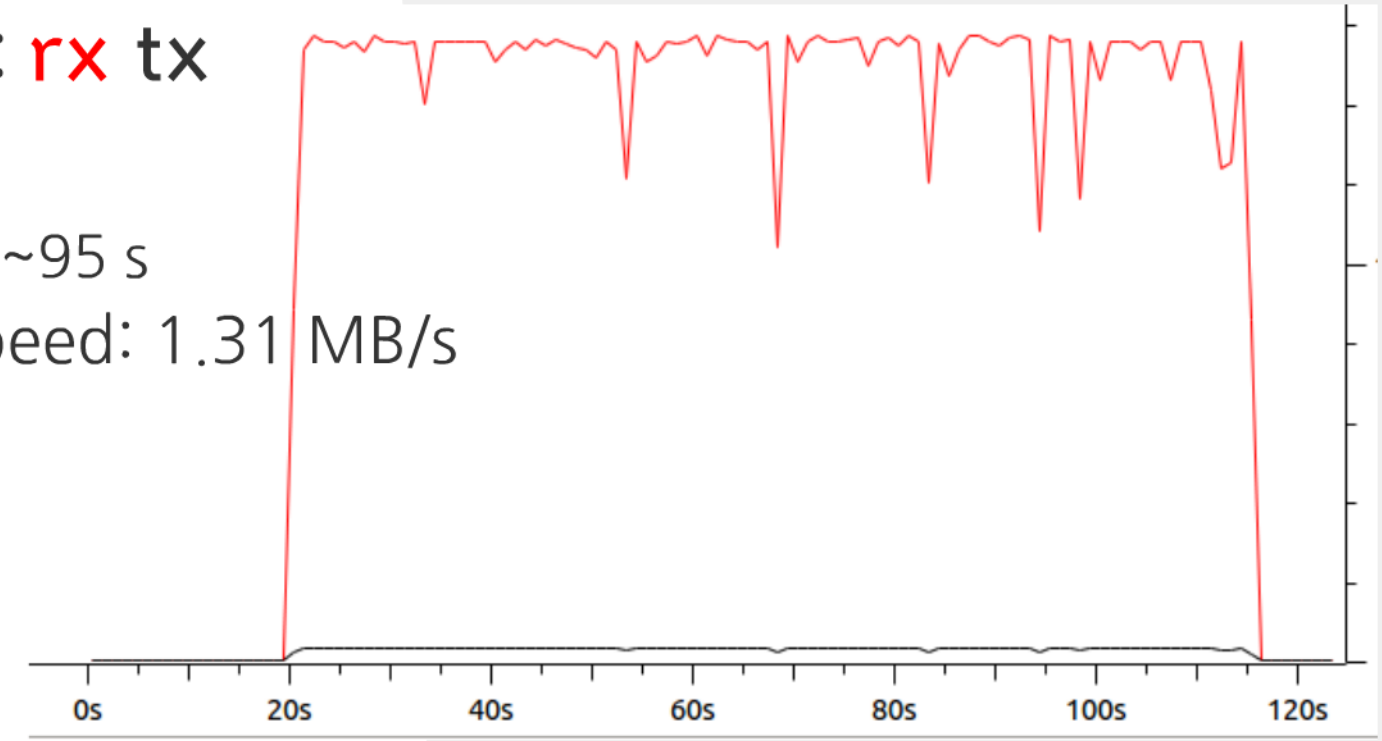

netgroup
network

mitm test result

host: rx tx

length: ~95 s

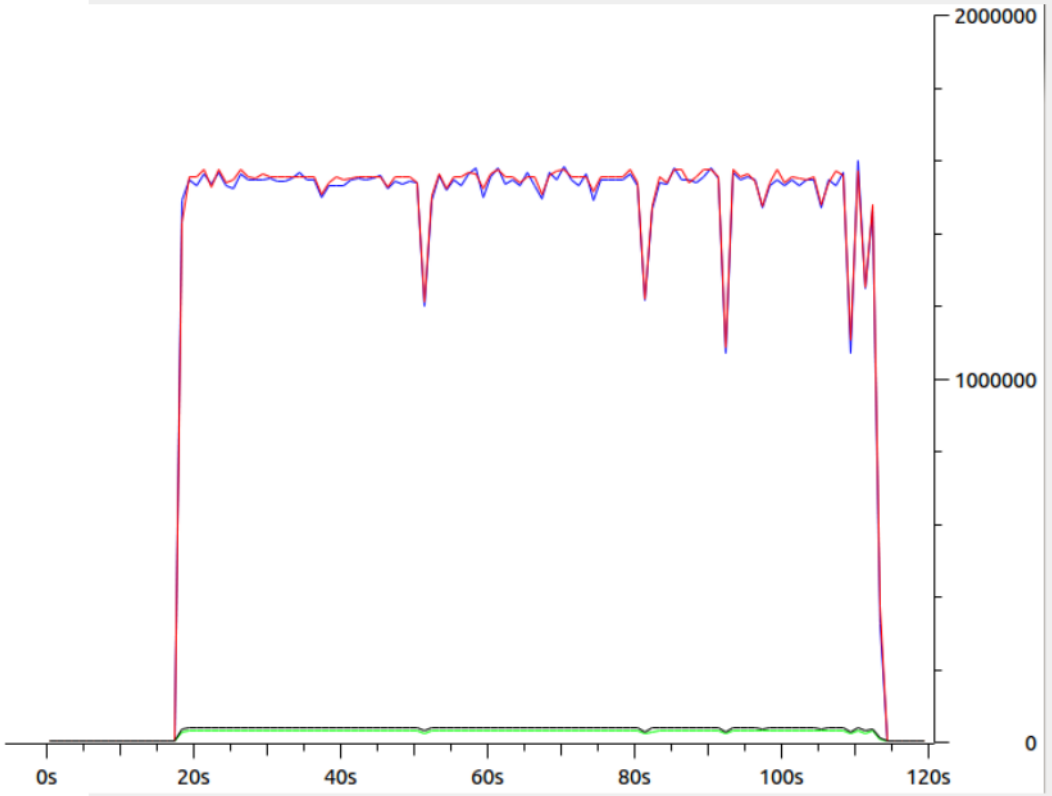
max speed: 1.31 MB/s



mitm test result

host: rx tx

server: tx rx



mitm test result

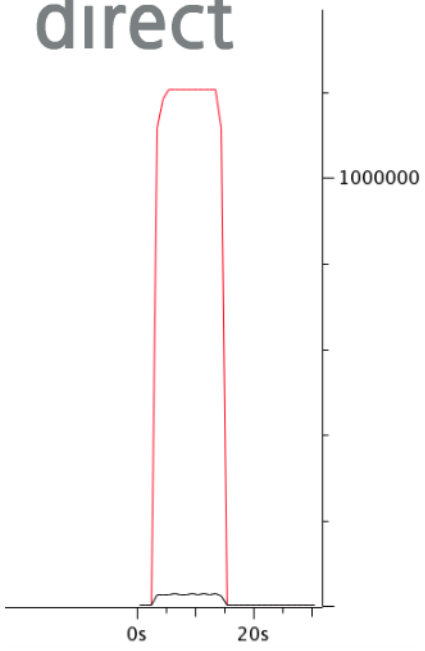
host: rx tx

length: ~ 12 s

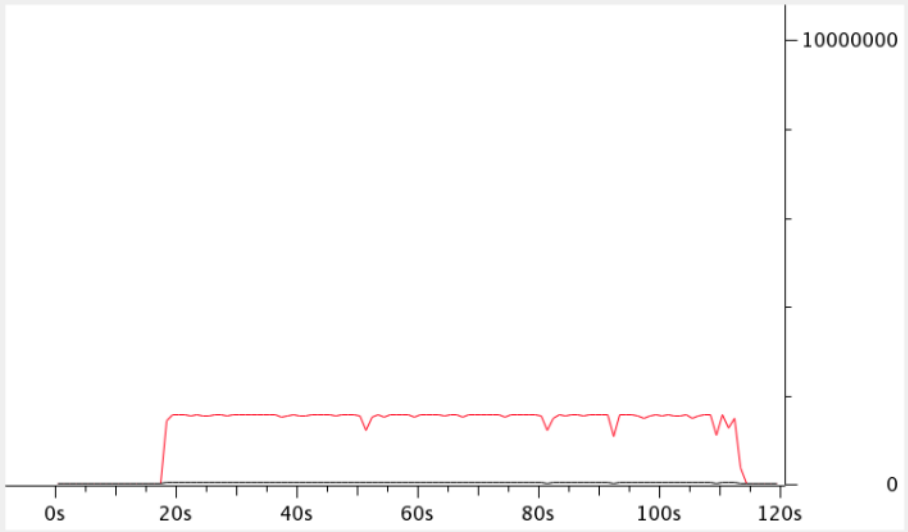
max speed: 11.2 MB/s

mitm test result

direct

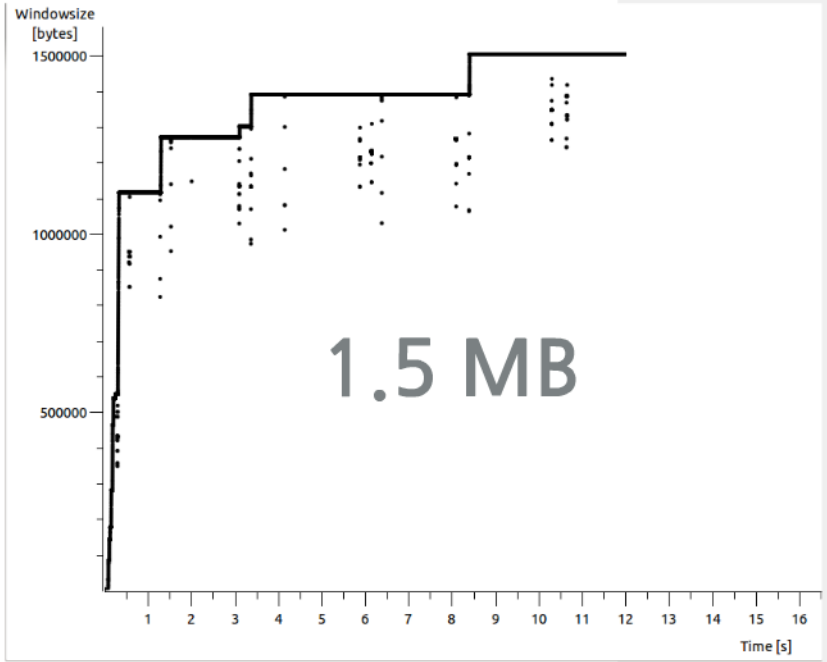


mitm

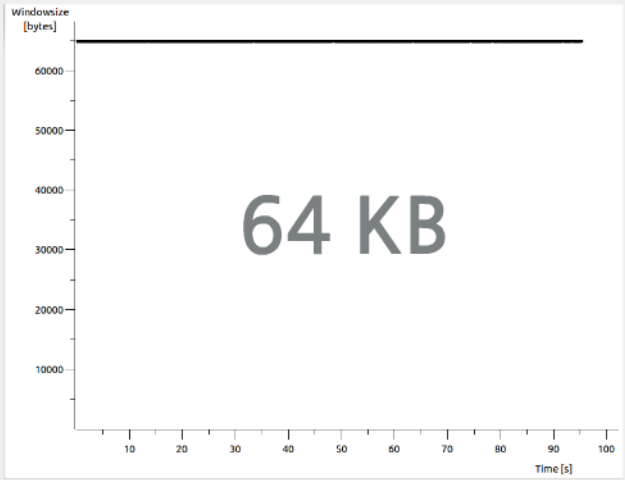


mitm test result

direct



mitm



lwip pros and cons

simplicity

30 memory

31 assert

It lacks of error handling.

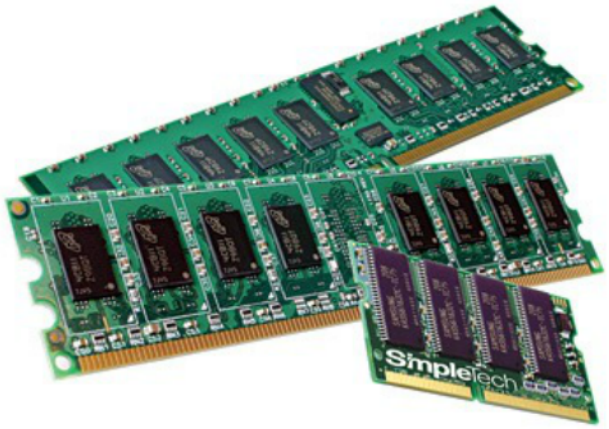
32 8 bit constants

(Max number of TCP connections, PBUF..)

33 thread

30

memory



31

assert

It lacks of error handling.

8 bit constants

(Max number of TCP connections, PBUF..)

33

thread

lwip pros and cons

http://www.ecoscentric.com/ecospro/doc/html/ref/lwip-basics-limitations.html

"No complex data structures, caches and search trees to optimise speed. Generally simple lists are used."

"The BSD sockets compatibility API does not implement all socket options, API functions, nor API semantics."

"Selective Acknowledgements (SACKs) (from RFC2018) are not provided in the TCP implementation. SACKs are a commonly implemented approach to increasing performance on links subject to packet loss, packet errors or congestion."

"Retransmission and windowing algorithms are implemented simply, at the expense of some performance."

"Thread safety (for the sequential and BSD compatibility API) is implemented in a very simple form. Individual connections should not be operated on by multiple threads simultaneously. The mutual exclusion that is provided is at a very coarse grain - the network processing operations themselves are not multi-threaded."

"Error handling for application errors is frequently only handled with asserts - used only during debug builds during development, allowing for smaller production code in release builds."

lwip summary

Experimental results proved that lwip can be used as a man in the middle user space stack.

Some bugs still has to be fixed (memory and mailbox).

Integration of every kind of stack in the FROG architecture introduces new challenges due to the non-conventional FROG's structure.

lwip question