FOSDEM 2018

IPC in 1-2-3

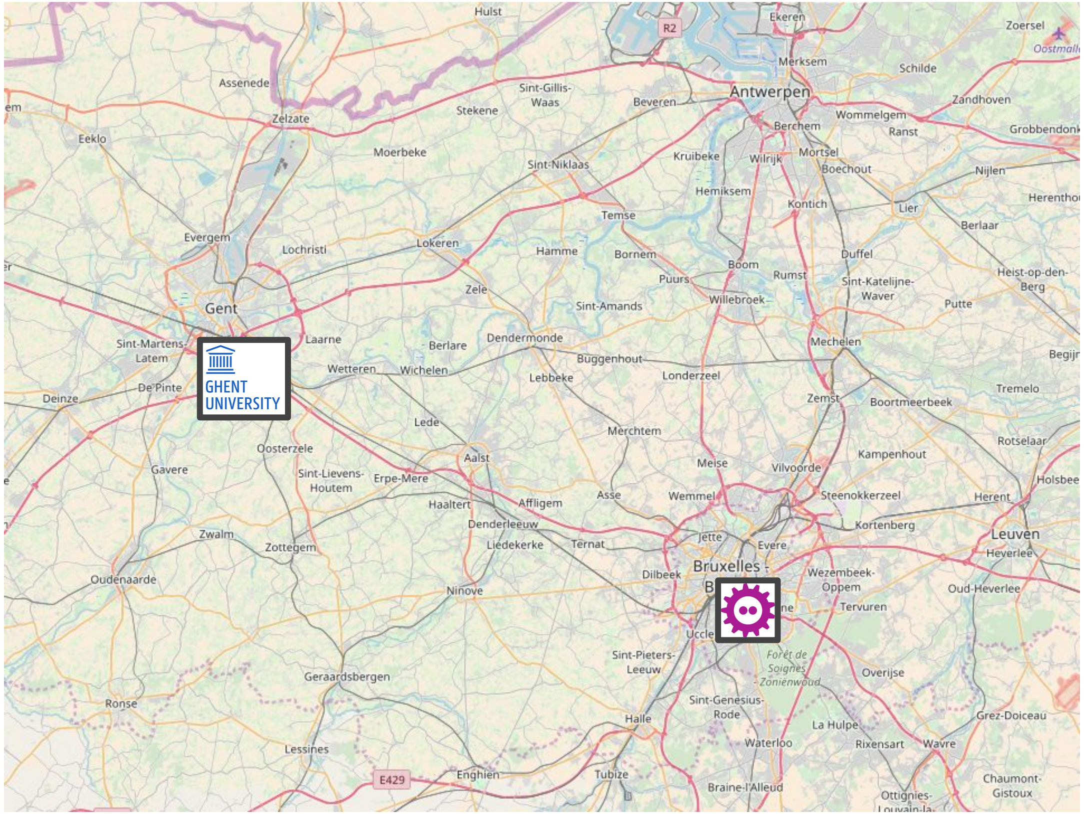
Everything you always wanted from a network

(but were afraid to ask)

Dimitri Staessens
Sander Vrijders

IDLab
INTERNET & DATA LAB
imec
GHENT UNIVERSITY

Who are we?



© OpenStreetMap contributors

Next Generation Internet

IRATI
INVESTIGATING RINA

Pristine

FIRE
Future Internet Research and Experimentation

HORIZON 2020

ARCFIRE
Experimenting with RINA

geni
Exploring Networks of the Future

Internet reliability?

DDoS attack that disrupted internet was largest of its kind in history, experts say

Dyn, the victim of last week's denial of service attack, said it was orchestrated using a weapon called the Mirai botnet as the 'primary source of malicious attack'

Major cyber attack disrupts internet service across Europe and US

Source: The Guardian

Internet boffins take aim at BGP route leaks

Routers should know their place

By Richard Chirgwin 19 Jun 2017 at 03:57 12 SHARE ▼

ent bugs in internet infrastructure, route leaks in tocol (BGP), is in the sights of a group of 'net ternet-Draft.

et's persistent trouble-spots: ineradicable because erable because it's ancient, a relic of a collegiate s knew each other by name.

Source: The Register

BUSINESS

BGP Routing Table Size Limit Blamed for Tuesday's Website Outages

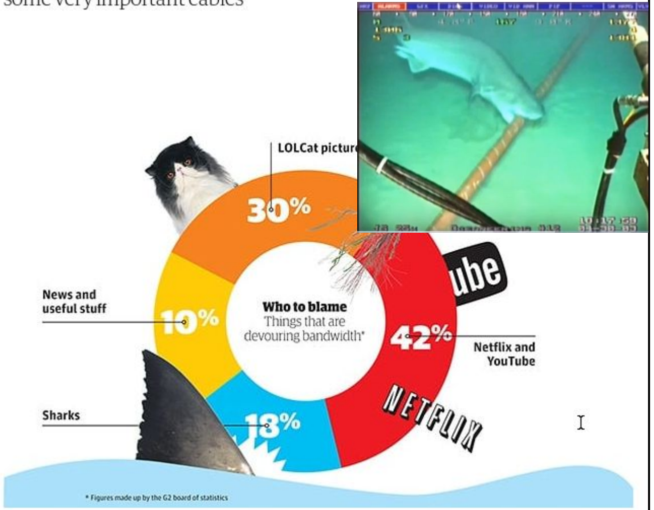
Numerous service pro maintainin global BGP reached a (Photo by S

Many service providers experience outages, causing downtime for hosting firms and customer websites.

Source: DataCenter Knowledge

The internet is broken. You can blame sharks. And Netflix

Don't panic, the situation isn't as awful as it sounds, but online space is running out and undersea creatures are nibbling at some very important cables



What's eating the internet? Photograph: /Guardian

Bad news: The internet ran out of space on Tuesday. Worse news: Sharks are eating what is left.

s it sounds. Yet. But there are some existential threats n, and there are worse ways of putting it than to point ll up.

RUSSIAN SHIPS COULD CAUSE 'CATASTROPHE' FOR WEST BY CUTTING TRANSATLANTIC INTERNET CABLES

BY DAMIEN SHARKOV ON 12/15/17 AT 5:08 AM

Source: Newsweek

Source: The Guardian

Internet security?

Massive Phishing Attack Targets Gmail Users

by Alex Johnson / May.04.2017 / 7:19 PM ET

Google apps on an Android smartphone, including Chrome, Gmail and

Source: NBC News

The Guardian view on internet security: a huge and growing problem

Editorial

The power of smartphones is too easily turned against their users. Governments, companies and users must all work together to keep themselves safe

Source: The Guardian

The Internet Is Mostly Bots

More than half of web traffic comes from automated programs—many of them malicious.

Source: The Atlantic

Hacker makes $84k hijacking Bitcoin mining pool

Researchers investigated after their own Bitcoin mining pool was tapped, though how hackers accessed ISP infrastructure is still not known

erated $84,000 worth of the Bitcoin cryptocurrency by gaini
nadian internet provider and diverting the computing powe
n "mines".

Source: The Guardian

Heartbleed bug: What you need to know

By Jane Wakefield
Technology reporter

10 April 2014

Share

This week it has emerged that a major security flaw at the heart of the internet may have been exposing users' personal information and passwords to hackers for the past two years.

It is not known how widely the bug has been exploited, if at all, but what is clear is that it is one of the biggest security issues to have faced the internet to date.

Security expert Bruce Schneier described it as "catastrophic". He said: "On the scale of one to 10, this is an 11."

The BBC has attempted to round up everything you need to know about Heartbleed.

CODENOMICON

The bug could be a huge problem

Source: BBC

Internet privacy?

Edward Snowden: Leaks that exposed US spy programme

17 January 2014

f Share

Edward Snowden, a former contractor for the CIA, left the US in late May after leaking to the media details of extensive internet and phone surveillance by American intelligence. Mr Snowden, who has been granted temporary asylum in Russia, faces espionage charges over his actions.

As the scandal widens, BBC News looks at th
activities to light.

US spy agency 'collects phone records'

Source: BBC

They Have, Right Now, Another You

Sue Halpern

DECEMBER 22, 2016 ISSUE

Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy
by Cathy O'Neil
Crown, 259 pp., $26.00

Virtual Competition: The Promise and Perils of the Algorithm-Driven Economy
by Ariel Ezrachi and Maurice E. Stucke
Harvard University Press, 356 pp., $29.95

Source: New York review of books

Encryption Won't Stop Your Internet Provider From Spying on You

Data patterns alone can be enough to give away what video you're watching on YouTube.

Source: The Atlantic

PGP creator Phil Zimmermann: 'Intelligence agencies have never had it so good'

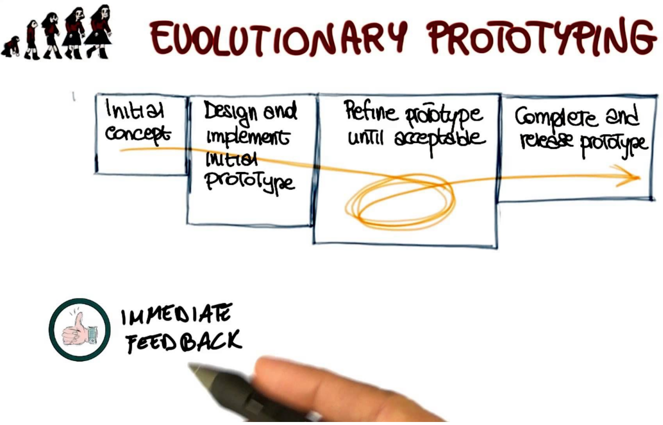
Encryption expert says Sony Pictures hack shows why companies should value privacy as well as security, too, and is unimpressed by David Cameron's encryption stance

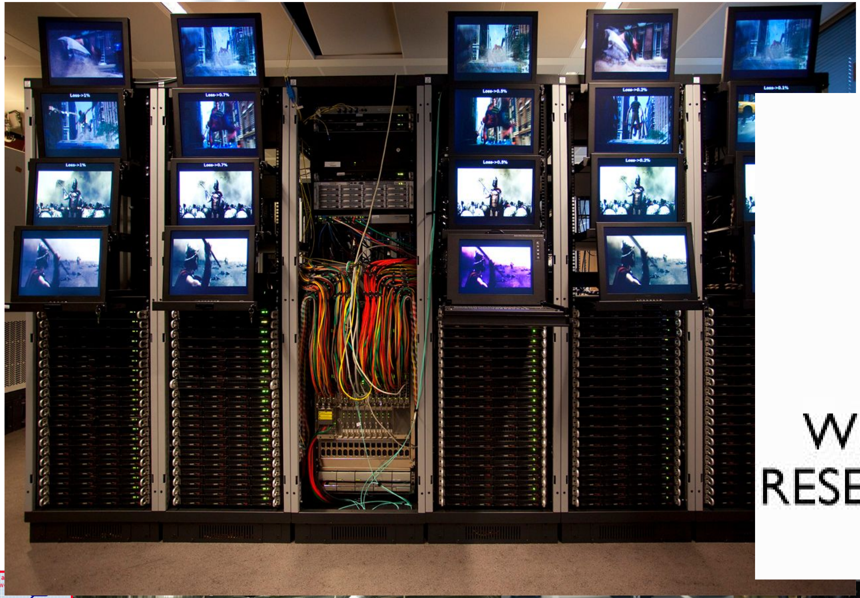
Source: The Guardian

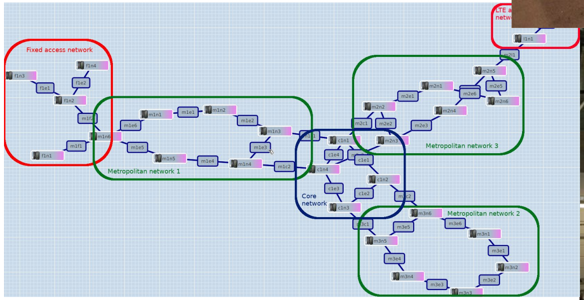
Verizon and AT&T accused of selling your phone number and location to almost anyone

Source: Android Authority

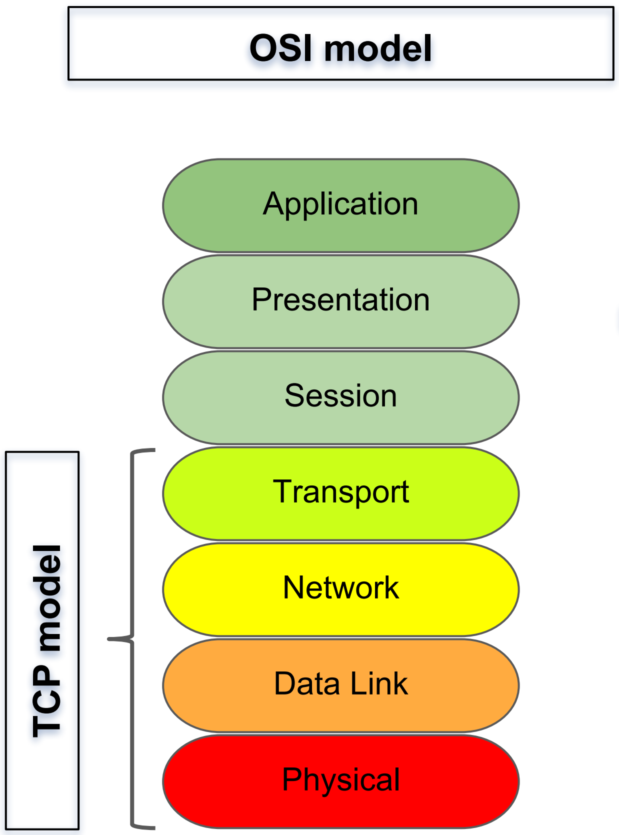
Our research methodology





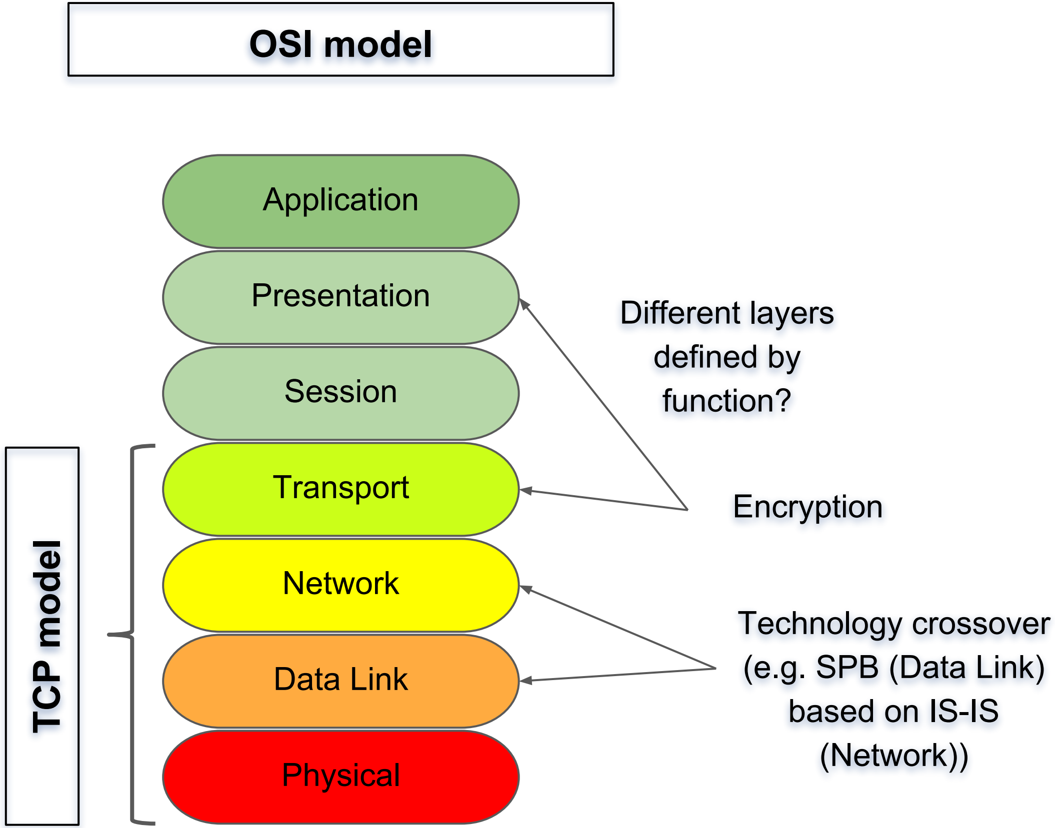


Network Architectures

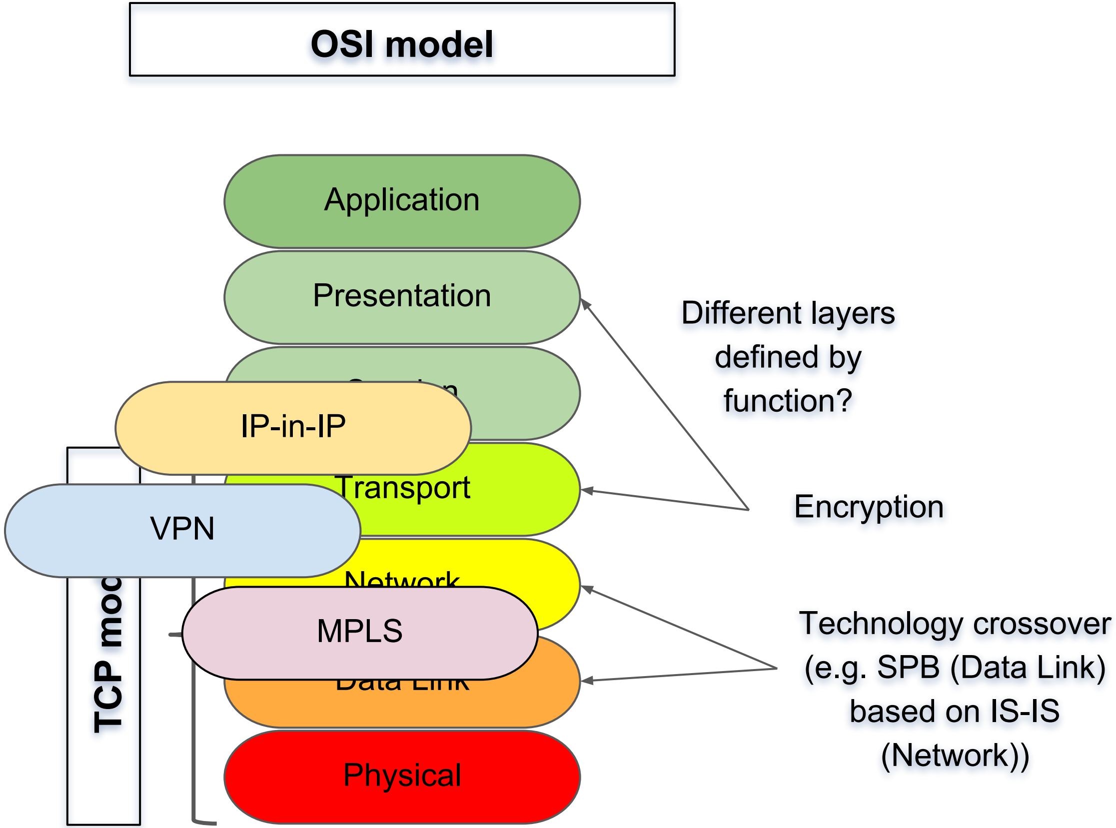


Different layers
defined by
function

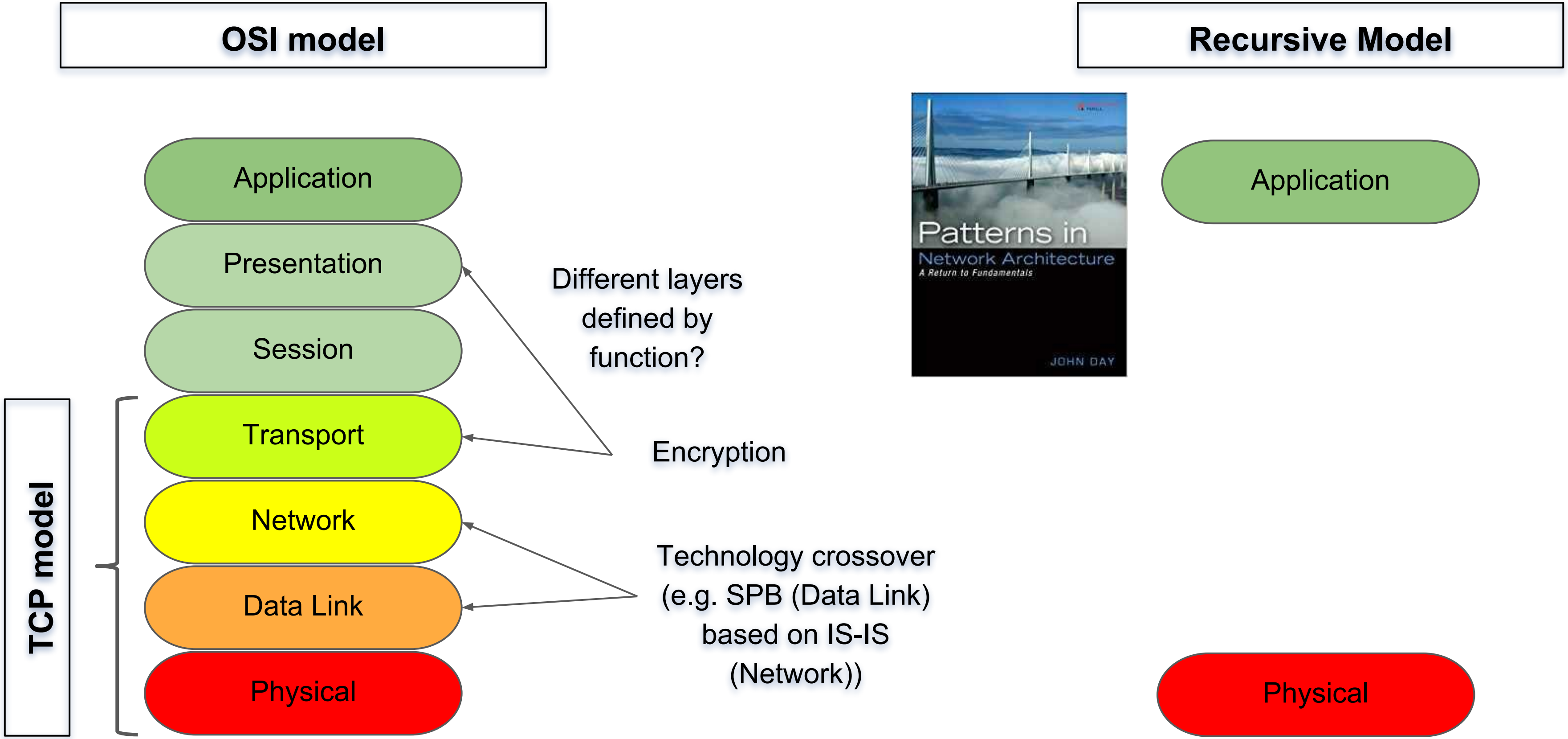
Network Architectures



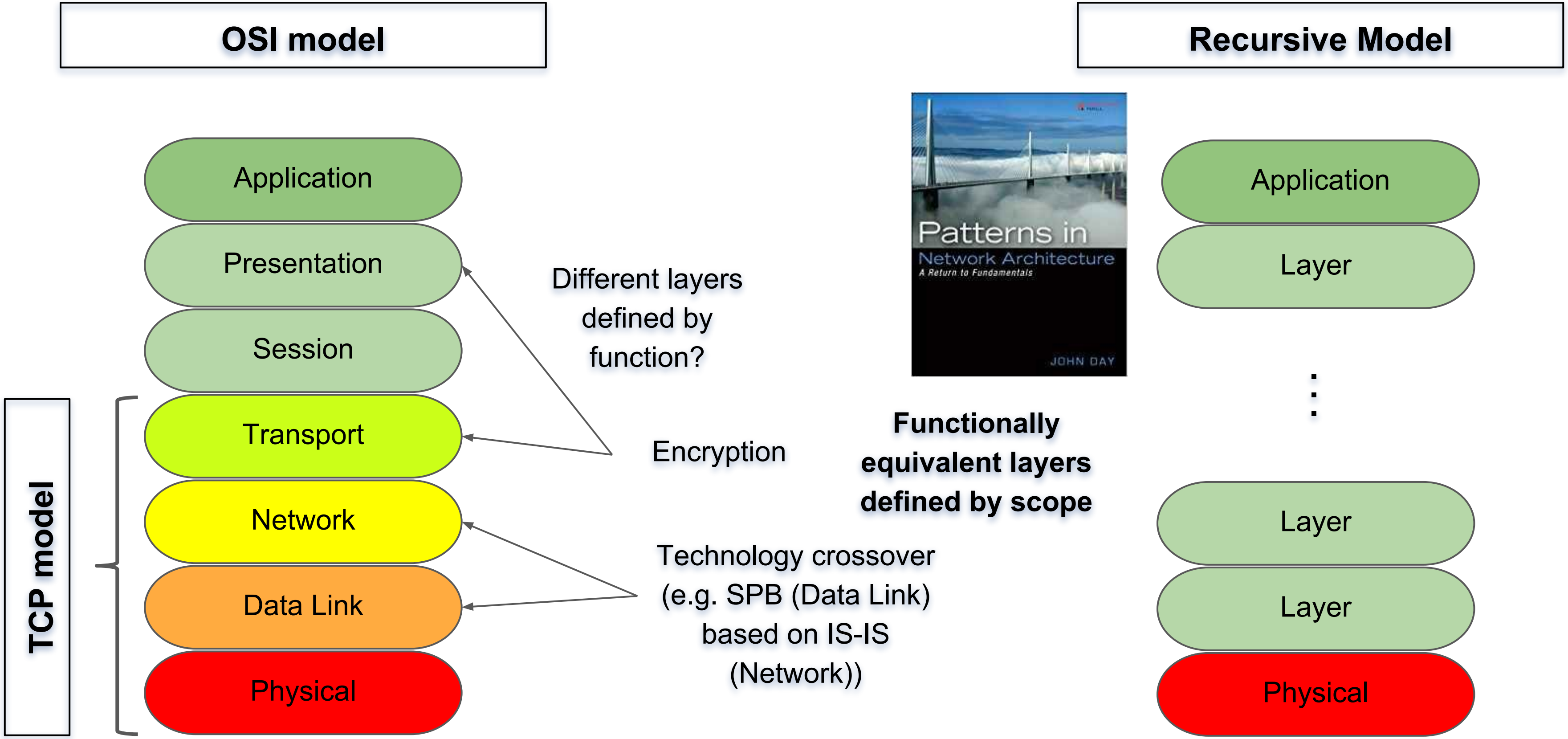
Network Architectures



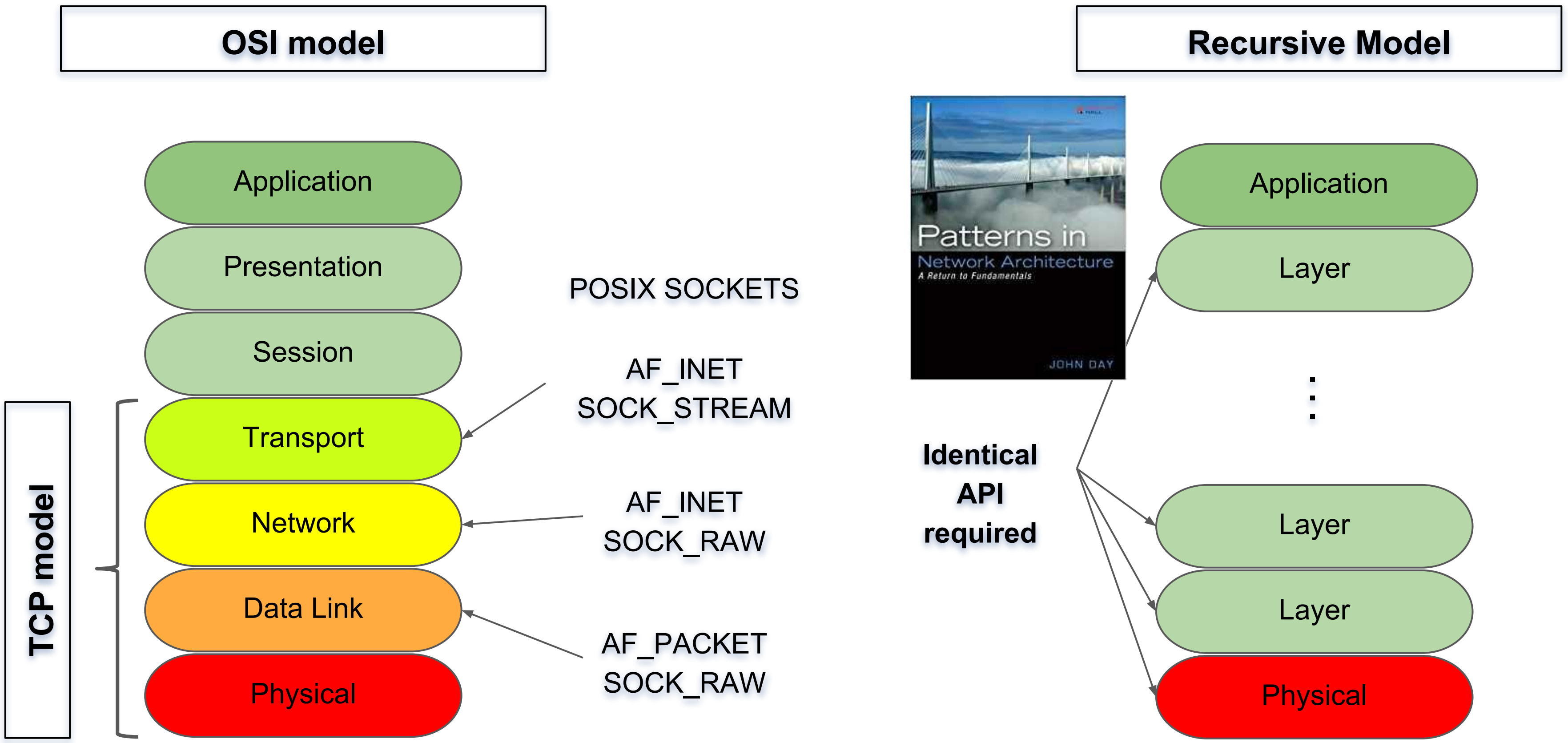
Network Architectures



Network Architectures



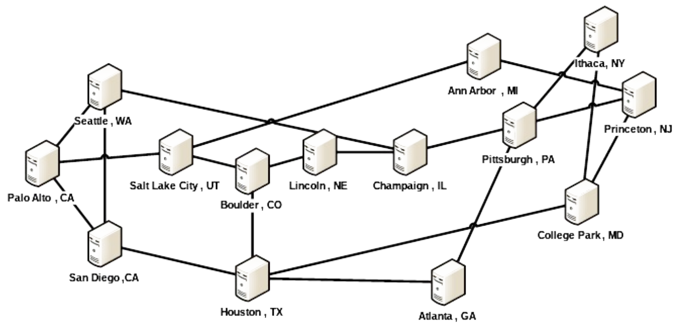
Network APIs



Ouroboros

https://ouroboros.ilabt.imec.be

What is Ouroboros?

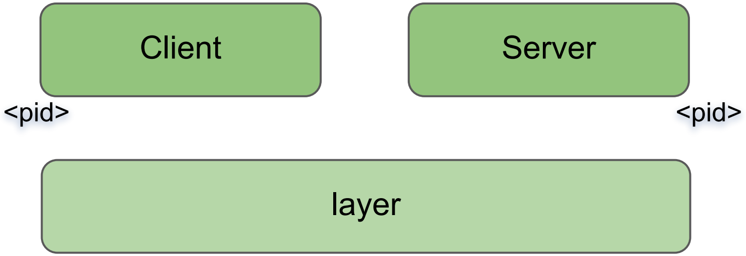


- A decentralised packet switched network
- Redesigned from the ground up, following a recursive model
- That blurs the differences between LANs, MANs, WANs and VPNs
- And provides you better services than you are used to from TCP and UDP
- With increased privacy, security and anonymity
- Using the simplest network API known to (this) man

Ouroboros API

Server

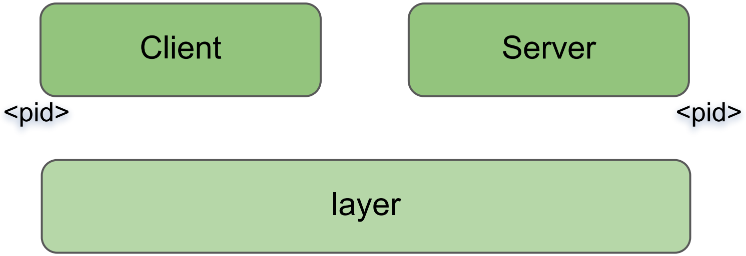
Client

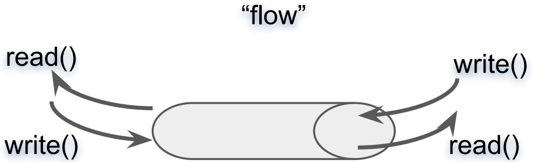


Ouroboros API

Server

Client

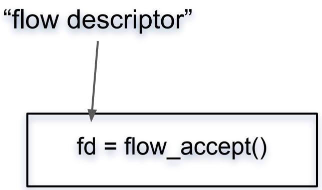


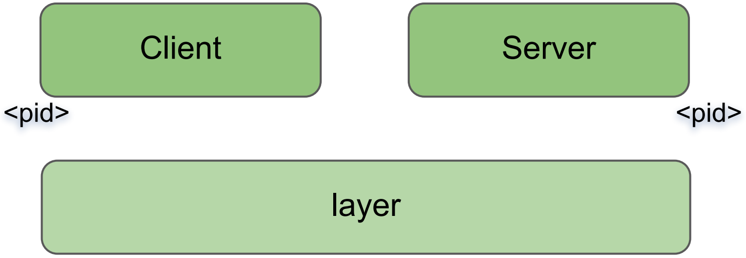


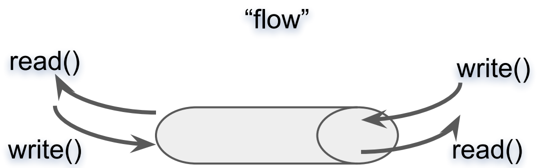
Ouroboros API

Server

Client







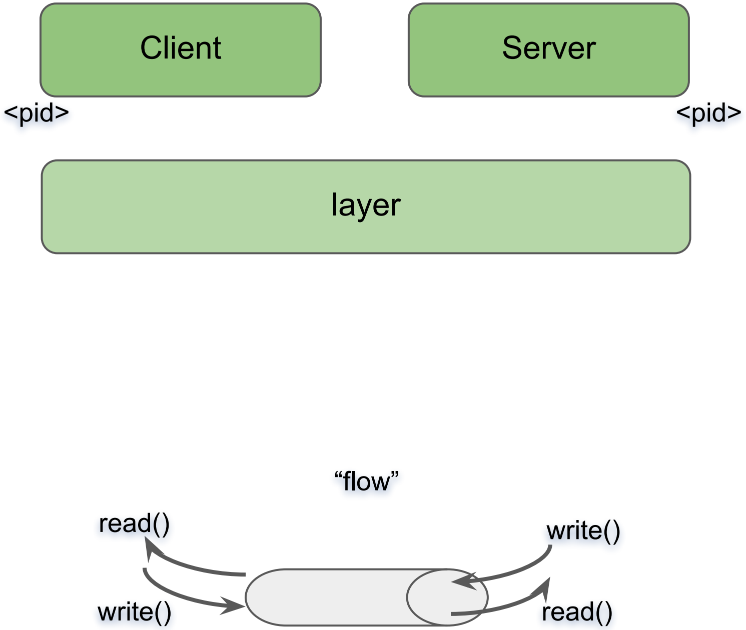
Ouroboros API

Server

Client

fd = flow_accept()

fd = flow_alloc(pid)



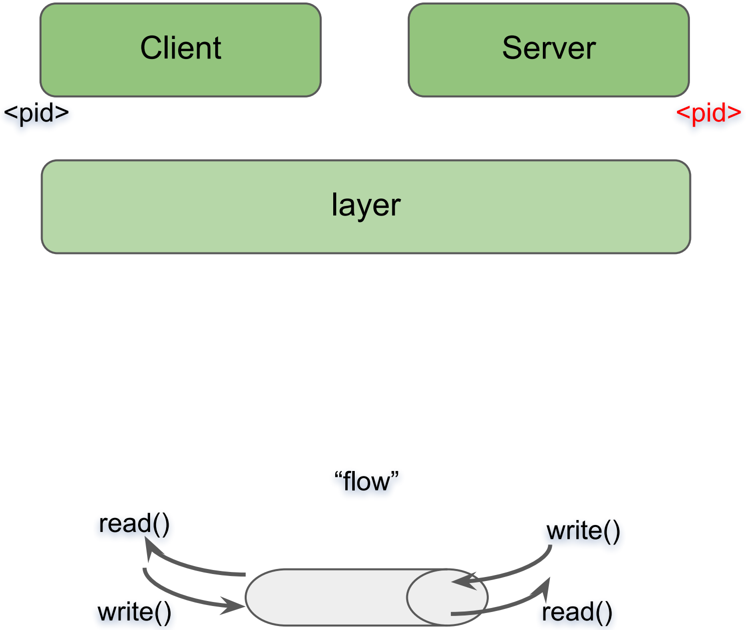
Ouroboros API

Server

Client

fd = flow_accept()

fd = flow_alloc(pid)



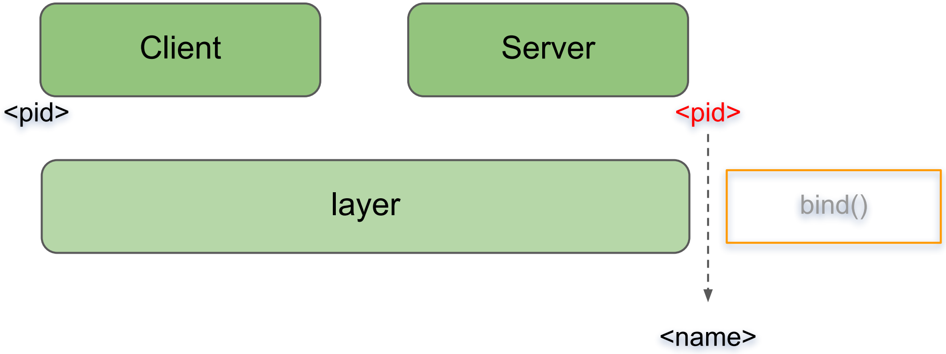
Ouroboros API

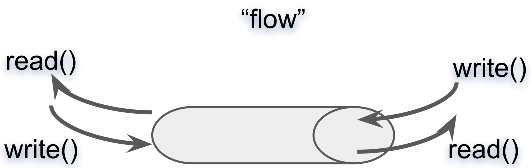
Server

Client

fd = flow_accept()

fd = flow_alloc(pid)





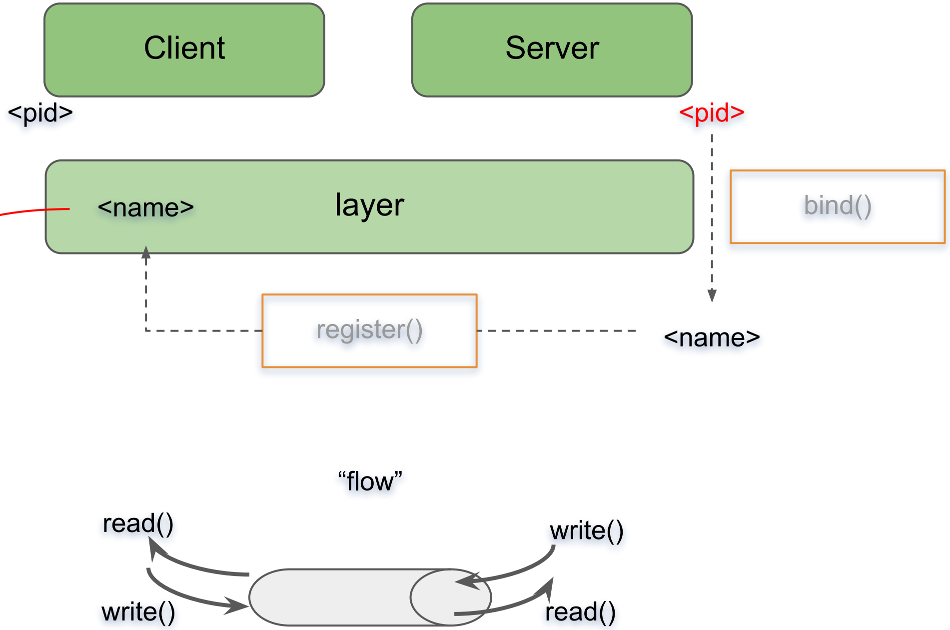
Ouroboros API

Server

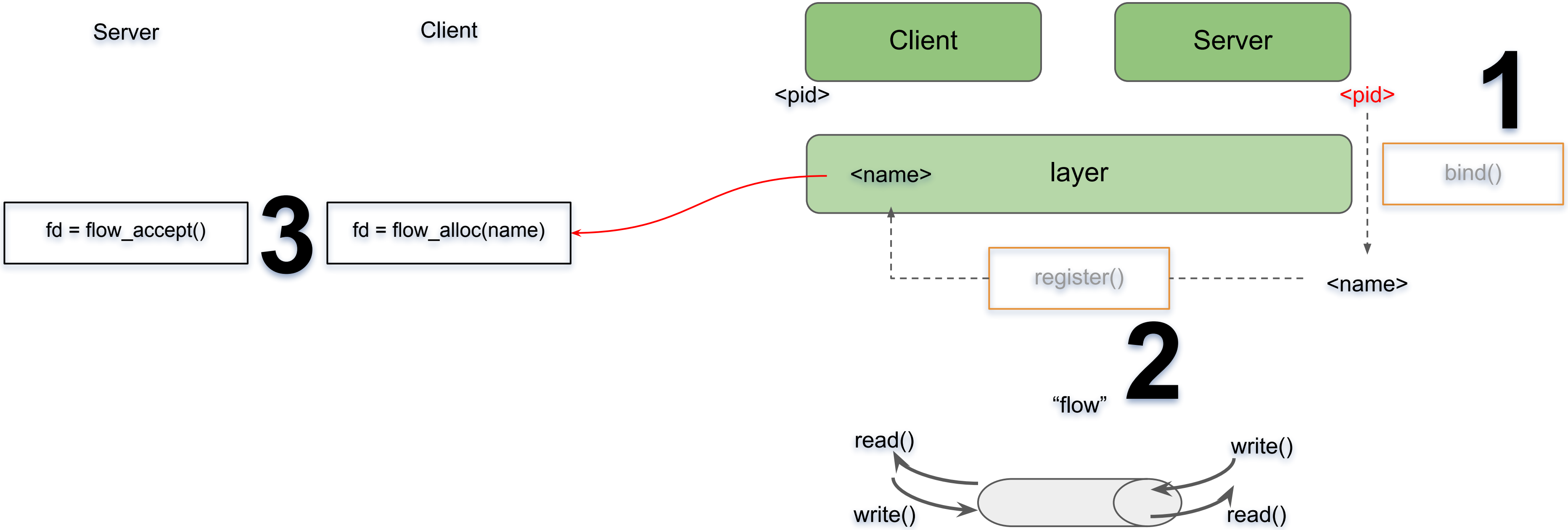
Client

fd = flow_accept()

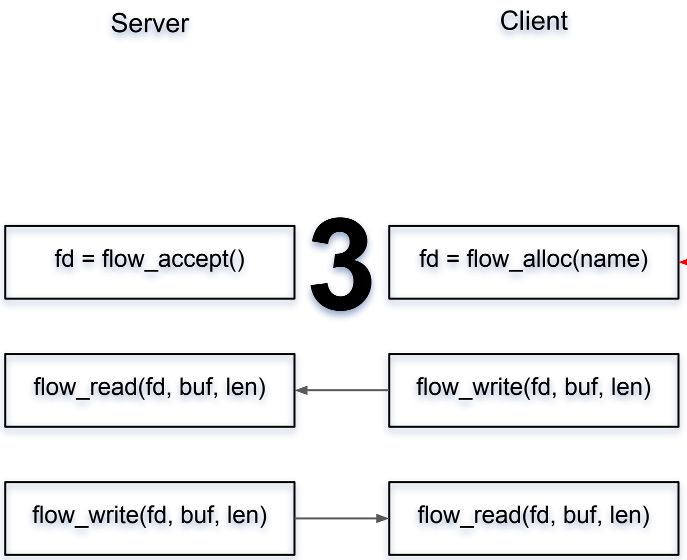
fd = flow_alloc(name)

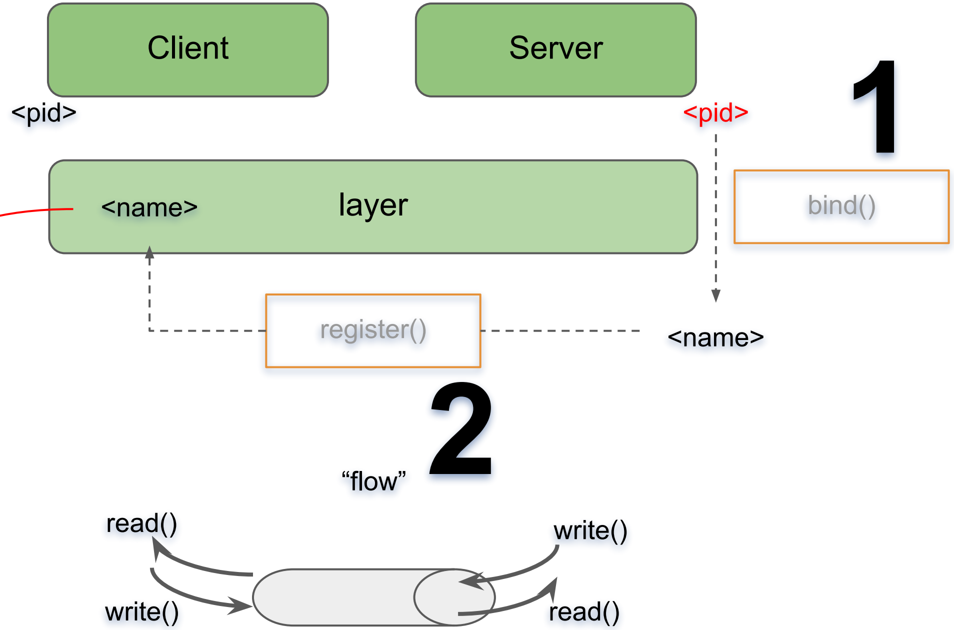


Ouroboros API

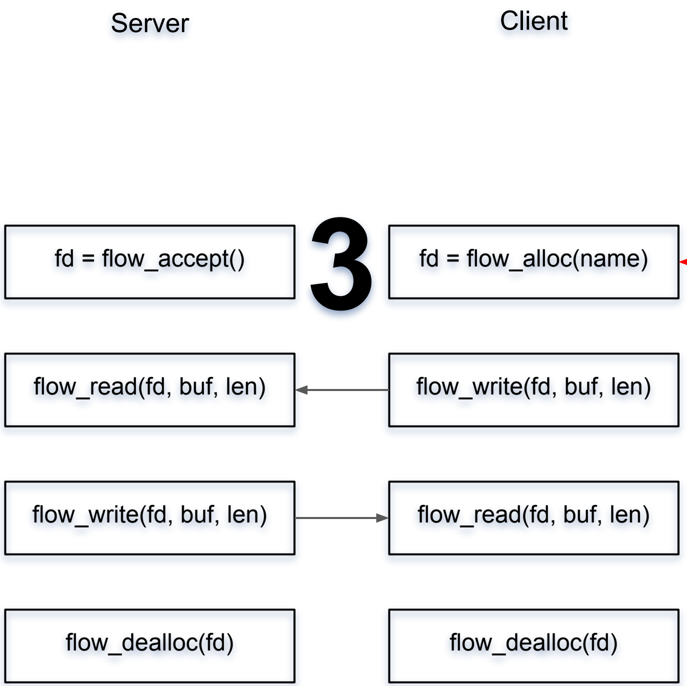


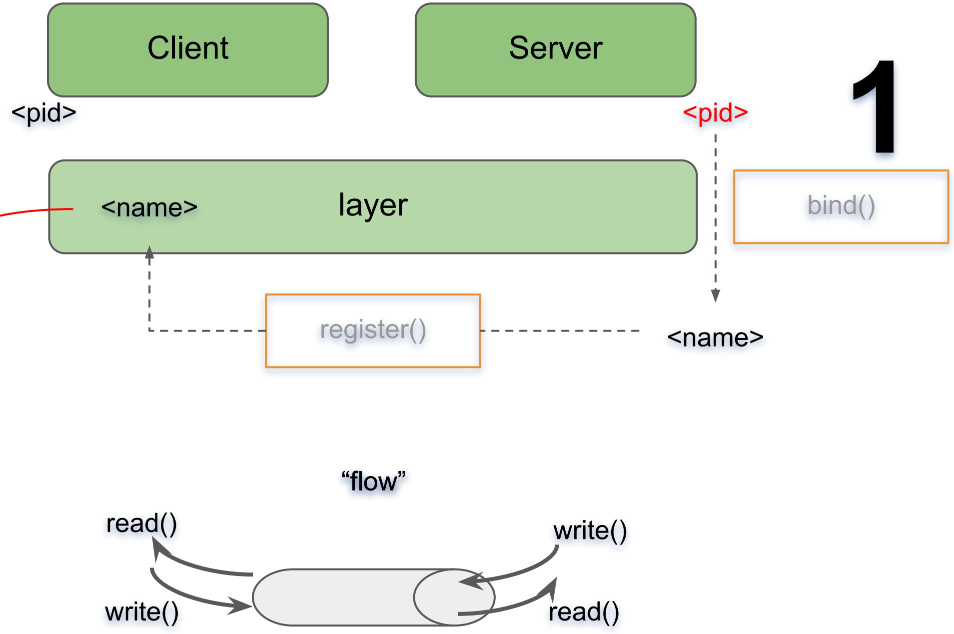
Ouroboros API



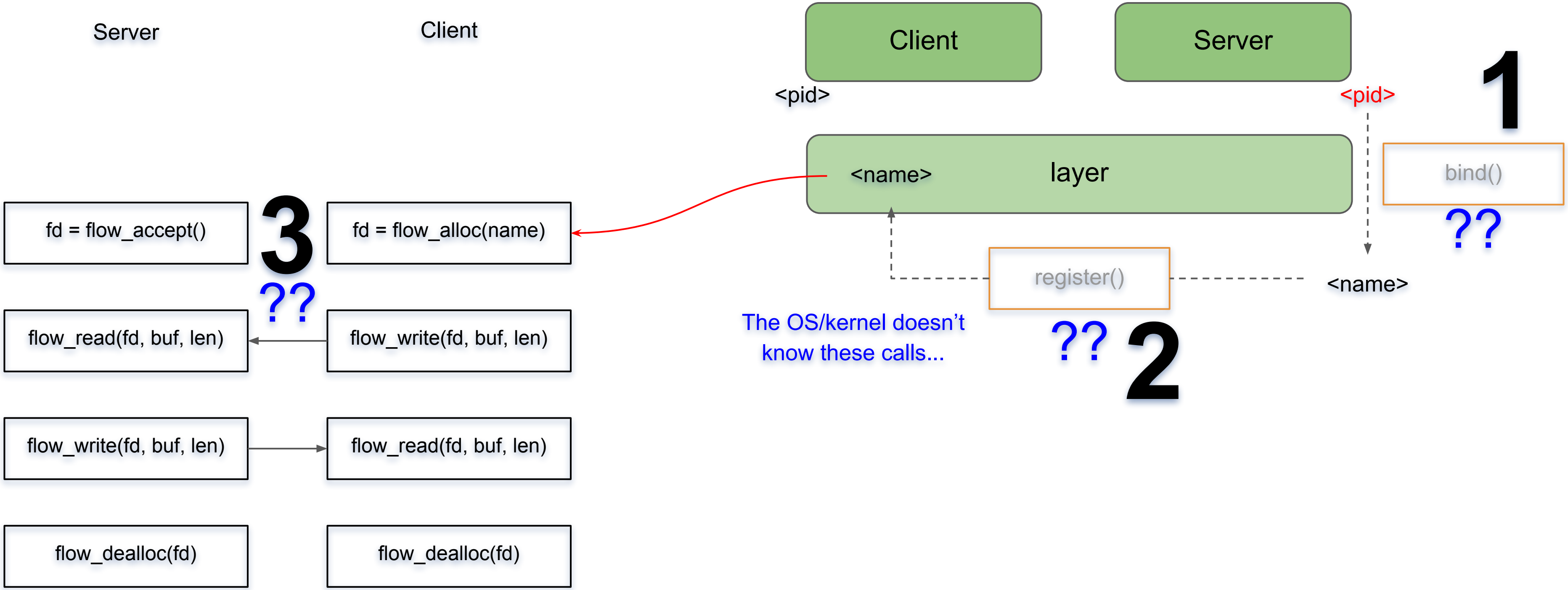


Ouroboros API

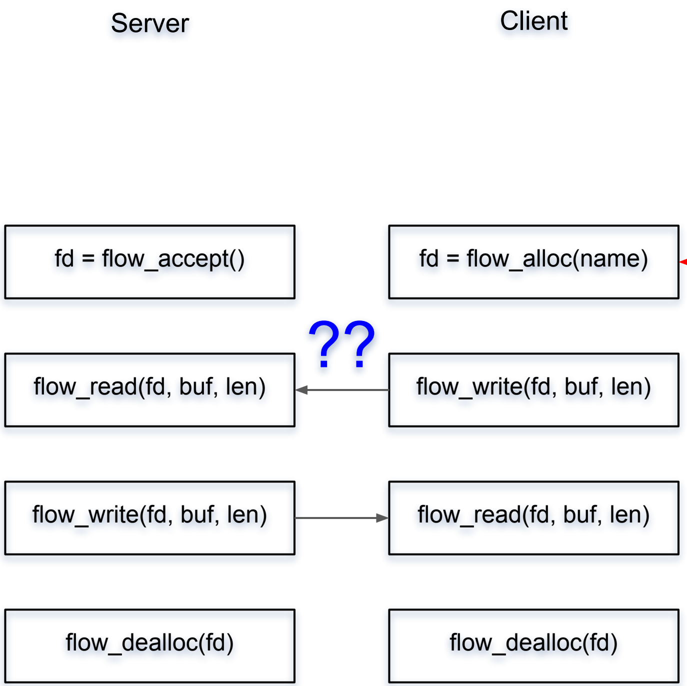


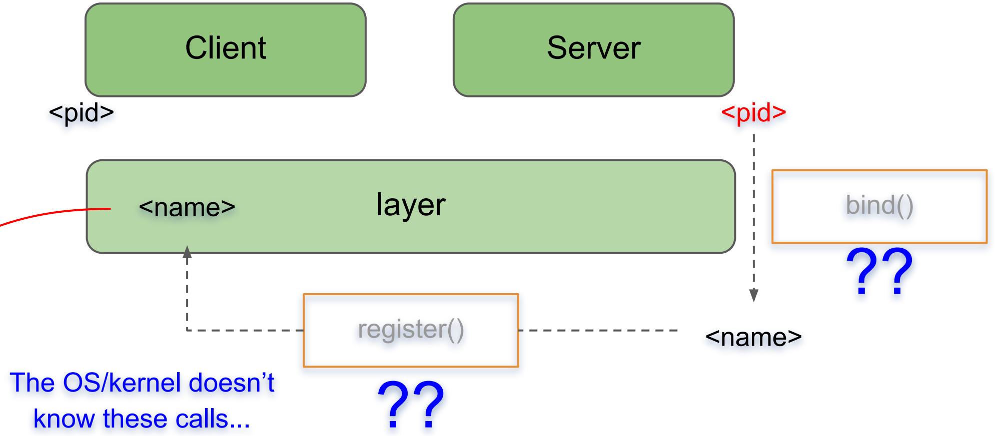


Ouroboros API



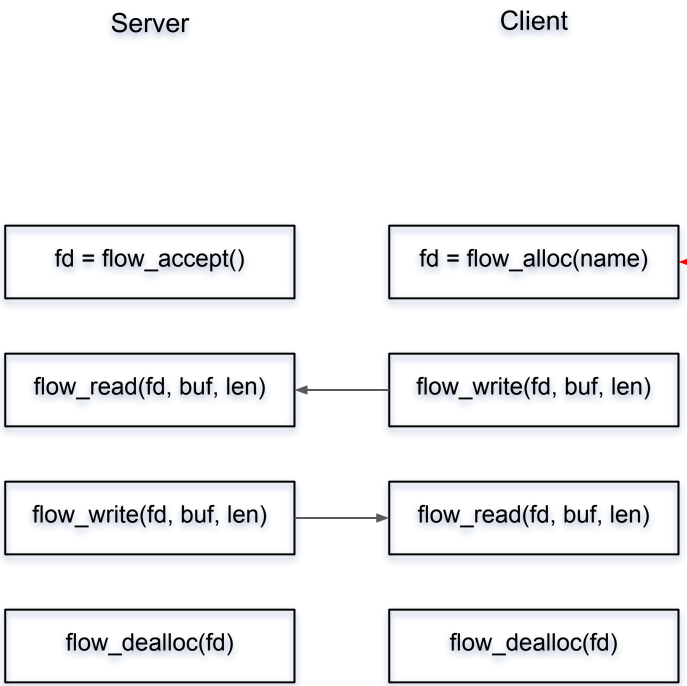
Ouroboros API

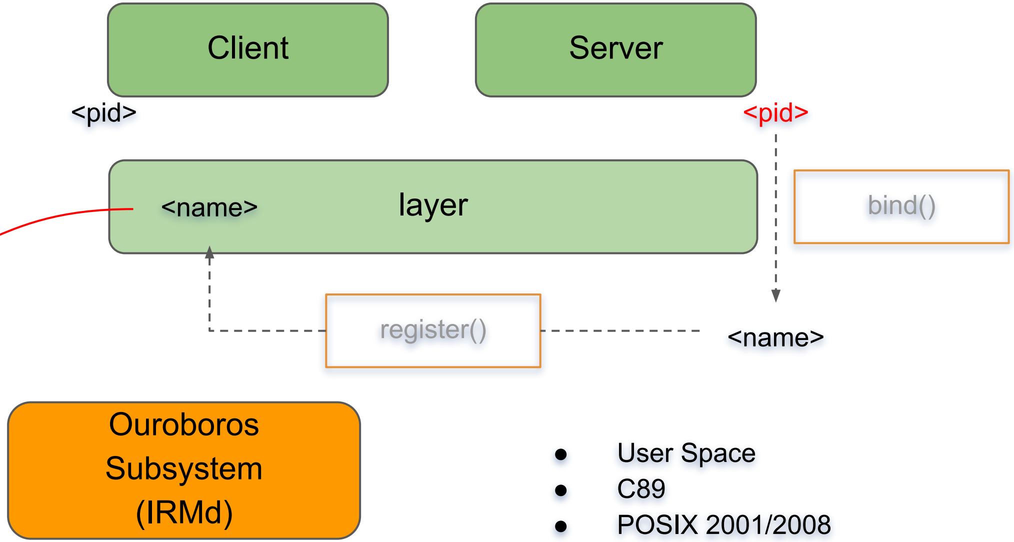




- User Space
- C89
- POSIX 2001/2008

Ouroboros API





- User Space
- C89
- POSIX 2001/2008

```
$ sudo irmd --stdout
irmd(II): Ouroboros IPC Resource Manager daemon started…
```

```
$ systemctl enable irmd
$ sudo journalctl -t irmd
-- Logs begin at Thu 2018-02-01 16:55:03 CET, end at Fri 2018-02-02 14:11:31 CET. --
Feb 02 14:10:06 phoneutria irmd[5243]: irmd: Ouroboros IPC Resource Manager
daemon started...
```

```c
int server_main(void)
{
        int         fd = 0;
        char        buf[BUF_SIZE];
        ssize_t count = 0;

        printf("Starting the server.\n");

        while (true) {
                fd = flow_accept(NULL, NULL);
                if (fd < 0) {
                        printf("Failed to accept flow.\n");
                        break;
                }

                printf("New flow.\n");

                count = flow_read(fd, &buf, BUF_SIZE);
                if (count < 0) {
                        printf("Failed to read SDU.\n");
                        flow_dealloc(fd);
                        continue;
                }

                printf("Message from client is %.*s.\n", (int) count, buf);

                if (flow_write(fd, buf, count) == -1) {
                        printf("Failed to write SDU.\n");
                        flow_dealloc(fd);
                        continue;
                }

                flow_dealloc(fd);
        }

        return 0;
}
```

"Client says hi!"

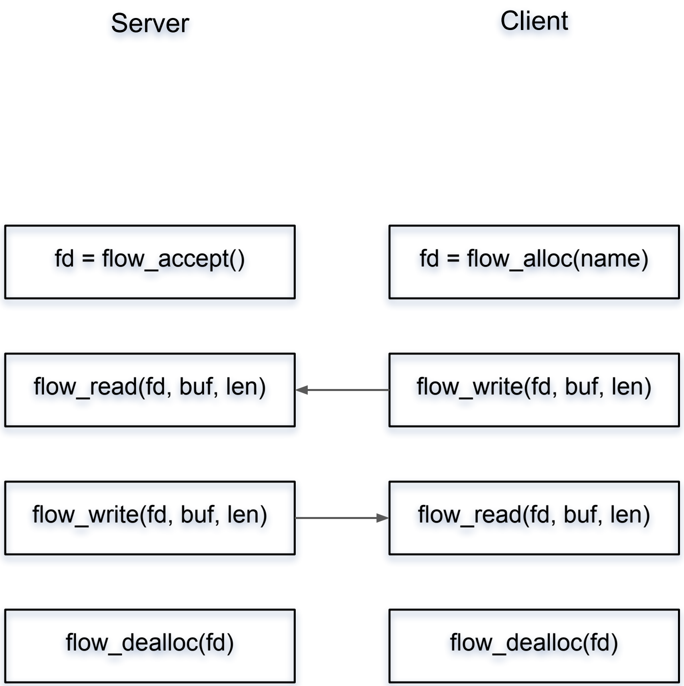
"Client says hi!"

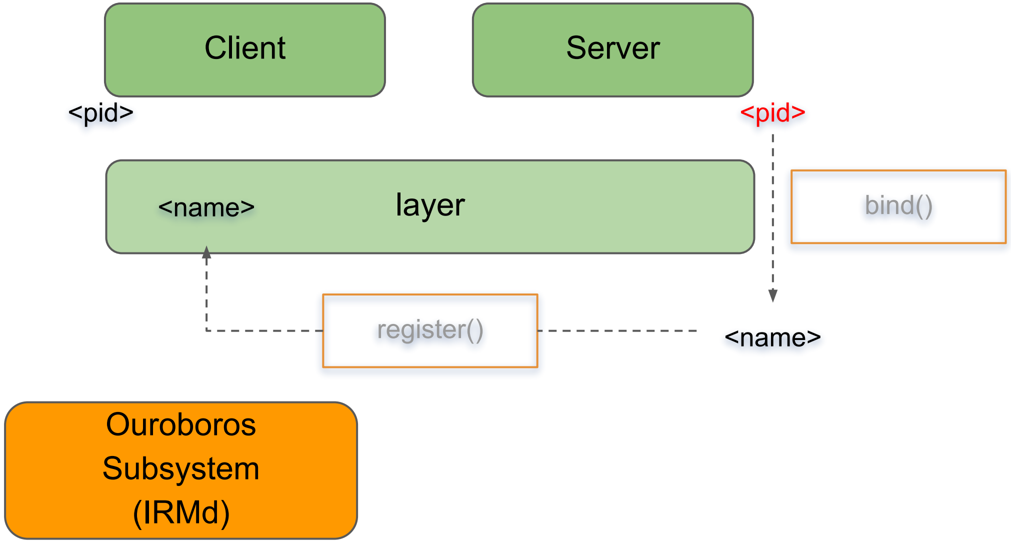
```
$ echo-app -l
Starting the server.
New flow.
Message from client is Client says hi!.
```

```c
int client_main(void)
{
        int         fd          = 0;
        char        buf[BUF_SIZE];
        char *  message = "Client says hi!";
        ssize_t count   = 0;

        fd = flow_alloc("echo", NULL, NULL);
        if (fd < 0) {
                printf("Failed to allocate flow.\n");
                return -1;
        }

        if (flow_write(fd, message, strlen(message) + 1) < 0) {
                printf("Failed to write SDU.\n");
                flow_dealloc(fd);
                return -1;
        }

        count = flow_read(fd, buf, BUF_SIZE);
        if (count < 0) {
                printf("Failed to read SDU.\n");
                flow_dealloc(fd);
                return -1;
        }

        printf("Server replied with %.*s\n", (int) count, buf);

        flow_dealloc(fd);

        return 0;
}
```

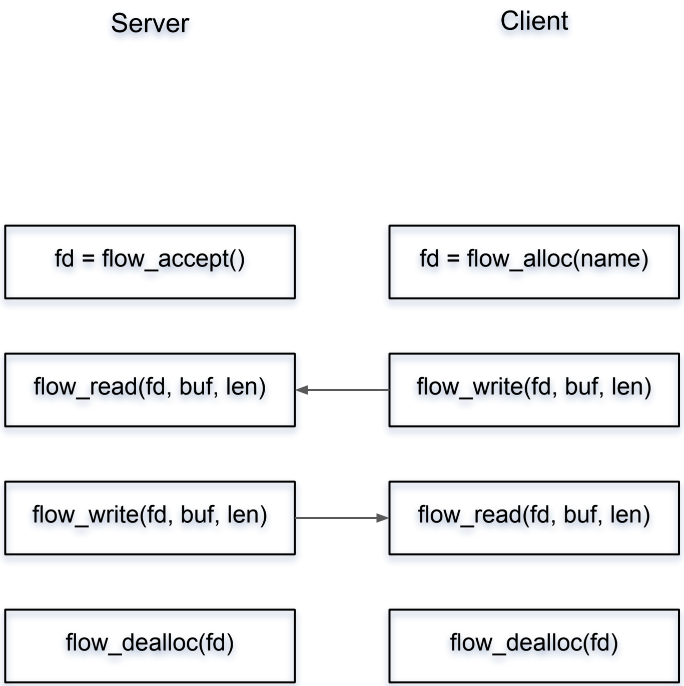
```
$ echo-app
Server replied with Client says hi!
```

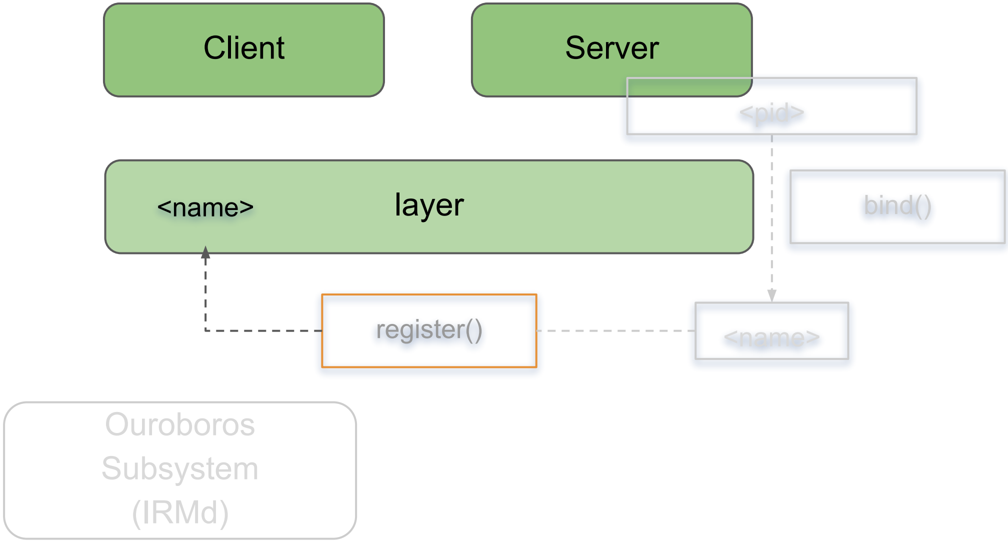
Functions of a layer





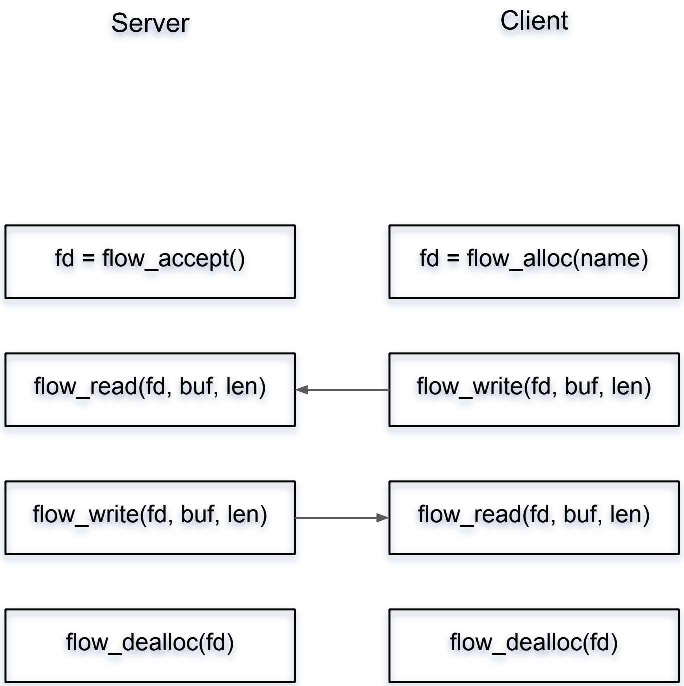
Functions of a layer

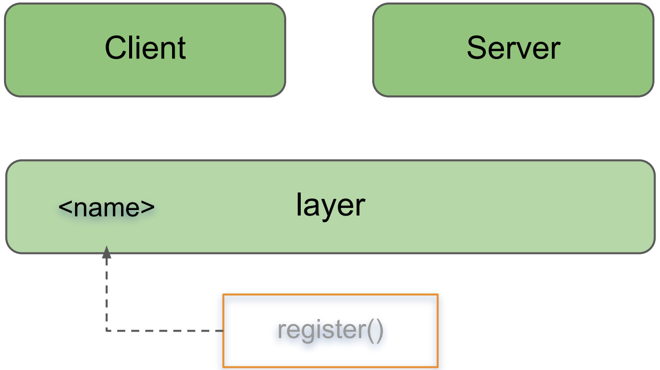




NB1) The bind operation is local to the IRMd

Functions of a layer





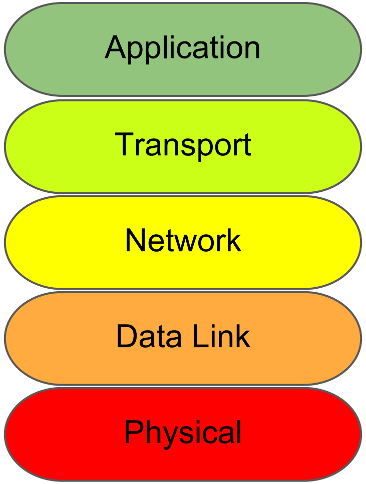
- Keep track and figure out where there are endpoints for a certain name: **directory**
- Figure out how to get packets from one point to another: **routing**
- Forward the packets from one point to another: **forwarding**
- Allocate and release resources: **flow allocation/deallocation**

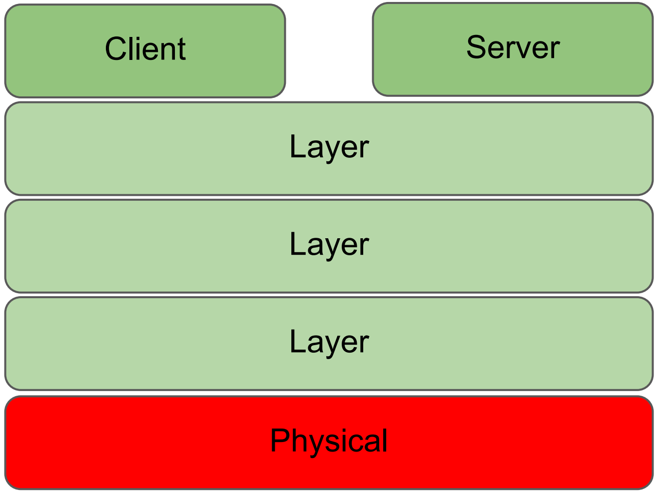
NB1) The bind operation is local to the IRMd
NB2) This is not be an exhaustive list

Ouroboros

local IPC

How do two processes on a PC communicate?

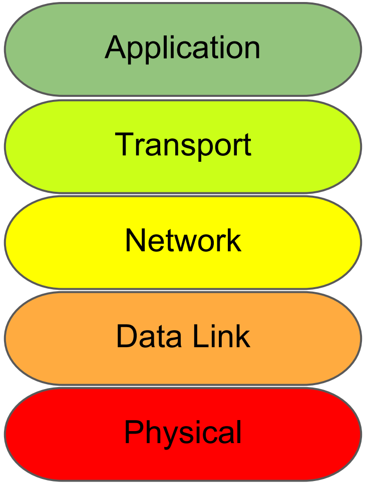


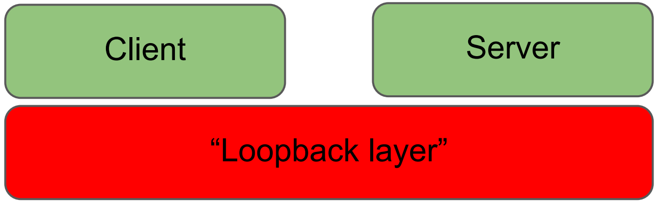


“Loopback interface”

127.0.0.1

How do two processes on a PC communicate?



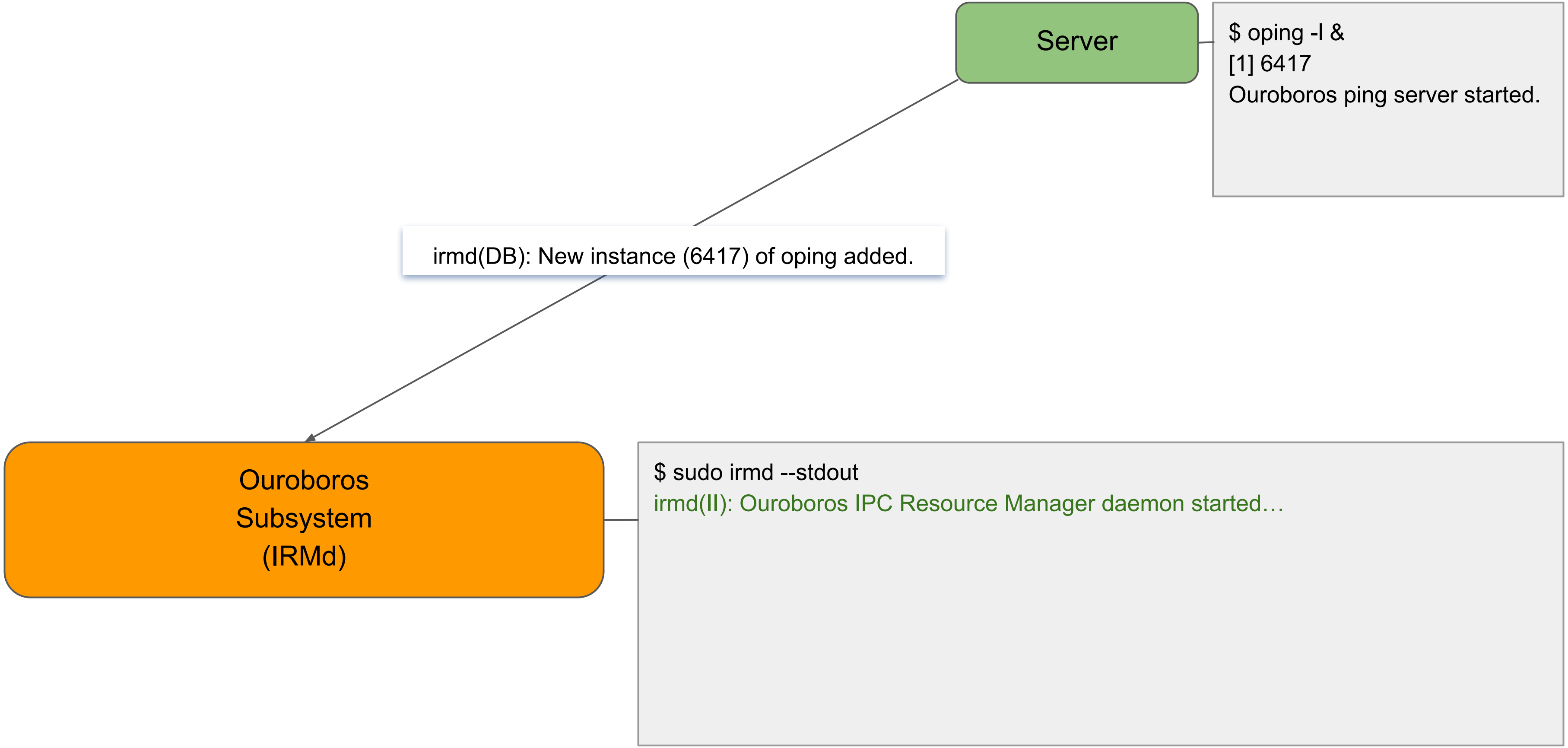


Ouroboros Local Inter-Process Communication

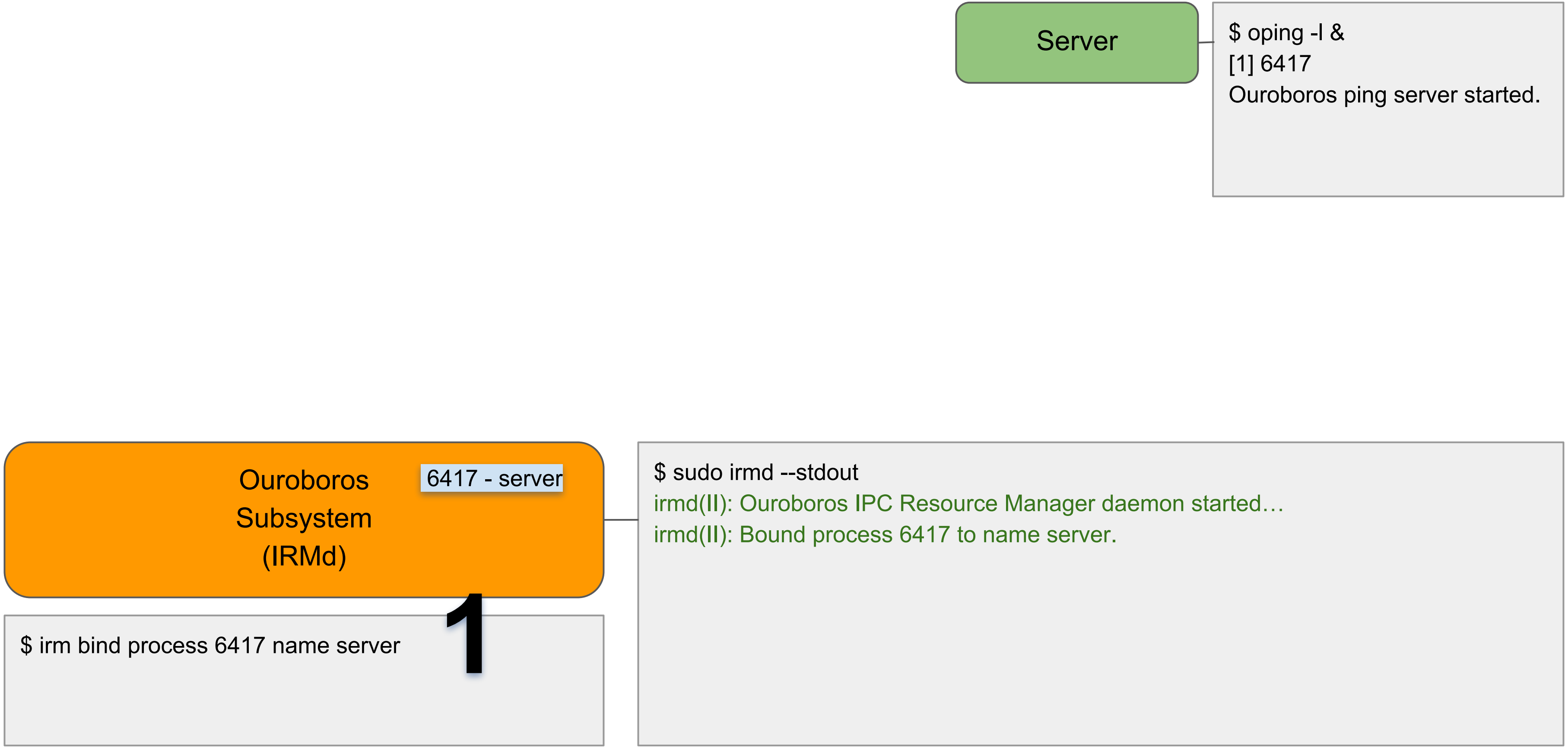
Ouroboros
Subsystem
(IRMd)

```
$ sudo irmd --stdout
irmd(II): Ouroboros IPC Resource Manager daemon started…
```

Ouroboros Local Inter-Process Communication



Ouroboros Local Inter-Process Communication



Ouroboros Local Inter-Process Communication

Server

```
$ oping -l &
[1] 6417
Ouroboros ping server started.
```

Local layer

```
ipcpd-local(II): Bootstrapped local IPCP with pid 6532.
```

Ouroboros Subsystem (IRMd)

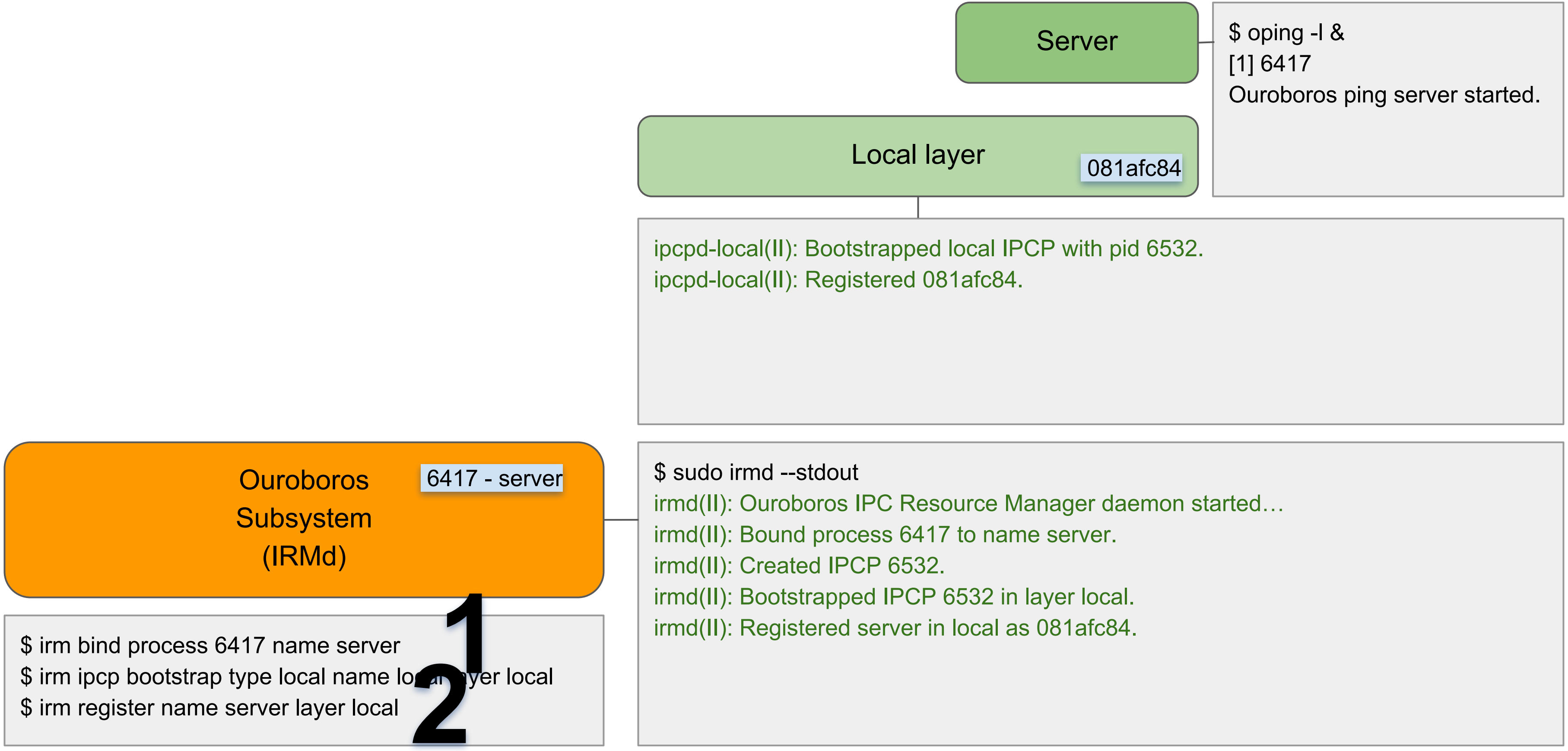
6417 - server

```
$ sudo irmd --stdout
irmd(II): Ouroboros IPC Resource Manager daemon started…
irmd(II): Bound process 6417 to name server.
irmd(II): Created IPCP 6532.
irmd(II): Bootstrapped IPCP 6532 in layer local.
```

1

```
$ irm bind process 6417 name server
$ irm ipcp bootstrap type local name local layer local
```

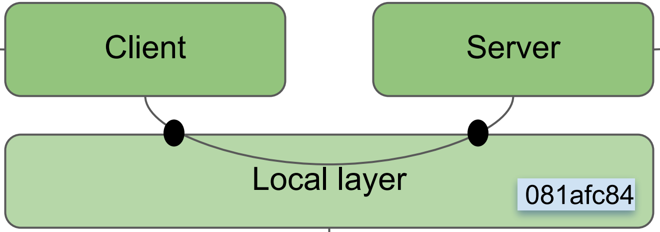
Ouroboros Local Inter-Process Communication



Ouroboros Local Inter-Process Communication

3

```
$ oping -n server
Pinging server with 64 bytes of data:

64 bytes from server: seq=0 time=0.480 ms
64 bytes from server: seq=1 time=0.268 ms
64 bytes from server: seq=2 time=0.239 ms
```




```
$ oping -l &
[1] 6417
Ouroboros ping server started.
New flow 64.
```

```
ipcpd-local(II): Bootstrapped local IPCP with pid 6532.
ipcpd-local(II): Registered 081afc84.
ipcpd-local(II): Pending local allocation request on fd 64.
ipcpd-local(II): Flow allocation completed, fds (64, 65).
```

Ouroboros Subsystem (IRMd)

6417 - server

1

2

```
$ irm bind process 6417 name server
$ irm ipcp bootstrap type local name local layer local
$ irm register name server layer local
```

```
$ sudo irmd --stdout
irmd(II): Ouroboros IPC Resource Manager daemon started…
irmd(II): Bound process 6417 to name server.
irmd(II): Created IPCP 6532.
irmd(II): Bootstrapped IPCP 6532 in layer local.
irmd(II): Registered server in local as 081afc84.
irmd(II): Flow request arrived for server.
irmd(II): Flow on port_id 0 | 1 allocated.
```

Ouroboros Local Inter-Process Communication

3

```
$ oping -n server
Pinging server with 64 bytes of data:

64 bytes from server: seq=0 time=0.480 ms
64 bytes from server: seq=1 time=0.268 ms
64 bytes from server: seq=2 time=0.239 ms
^C64 bytes from server: seq=3 time=0.263 ms

--- server ping statistics ---
4 SDUs transmitted, 4 received, 0% packet loss, time: 4001.325 ms
rtt min/avg/max/mdev = 0.239/0.312/0.480/0.112 ms
```

Client

Server

```
$ oping -l &
[1] 6417
Ouroboros ping server started.
New flow 64.
Flow 64 timed out.
```

Local layer 081afc84

```
ipcpd-local(II): Bootstrapped local IPCP with pid 6532.
ipcpd-local(II): Registered 081afc84.
ipcpd-local(II): Pending local allocation request on fd 64.
ipcpd-local(II): Flow allocation completed, fds (64, 65).
ipcpd-local(II): Flow with fd 64 deallocated.
ipcpd-local(II): Flow with fd 65 deallocated.
```

Ouroboros Subsystem (IRMd) 6417 - server

1

```
$ irm bind process 6417 name server
$ irm ipcp bootstrap type local name local layer local
$ irm register name server layer local
```

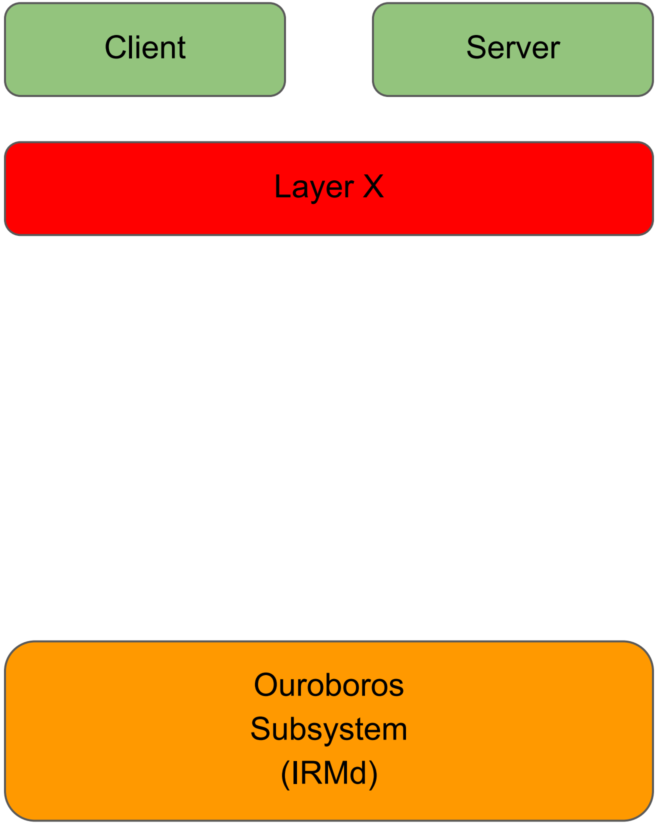
2

```
$ sudo irmd --stdout
irmd(II): Ouroboros IPC Resource Manager daemon started…
irmd(II): Bound process 6417 to name server.
irmd(II): Created IPCP 6532.
irmd(II): Bootstrapped IPCP 6532 in layer local.
irmd(II): Registered server in local as 081afc84.
irmd(II): Flow request arrived for server.
irmd(II): Flow on port_id 0 | 1 allocated.
irmd(II): Completed deallocation of port_id 0 | 1 by process 6532.
```

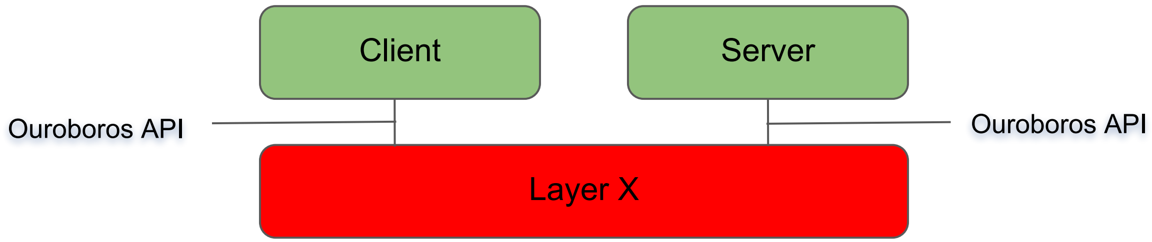
Ouroboros

over layer X

Networking Ouroboros



Networking Ouroboros



Ouroboros over Ethernet

```
$ sudo irmd --stdout
irmd(II): Ouroboros IPC Resource Manager daemon started…
```

Ouroboros Subsystem (IRMd)

Ouroboros over Ethernet

Eth IPCP

```
$ sudo irmd --stdout
irmd(II): Ouroboros IPC Resource Manager daemon started…
irmd(II): Created IPCP 19591.
ipcpd/eth-llc(II): Using raw socket device.
irmd(II): Bootstrapped IPCP 19591 in layer ethernet.
```

Ouroboros Subsystem (IRMd)

```
$ irm i b t eth-llc l ethernet n eth if wlp2s0
```

Ouroboros over Ethernet

c8a3f205

Eth IPCP

```
$ sudo irmd --stdout
irmd(II): Ouroboros IPC Resource Manager daemon started…
irmd(II): Created IPCP 19591.
ipcpd/eth-llc(II): Using raw socket device.
irmd(II): Bootstrapped IPCP 19591 in layer ethernet.
irmd(II): Registered ioq3 in ethernet as c8a3f205.
```

Ouroboros Subsystem (IRMd)

```
$ irm i b t eth-llc l ethernet n eth if wlp2s0
$ irm reg n ioq3 l ethernet
```

2

Ouroboros over Ethernet

c8a3f205

Eth IPCP

```
$ sudo irmd --stdout
irmd(II): Ouroboros IPC Resource Manager daemon started…
irmd(II): Created IPCP 19591.
ipcpd/eth-llc(II): Using raw socket device.
irmd(II): Bootstrapped IPCP 19591 in layer ethernet.
irmd(II): Registered ioq3 in ethernet as c8a3f205.
irmd(II): Bound program <path>/ioq3ded.x86_64 to name ioq3.
```

Ouroboros Subsystem (IRMd)

```
$ irm i b t eth-llc l ethernet n eth if wlp2s0
$ irm reg n ioq3 l ethernet
$ irm b prog ./ioq3ded.x86_64 n ioq3
```

2

1

Ouroboros over Ethernet

Server

c8a3f205

Eth IPCP

```
$ ./ioq3ded.x86_64 <params>
ioq3 1.36_GIT_71bd8d10-2018-02-02 linux-x86_64 Feb  2 2018
...
```

```
$ sudo irmd --stdout
irmd(II): Ouroboros IPC Resource Manager daemon started…
irmd(II): Created IPCP 19591.
ipcpd/eth-llc(II): Using raw socket device.
irmd(II): Bootstrapped IPCP 19591 in layer ethernet.
irmd(II): Registered ioq3 in ethernet as c8a3f205.
irmd(II): Bound program <path>/ioq3ded.x86_64 to name ioq3.
irmd(DB): Process 8976 inherits name ioq3 from program ioq3ded.x86_64.
```

Ouroboros Subsystem (IRMd)

```
$ irm i b t eth-llc l ethernet n eth if wlp2s0
$ irm reg n ioq3 l ethernet
$ irm b prog ./ioq3ded.x86_64 n ioq3
```

2

1

Ouroboros over Ethernet

Server

$ ./ioq3ded.x86_64 <params>
ioq3 1.36_GIT_71bd8d10-2018-02-02 linux-x86_64 Feb 2 2018
...

c8a3f205

Eth IPCP

$ sudo irmd --stdout
irmd(II): Ouroboros IPC Resource Manager daemon started...

$ sudo irmd --stdout
irmd(II): Ouroboros IPC Resource Manager daemon started…
irmd(II): Created IPCP 19591.
ipcpd/eth-llc(II): Using raw socket device.
irmd(II): Bootstrapped IPCP 19591 in layer ethernet.
irmd(II): Registered ioq3 in ethernet as c8a3f205.
irmd(II): Bound program <path>/ioq3ded.x86_64 to name ioq3.
irmd(DB): Process 8976 inherits name ioq3 from program ioq3ded.x86_64.

Ouroboros Subsystem (IRMd)

Ouroboros Subsystem (IRMd)

$ irm i b t eth-llc l ethernet n eth if wlp2s0
$ irm reg n ioq3 l ethernet
$ irm b prog ./ioq3ded.x86_64 n ioq3

2 1

Ouroboros over Ethernet

Server

$ ./ioq3ded.x86_64 <params>
ioq3 1.36_GIT_71bd8d10-2018-02-02 linux-x86_64 Feb 2 2018
...

c8a3f205

Eth IPCP | Ethernet Layer | Eth IPCP

$ sudo irmd --stdout
irmd(II): Ouroboros IPC Resource Manager daemon started...
irmd(II): Created IPCP 6268.
ipcpd/eth-llc(II): Using raw socket device.
irmd(II): Bootstrapped IPCP 6268 in layer ethernet.

$ sudo irmd --stdout
irmd(II): Ouroboros IPC Resource Manager daemon started…
irmd(II): Created IPCP 19591.
ipcpd/eth-llc(II): Using raw socket device.
irmd(II): Bootstrapped IPCP 19591 in layer ethernet.
irmd(II): Registered ioq3 in ethernet as c8a3f205.
irmd(II): Bound program <path>/ioq3ded.x86_64 to name ioq3.
irmd(DB): Process 8976 inherits name ioq3 from program ioq3ded.x86_64.

$ irm i b t eth-llc l ethernet n eth if wlp2s0

Ouroboros Subsystem (IRMd)

Ouroboros Subsystem (IRMd)

$ irm i b t eth-llc l ethernet n eth if wlp2s0
$ irm reg n ioq3 l ethernet
$ irm b prog ./ioq3ded.x86_64 n ioq3

21

Ouroboros over Ethernet

3

Client

Server

$ ./ioq3ded.x86_64 <params>
ioq3 1.36_GIT_71bd8d10-2018-02-02 linux-x86_64 Feb 2 2018
...

Eth IPCP Ethernet Layer Eth IPCP

$ sudo irmd --stdout
irmd(II): Ouroboros IPC Resource Manager daemon started...
irmd(II): Created IPCP 6268.
ipcpd/eth-llc(II): Using raw socket device.
irmd(II): Bootstrapped IPCP 6268 in layer ethernet.

$ sudo irmd --stdout
irmd(II): Ouroboros IPC Resource Manager daemon started…
irmd(II): Created IPCP 19591.
ipcpd/eth-llc(II): Using raw socket device.
irmd(II): Bootstrapped IPCP 19591 in layer ethernet.
irmd(II): Registered ioq3 in ethernet as c8a3f205.
irmd(II): Bound program <path>/ioq3ded.x86_64 to name ioq3.
irmd(DB): Process 8976 inherits name ioq3 from program ioq3ded.x86_64.
irmd(II): Flow request arrived for ioq3.

$ ./ioquake3.x86_64 +set com basegame baseoa

] connect -O ioq3

$ irm i b t eth-llc l ethernet n eth if wlp2s0

Ouroboros Subsystem (IRMd)

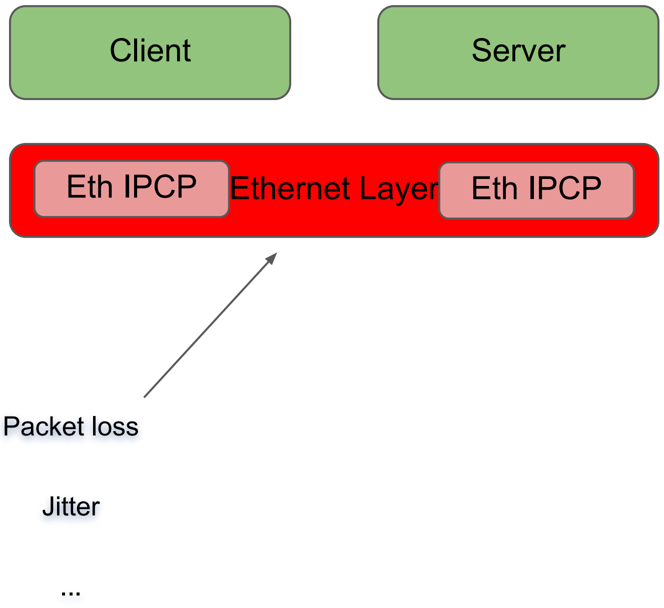
Ouroboros Subsystem (IRMd)

$ irm i b t eth-llc l ethernet n eth if wlp2s0
$ irm reg n ioq3 l ethernet
$ irm b prog ./ioq3ded.x86_64 n ioq3

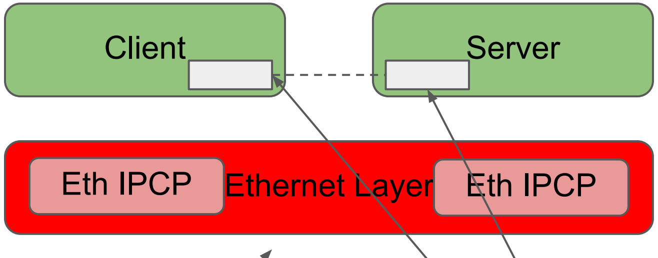
2

1

Reliability



Reliability



Packet loss

Jitter

...

Connection management

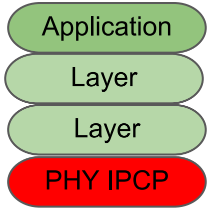
- **SEQ**
- **ACKs**
- **REORDER**
- **FC**
- **RTX**

- **FRAGMENTATION**
- **ENCRYPTION**
- **CHECKSUMMING**

NB3) connection management and fragmentation/encryption/checksumming is available in every process and thus not a distinct function of a layer

Ouroboros over X

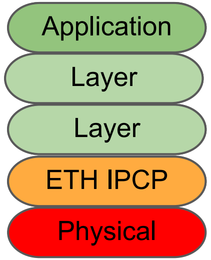
L1



RAPTOR NetFPGA 10G

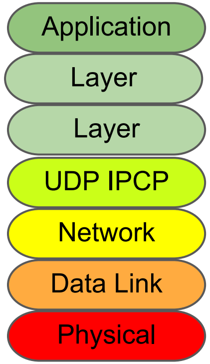
L2

- **RAW SOCKETS**
- **BPF**
- **NETMAP**



L3/4

- **POSIX SOCKETS**
- **IPv4 / UDP / DDNS**



	flow allocation	routing	forwarding	directory
raptor	ouroboros	N/A	N/A	ouroboros
eth-llc	ouroboros	RSTP	Ethernet	ouroboros
udp	ouroboros	OSPF	IP	DDNS

Ouroboros

over Ouroboros

Ouroboros

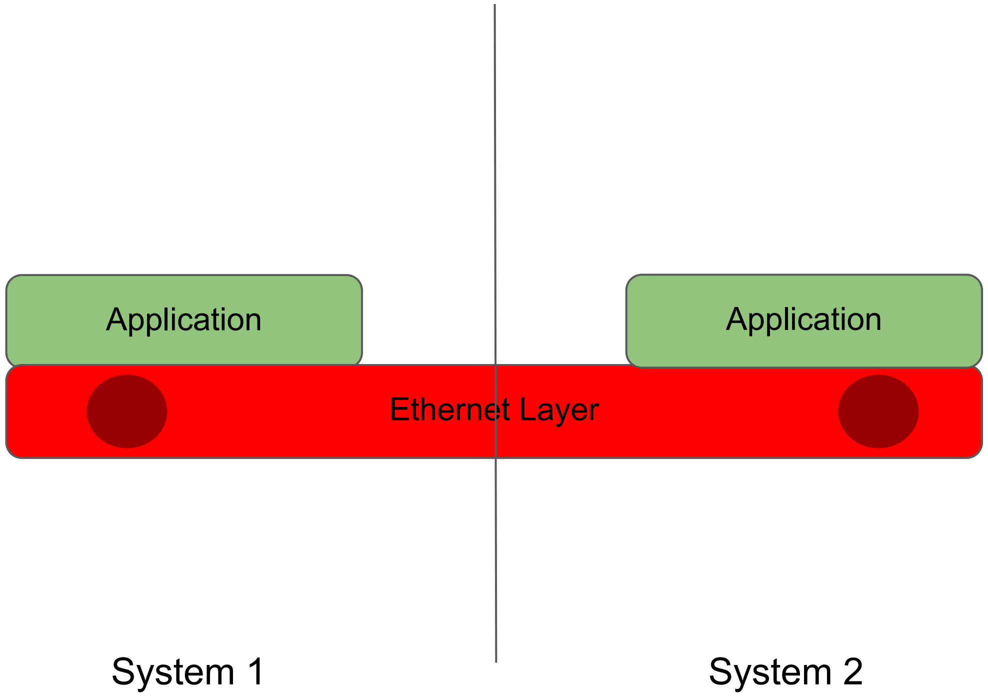
over Ouroboros

over Ouroboros

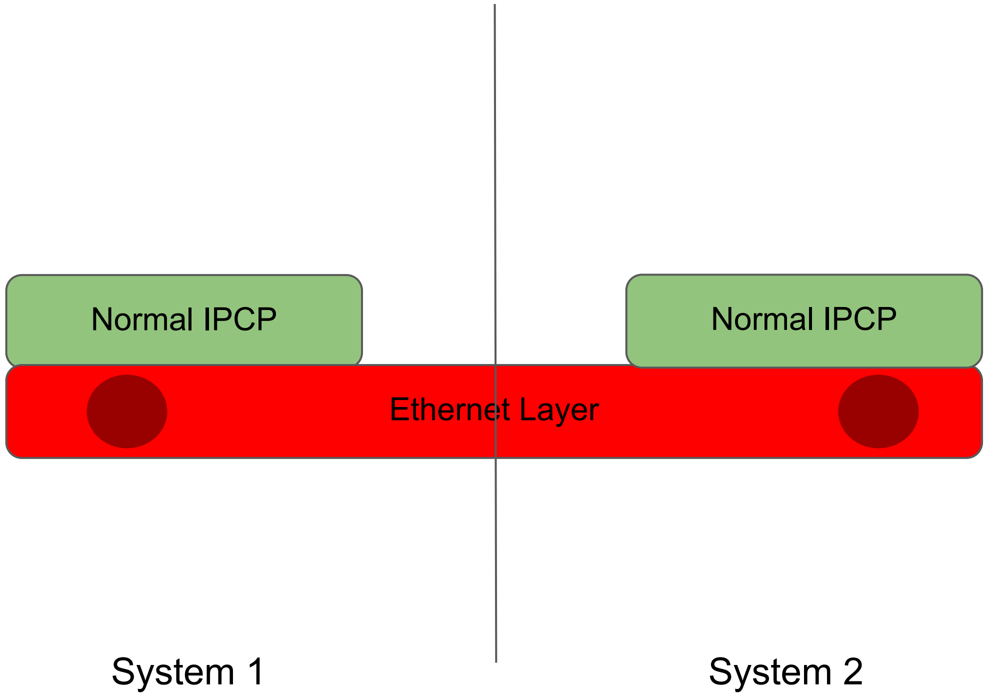
over Ouroboros

over Ouroboros

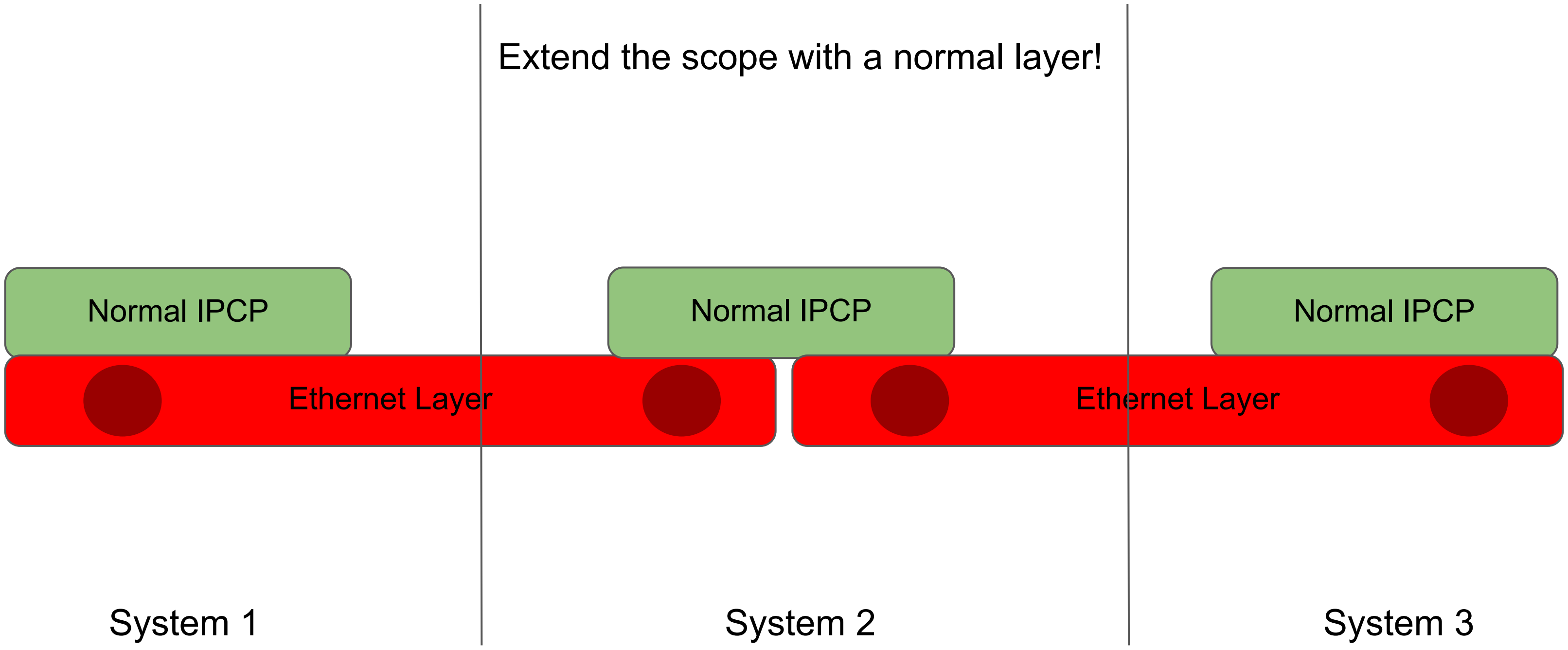
Recursive model



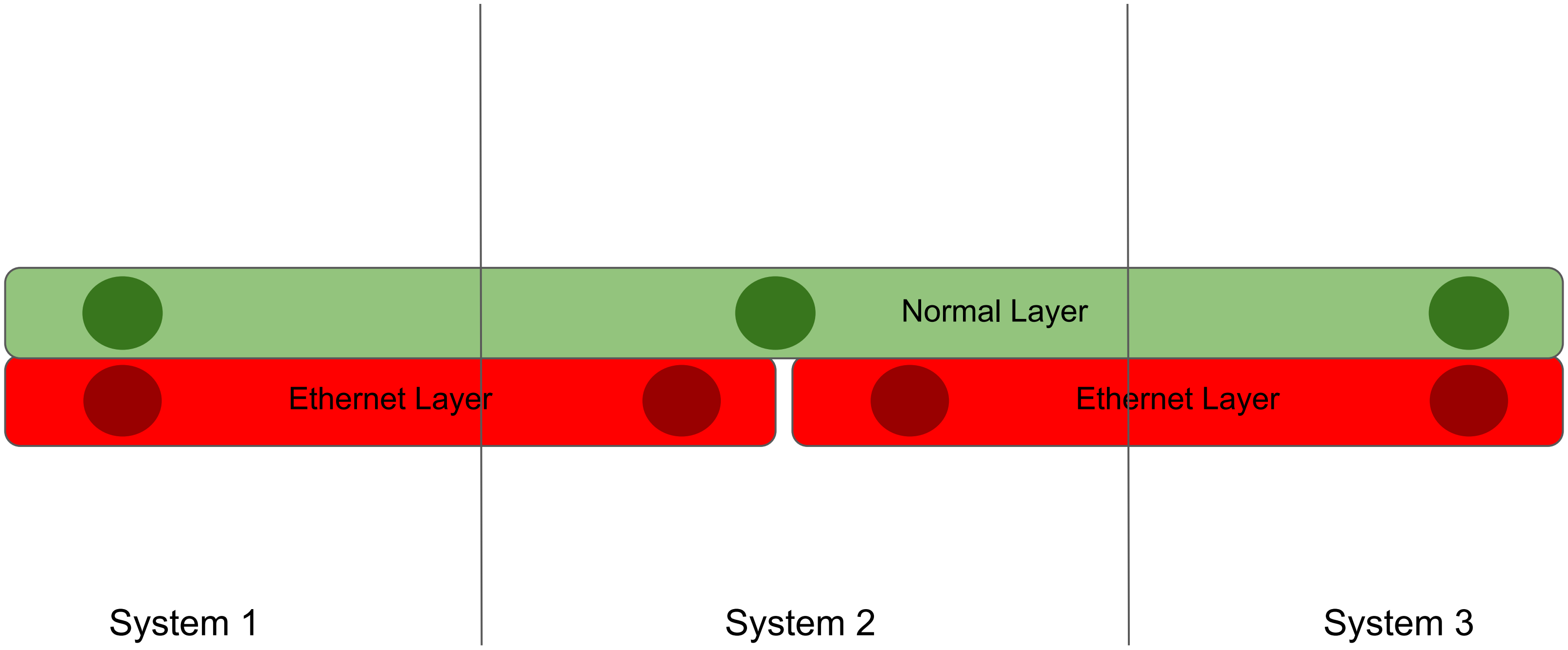
Recursive model



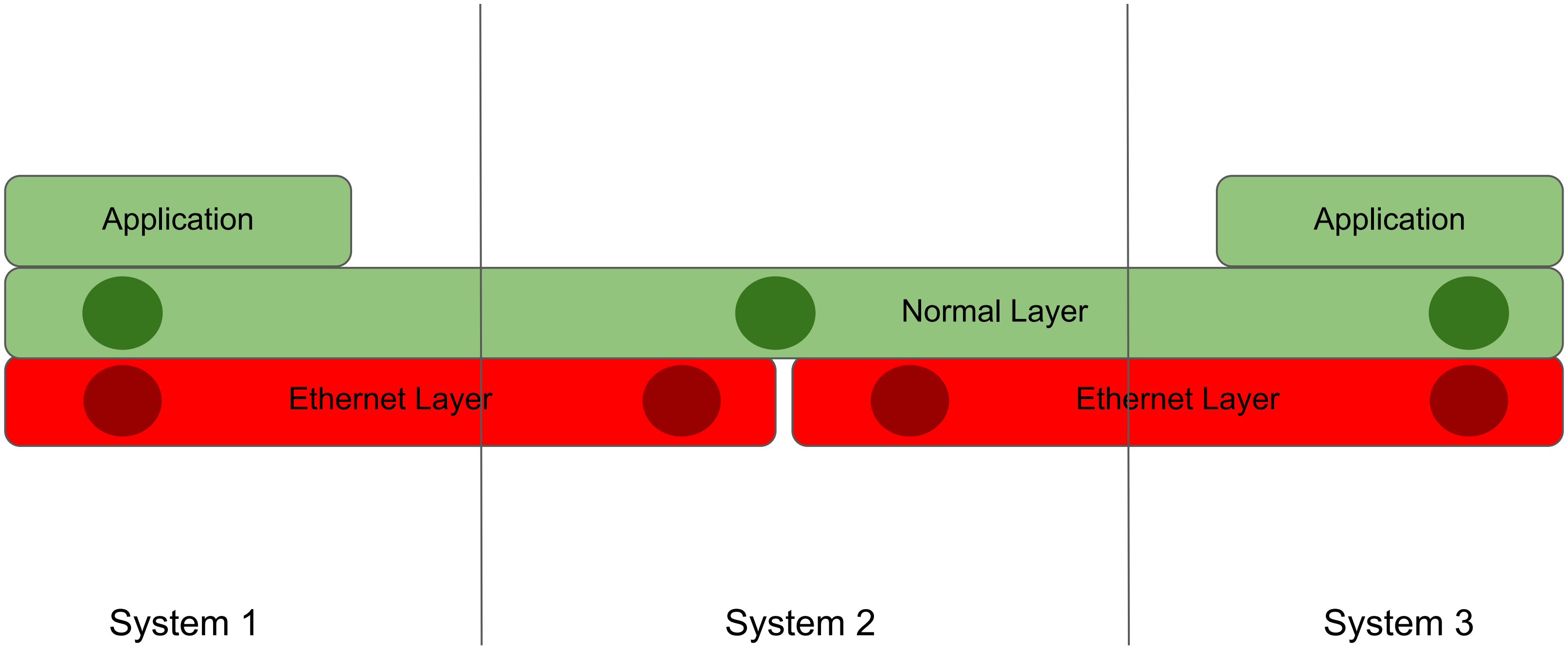
Recursive model



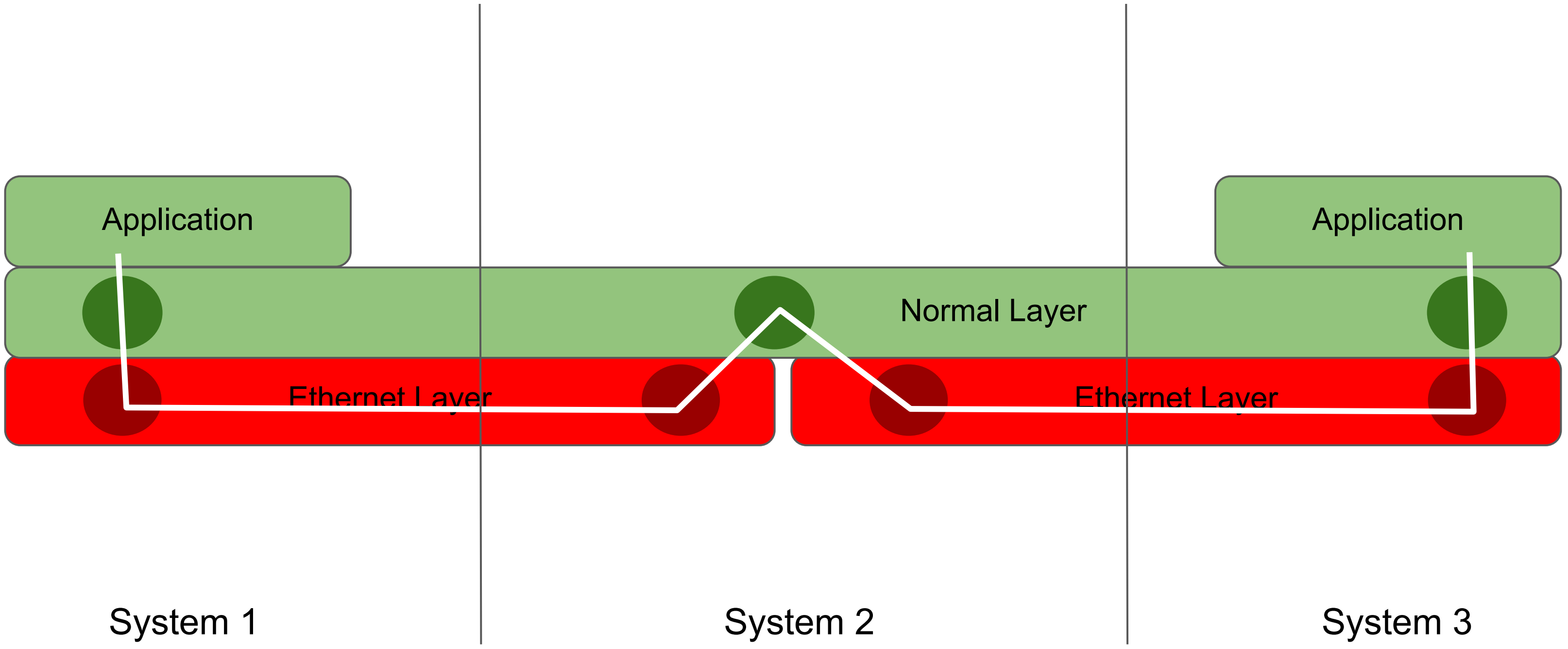
Recursive model



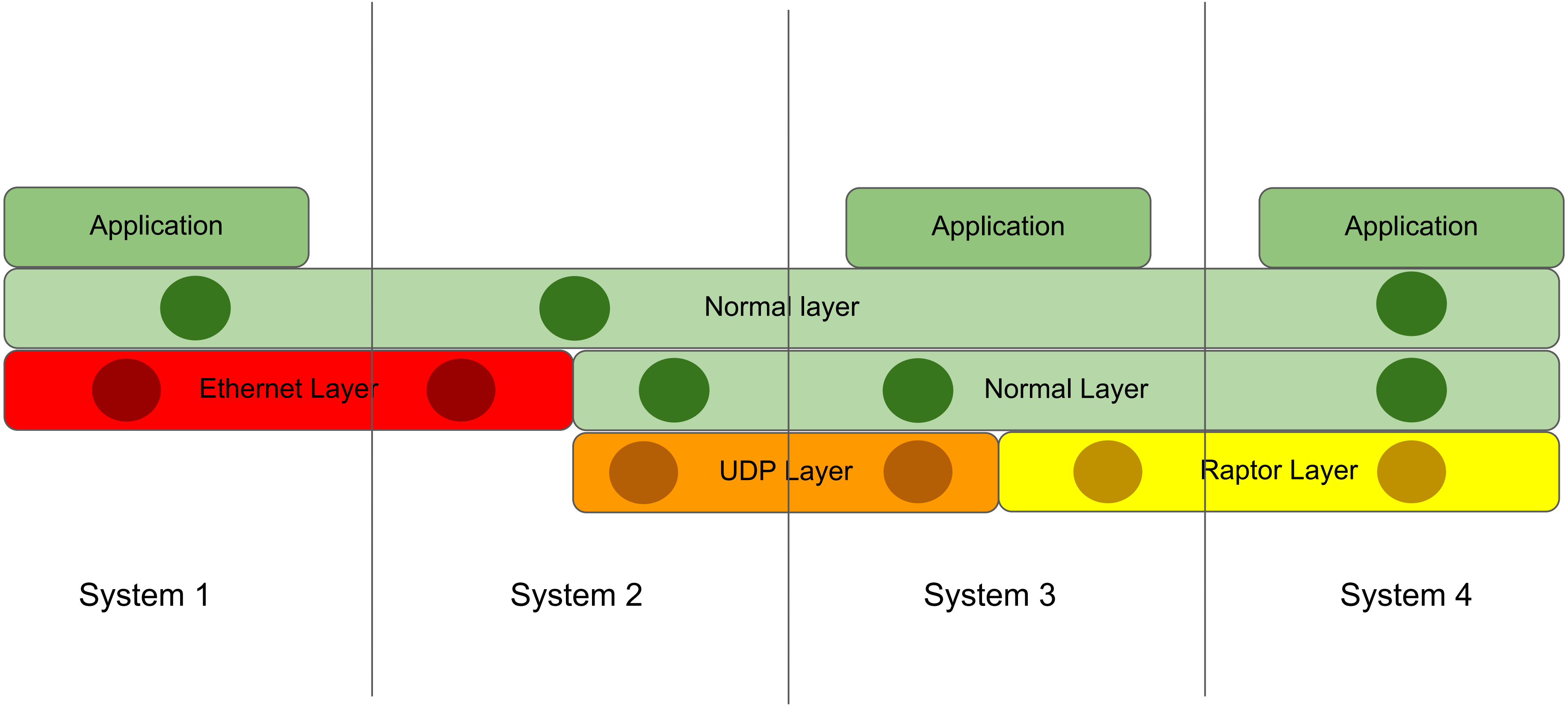
Recursive model



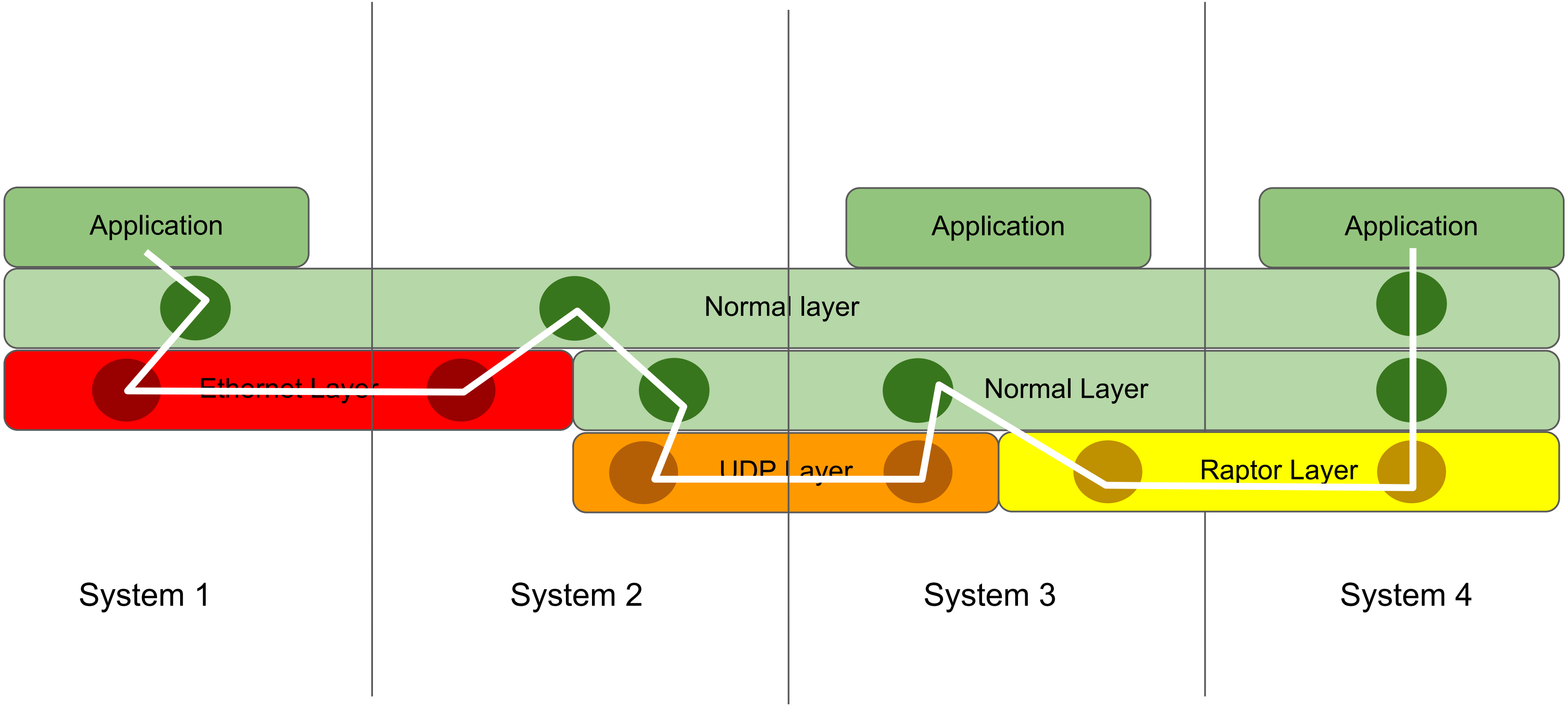
Recursive model



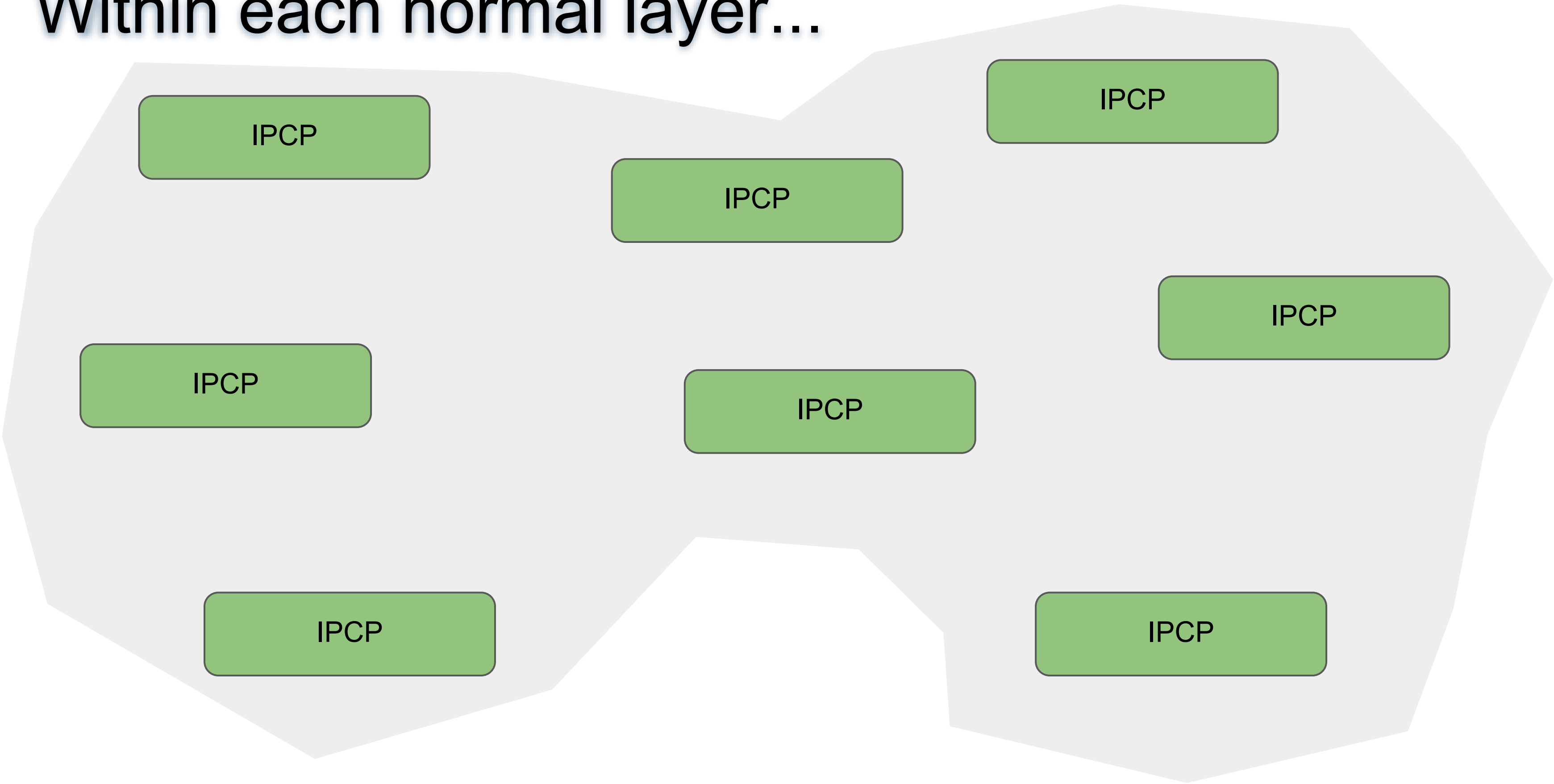
Let's keep recursing!



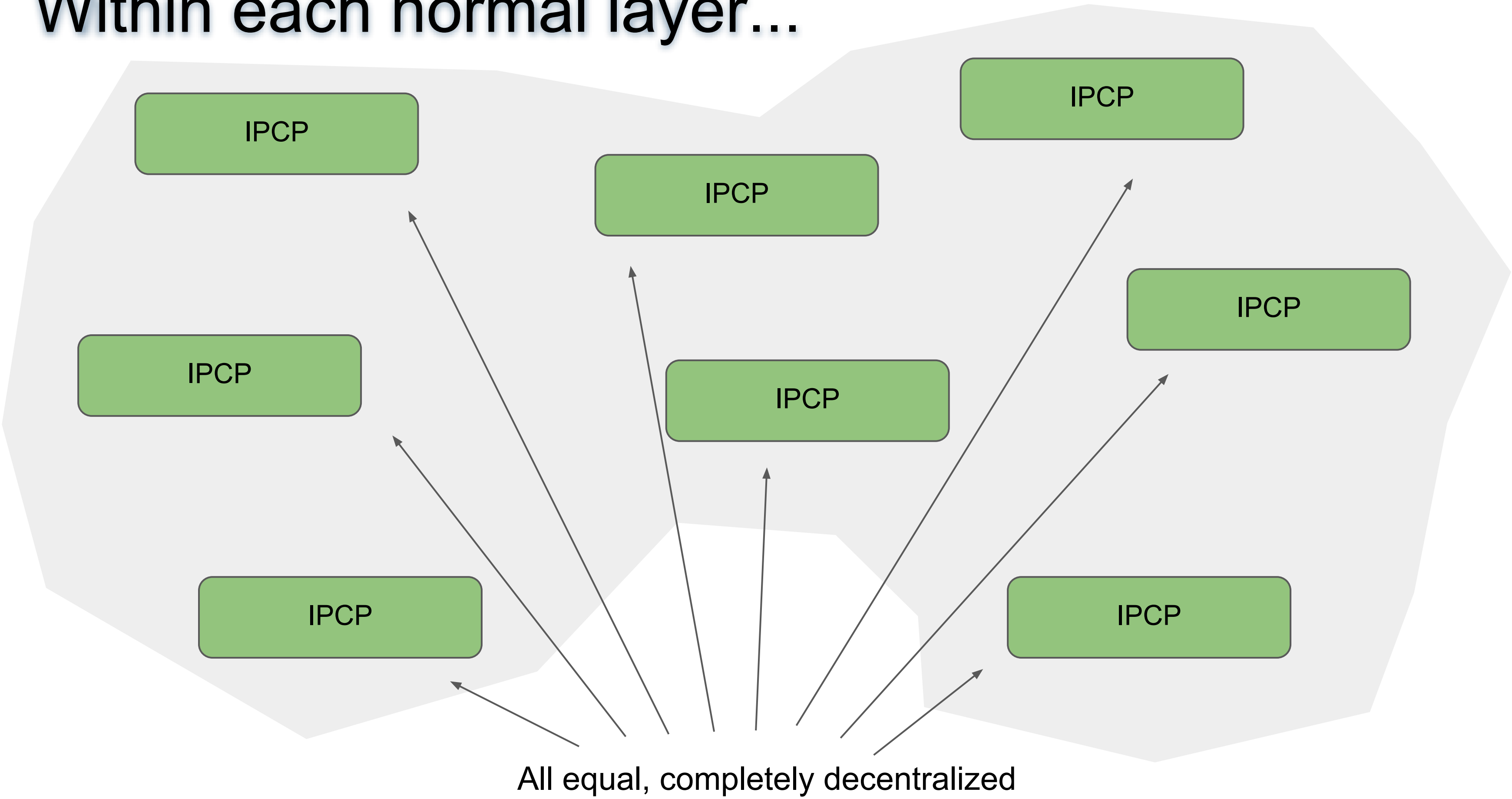
Let's keep recursing!



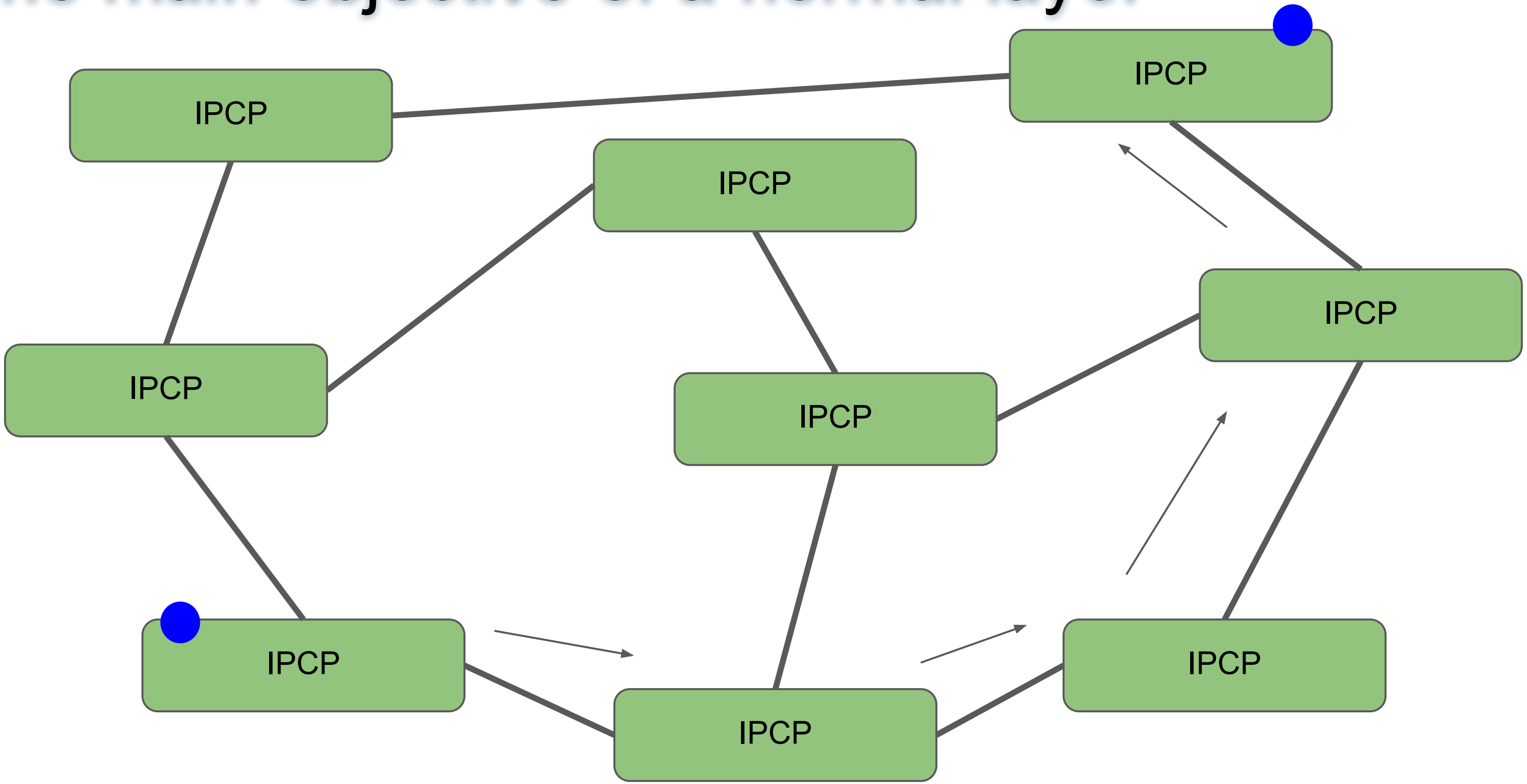
Within each normal layer...



Within each normal layer...



The main objective of a normal layer

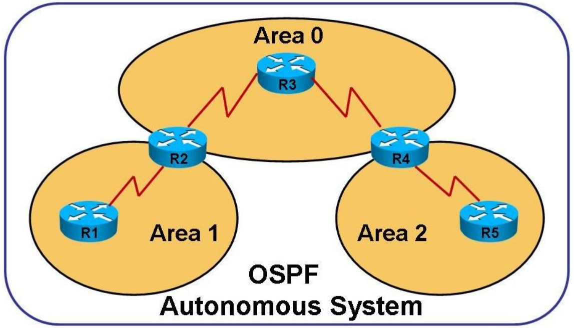


Complexity reduced through synergy

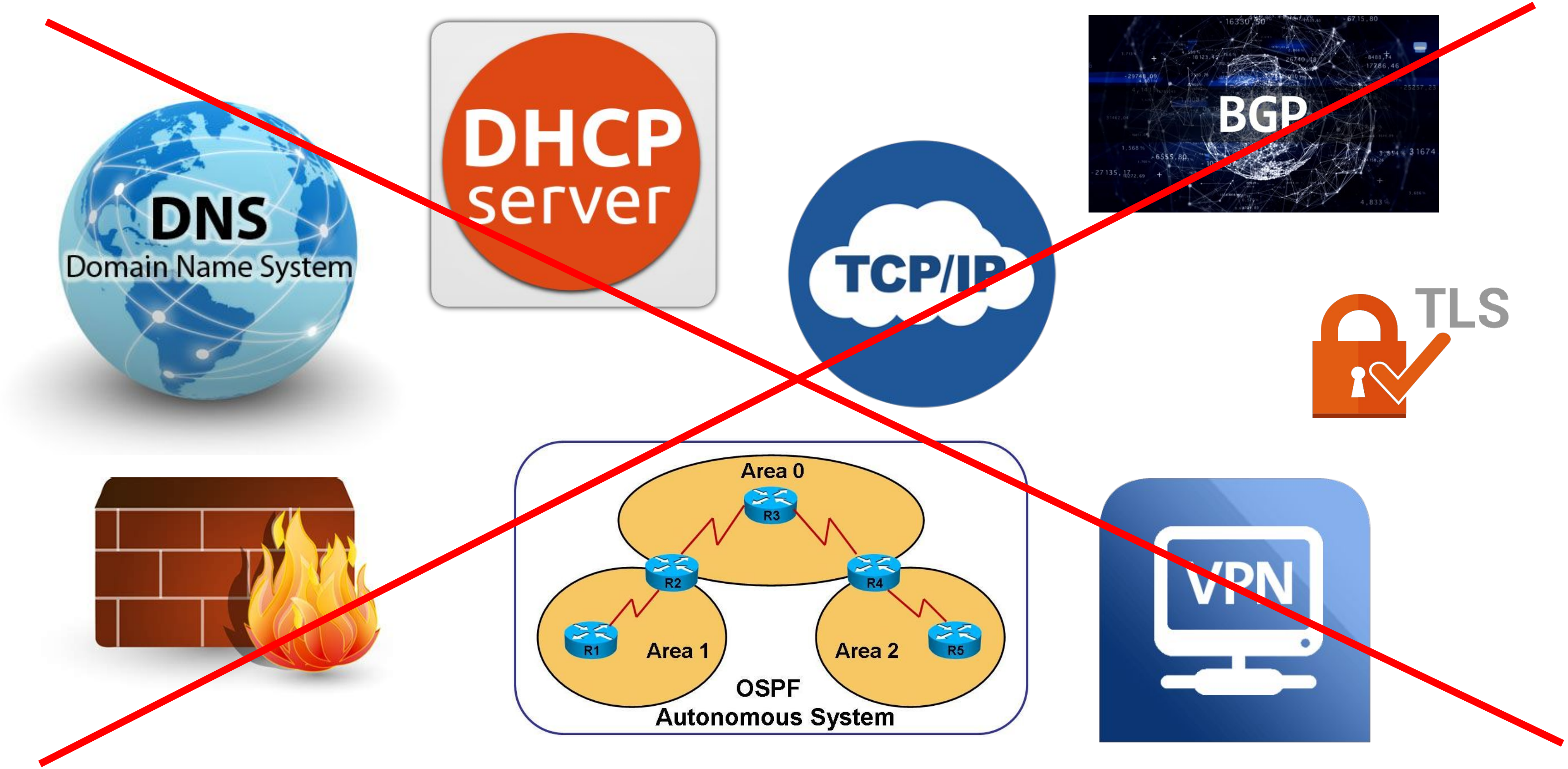
DHCP server

TCP/IP

TLS



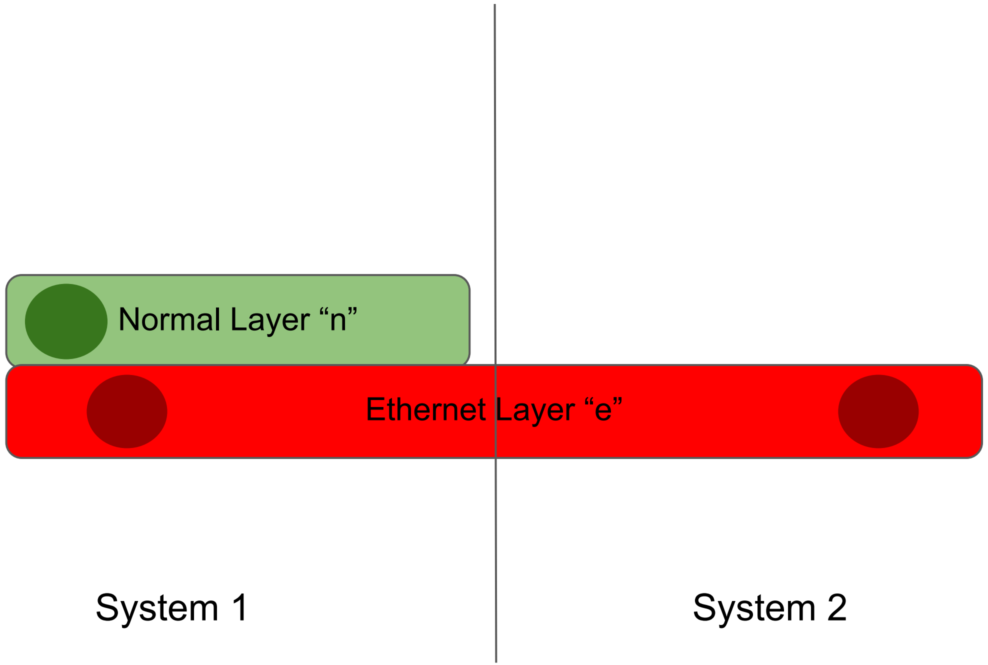
Complexity reduced through synergy



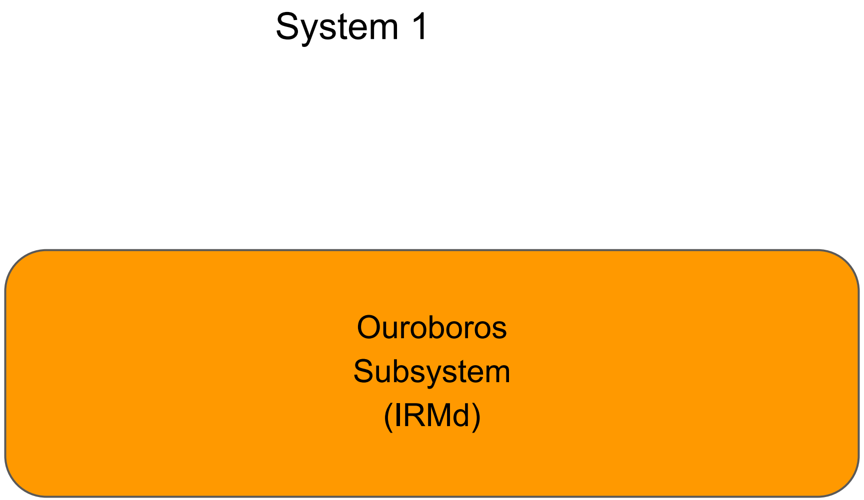
Complexity reduced through synergy

IPCP

Bootstrapping the normal layer



Bootstrapping the normal layer




```
$ irm ipcp bootstrap
```

```
Usage: irm ipcp bootstrap  name <ipcp name>
           layer <layer name>
           type [TYPE]
where TYPE = {normal local udp eth-llc raptor},

if TYPE == normal
           [addr <address size> (default: 4)]
           [fd <fd size> (default: 2)]
           [ttl (add time to live value in the PCI)]
           [addr_auth <ADDRESS_POLICY> (default: flat)]
           [routing <ROUTING_POLICY> (default: link_state)]
           [pff [PFF_POLICY] (default: simple)]
           [hash [ALGORITHM] (default: SHA3_256)]
           [autobind]
where ADDRESS_POLICY = {flat}
           ROUTING_POLICY = {link_state lfa}
           PFF_POLICY = {simple alternate}
           ALGORITHM = {SHA3_224 SHA3_256 SHA3_384 SHA3_512}
...
```

Bootstrapping the normal layer

Ethernet IPCP “e1”

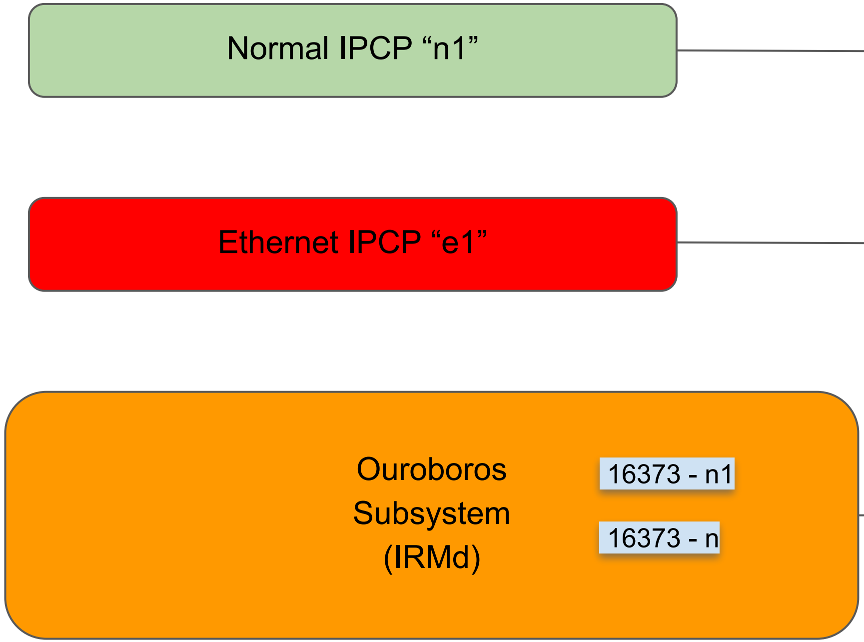
```
ipcpd/eth-llc(II): Using raw socket device.
ipcpd/eth-llc(DB): Bootstrapped IPCP over Ethernet with LLC with pid 16357.
```

Ouroboros
Subsystem
(IRMd)

```
$ sudo irmd --stdout
irmd(II): Created IPCP 16357.
irmd(II): Bootstrapped IPCP 16357 in layer e.
```

```
$ irm ipcp bootstrap name e1 type eth-llc if eth0 layer e
```

Bootstrapping the normal layer



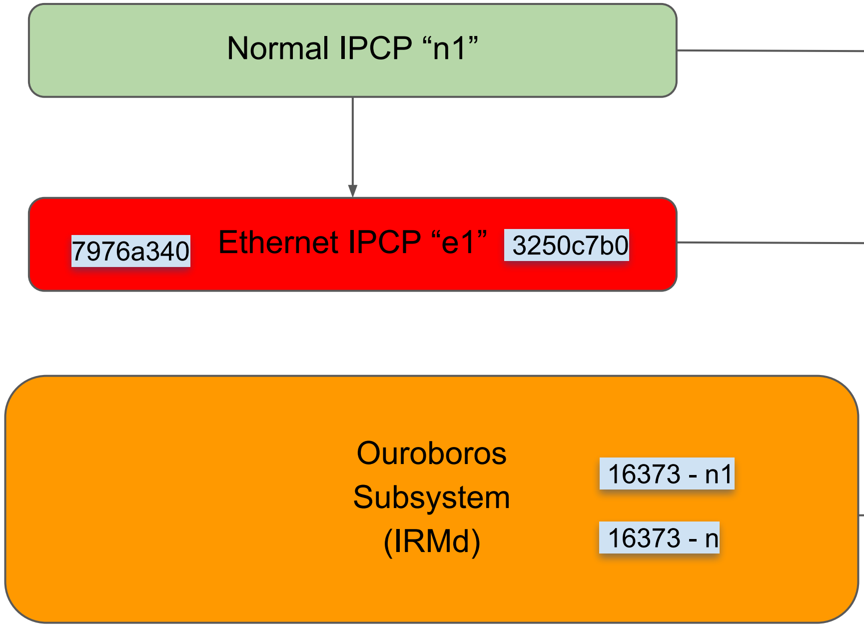

```
normal-ipcp(DB): IPCP got address 1860022337.
directory(DB): Bootstrapping directory.
directory(II): Directory bootstrapped.
```

```
ipcpd/eth-llc(II): Using raw socket device.
ipcpd/eth-llc(DB): Bootstrapped IPCP over Ethernet with LLC with
pid 16357.
```

```
$ sudo irmd --stdout
irmd(II): Created IPCP 16357.
irmd(II): Bootstrapped IPCP 16357 in layer e.
irmd(II): Created IPCP 16373.
irmd(II): Bound process 16373 to name n1.
irmd(II): Bound process 16373 to name n.
irmd(II): Bootstrapped IPCP 16373 in layer n.
irmd(DB): New instance (16373) of ipcpd-normal added.
irmd(DB): This process accepts flows for:
irmd(DB):        n1
irmd(DB):        n
```

```
$ irm ipcp bootstrap name e1 type eth-llc if eth0 layer e
$ irm ipcp bootstrap name n1 type normal layer n autobind
```

Bootstrapping the normal layer



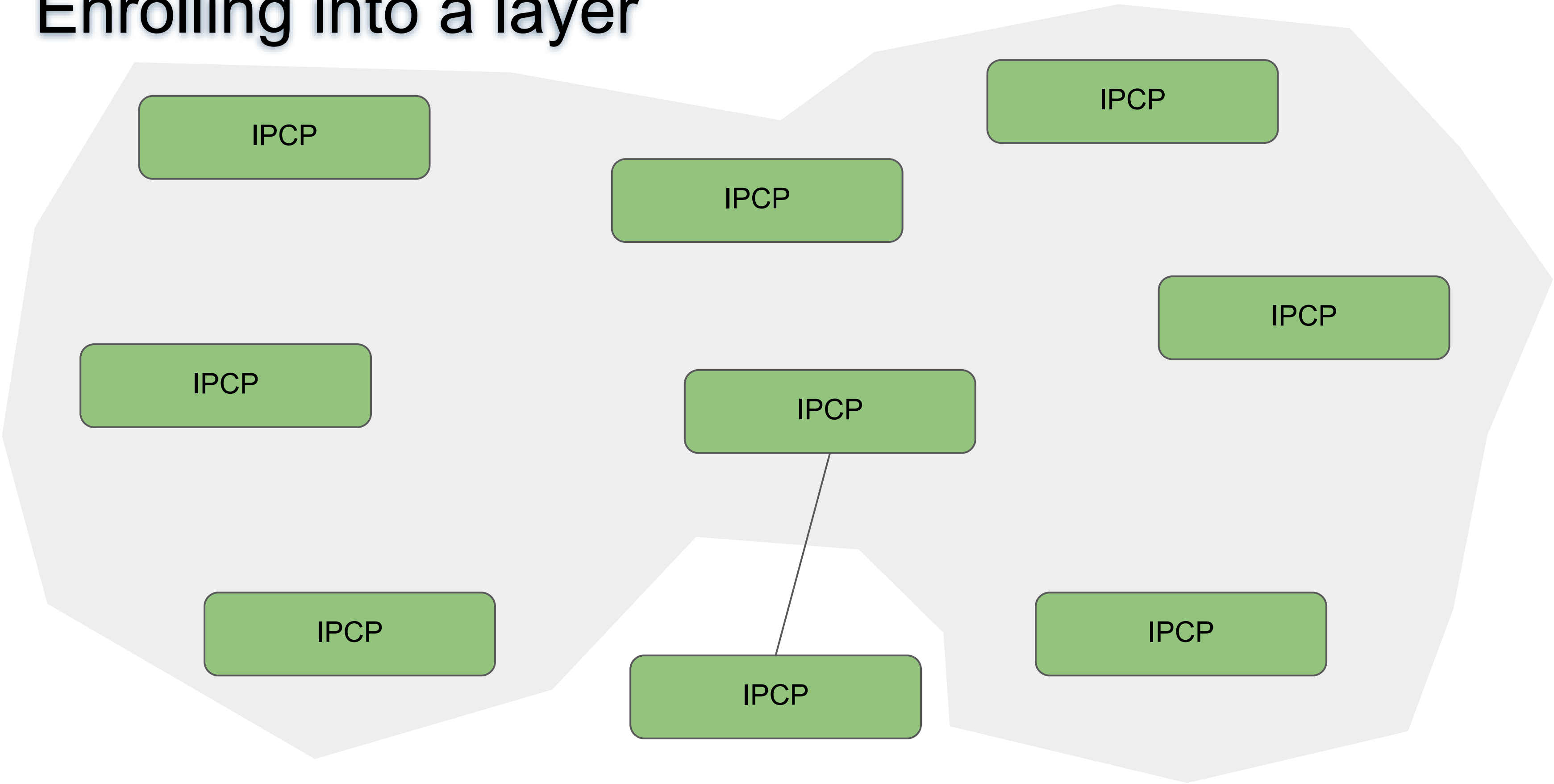

```
normal-ipcp(DB): IPCP got address 1860022337.
directory(DB): Bootstrapping directory.
directory(II): Directory bootstrapped.
```

```
ipcpd/eth-llc(II): Using raw socket device.
ipcpd/eth-llc(DB): Bootstrapped IPCP over Ethernet with LLC with pid 16357.
```

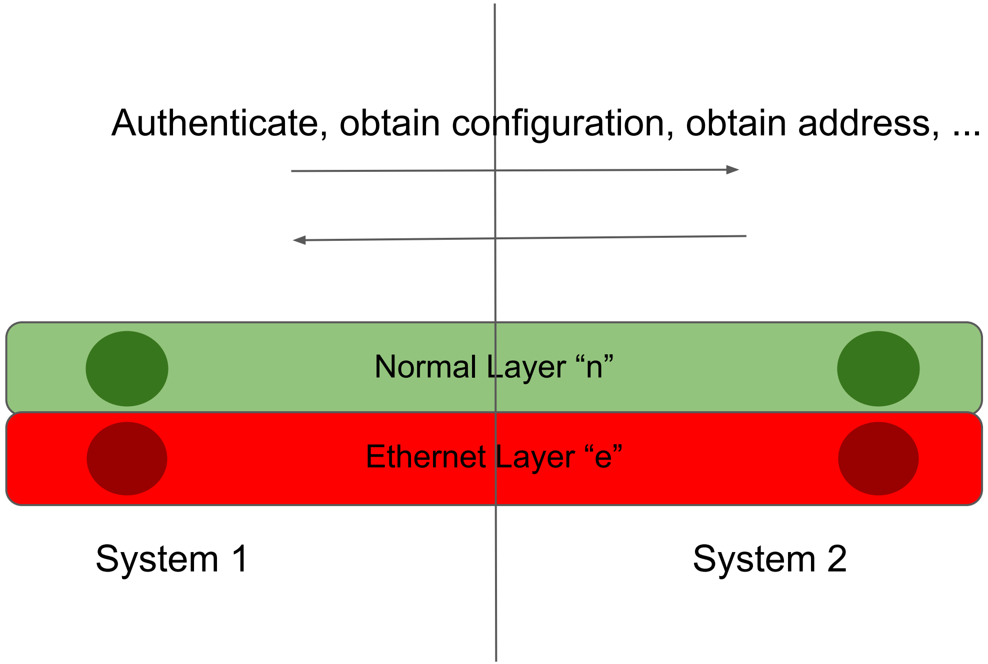
```
$ sudo irmd --stdout
...
irmd(II): Registered n1 in e as 7976a340.
irmd(II): Registered n in e as 3250c7b0.
```

```
$ irm ipcp bootstrap name e1 type eth-llc if eth0 layer e
$ irm ipcp bootstrap name n1 type normal layer n autobind
$ irm register name n1 layer e
$ irm register name n layer e
```

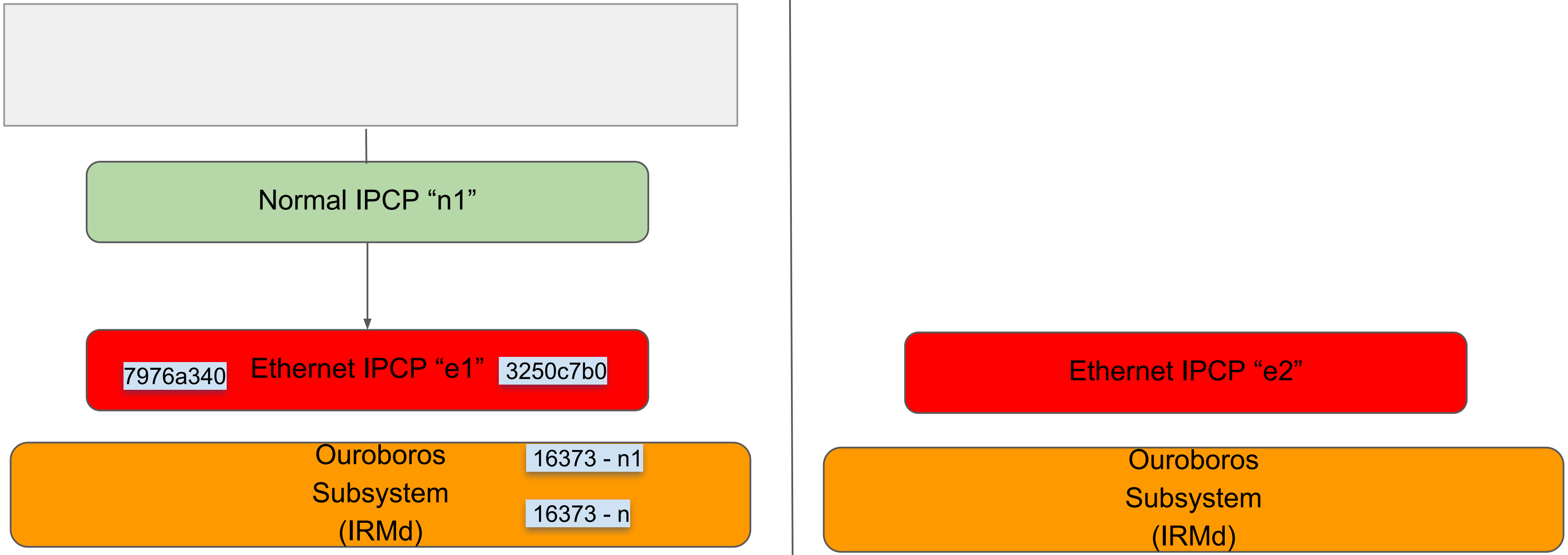
Enrolling into a layer



Enrolling the normal layer



Enrollment normal layer



$ irm ipcp bootstrap name e1 type eth-llc if eth0 layer e
$ irm ipcp bootstrap name n1 type normal layer n autobind
$ irm register name n1 layer e
$ irm register name n layer e

System 1

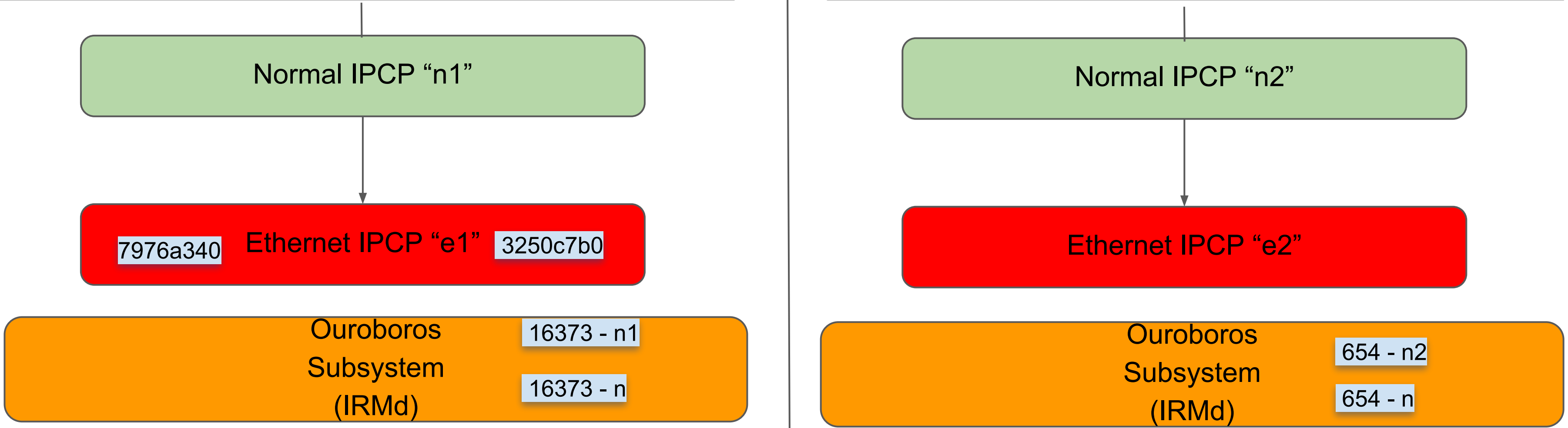
$ irm ipcp bootstrap name e2 type eth-llc if eth0 layer e

System 2

Enrollment normal layer

```
enrollment(DB): Enrolling a new neighbor.
enrollment(DB): Sending enrollment info (38 bytes).
enrollment(DB): Neighbor enrollment successful.
```

```
enrollment(DB): Getting boot information.
enrollment(DB): Received enrollment info (38 bytes).
normal-ipcp(DB): IPCP got address 65925404.
```




```
$ irm ipcp bootstrap name e1 type eth-llc if eth0 layer e
$ irm ipcp bootstrap name n1 type normal layer n autobind
$ irm register name n1 layer e
$ irm register name n layer e
```

System 1

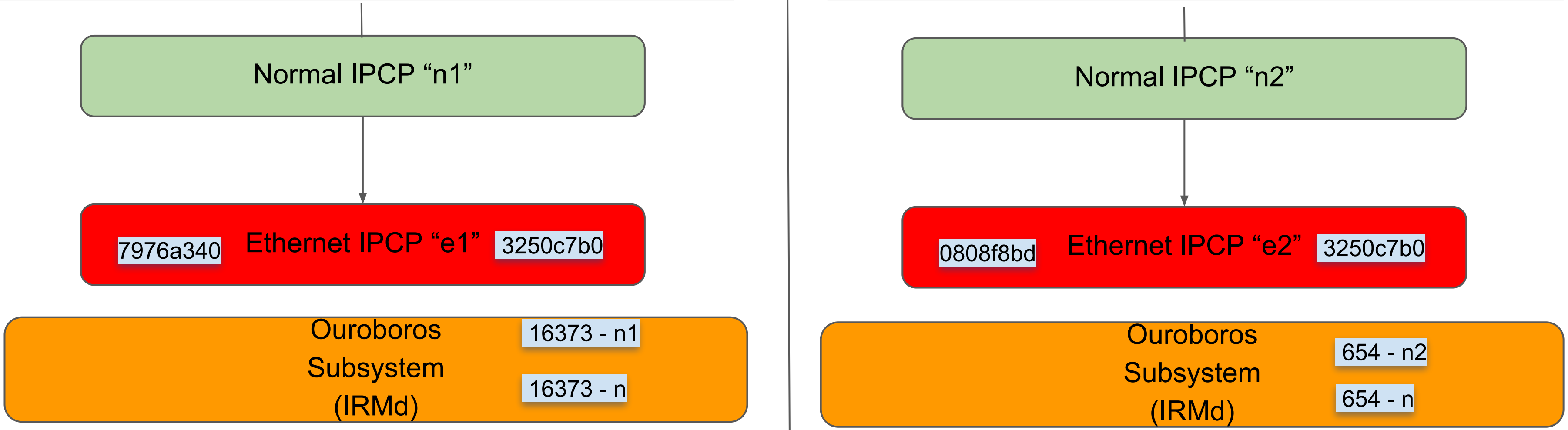
```
$ irm ipcp bootstrap name e2 type eth-llc if eth0 layer e
$ irm ipcp enroll name n2 layer n autobind
```

System 2

Enrollment normal layer

```
enrollment(DB): Enrolling a new neighbor.
enrollment(DB): Sending enrollment info (38 bytes).
enrollment(DB): Neighbor enrollment successful.
```

```
enrollment(DB): Getting boot information.
enrollment(DB): Received enrollment info (38 bytes).
normal-ipcp(DB): IPCP got address 65925404.
```



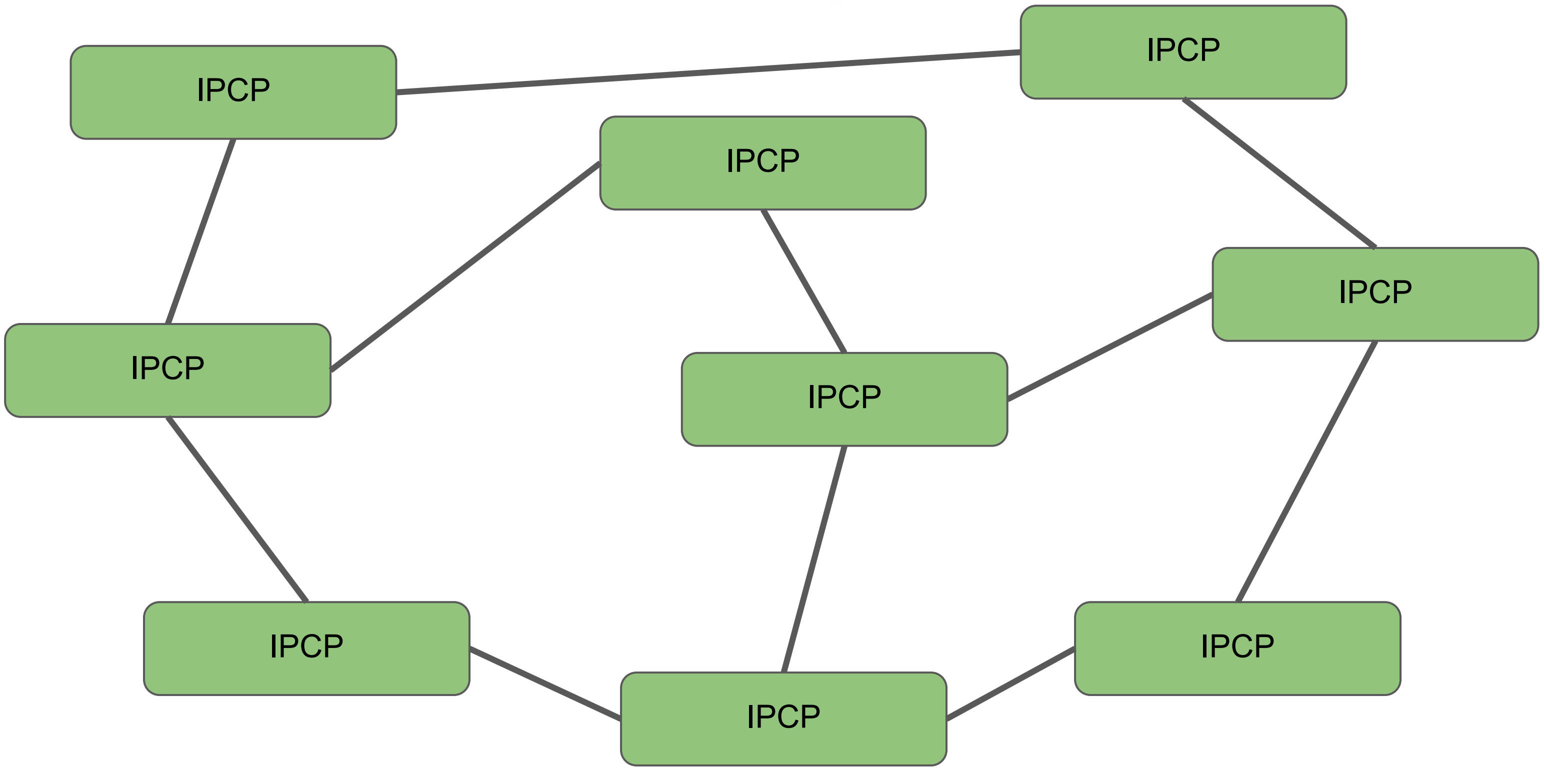

```
$ irm ipcp bootstrap name e1 type eth-llc if eth0 layer e
$ irm ipcp bootstrap name n1 type normal layer n autobind
$ irm register name n1 layer e
$ irm register name n layer e
```

System 1

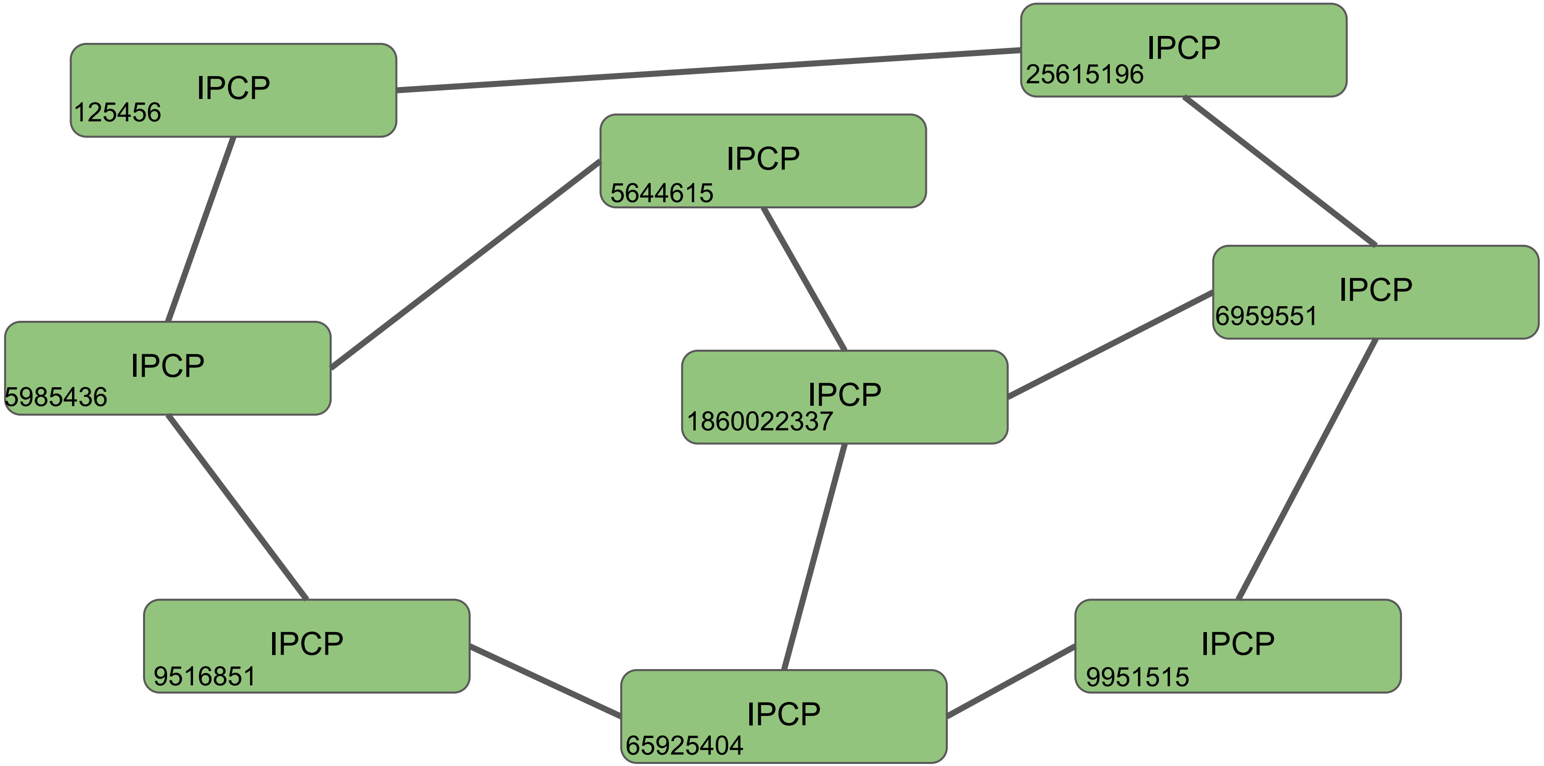
```
$ irm ipcp bootstrap name e2 type eth-llc if eth0 layer e
$ irm ipcp enroll name n2 layer n autobind
$ irm register name n2 layer e
$ irm register name n layer e
```

System 2

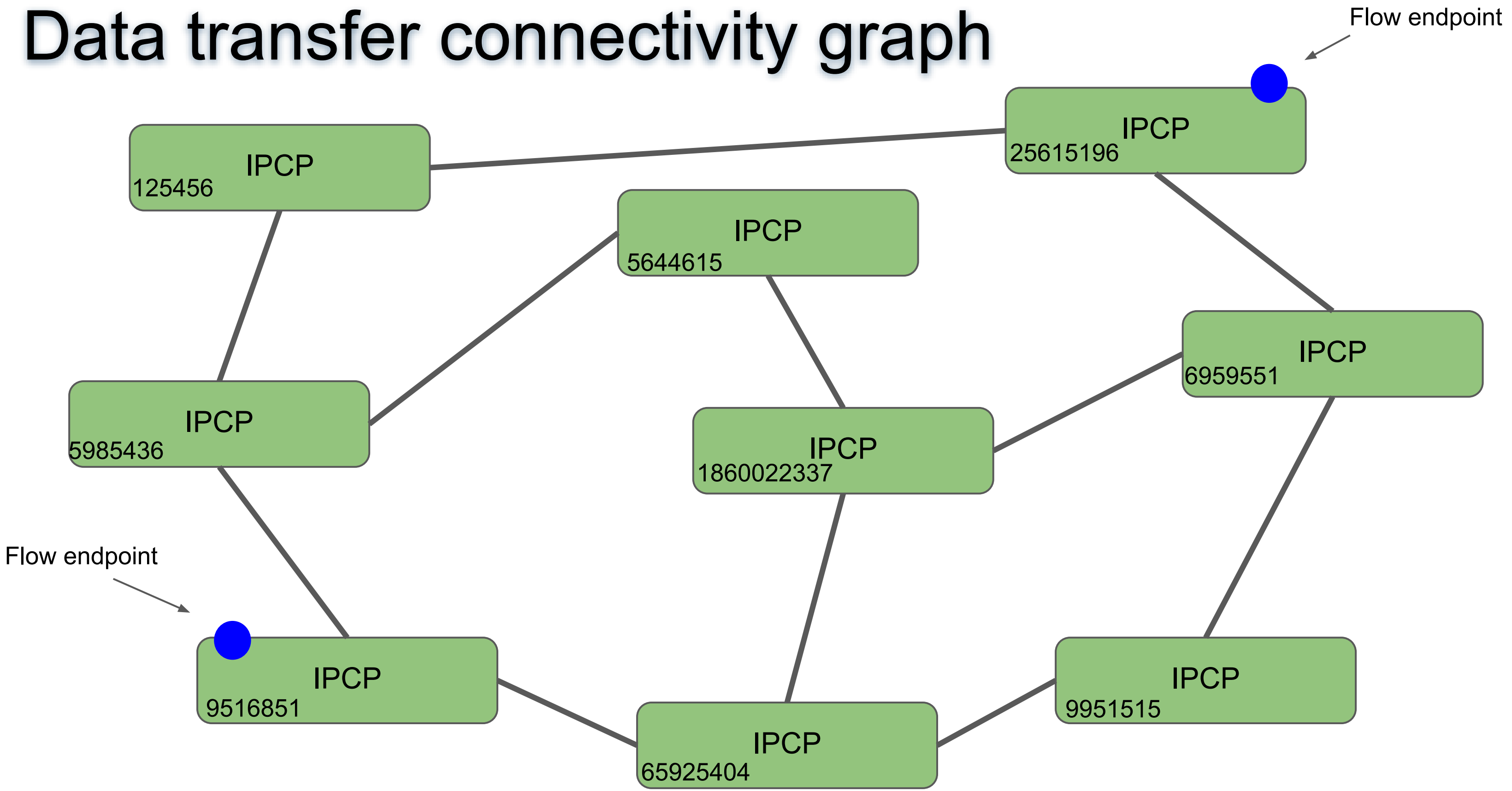
Data transfer connectivity graph



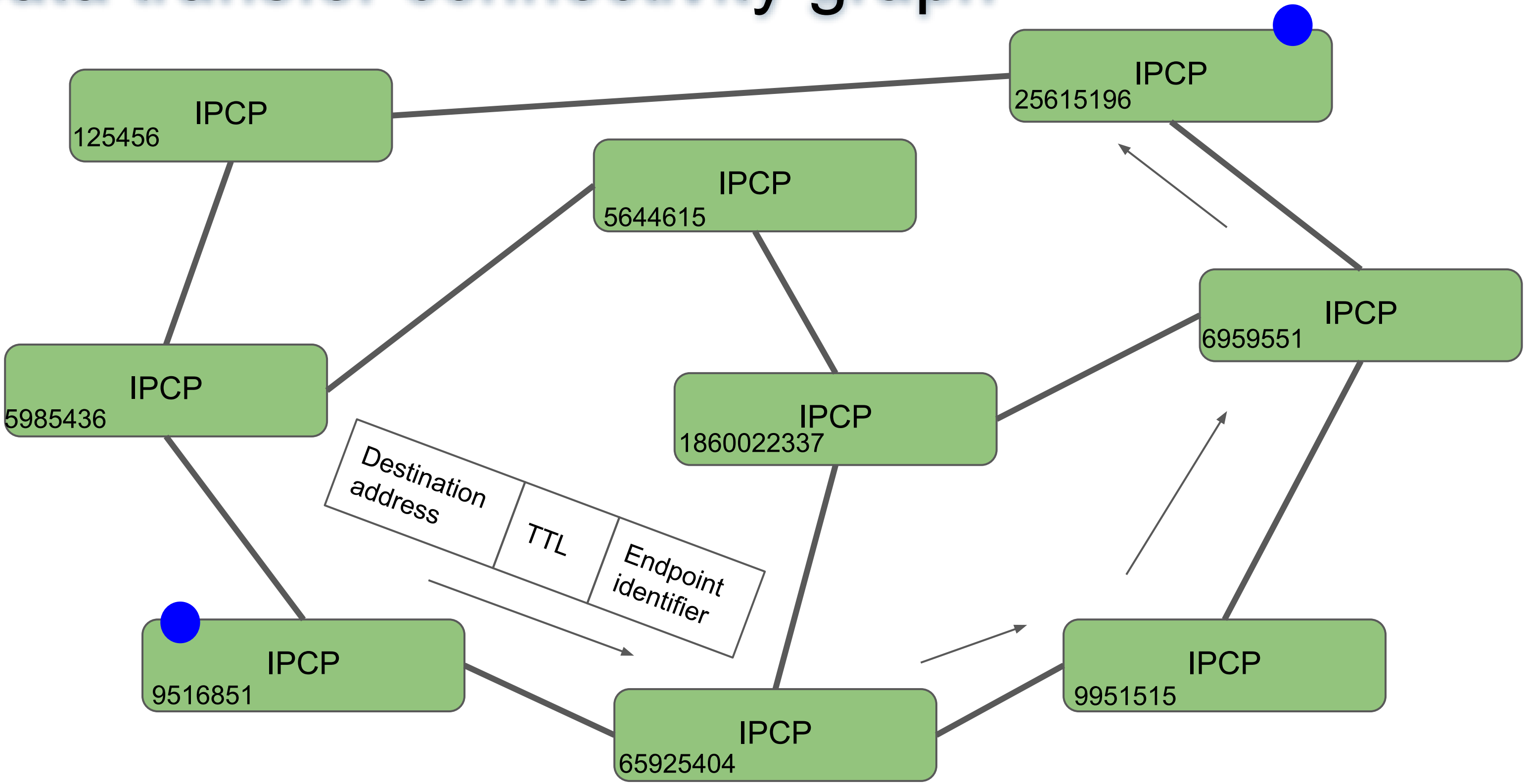
Data transfer connectivity graph



Data transfer connectivity graph



Data transfer connectivity graph



Data transfer connection

connection-manager(DB): Sending cacep info for protocol dtp to fd 65.
dt-ae(DB): Added fd 65 to SDU scheduler.
link-state-routing(DB): Type dt neighbor 142626484 added.

Normal IPCP “n1”

7976a340 Ethernet IPCP “e1” 3250c7b0

Ouroboros Subsystem (IRMd)
16373 - n1
16373 - n

$ irm ipcp connect name n1 component dt dst n2

System 1

dt-ae(DB): Added fd 65 to SDU scheduler.
link-state-routing(DB): Type dt neighbor 559955924 added.
dt-ae(DB): Could not get nhop for addr 559955924.
dt-ae(DB): Could not get nhop for addr 559955924.
dht(DB): Enrollment of DHT completed.

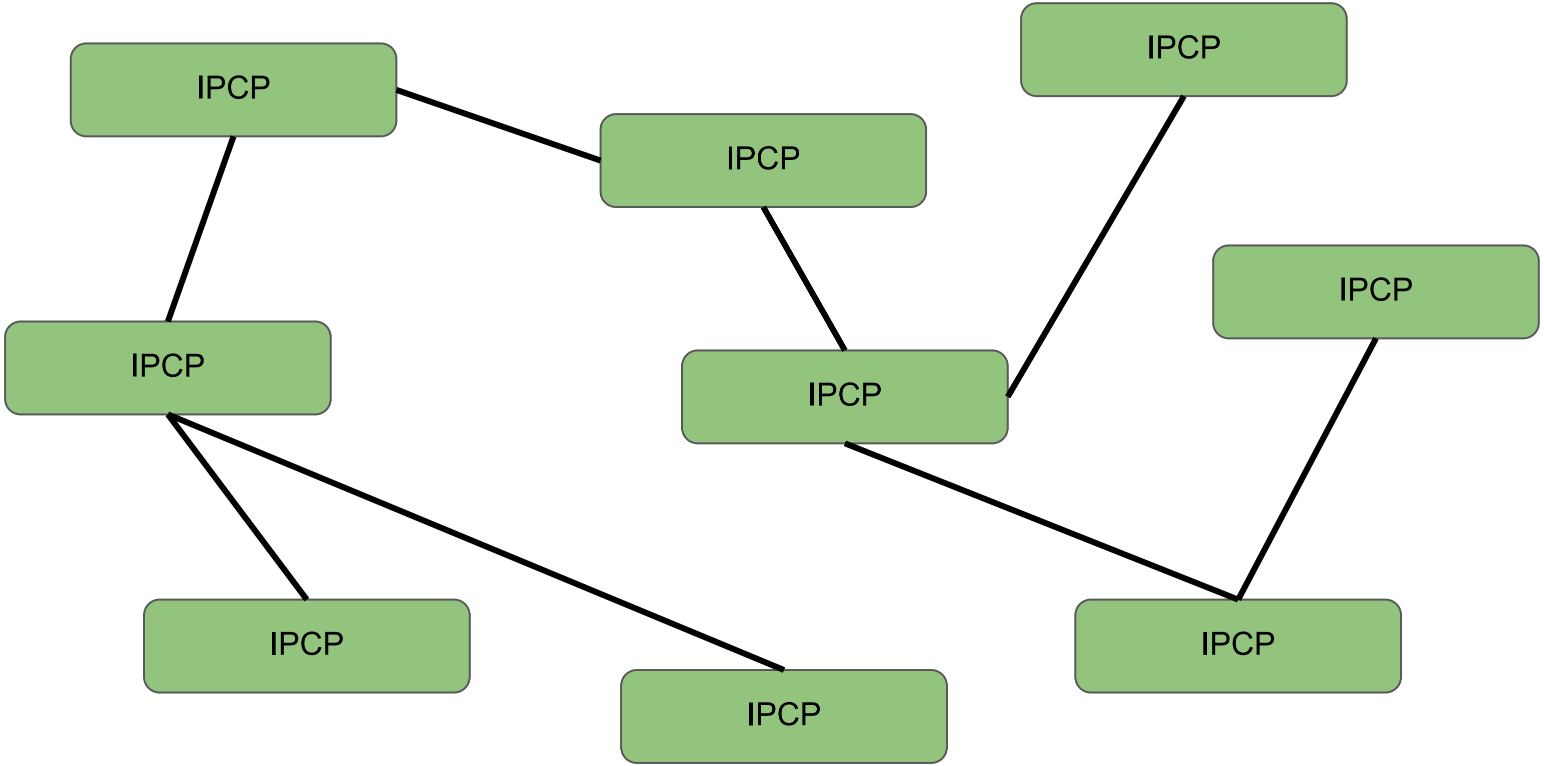
Normal IPCP “n2”

0808f8bd Ethernet IPCP “e2” 3250c7b0

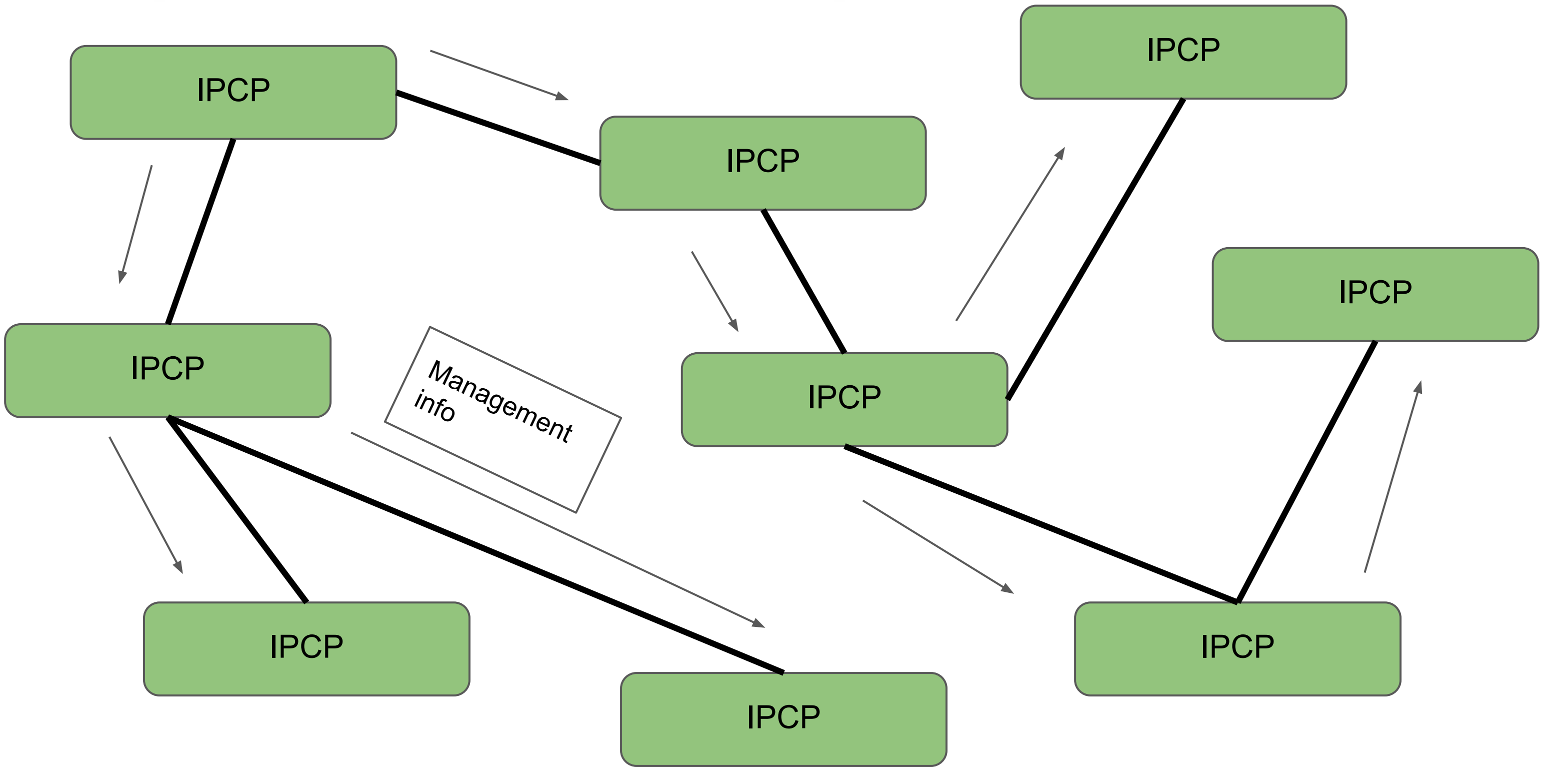
Ouroboros Subsystem (IRMd)
654 - n2
654 - n

System 2

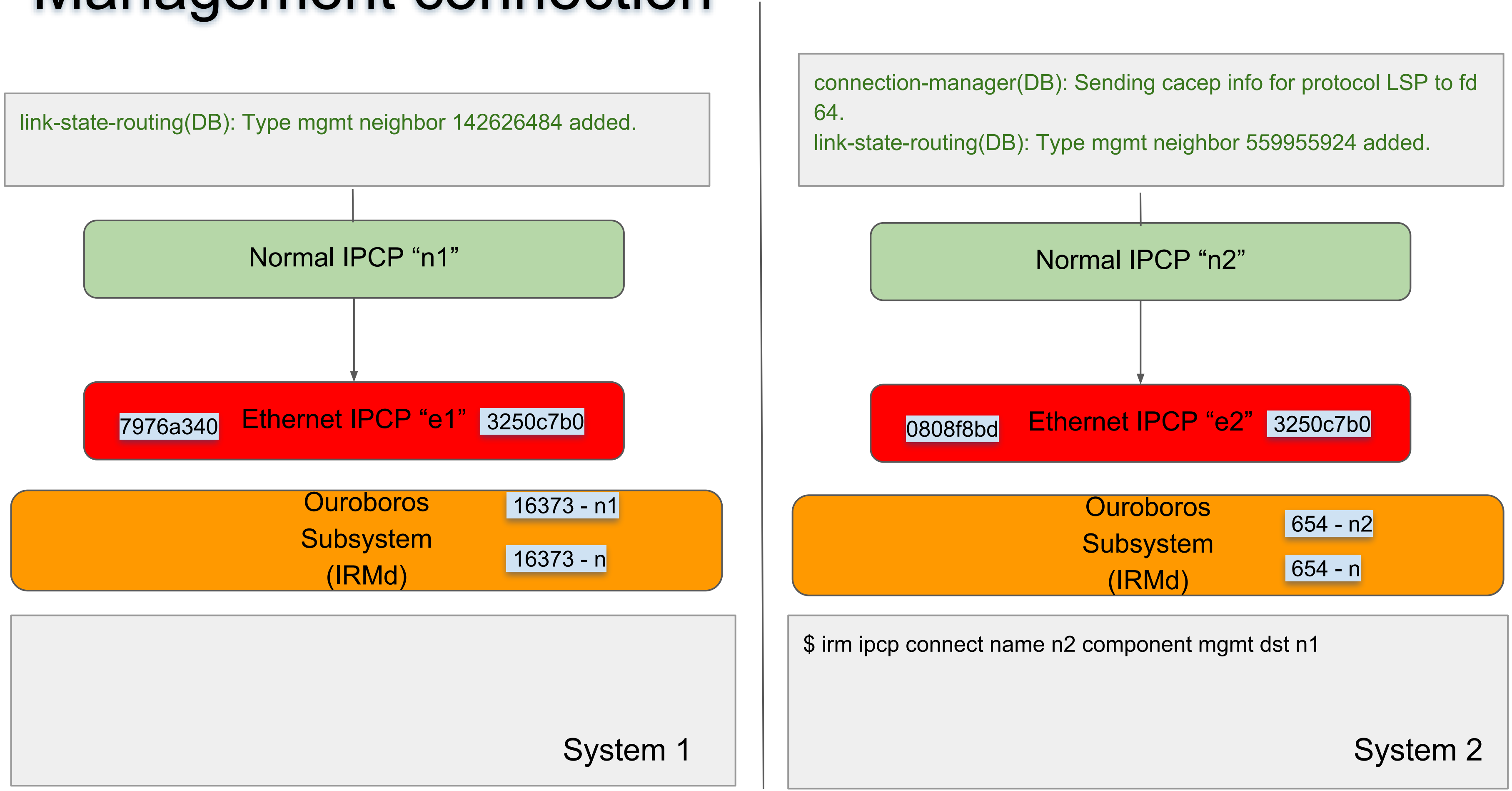
Management connectivity graph



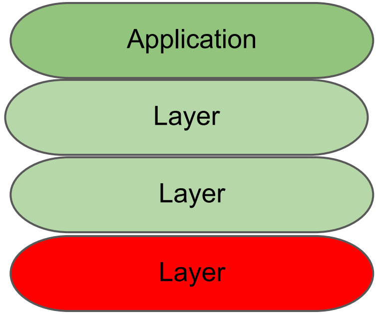
Management connectivity graph



Management connection

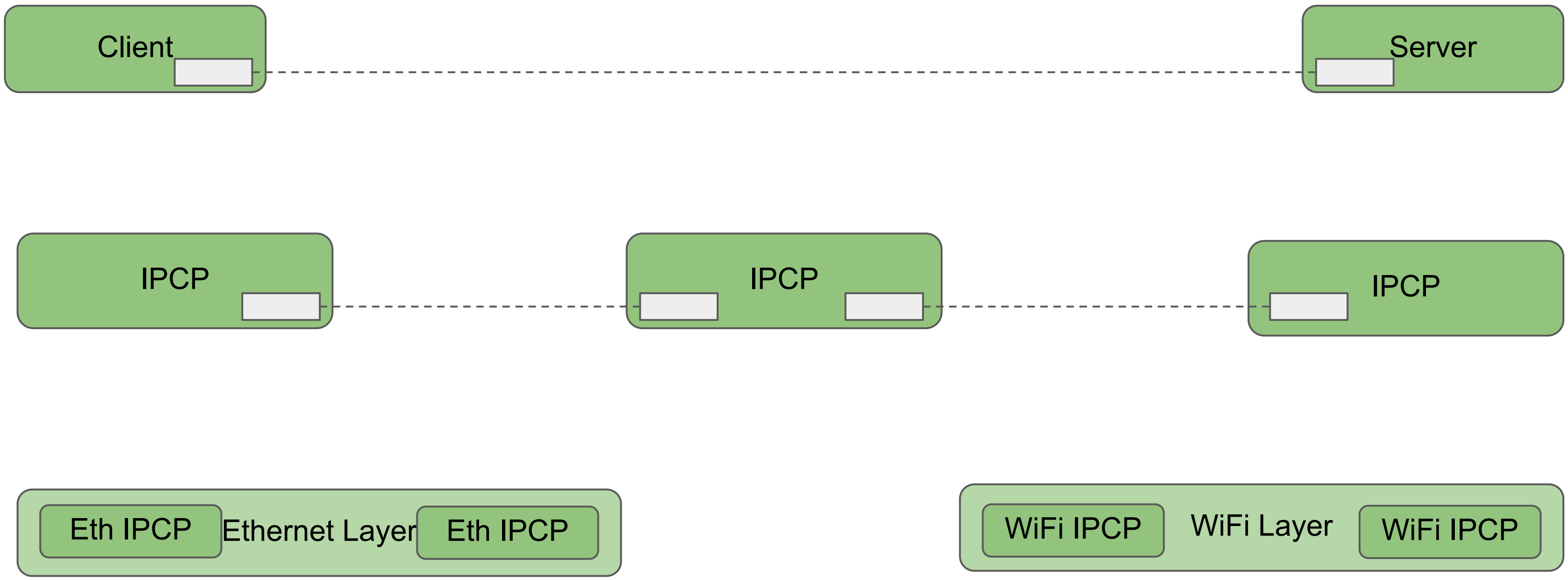


Ouroboros over Ouroboros



	flow allocation	routing	forwarding	directory	enrollment
raptor	ouroboros	N/A	N/A	ouroboros	N/A
eth-llc	ouroboros	RSTP	Ethernet	ouroboros	N/A or WiFi
udp	ouroboros	OSPF	IP	DDNS	N/A
ouroboros	ouroboros	IS-IS	ouroboros	DHT	Yes

Reliability (revisited)



NB3) connection management is available in every process and thus not a distinct function of a layer

Asynchronous I/O API

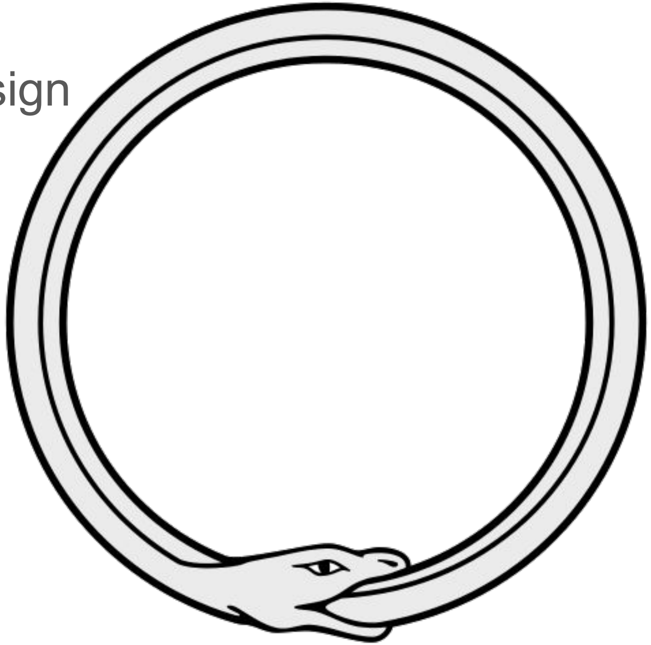
```
while (true) {
        fd = flow_accept(&qs, NULL);
        if (fd < 0) {
                printf("Failed to accept flow.\n");
                break;
        }

        printf("New flow %d.\n", fd);

        clock_gettime(CLOCK_REALTIME, &now);

        pthread_mutex_lock(&server.lock);
        fset_add(server.flows, fd);
        server.times[fd] = now;
        pthread_mutex_unlock(&server.lock);

        fccntl(fd, FLOWSFLAGS, FLOWFNONBLOCK | FLOWFRDWR);
}
```

```
while (true) {
        if (fevent(server.flows, server.fq, &timeout) == -ETIMEDOUT)
                continue;

        while ((fd = fqueue_next(server.fq)) >= 0) {
                msg_len = flow_read(fd, buf, OPING_BUF_SIZE);
                if (msg_len < 0)
                                continue;

                if (ntohl(msg->type) != ECHO_REQUEST) {
                                printf("Invalid message on fd %d.", fd);
                                continue;
                }

                clock_gettime(CLOCK_REALTIME, &now);

                pthread_mutex_lock(&server.lock);
                server.times[fd] = now;
                pthread_mutex_unlock(&server.lock);

                msg->type = htonl(ECHO_REPLY);

                if (flow_write(fd, buf, msg_len) < 0)
                                printf("Error writing to flow (fd %d).\n", fd);
}
```

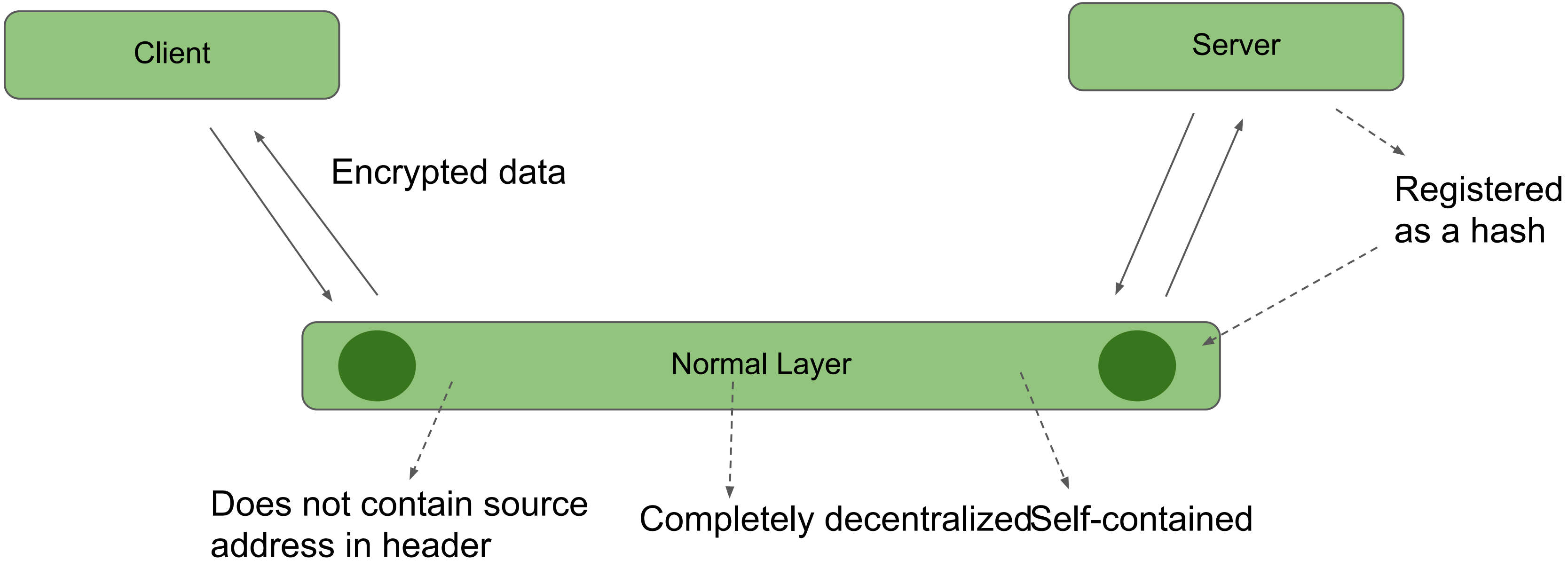
Wrapping Up

Ouroboros

- Provides a single layer abstraction
- Simplifies writing distributed applications
- Simplifies managing distributed applications
- Provides a secure and trustworthy network design
- Hides complexity



Ouroboros is more anonymous and secure



The future…   (03/02/2018)

- Research
  - Distributed address assignment
  - Efficient layer designs (routing, resource allocation, interactions)
  - Efficient congestion control
  - IoT devices
  - ...
- Implementation
  - Bug fixing
  - Optimization
  - Lockless data structures
  - Encryption (ECDH - AES)
  - …
- Deployment
  - Porting applications
  - Sockets emulator
  - ….

Join us...

freenode

#ouroboros

ouroboros@freelists.org

https://ouroboros.ilabt.imec.be

Acknowledgements

The development of Ouroboros was partly funded by the Flemish Government under grant no G045315N.

We would like to thank our colleagues for their feedback to improve this presentation.

We would like to thank our European and US project partners for all the valuable discussions on recursive network architectures.

We would like to thank our past and current master thesis students for their work on the prototypes.

We would like to thank our supervisors for the opportunity for us to work on this ambitious project.

That's all

we could cram into 50 minutes

folks!!