Minerva Connect

Michael Richardson

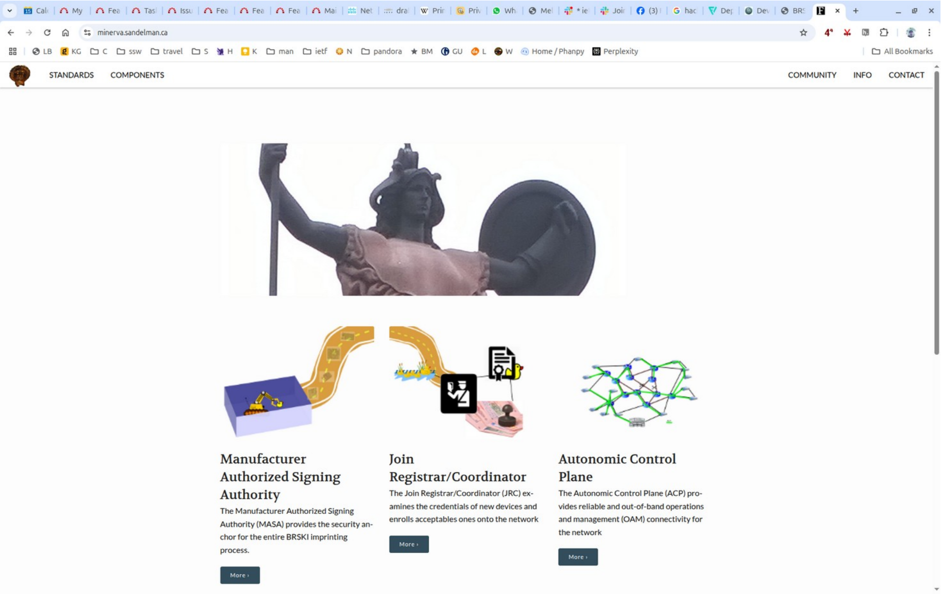
mcr+ietf@sandelman.ca

https://minerva.sandelman.ca

slides at https://www.sandelman.ca/SSW/talk/2025-ipsec-workshop

Goals

- A non-trivial (non-Masters-Thesis) implementation of RFC8994, RFC8995 and related IoT specifications
- Target: Linux/OpenWRT based routers, OpenBMC,
  - providing virtual out-of-band console server product
- Registrar Virtual Appliance



Manufacturer Authorized Signing Authority

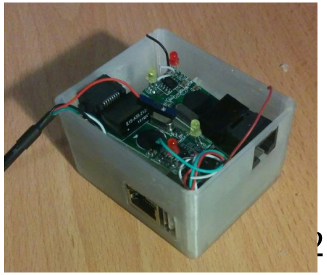
The Manufacturer Authorized Signing Authority (MASA) provides the security anchor for the entire BRSKI imprinting process.

Join Registrar/Coordinator

The Join Registrar/Coordinator (JRC) examines the credentials of new devices and enrolls acceptables ones onto the network

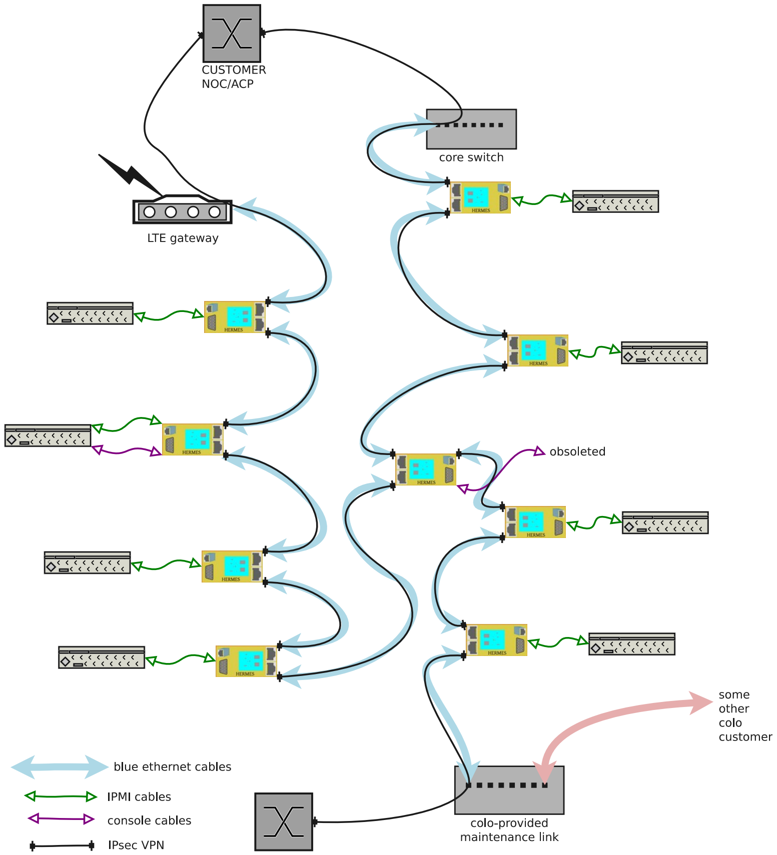
Autonomic Control Plane

The Autonomic Control Plane (ACP) provides reliable and out-of-band operations and management (OAM) connectivity for the network



Use Case: Data Center BMC / serial server

- This is an itch that I have.
- Remote management of physical servers
- *... debugging production systems with kernel issues*
    - *ancient Trauma due to 2000-era KLIPS*

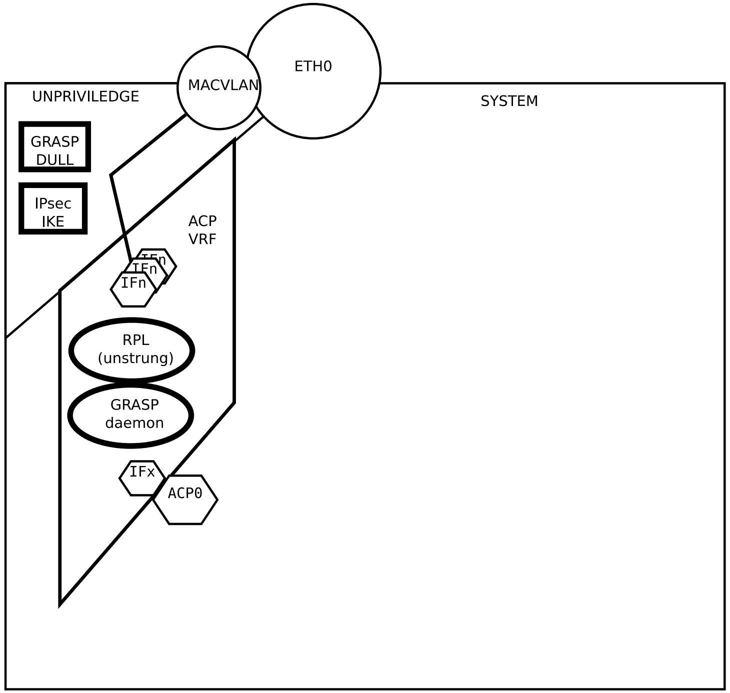


Minerva Components

- CONNECT
- BOOTSTRAP
- ROOSTER
- BRSKI
  - BEACH (Pledge)
  - FOUNTAIN (Registrar)
  - HIGHWAY (MASA)
- UNSTRUNG (RPL)
- Bluerose (*SWAN)

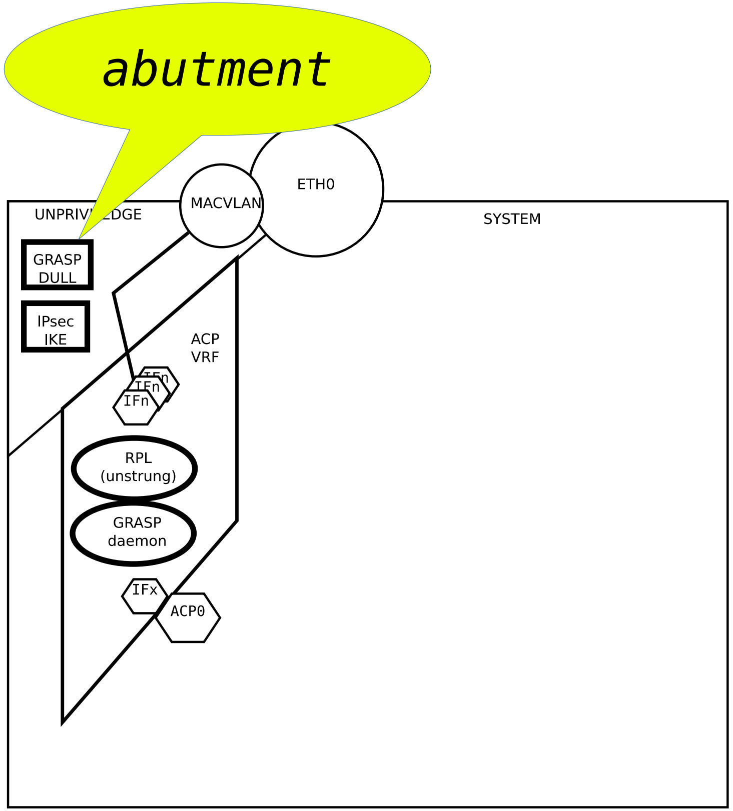
CONNECT Architecture - 1

- Uses Linux network namespaces
  - Not a fully isolated container.
- Parent process deals with network interfaces coming/going, and creates virtual interfaces that it pushes into the unpriviledged “dull” space.
  - Calling it the “abutment” space since early 2024.
  - IKE daemon runs in the abutment space
  - GRASP DULL daemon runs in the abutment space
- A second space is the “ACP” space
  - The RPL daemon runs in the ACP space.
  - Full GRASP daemon will run in the acp space
- System sees a single interface, “acp0”, which has an IPv6 address assigned by the Registrar, and a /48 route for the rest of the ACP



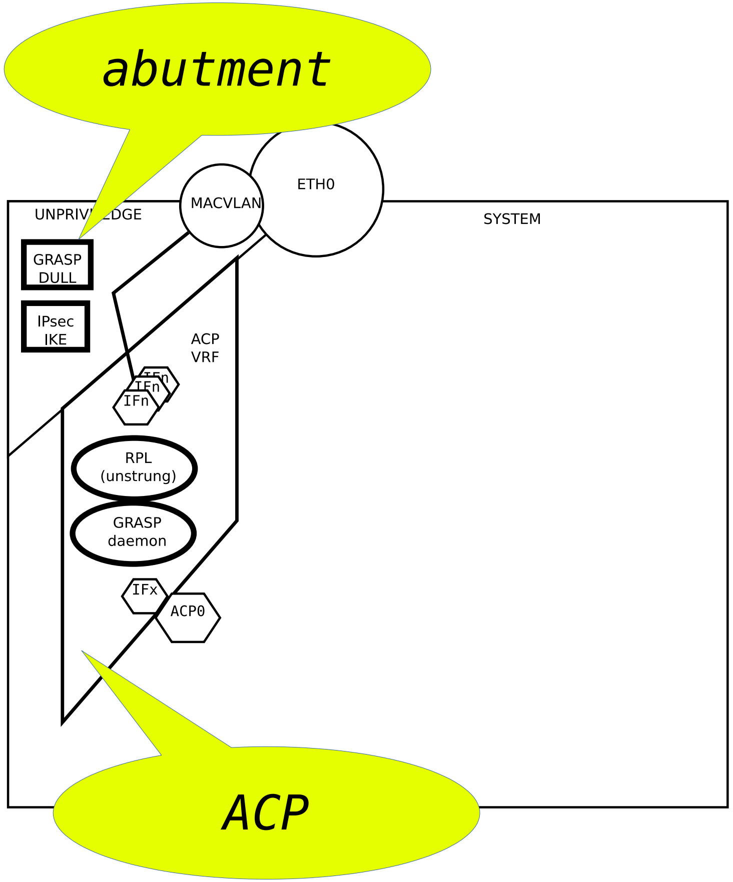
CONNECT Architecture - 1

- Uses Linux network namespaces
  - Not a fully isolated container.
- Parent process deals with network interfaces coming/going, and creates virtual interfaces that it pushes into the unpriviledged “dull” space.
  - Calling it the “abutment” space since early 2024.
  - IKE daemon runs in the abutment space
  - GRASP DULL daemon runs in the abutment space
- A second space is the “ACP” space
  - The RPL daemon runs in the ACP space.
  - Full GRASP daemon will run in the acp space
- System sees a single interface, “acp0”, which has an IPv6 address assigned by the Registrar, and a /48 route for the rest of the ACP



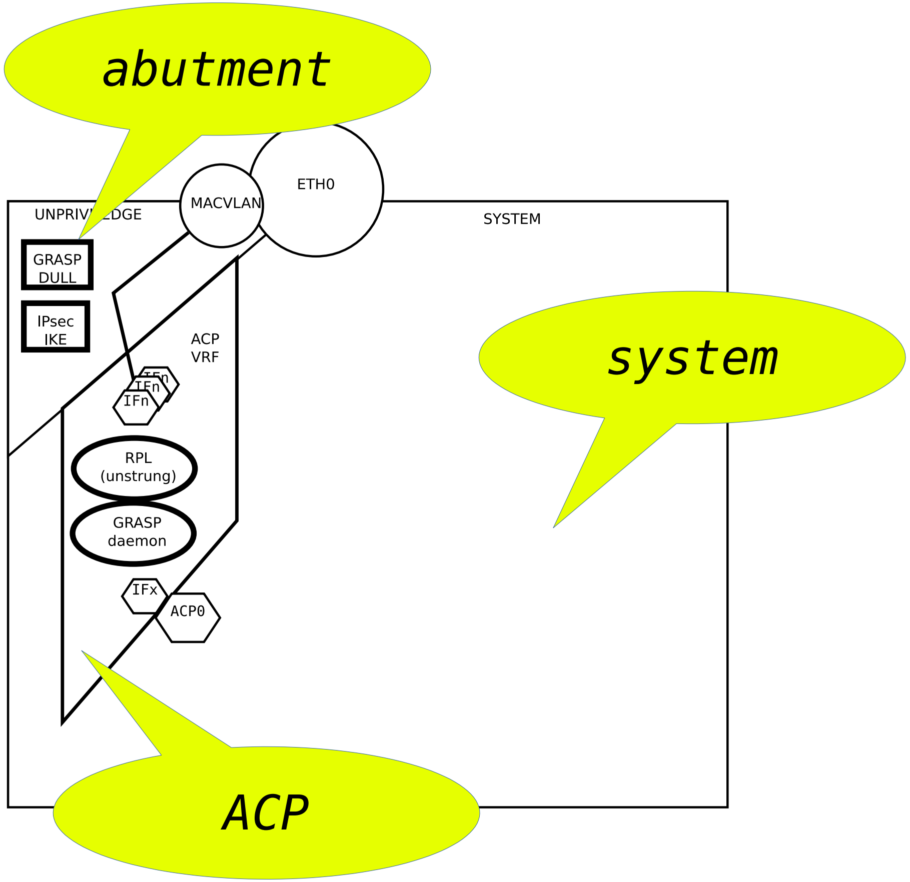
CONNECT Architecture - 1

- Uses Linux network namespaces
  - Not a fully isolated container.
- Parent process deals with network interfaces coming/going, and creates virtual interfaces that it pushes into the unpriviledged “dull” space.
  - Calling it the “abutment” space since early 2024.
  - IKE daemon runs in the abutment space
  - GRASP DULL daemon runs in the abutment space
- A second space is the “ACP” space
  - The RPL daemon runs in the ACP space.
  - Full GRASP daemon will run in the acp space
- System sees a single interface, “acp0”, which has an IPv6 address assigned by the Registrar, and a /48 route for the rest of the ACP

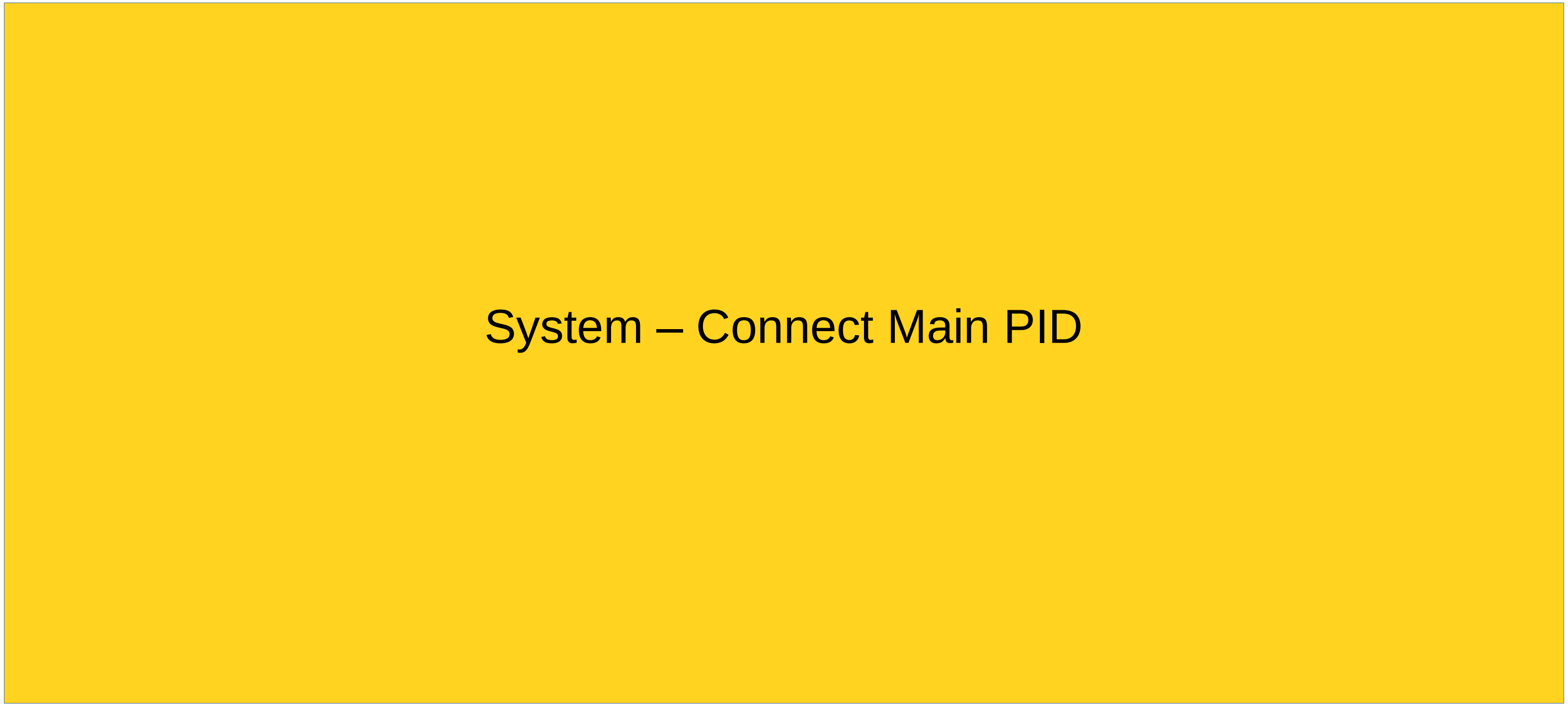


CONNECT Architecture - 1

- Uses Linux network namespaces
  - Not a fully isolated container.
- Parent process deals with network interfaces coming/going, and creates virtual interfaces that it pushes into the unpriviledged "dull" space.
  - Calling it the "abutment" space since early 2024.
  - IKE daemon runs in the abutment space
  - GRASP DULL daemon runs in the abutment space
- A second space is the "ACP" space
  - The RPL daemon runs in the ACP space.
  - Full GRASP daemon will run in the acp space
- System sees a single interface, "acp0", which has an IPv6 address assigned by the Registrar, and a /48 route for the rest of the ACP

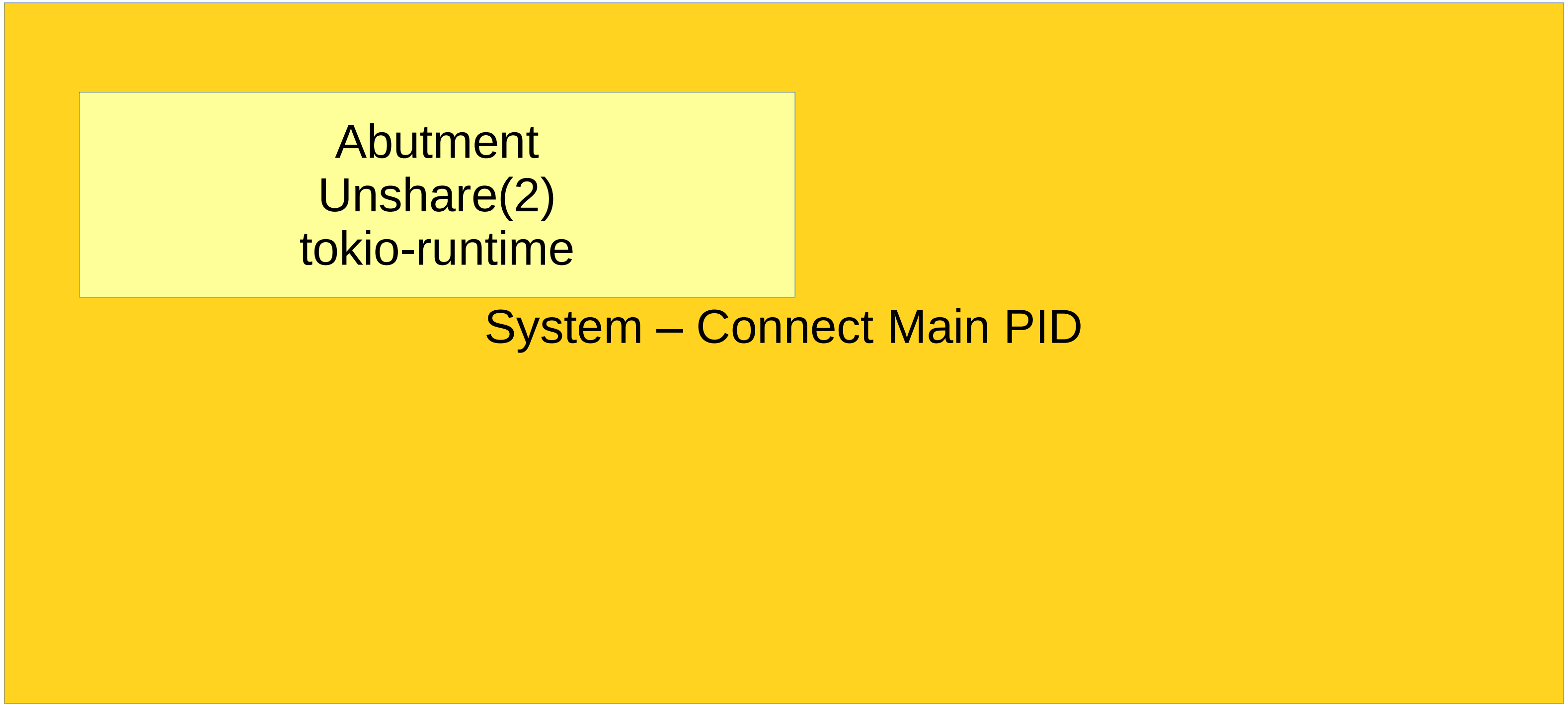


Architecture Diagram - 2



System – Connect Main PID

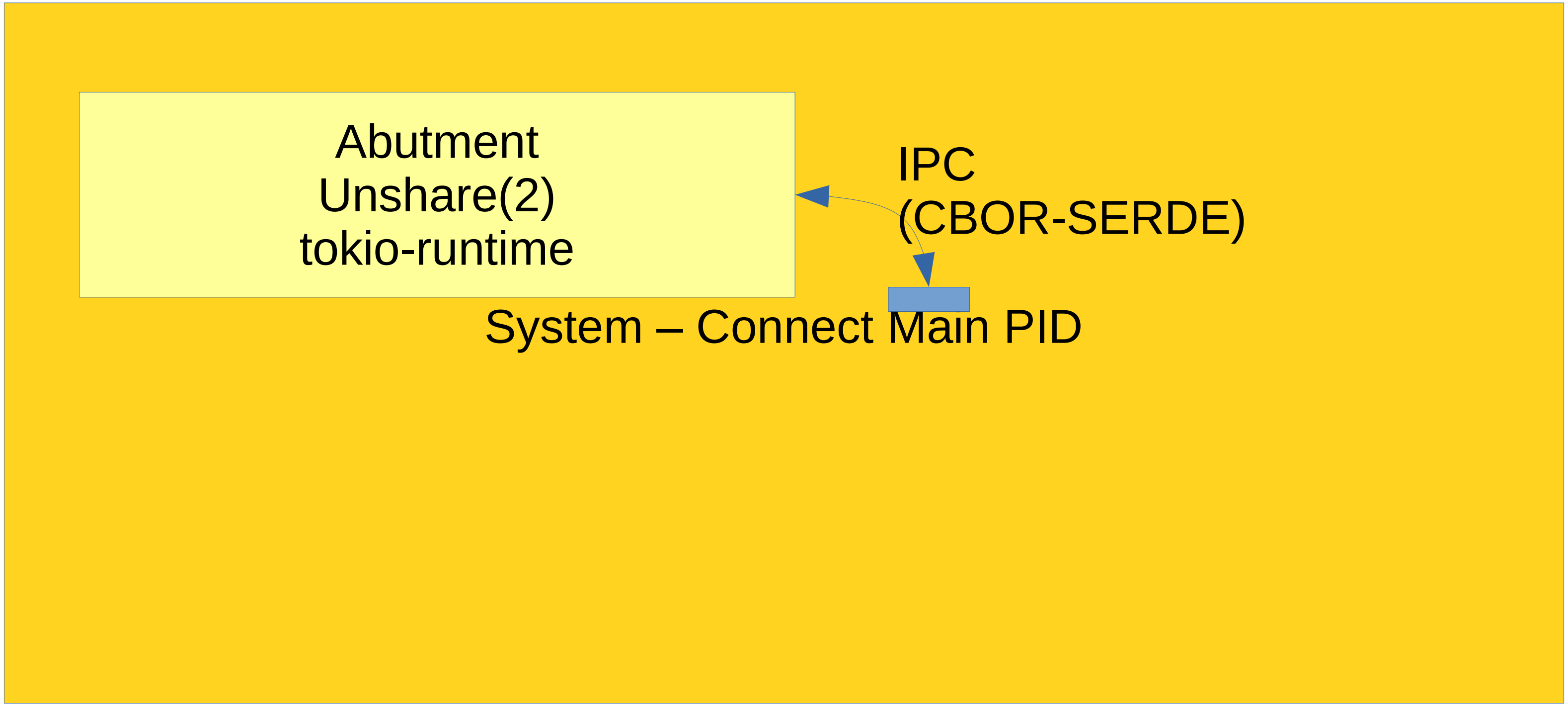
Architecture Diagram - 2



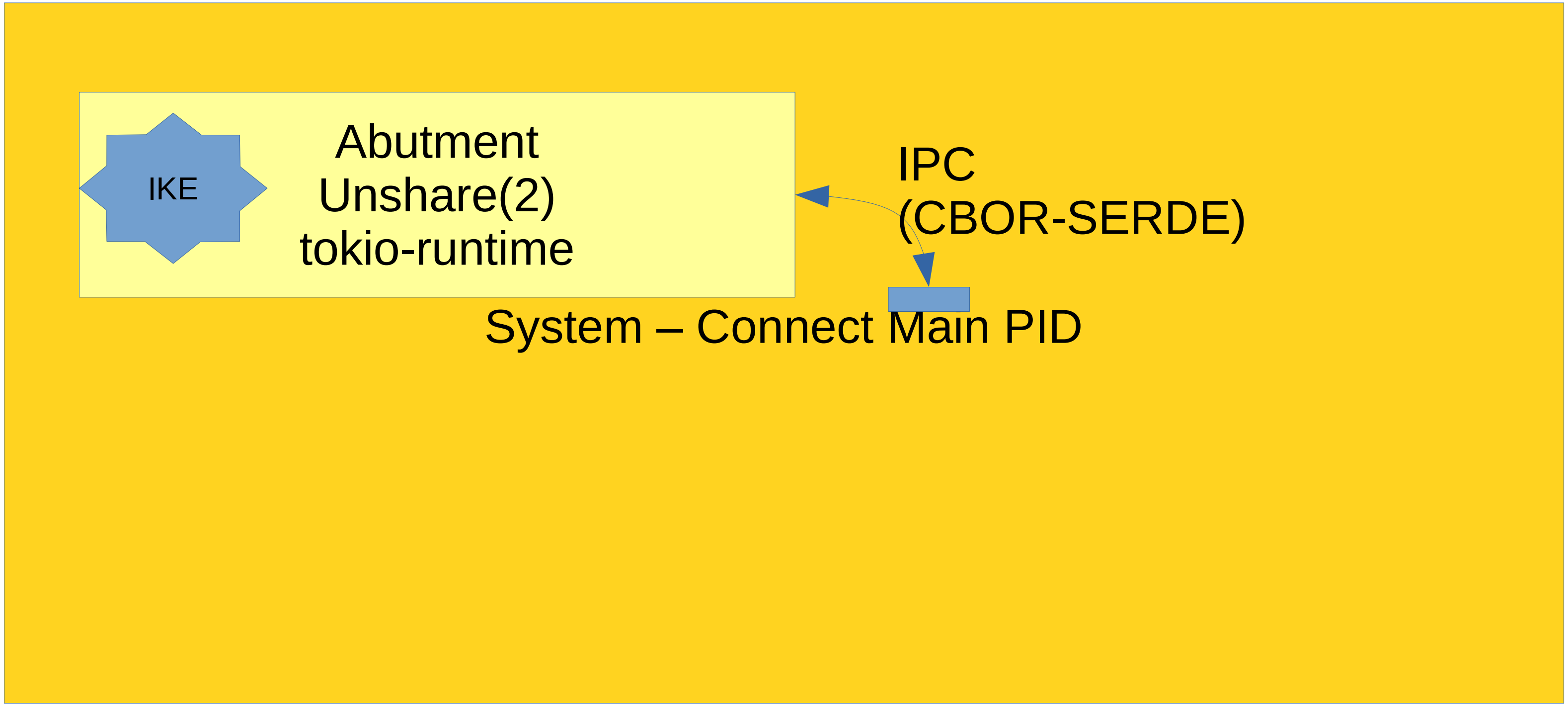
Abutment
Unshare(2)
tokio-runtime

System – Connect Main PID

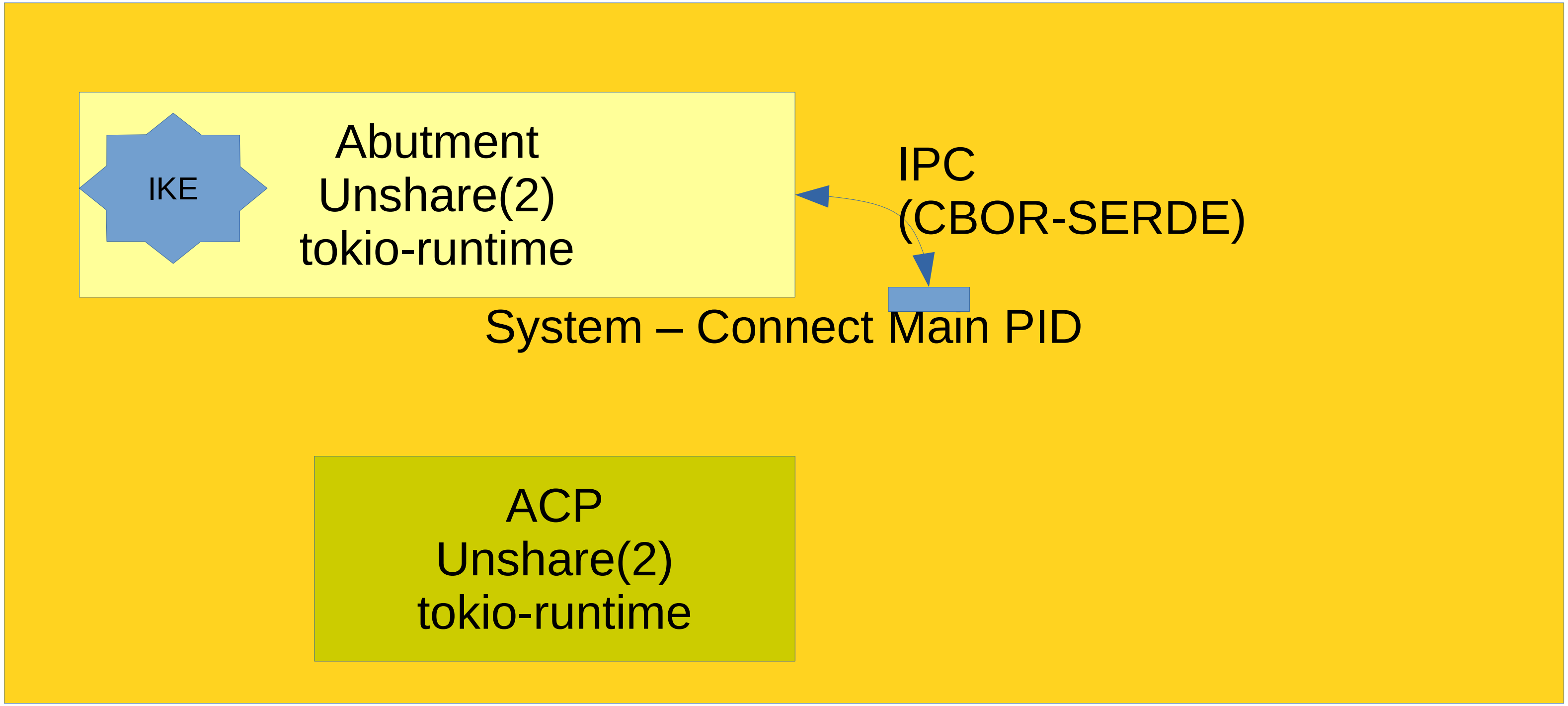
Architecture Diagram - 2



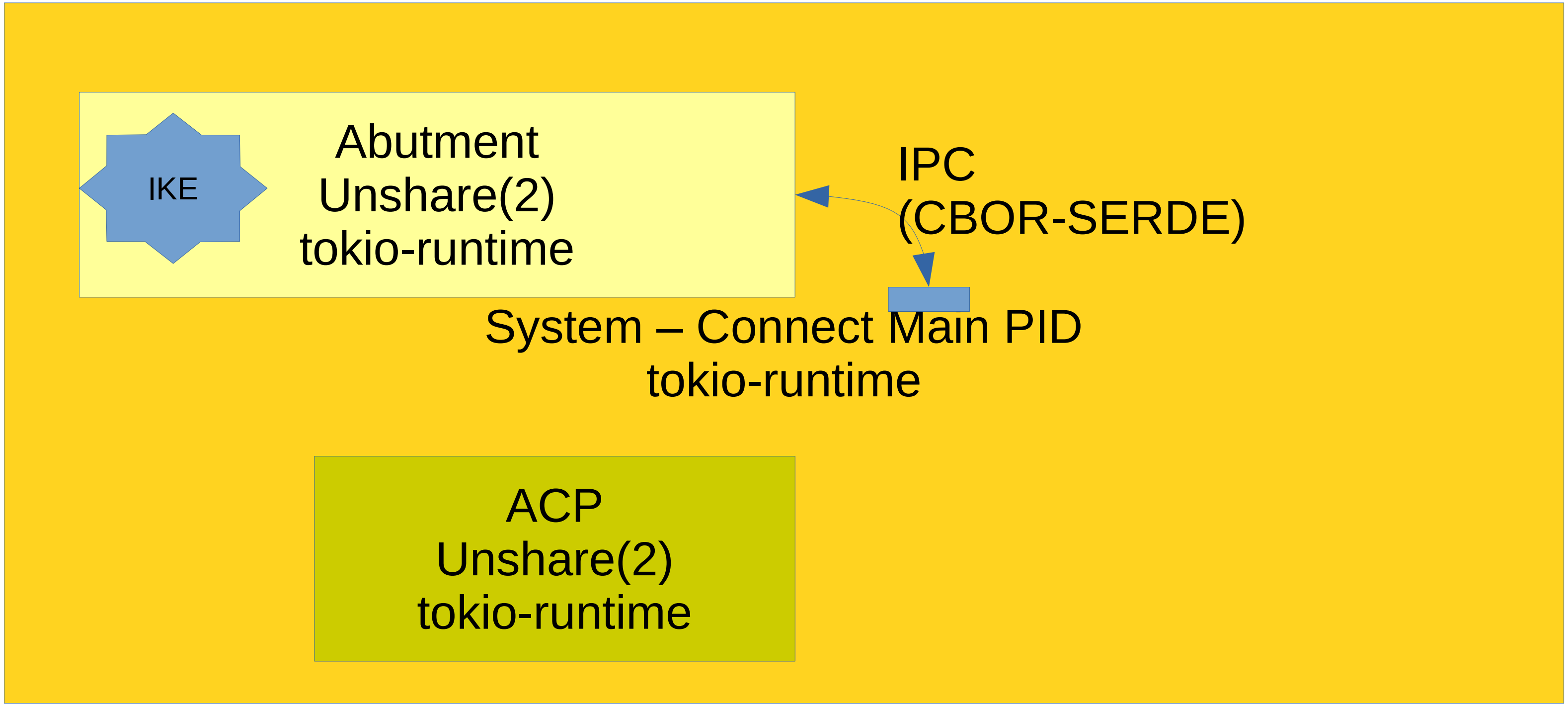
Architecture Diagram - 2



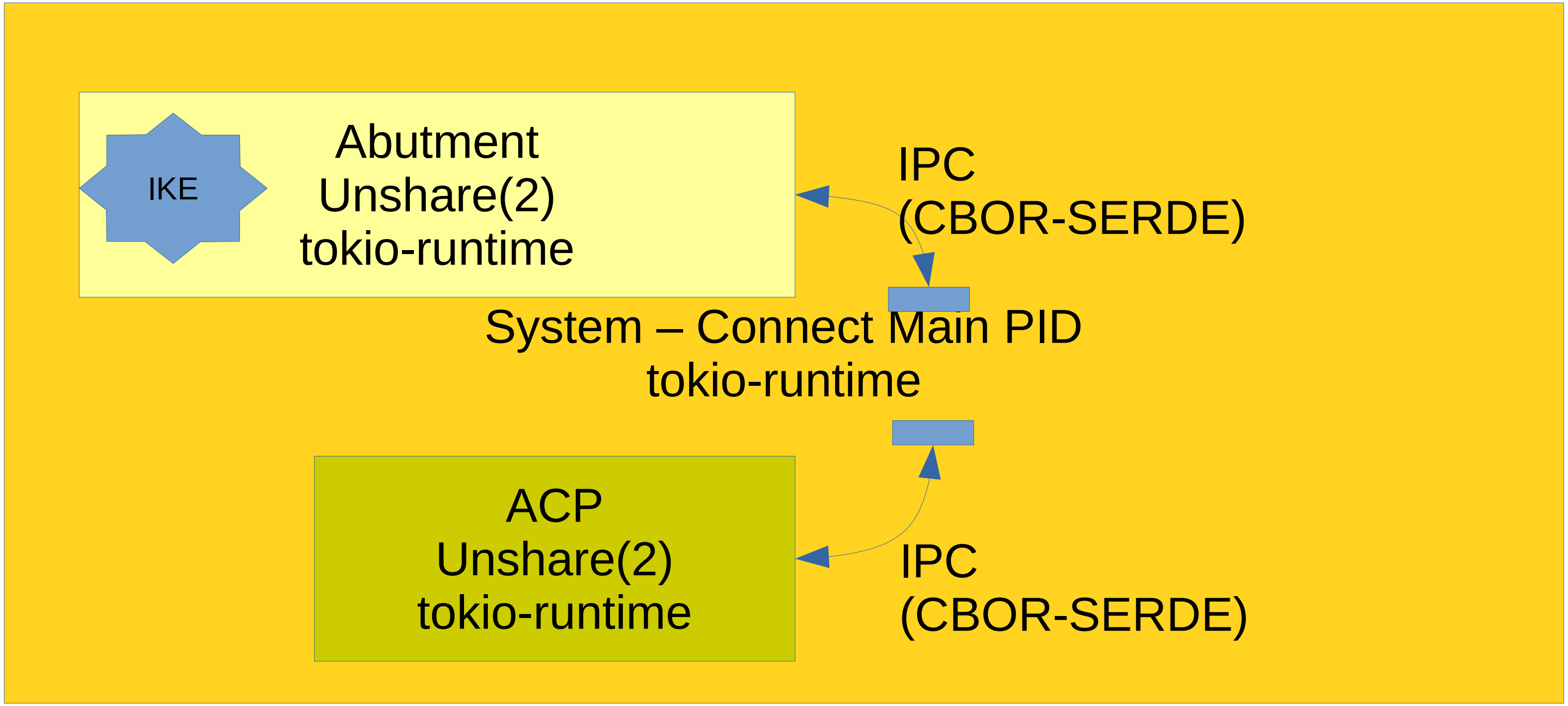
Architecture Diagram - 2



Architecture Diagram - 2



Architecture Diagram - 2



IKE

Abutment
Unshare(2)
tokio-runtime

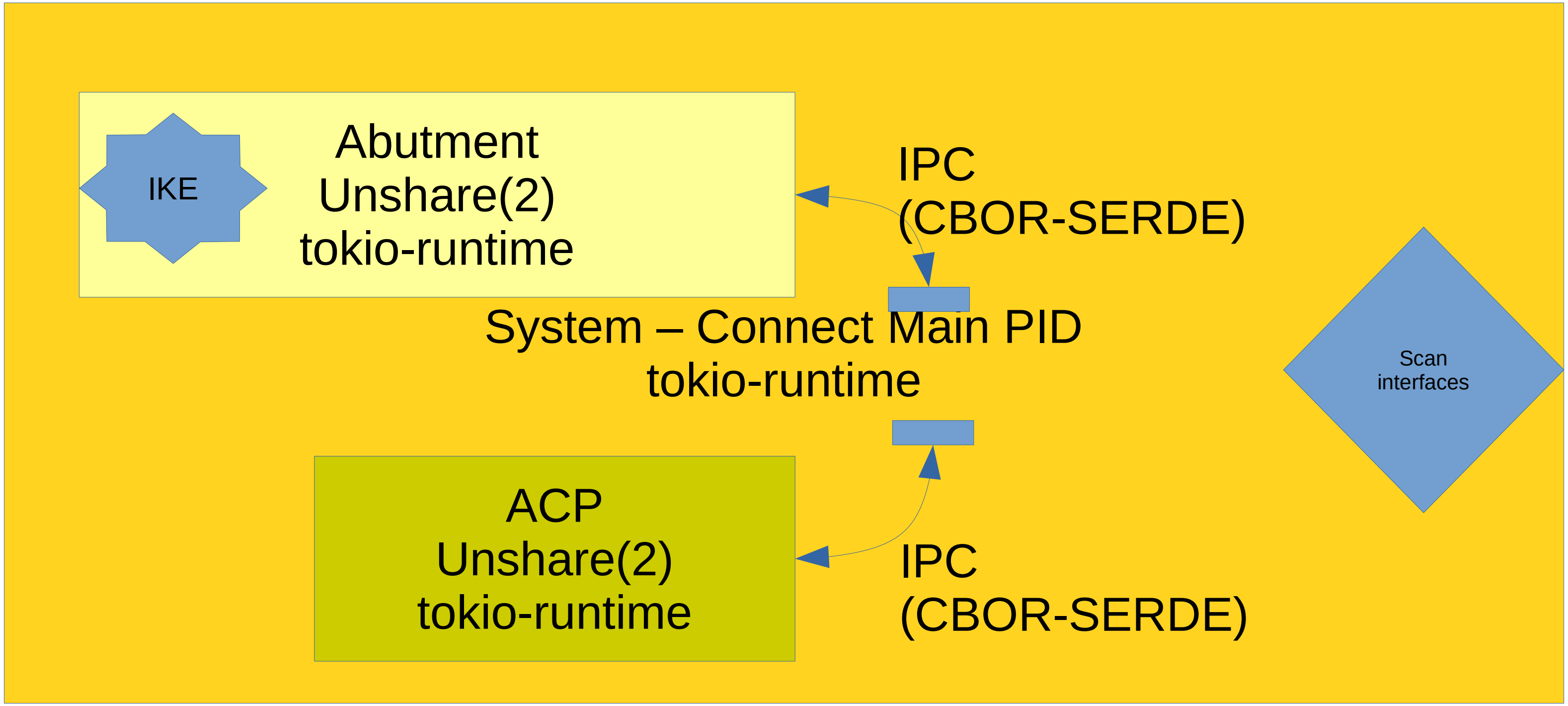
IPC
(CBOR-SERDE)

System – Connect Main PID
tokio-runtime

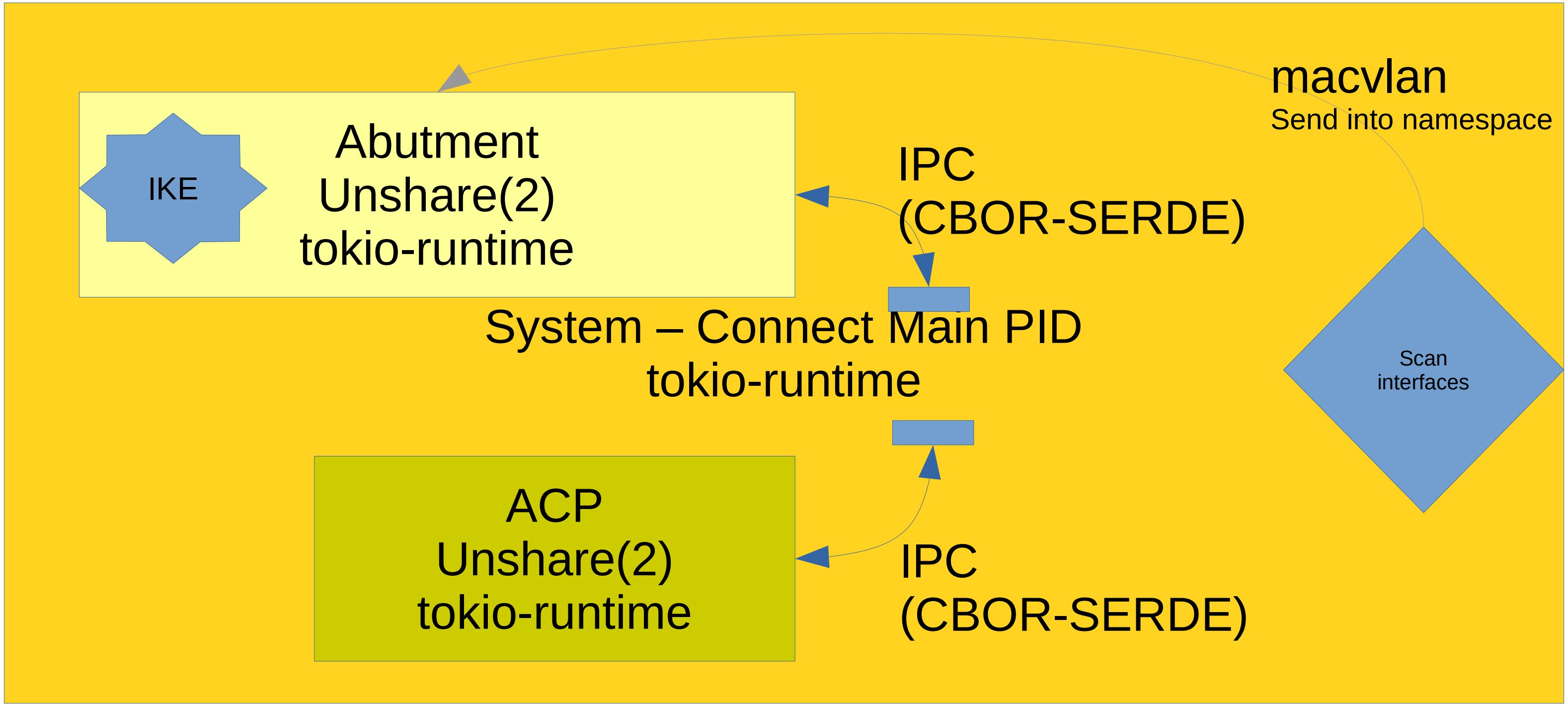
ACP
Unshare(2)
tokio-runtime

IPC
(CBOR-SERDE)

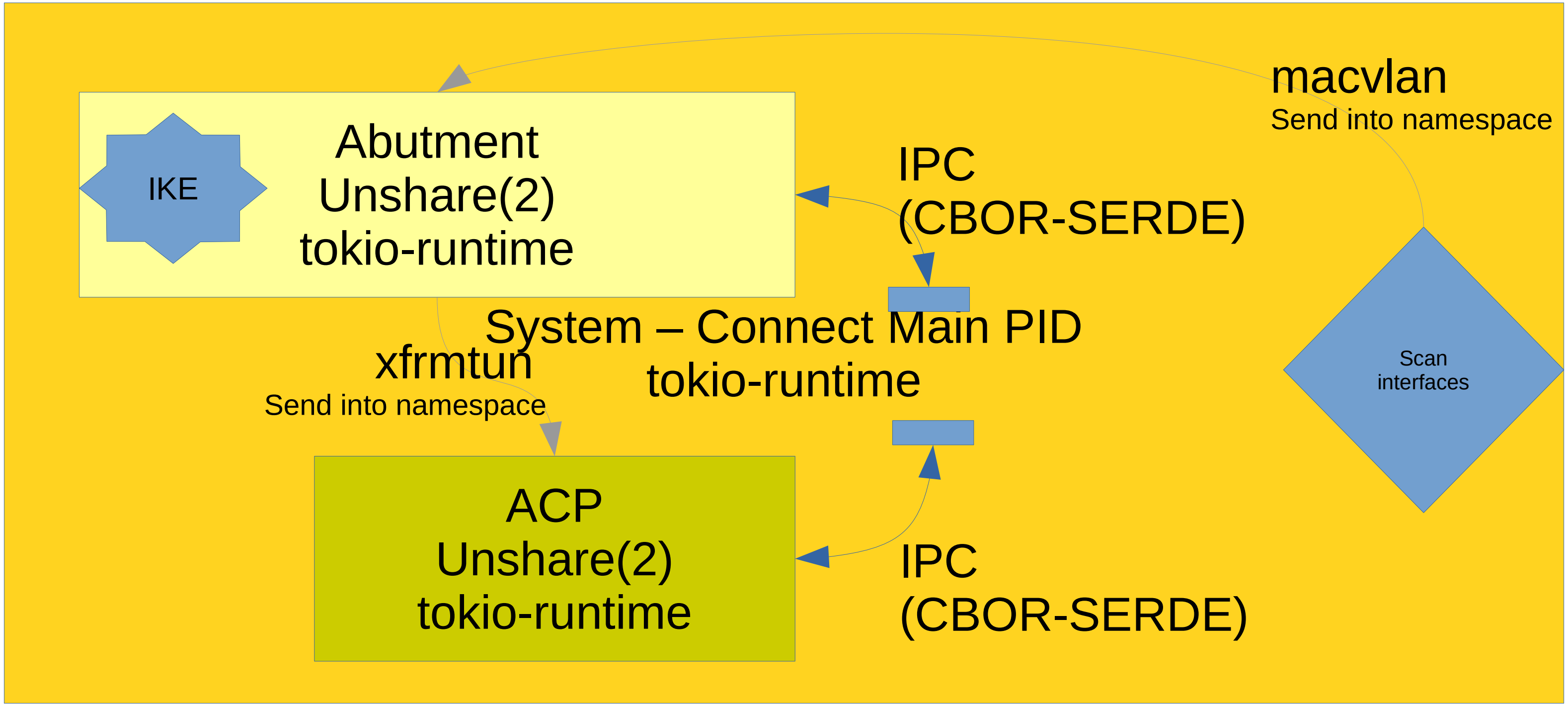
Architecture Diagram - 2



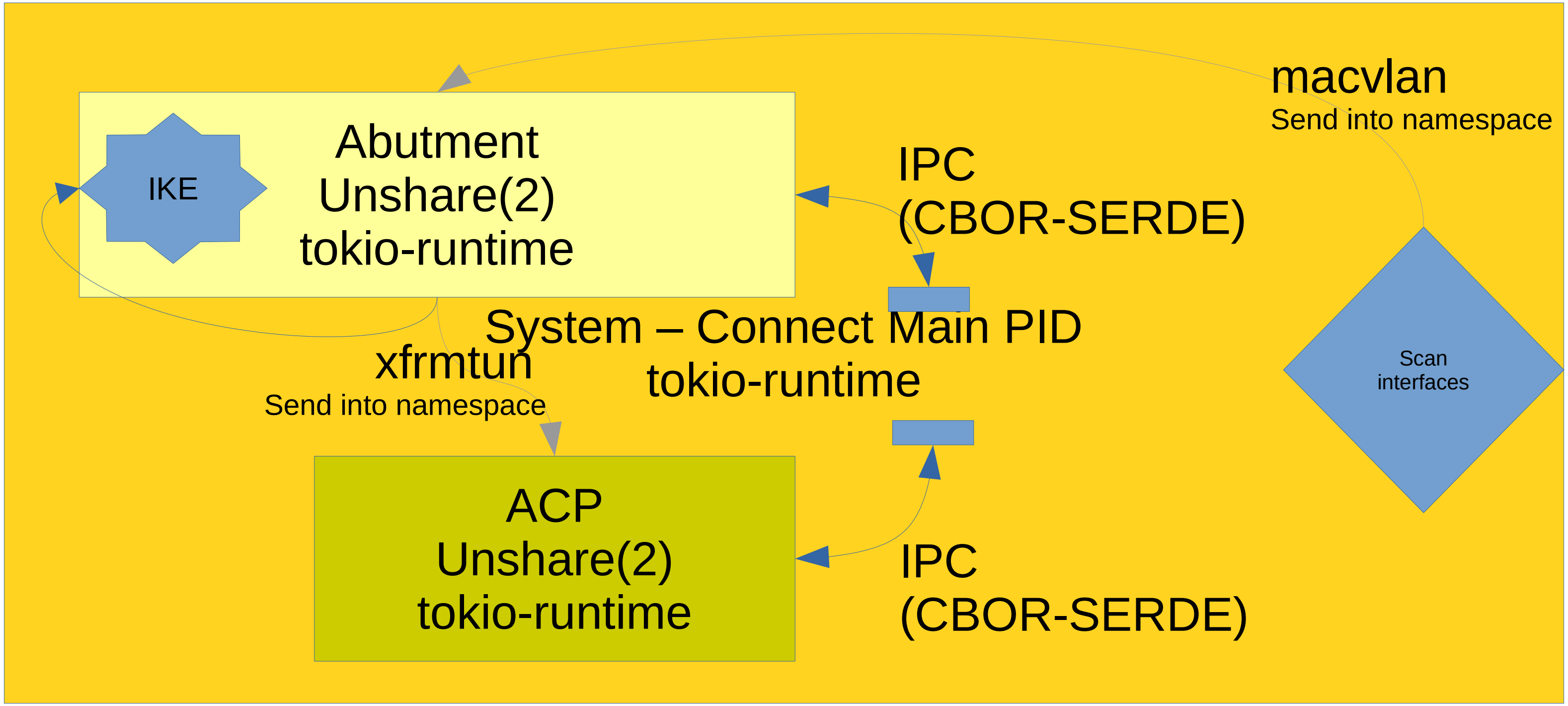
Architecture Diagram - 2



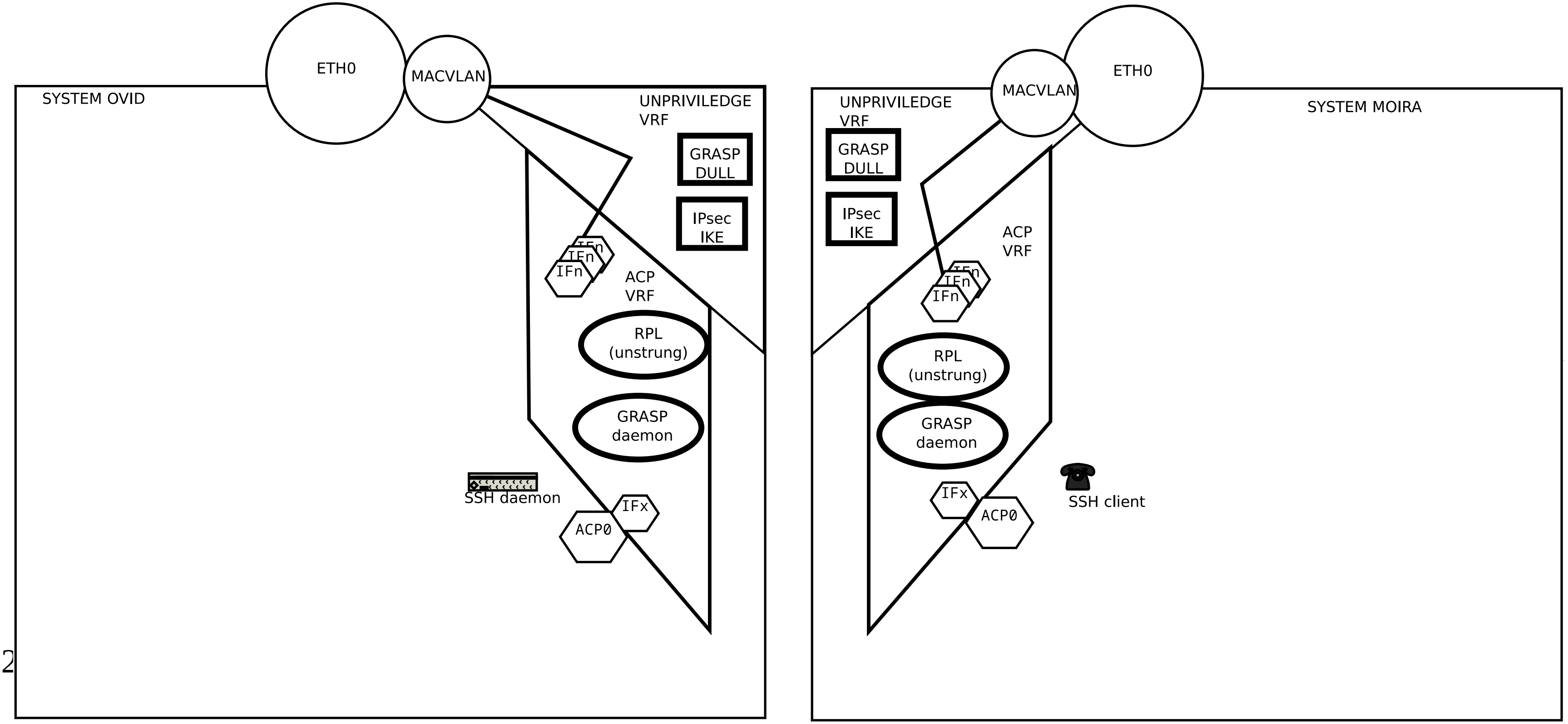
Architecture Diagram - 2



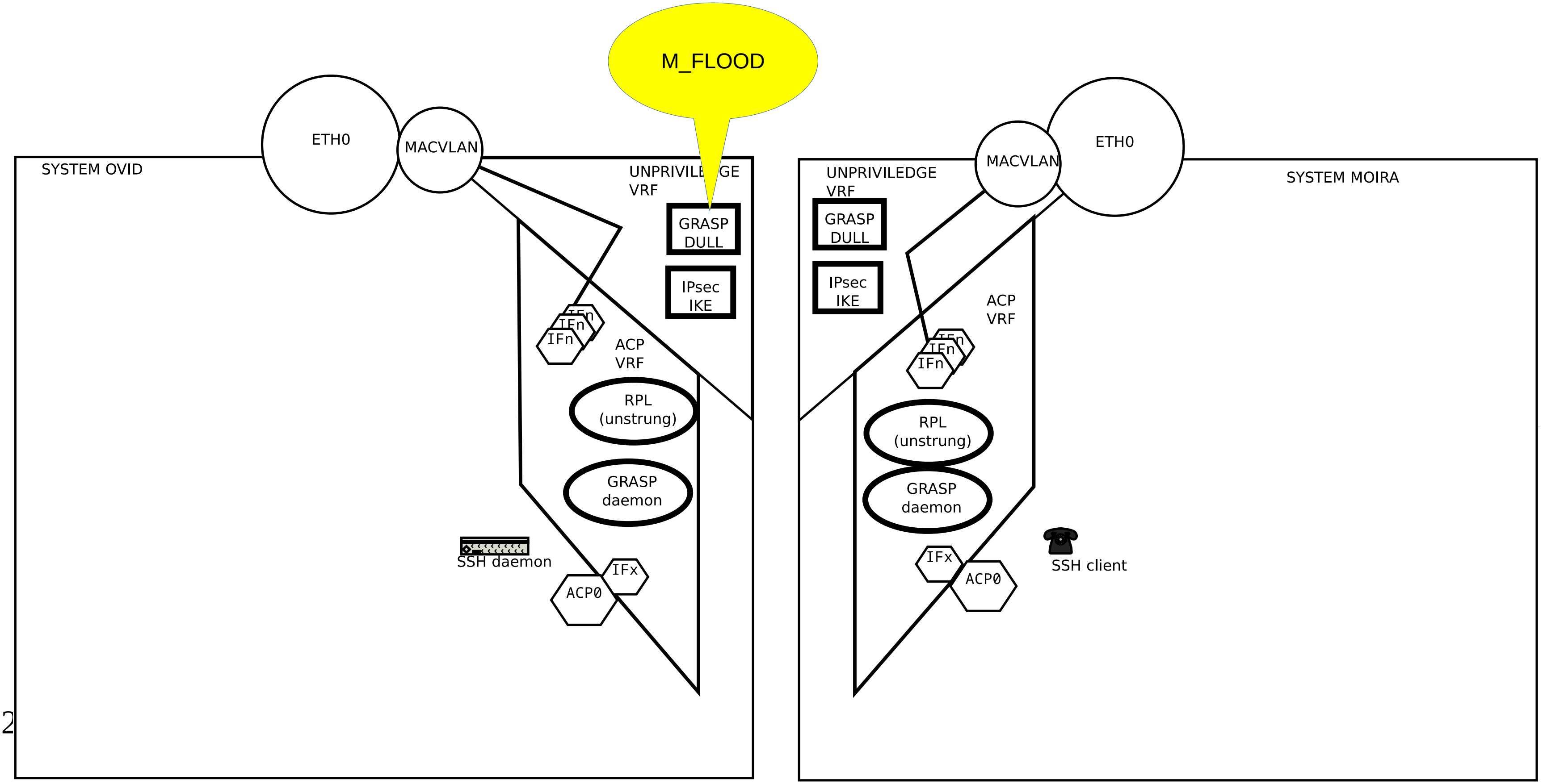
Architecture Diagram - 2



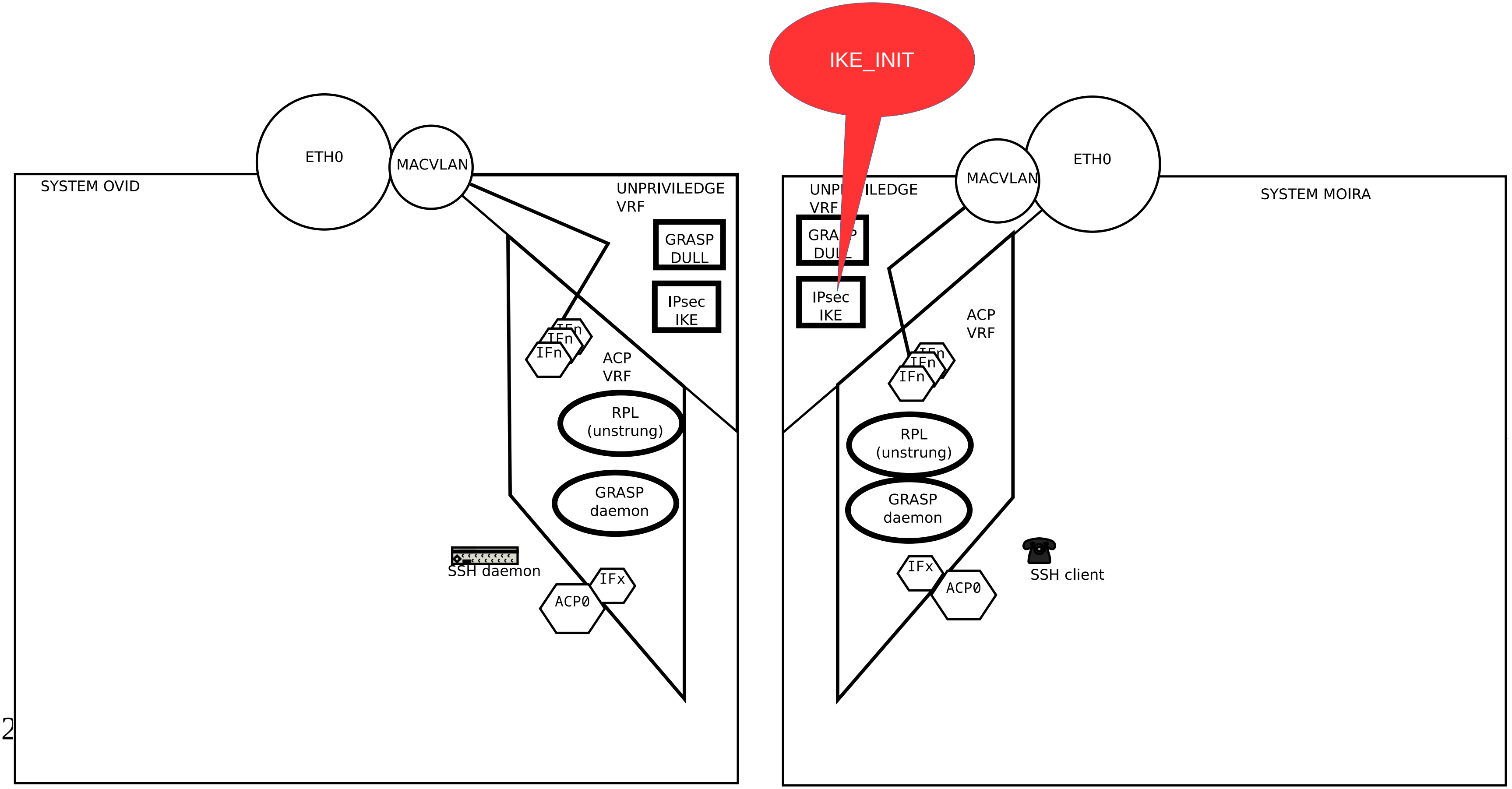
Architecture Diagram - 3



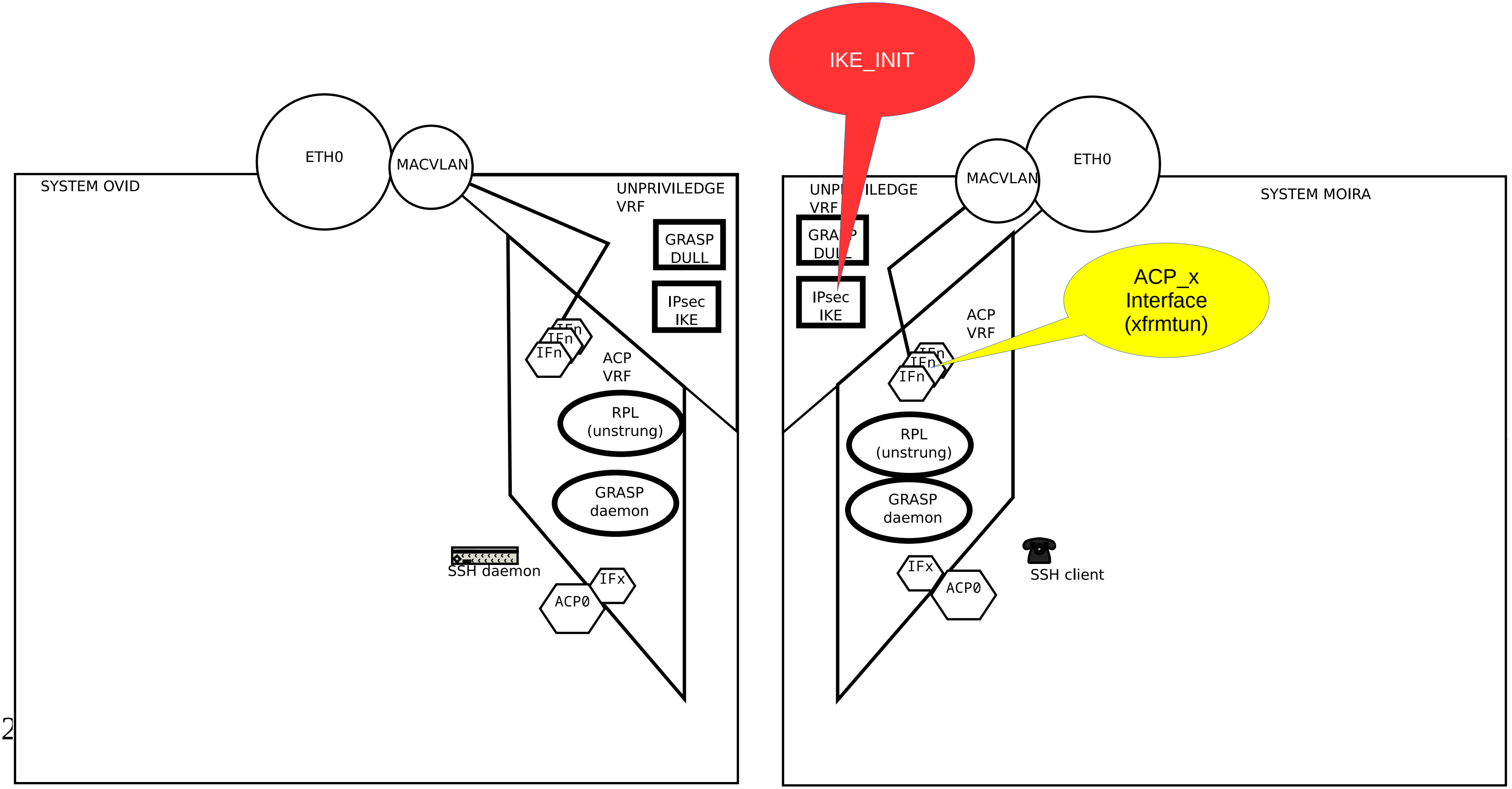
Architecture Diagram - 3



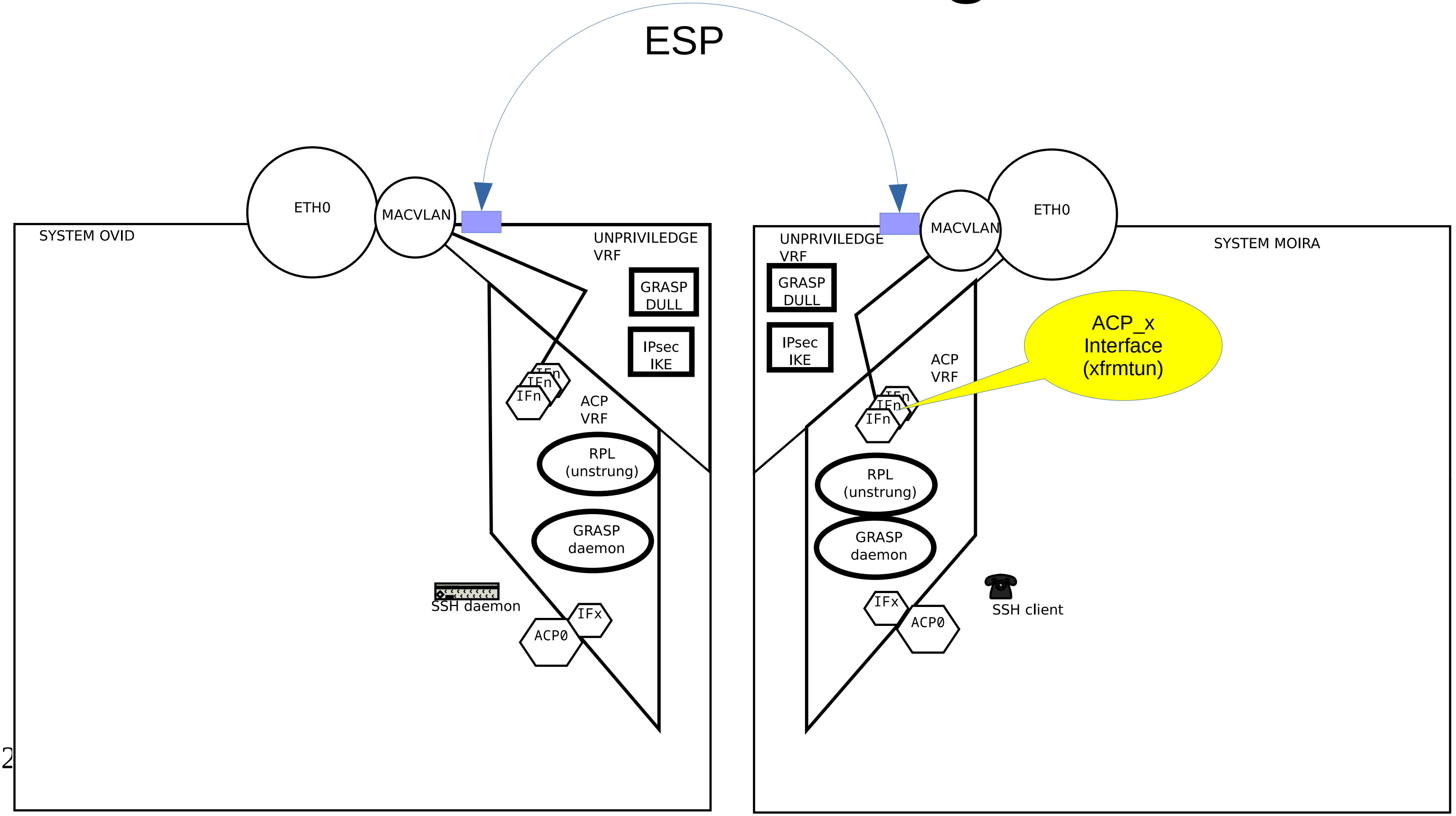
Architecture Diagram - 3



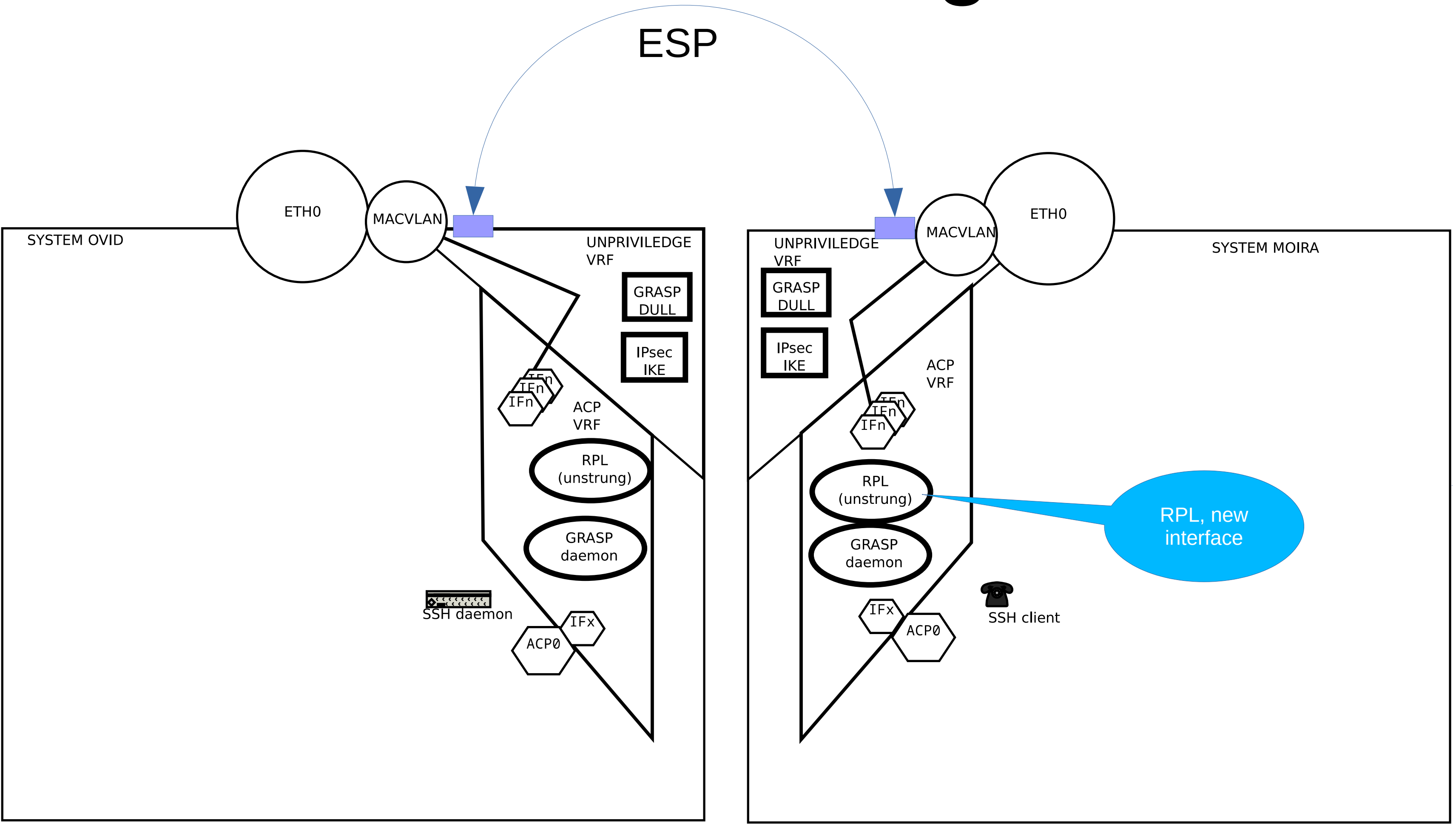
Architecture Diagram - 3



Architecture Diagram - 3

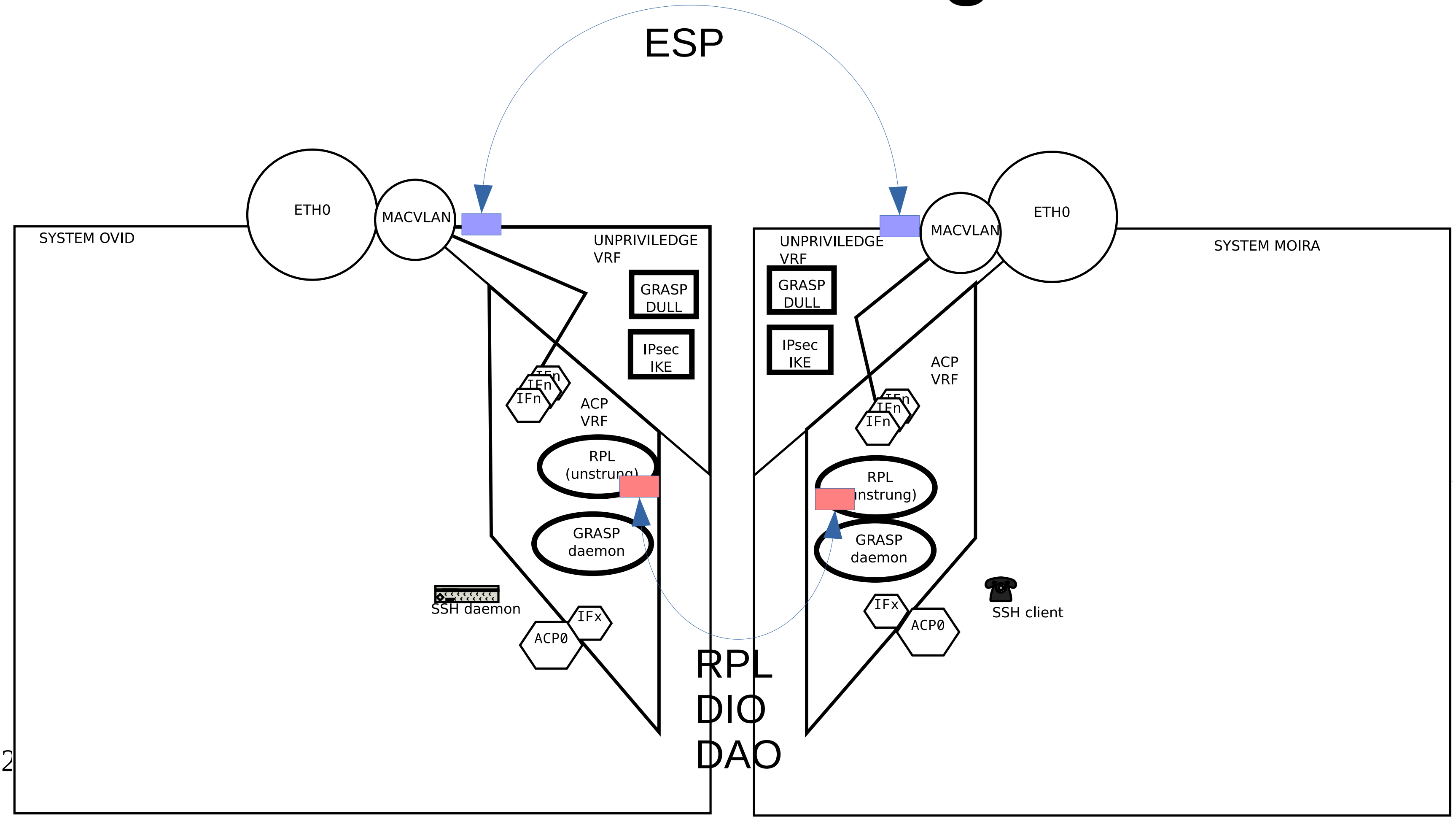


Architecture Diagram - 3



RPL, new interface

Architecture Diagram - 3



Challenges

- `Connect` written in rust
  - 5500 lines with some unit tests
  - Started in fall 2020
- `BlueroseSwan` written in C
  - History going back to 1997
    - (my history starts in 2001)
- `Unstrung` (RPL) written in C++
  - 11K lines, including tests
  - Started in 2009, gap from 2016 to 2021
- I’m uniquely steeped in these three technologies. Hah.

- Each daemon deals with lists of network interfaces
  - Using Netlink socket
  - Connect in three different namespaces
- Connect creates interfaces and moves them around network namespaces that it manages
- Bugs in systemd-login that makes it kill ssh if a namespace gets abandonned
  - Don’t use systemd for now.
- Linux IPsec turns out not to allow IPv6 scope-id to be set for the IPsec ESP SA
  - Discovered in fall 2022 after ruling out other annoying issues involving IPsec eating ICMP ND messages
  - Was obvious in hindsight

Would be nice

- Rewrite IKEv2 in RUST
- minimal non-configuration policy
- maybe some library for fundamental operations

- maybe merge IKEv2 code into connect
  - avoids much book-keeping
  - listening to netlink socket for interfaces up/down
    - five times!

ESP vs IPv6 Link Local

- while screwing around with **VTI** had many problems with ::0/0 <-> ::0/0 capturing ICMPv6 ND messages
- switched VTI to xfrmtun
  - now has third argument **link** argument
- did not notice at first that ESP SA has no scope_id (link_id) for connection
  - works with one interface, or randomly
- spoke about this at previous workshops

- needs new netlink code
  - remember discussion a few workshops ago about enumerating extensions
- netlink/IKE part is “easy”
- navigating the dst_lookup and friends to get the details down seems to elude me
  - looking for help to make this work

ULA numbering

- Each node makes up it’s own ULA
- each “physical” interface gets a /64 from the /48
- (SLAAC) used to generate a /128 from each interface
- the /128 goes as an alias on abutment **lo**
  - GRASP **M_FLOOD** announce tweaked
  - add /128 route to peer

- in fall 2023, was encouraged to use ULA
- but it’s an bits-on-the-wire change to RFC8994
- fixing still important
  - nice if it works on current *WRT, BMC kernels so...

Hacking around with ULAs

Was v6-LL
Now can be
ULA

```
M_FLOOD
        let ike_locator = grasp::GraspLocator::O_IPv6_LOCATOR { v6addr: myv6,
                                                              transport_proto: IPPROTO_UDP,
                                                              port_number: 500 };
        let acp_objective =grasp::GraspObjective { objective_name: "AN_ACP".to_string(),
                                                        objective_flags: grasp::F_SYNC,
                                                        loop_count: 1,  /* do not leave link */
                                                        objective_value:
Some("IKEv2".to_string()),
                                                        locator: Some(ike_locator) };
        let flood = grasp::GraspMessage { mtype: GraspMessageType::M_FLOOD,
                                          session_id: sesid,
                                          initiator: myllv6,
                                          ttl: 10000,
                                          objectives: vec![acp_objective] };
```

Remains
IPv6-LL

Hacking around with ULAs - 2

```
./dull ip –6 route ls

fdcc:aeab:2346:e:50ab:93ff:fee8:8dd4(/128)
via fe80::50ab:93ff:fee8:8dd4 dev
dull014 proto static metric 1024 pref
medium

fdcc:aeae:1234:24:9041:4eff:fe17:3e6b
via fe80::9041:4eff:fe17:3e6b dev
dull014 proto static metric 1024 pref
medium
```

More Challenges

- Macvlan does not mix with bridges (same internal hooks)
  - So connect creates ethernet pairs, and adds them to the bridge, if it finds a bridge.

```
hermes-[~] mcr 10016 %brctl show
bridge name     bridge id               STP enabled
interfaces
trusted         8000.52540051dafb       no              eth0

pull014
hermes-[~] mcr 10017 %./dull ifconfig
dull014: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu
        inet6 fe80::44c7:b2ff:fea2:6bbc  prefixlen 64
        inet6 fdc2:ae5d:4f12:23:44c7:b2ff:fea2:6bbc
       ether 46:c7:b2:a2:6b:bc  txqueuelen 1000  (Ethernet)
```

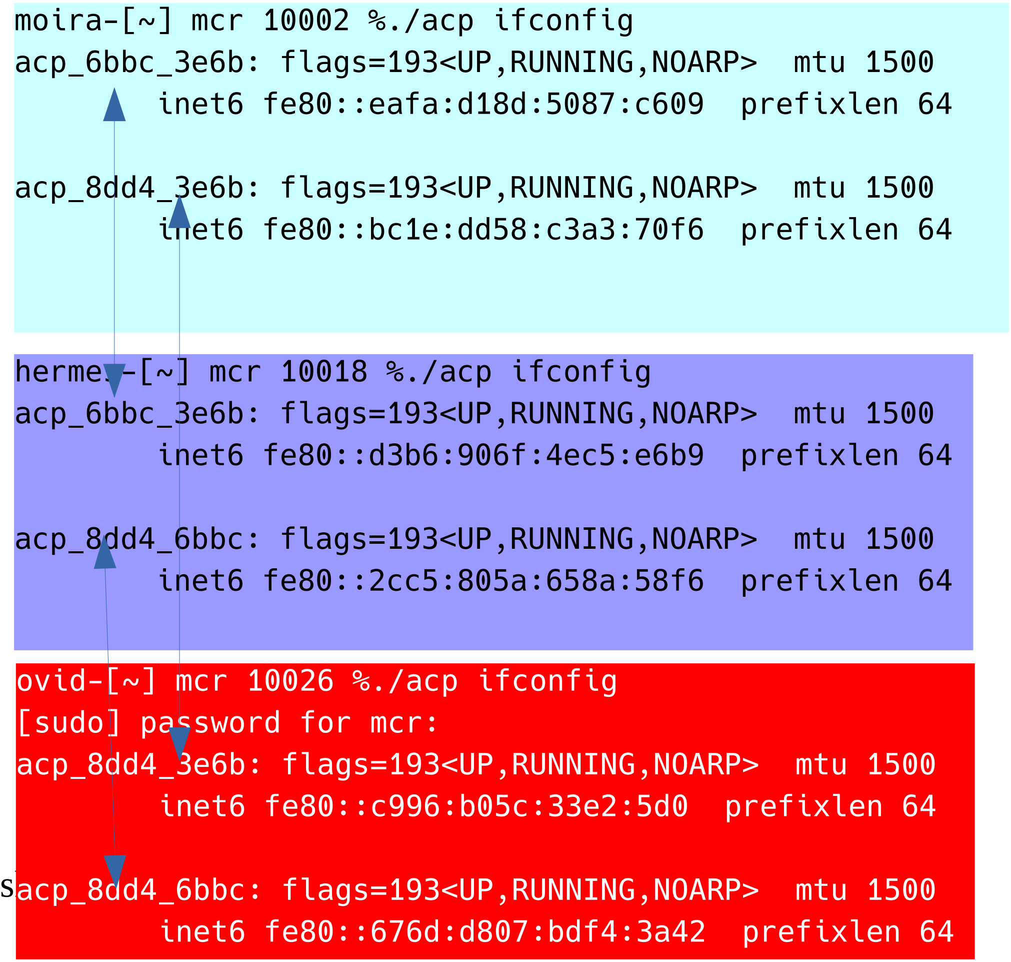
- Ethernet pairs have randomly assigned layer-2 addresses
- So have random IIDs for Ipv6-LL.
- ULA is reusing the IID too!

Every run has new values, which makes debugging annoying.

```
moira-[~] mcr 10038 %./dull ping6 fe80::44c7:b2ff:fea2:6bbc%dull013
PING fe80::44c7:b2ff:fea2:6bbc%dull013(fe80::44c7:b2ff:fea2:6bbc%dull013) 56 data bytes
64 bytes from fe80::44c7:b2ff:fea2:6bbc%dull013: icmp_seq=1 ttl=64 time=5.36 ms
64 bytes from fe80::44c7:b2ff:fea2:6bbc%dull013: icmp_seq=2 ttl=64 time=5.64 ms
64 bytes from fe80::44c7:b2ff:fea2:6bbc%dull013: icmp_seq=3 ttl=64 time=5.61 ms
```

Inside ESP debugging

- To test ACP interface to ACP interface, can use ping6 LL with interface.
- Wound up naming ACP interfaces for two ends of v6-LL outside (abutment) interface



```
moira-[~] mcr 10002 %./acp ifconfig
acp_6bbc_3e6b: flags=193<UP,RUNNING,NOARP>  mtu 1500
        inet6 fe80::eafa:d18d:5087:c609  prefixlen 64

acp_8dd4_3e6b: flags=193<UP,RUNNING,NOARP>  mtu 1500
        inet6 fe80::bc1e:dd58:c3a3:70f6  prefixlen 64
```

```
hermes-[~] mcr 10018 %./acp ifconfig
acp_6bbc_3e6b: flags=193<UP,RUNNING,NOARP>  mtu 1500
        inet6 fe80::d3b6:906f:4ec5:e6b9  prefixlen 64

acp_8dd4_6bbc: flags=193<UP,RUNNING,NOARP>  mtu 1500
        inet6 fe80::2cc5:805a:658a:58f6  prefixlen 64
```

```
ovid-[~] mcr 10026 %./acp ifconfig
[sudo] password for mcr:
acp_8dd4_3e6b: flags=193<UP,RUNNING,NOARP>  mtu 1500
        inet6 fe80::c996:b05c:33e2:5d0  prefixlen 64

acp_8dd4_6bbc: flags=193<UP,RUNNING,NOARP>  mtu 1500
        inet6 fe80::676d:d807:bdf4:3a42  prefixlen 64
```

```
23:48:51.211012 IP6 fdcc:aeae:1234:24:9041:4eff:fe17:3e6b >
       fdcc:aeab:2346:e:50ab:93ff:fee8:8dd4: ESP(spi=0x40f1ad64,seq=0x65)
23:49:22.571054 IP6 fdc2:ae5d:4f12:23:44c7:b2ff:fea2:6bbc >
       fdcc:aeae:1234:24:9041:4eff:fe17:3e6b: ESP(spi=0xef984590,seq=0x73e)
```

rks

IKE degenerate policy

```
000 "c_77e5_15b1": ::/0===fdcc:aeae:1234:10:5041:60ff:fe58:77e5
                                        [E=rfcSELF+fd739fc23c3440112233445500000100+@acp.example.com]...
                                        fd9e:4189:b0f:13:a4a0:58ff:fef0:15b1[E=*]===::/0; erouted; eroute owner:
#2
000 "c_77e5_15b1":     myip=unset; hisip=unset; myup=/bin/true; hisup=/bin/true;
mycert=hostcert.pem;
000 "c_77e5_15b1":   CAs: 'DC=ca, DC=sandelman, CN=fountain-test.example.com Unstrung
Fountain Root CA'...'%any'
000 "c_77e5_15b1":   policy: RS... ; kind=CK_TEMPLATE
000 "c_77e5_15b1":   newest ISAKMP SA: #1; newest IPsec SA: #2; eroute owner: #2;
```

```
000 "c_77e5_15b1": ::/0===fd9e:4189:b0f:13:a4a0:58ff:fef0:15b1
                      [E=rfcSELF+fd739fc23c3440112233445500000200+@acp.example.com]...
                      fdcc:aeae:1234:10:5041:60ff:fe58:77e5[E=*]===::/0; erouted; eroute
owner: #2
000 "c_77e5_15b1":     myip=unset; hisip=unset; myup=/bin/true; hisup=/bin/true;
mycert=hostcert.pem;
000 "c_77e5_15b1":   CAs: 'DC=ca, DC=sandelman, CN=fountain-test.example.com Unstrung Fountain
Root CA'...'%any'
0
000 "c_77e5_15b1":   policy: RSASIG+...; kind=CK_TEMPLATE
000 "c_77e5_15b1":   newest ISAKMP SA: #1; newest IPsec SA: #2; eroute owner: #2;
```

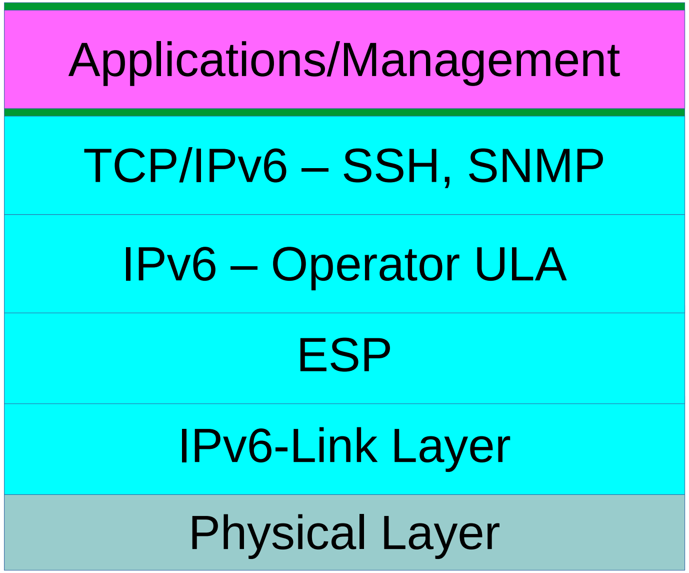
Upcoming challenges

- want no (SPD) policies for xfrmtun
  - net.ipv6.conf.acp_6bbc_3e6b.disable_policy
- tunkey used to link to SA by reqid
- rekeying might have bugs in kernel.. unclear

- simultaneous key/rekey implemented, but wildcard policy can confuse

- full (n^2) mesh is a problem
- connect Dead Peer to routing daemon

ACP: Architecture



Applications/Management

TCP/IPv6 – SSH, SNMP

IPv6 – Operator ULA

ESP

IPv6-Link Layer

Physical Layer

- Laser would ideally stay on even when port is administratively “down”
- Each port of switch would have its own interface logical interface, even if switch is really L2 only
- ESP is hop-by-hop, ideally **L2** hop-by-hop.
- Overlay creates “full” mesh across network
- Authentication is all PKIX certificates, from a common (private) CA
  - authorization is private CA == good
- “IP over Transport Mode”, but really it’s IP ::/0<--> IP ::/0 over ESP tunnel mode.
- IP addresses are not strongly filtered