RFC8994 – Autonomic Control Plane

Implementation Report

IETF120 – Vancouver – July 2024

Michael Richardson
<mcr+ietf@sandelman.ca>

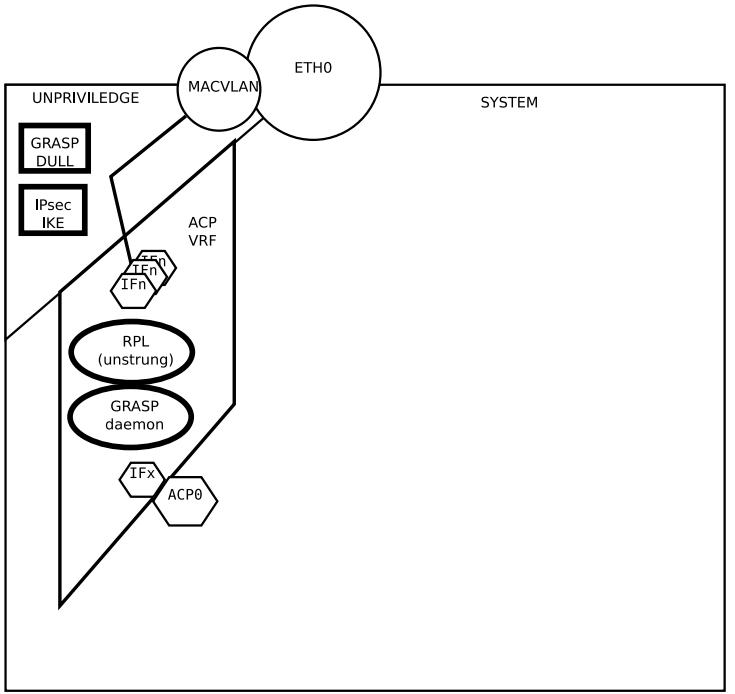
https://github.com/ANIMAgus-minerva/
https://minerva.sandelman.ca/

Cast List

- RFC8995 components
  - Minerva Highway (MASA)
  - Minerva Fountain (Registrar)
  - Minerva Reach (test pledge)
  - Minerva RustyBeach (pledge written in Rust)
    - no_std goal
    - bootstrap
- RFC8994 components
  - **Minerva Connect**
  - Minerva Rooster
  - Unstrung (RPL daemon)
  - Bluerose Openswan (IPsec IKEv2 daemon)

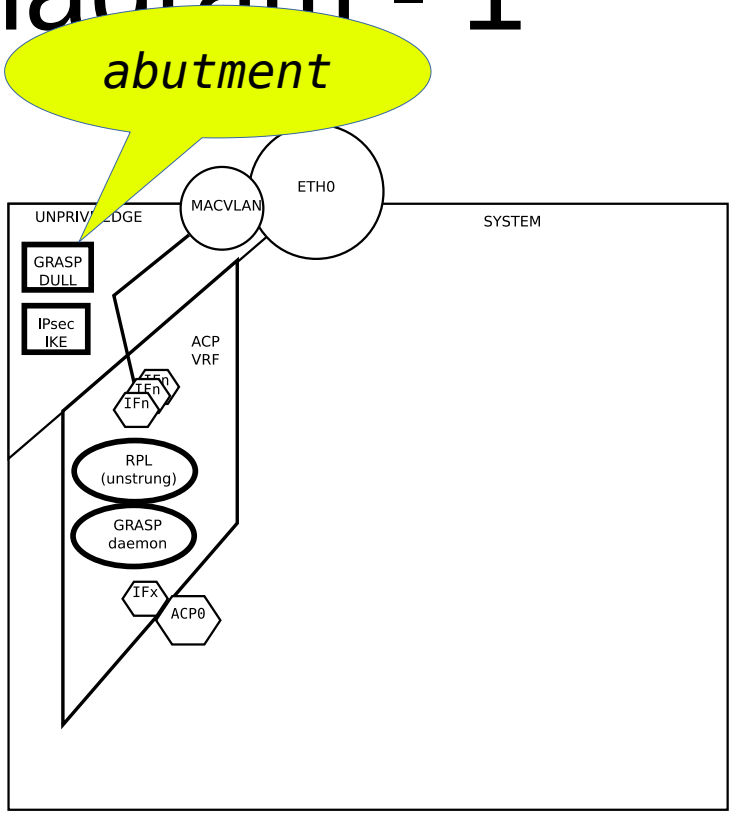
Architectural Diagram - 1

- 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



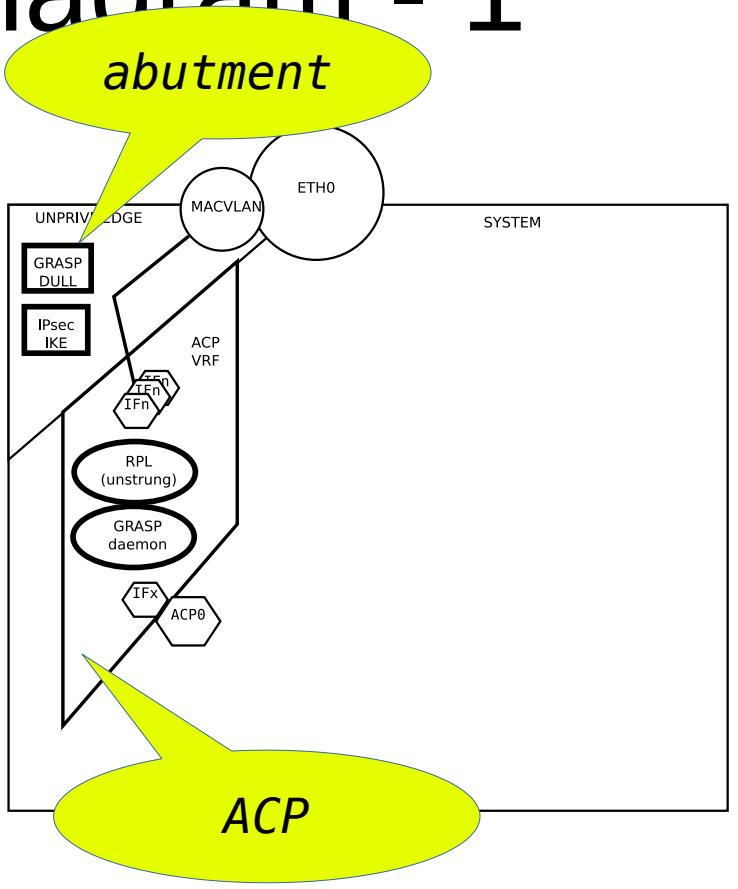
Architectural Diagram - 1

- 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



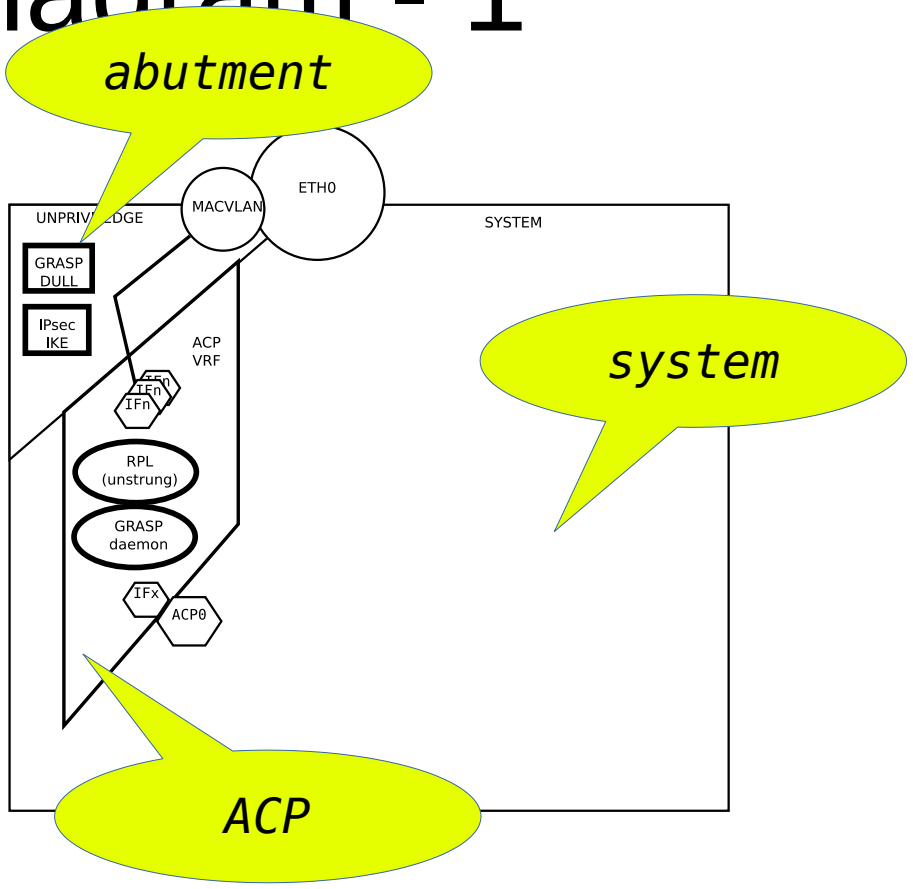
Architectural Diagram - 1

- 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

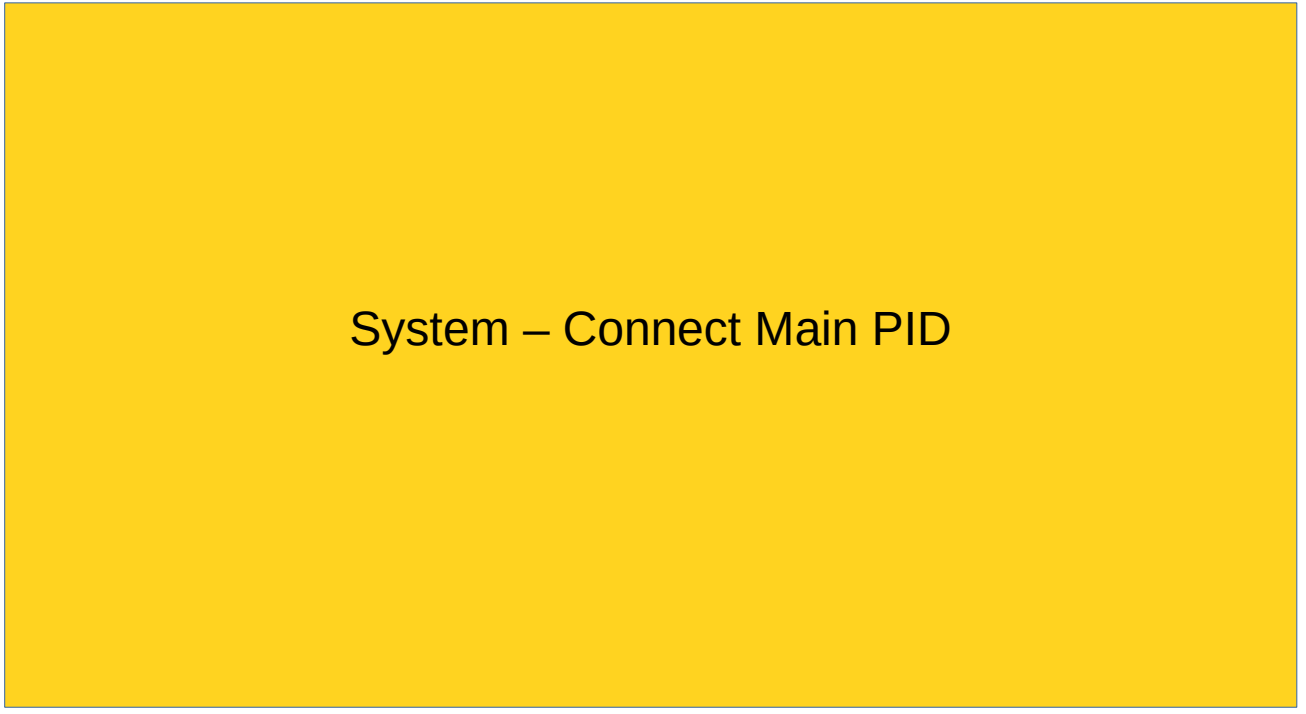


Architectural Diagram - 1

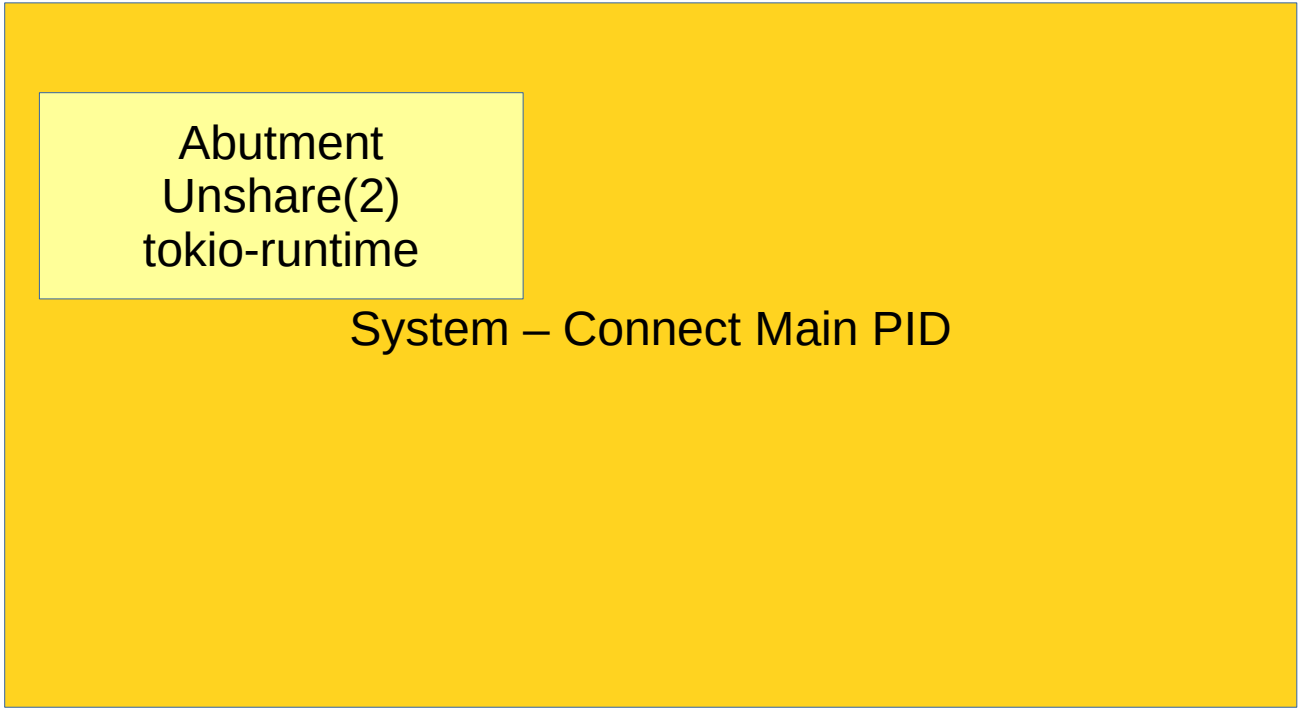
- Uses Linux network namespaces
  - Not a fully isolated container.
  -
- Parent process deals with network interfaces coming/going, and creates virtual interfaces that it pushes into the unpriviledged “dull” space.
  - Calling it the “abutment” space since early 2024.
  - IKE daemon runs in the abutment 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



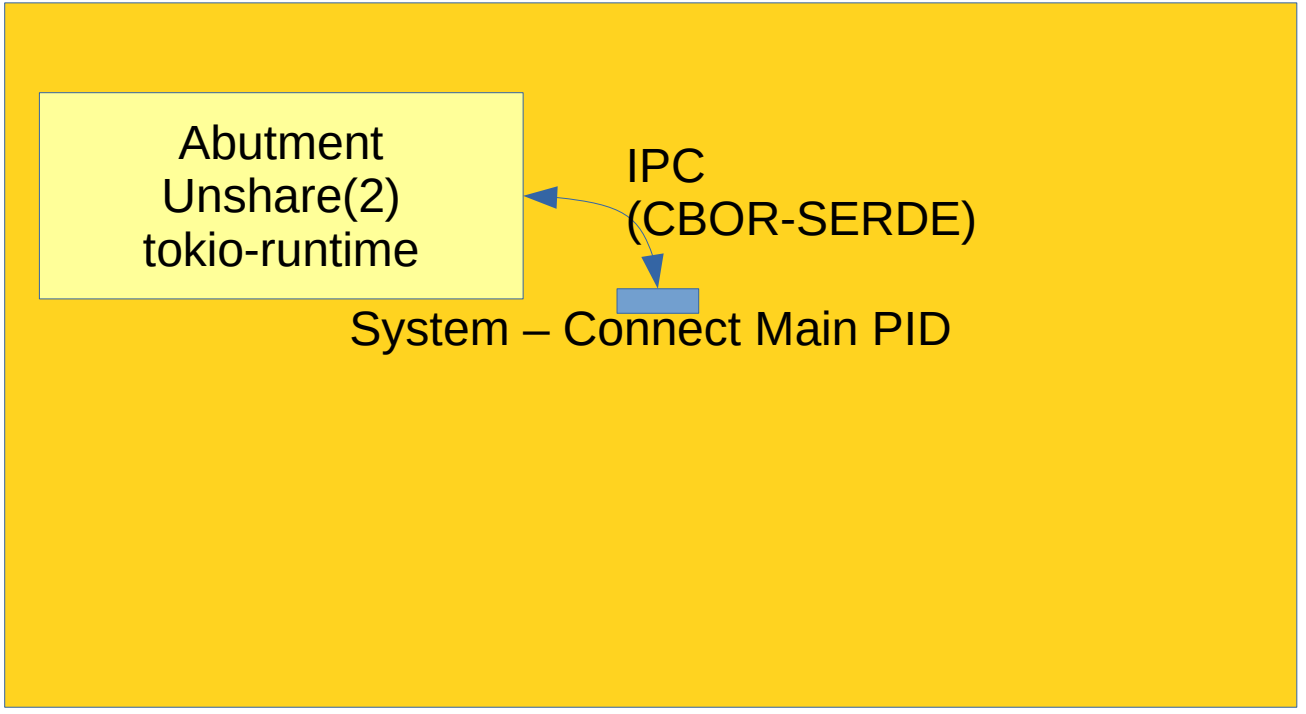
Architecture Diagram - 2



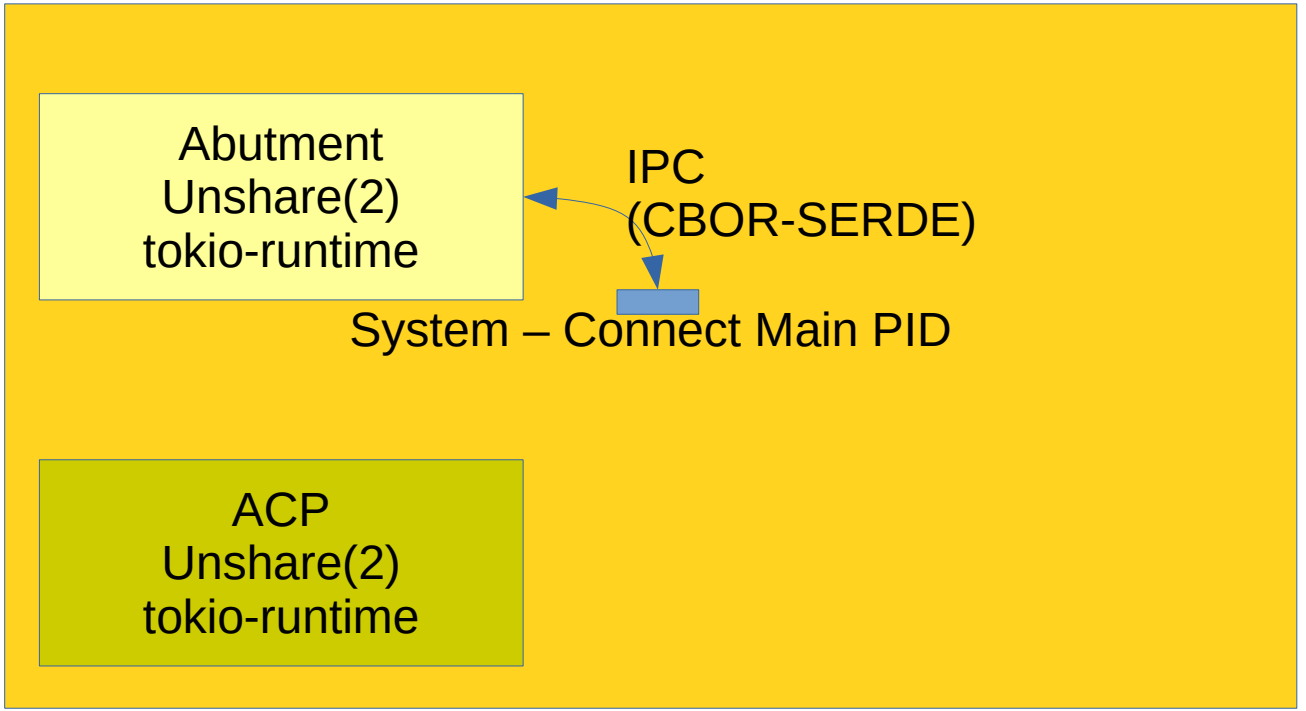
Architecture Diagram - 2



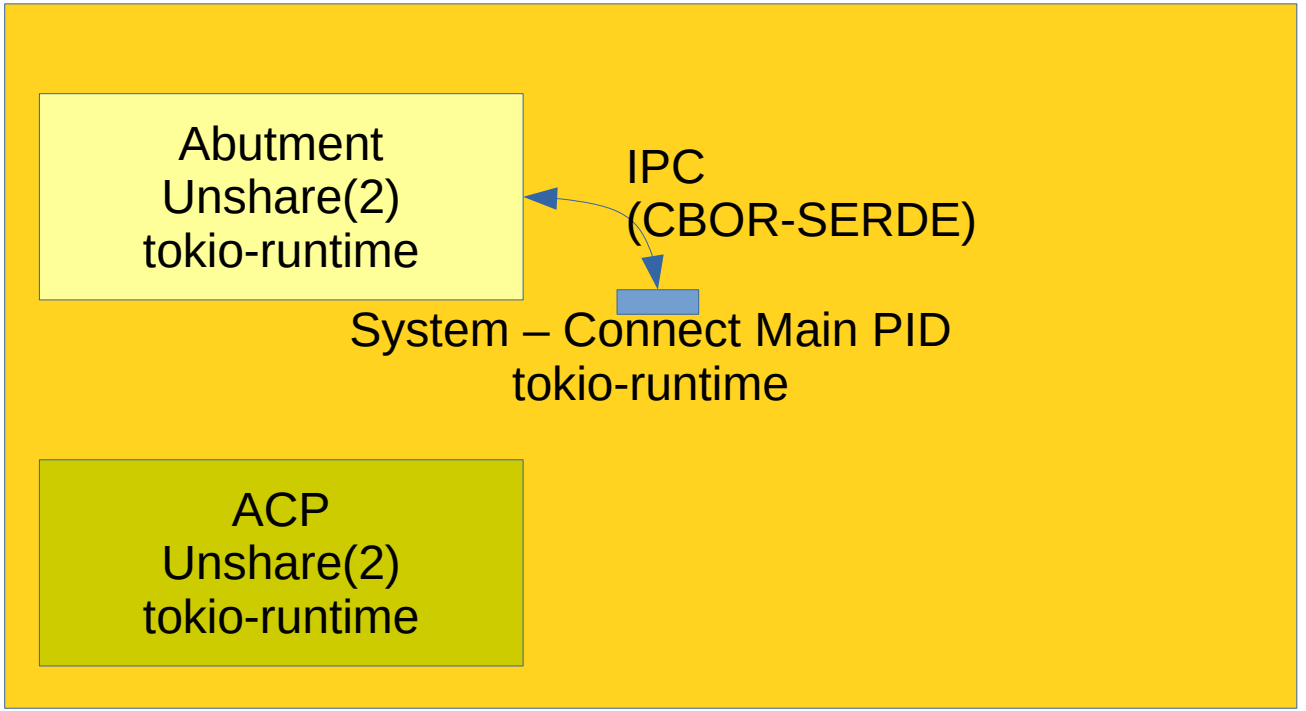
Architecture Diagram - 2



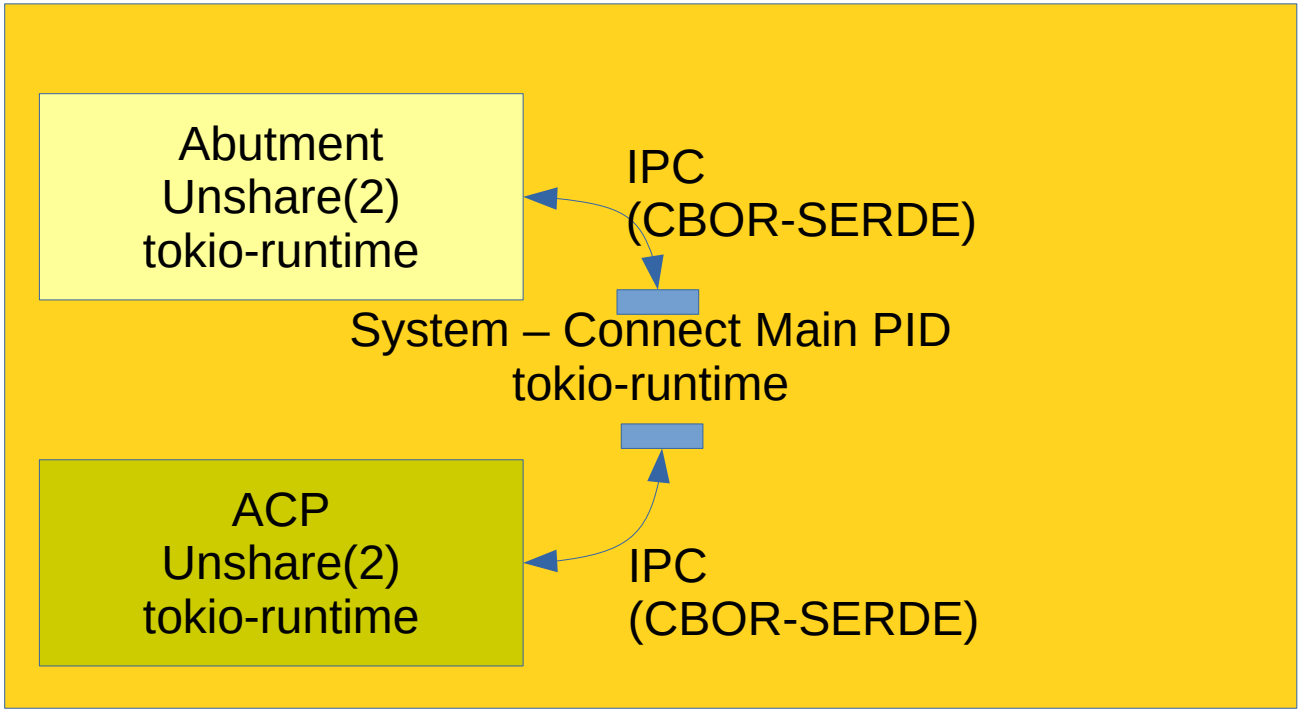
Architecture Diagram - 2



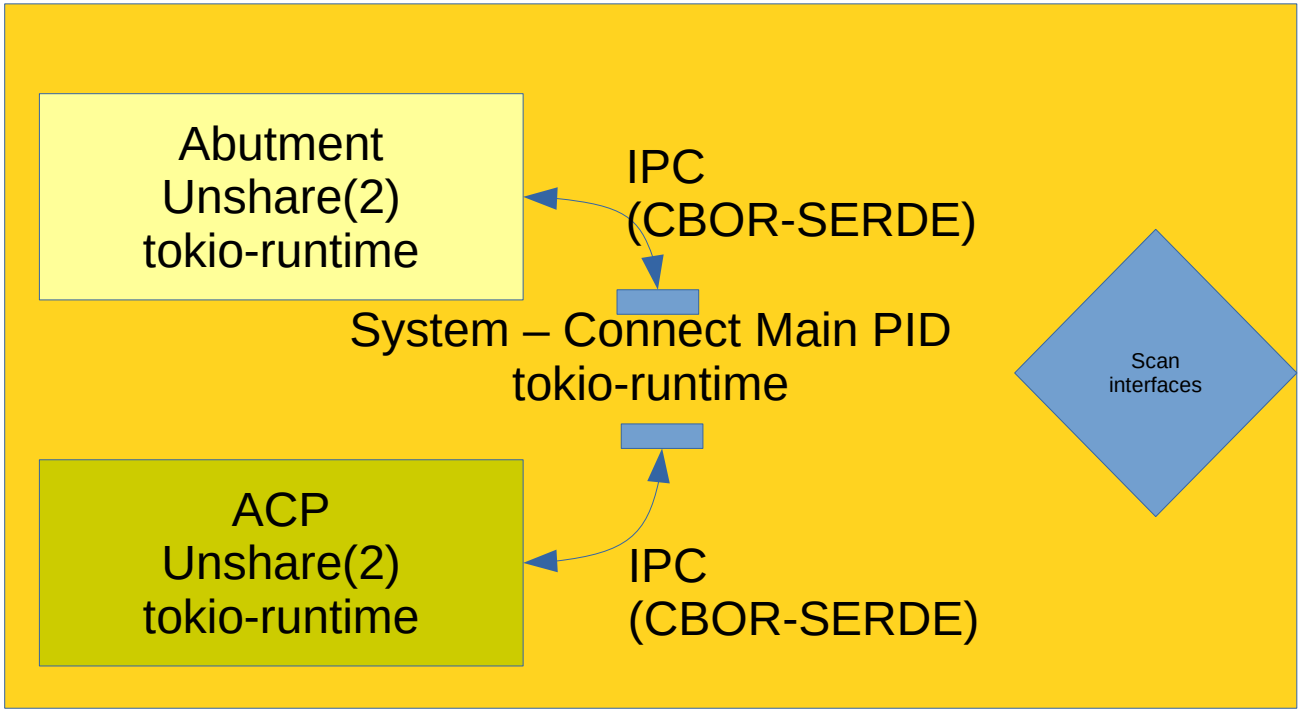
Architecture Diagram - 2



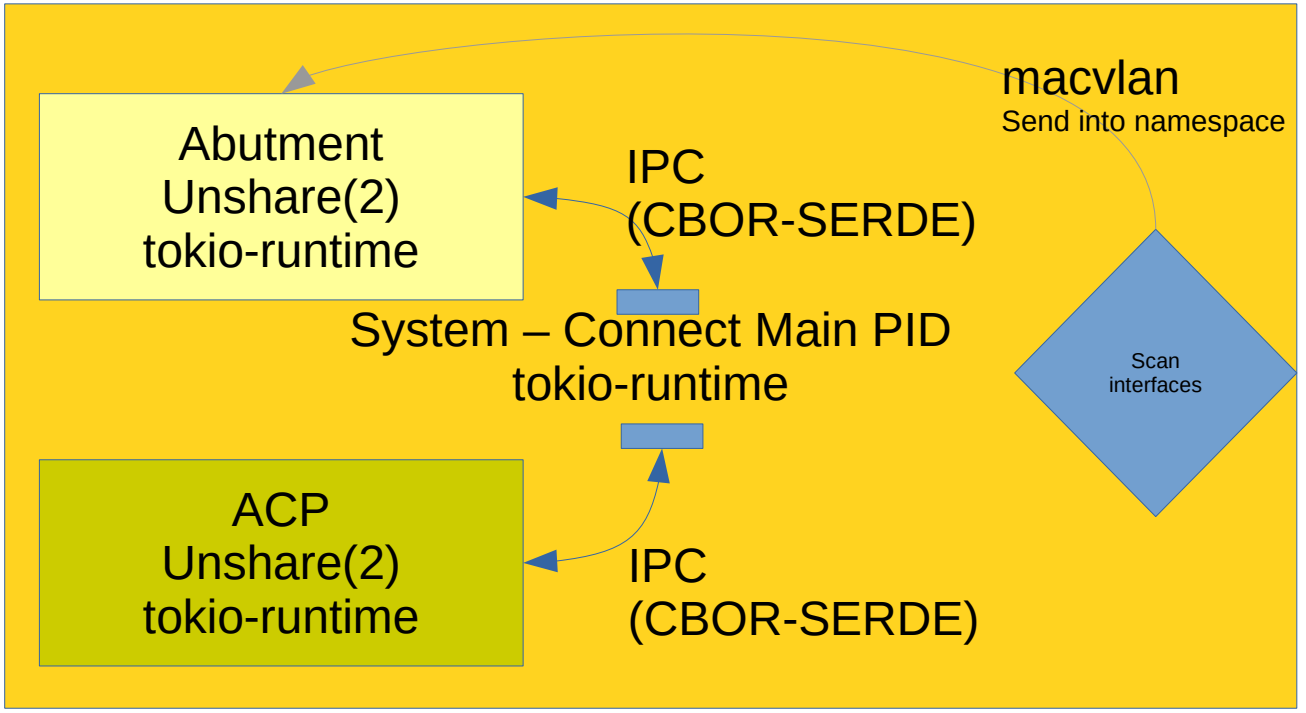
Architecture Diagram - 2



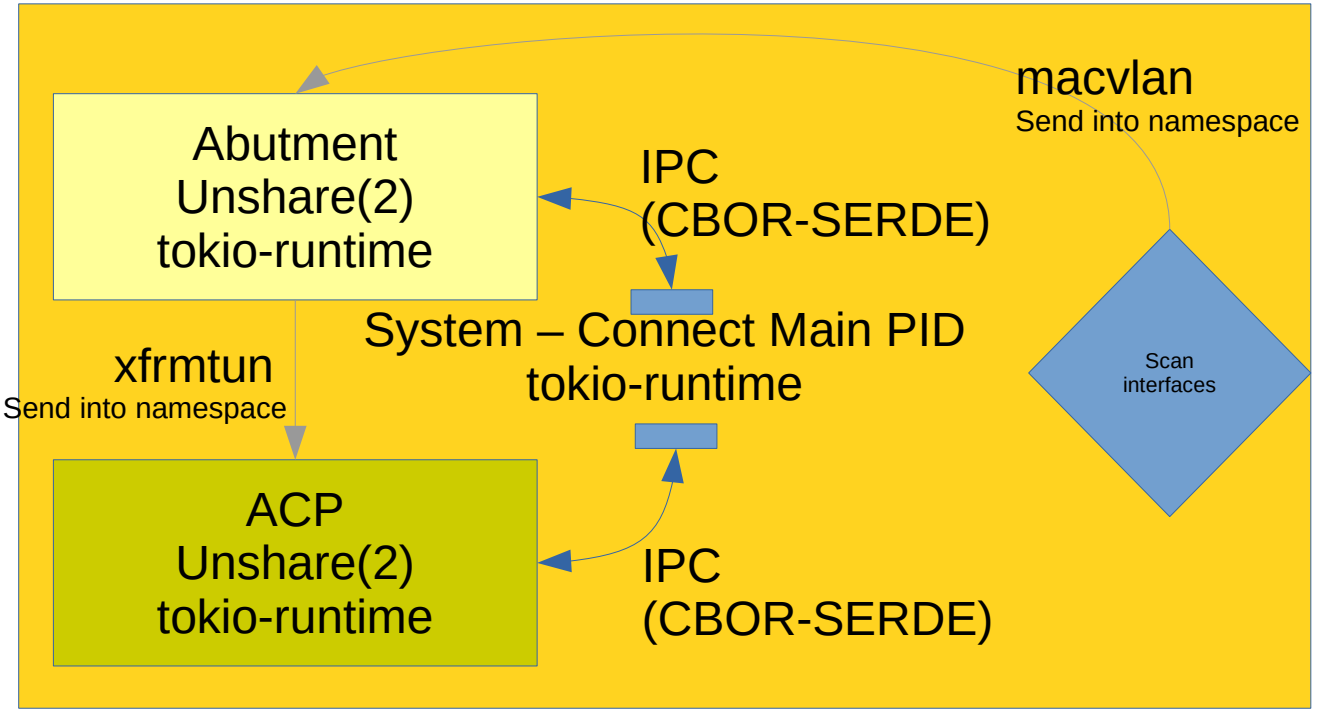
Architecture Diagram - 2



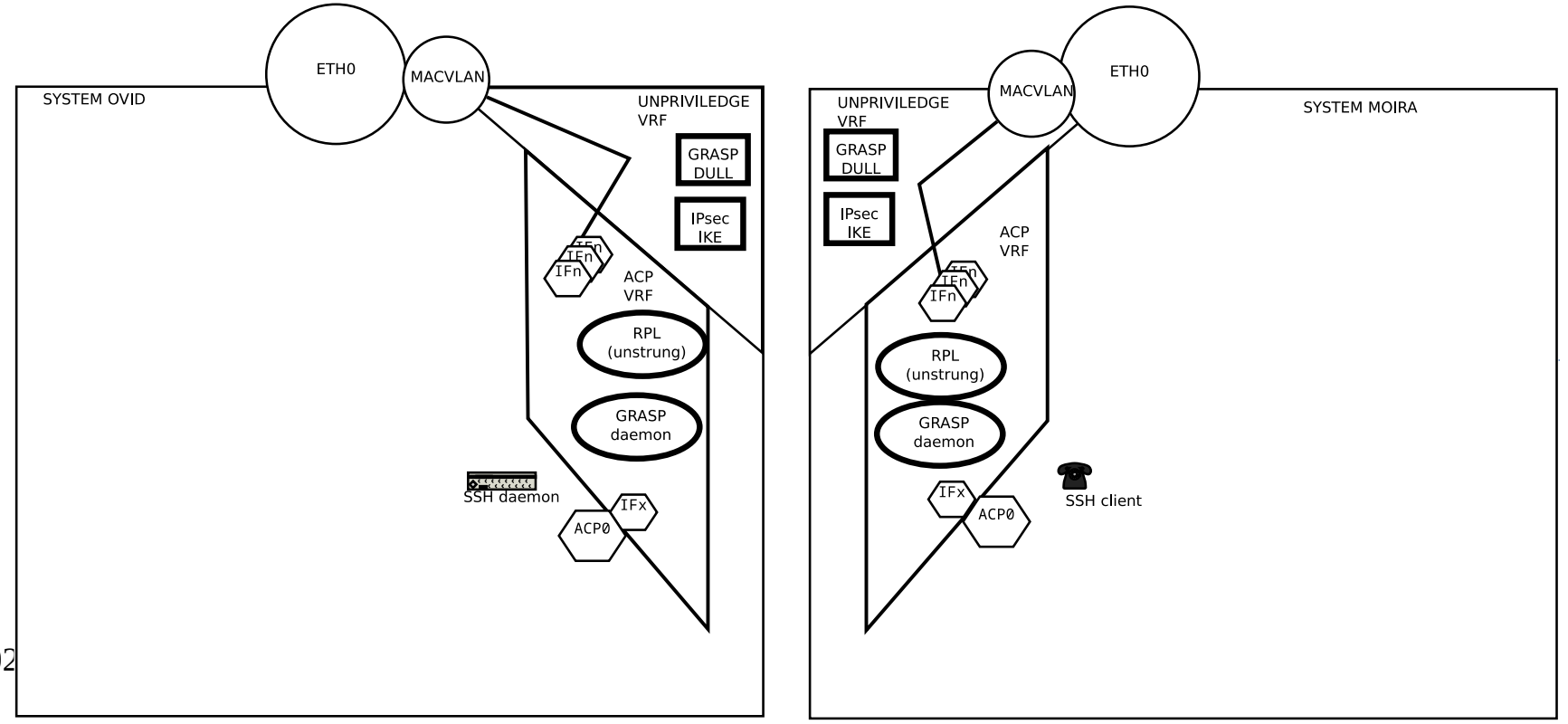
Architecture Diagram - 2



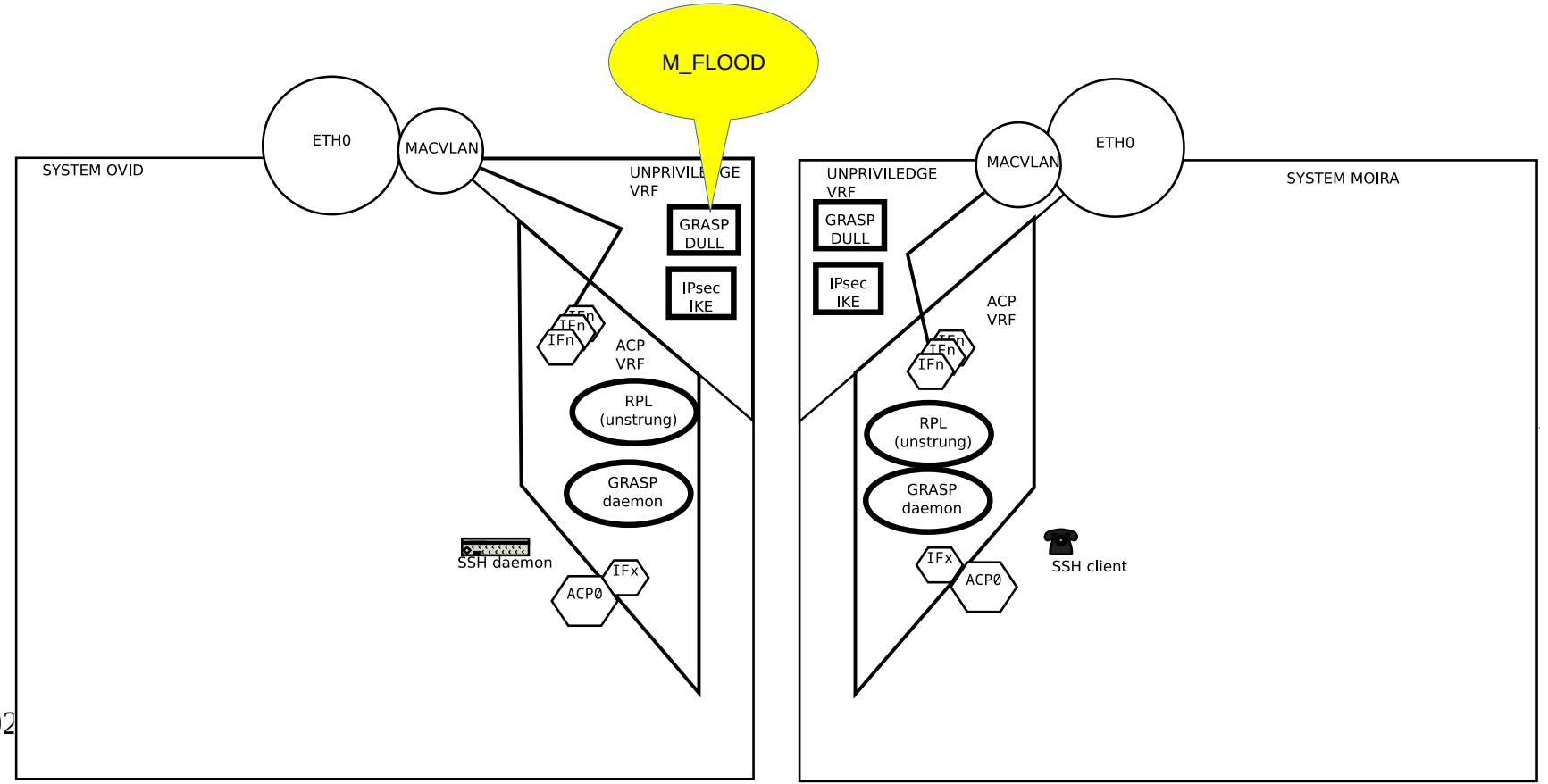
Architecture Diagram - 2



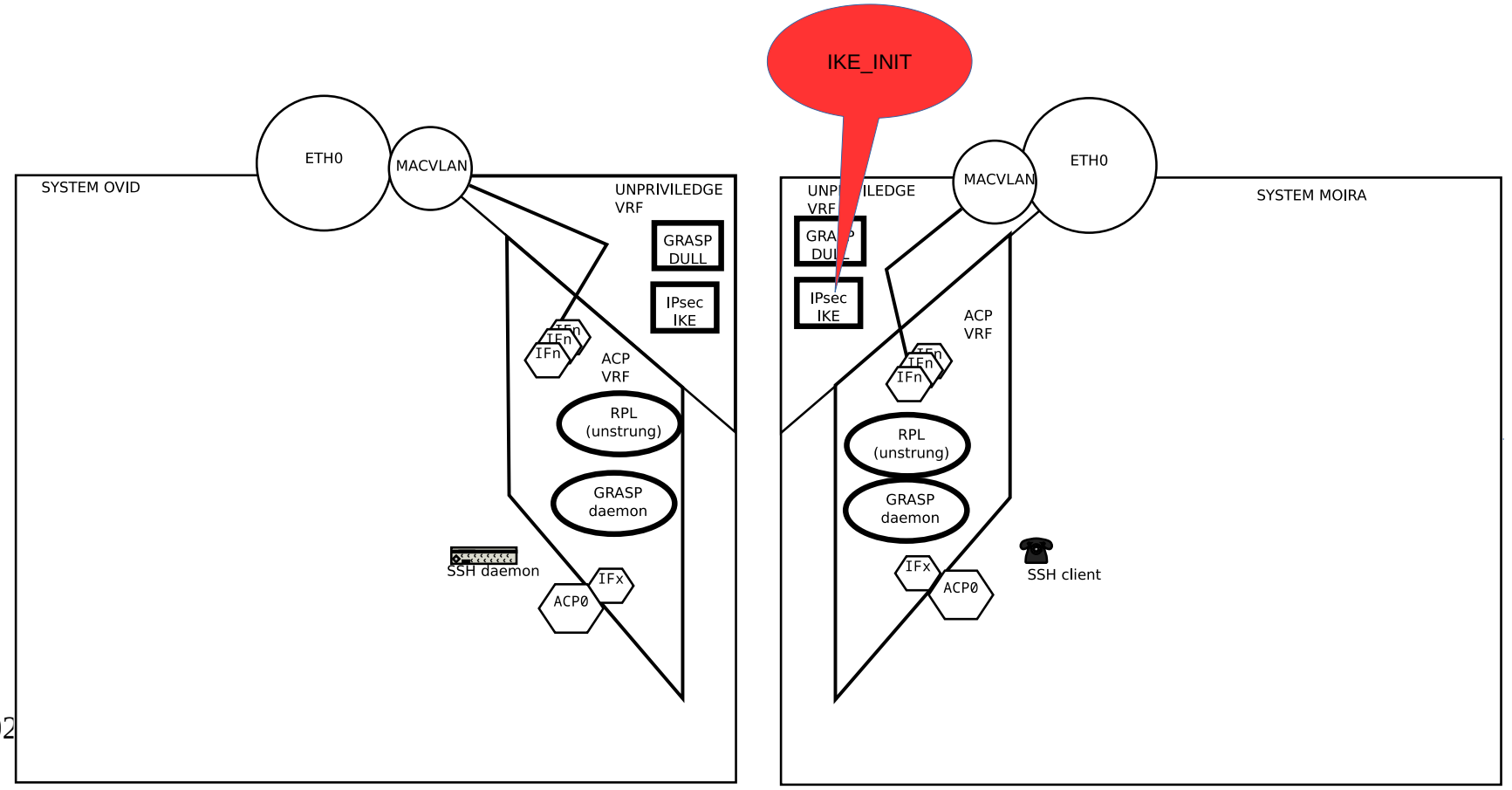
Architecture Diagram - 3



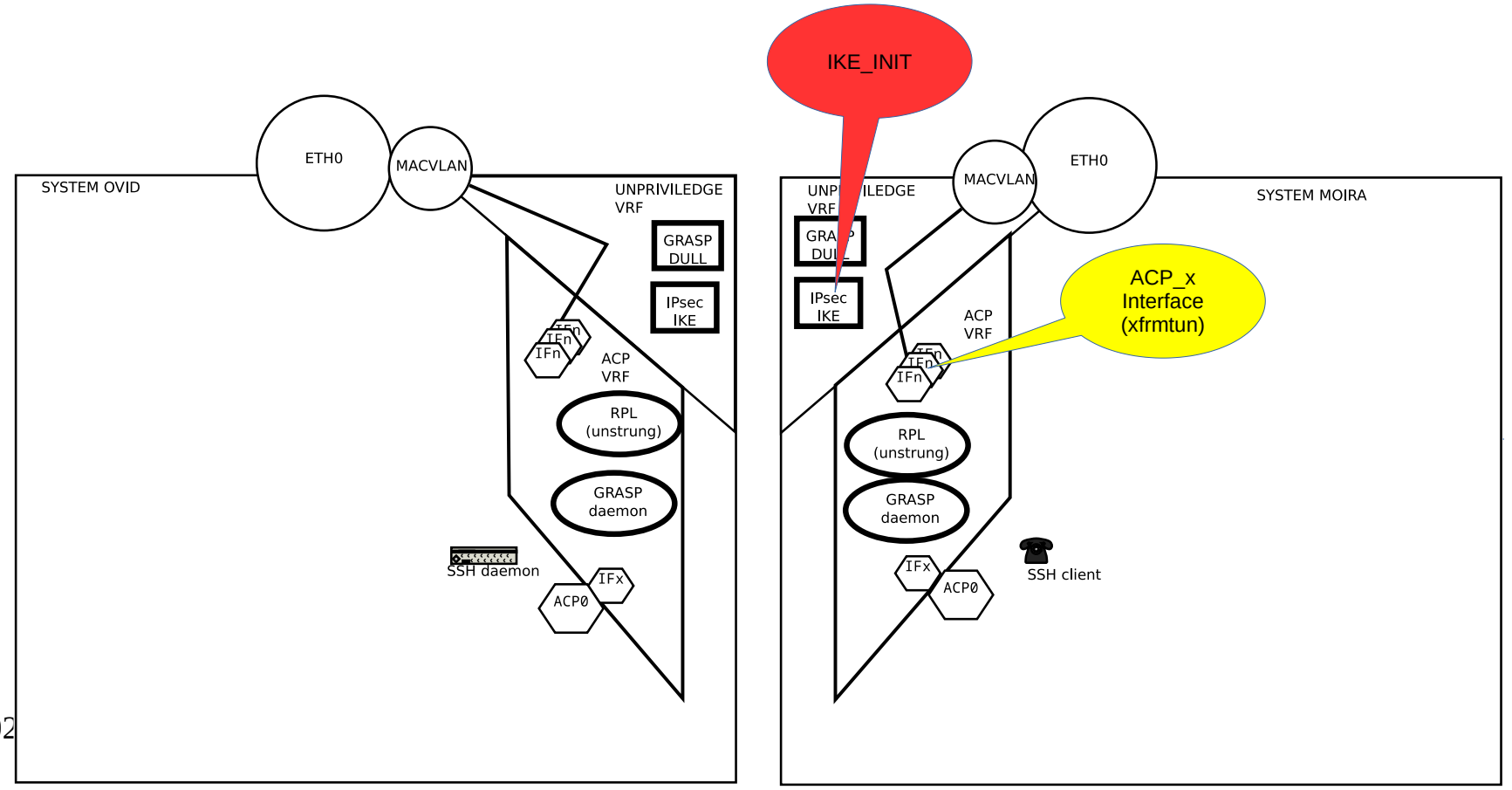
Architecture Diagram - 3



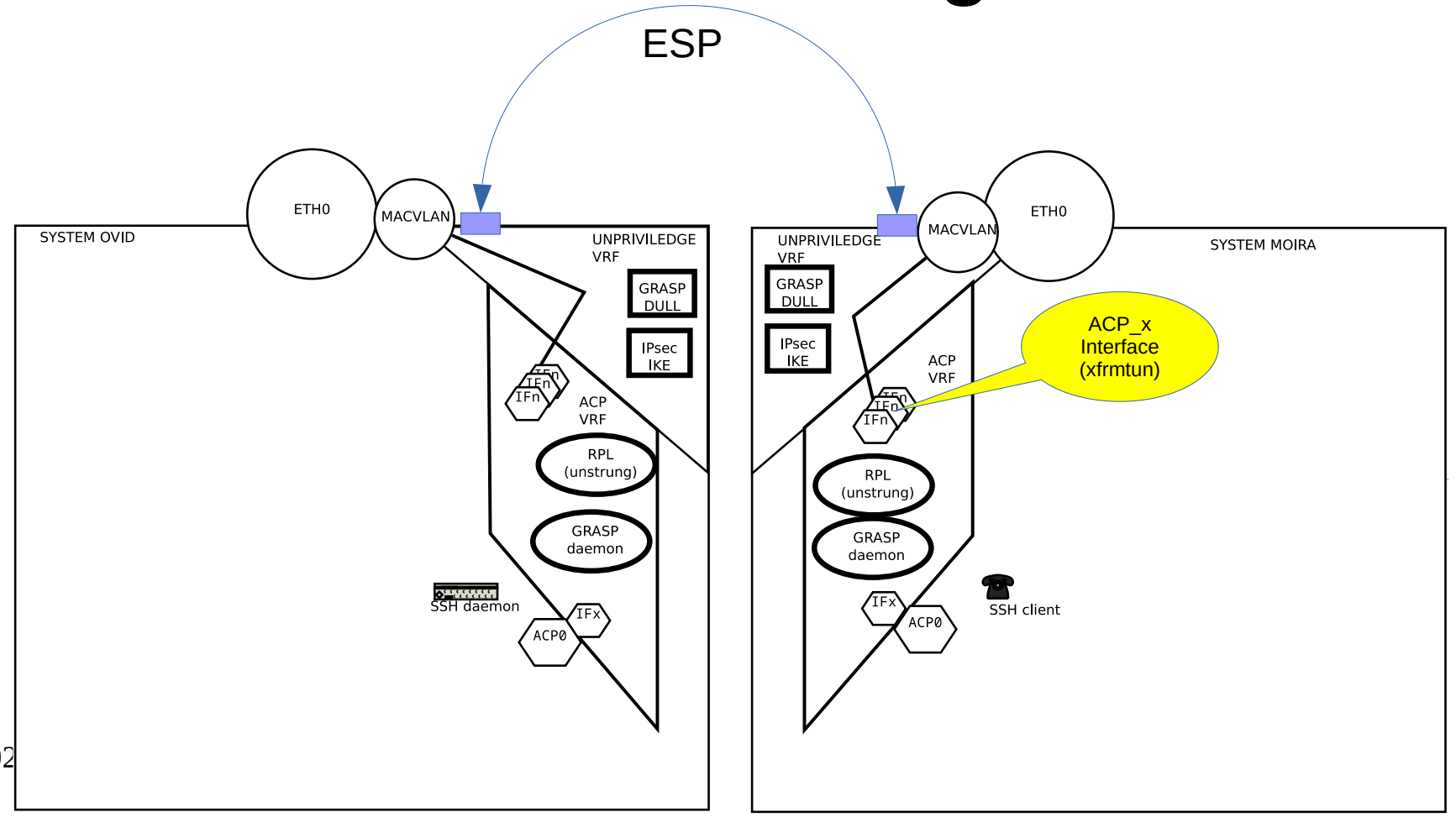
Architecture Diagram - 3



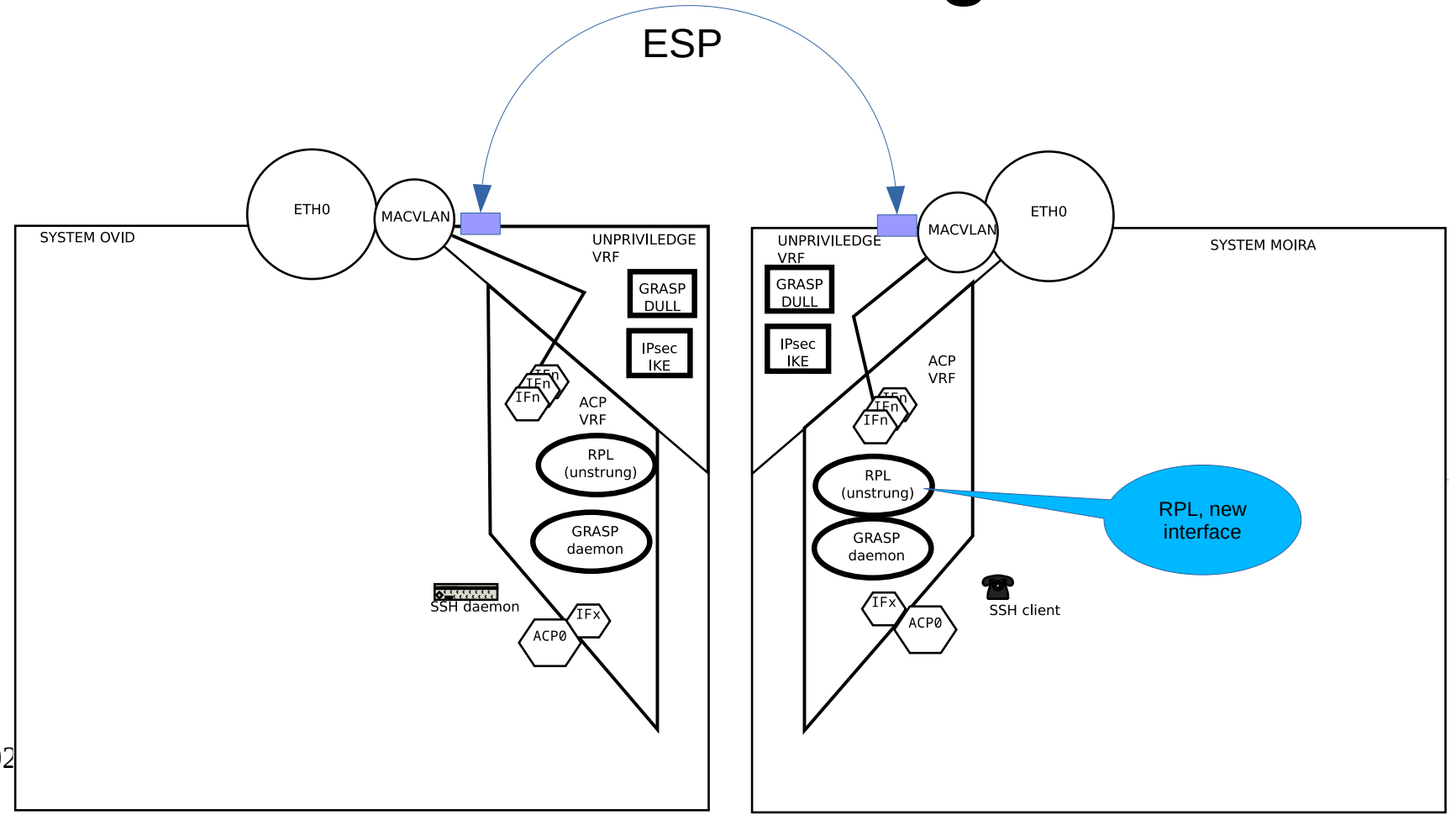
Architecture Diagram - 3



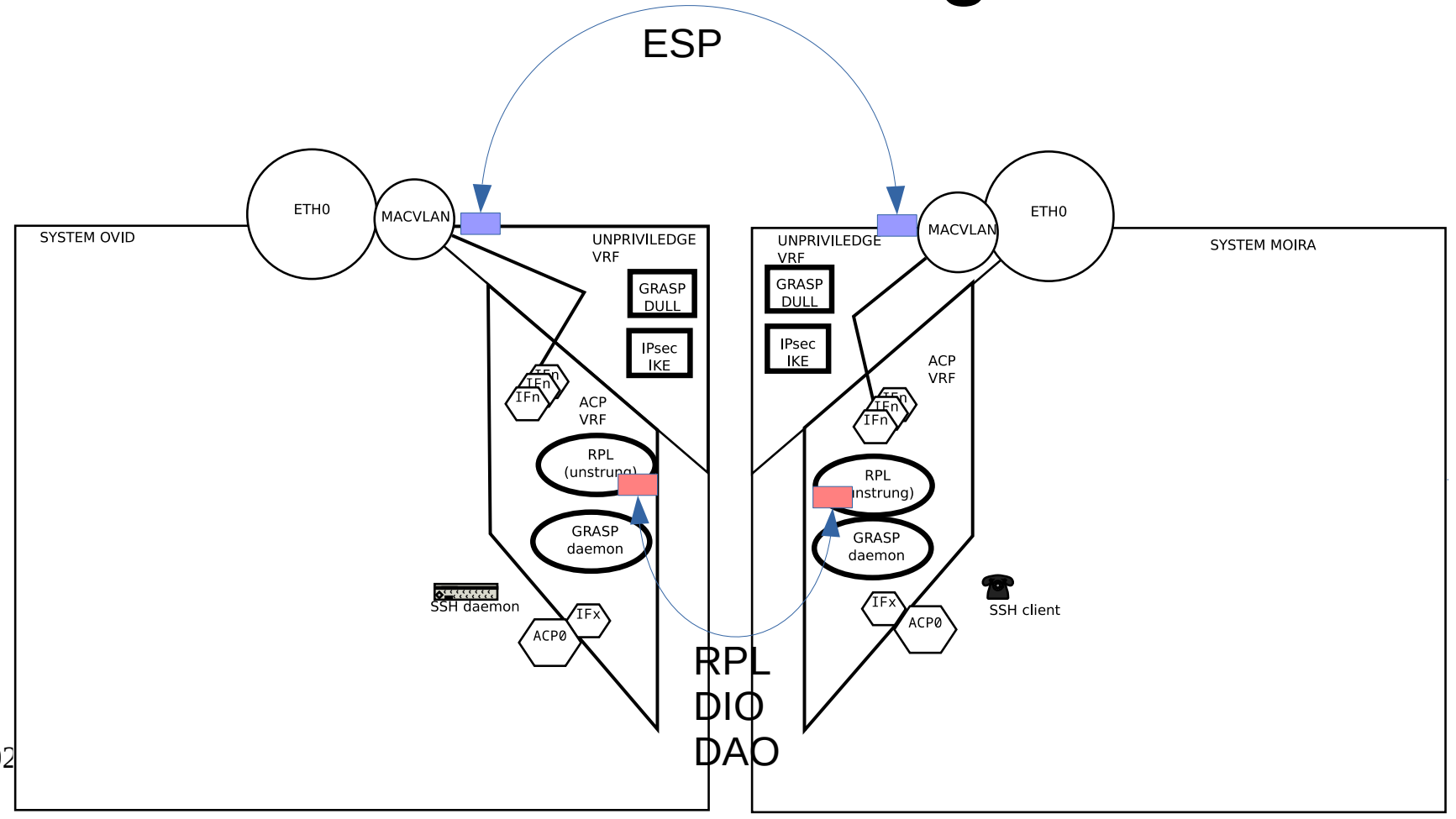
Architecture Diagram - 3



Architecture Diagram - 3



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

Solutions

- Teach ESP about scope-id for IPv6-LL
  - Worth doing, but lots of target systems won’t get new kernels tomorrow
- Number things with ULA and try that
  - Okay, but too much coordination required?
- Each systems numbers its abutment interfaces using a local ULA/48.
  - fdab:1234:3456:ifindex::IID/128

Hacking around with ULAs

```
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] };
```

Was v6-LL
Now can be
ULA

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 150
        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)
```

```
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
```

- 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.

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
```

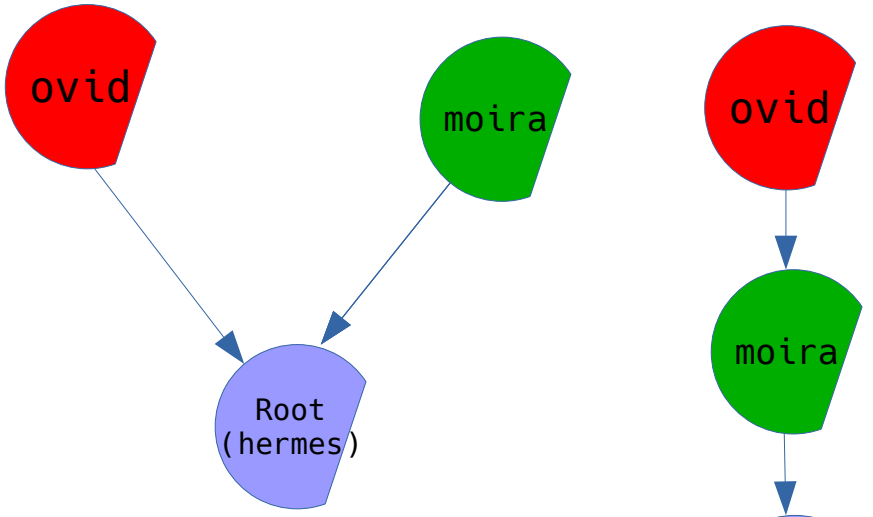
```
ovid-[~] mcr 10026 %./acp ifconfig
[sudo] password for mcr:
acp_8dd4_3e6b: flags=193<UP,RUNNING,NOARP>  mtu 1500
        inet6 fe80::c996:b05c:33e2:5d0  prefixlen 64

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)
```

MA

```
moira-[~] mcr 10006 %./acp ping6  fe80::ca88:90a3:49ff:7fd5%acp_fca9_c740
PING fe80::ca88:90a3:49ff:7fd5%acp_fca9_c740(fe80::ca88:90a3:49ff:7fd5%acp_fca9_c740) 56 data bytes
From fe80::7170:bf9d:fdd8:6b04%acp_fca9_c740 icmp_seq=1 Destination unreachable: Address unreachable
```

RPL layer



Conclusions 1

- ULA numbering is a 2nd version of IPsec.
  - Fundamentally non-interoperable with IPv6-LL version
  - GRASP announcement can announce both, mind you.
  - If we introduce this hack it probably won't go away
- Debugging is really difficult/tidious, we need to include some additional sanity checks
  - Need to get telemetry back (over ACP!)
    - So really, this all needs YANG modules that can describe topology: ACP, RPL,
  - Some other ways/conventions to allow this to be more easily understood
    - **Or just make it so reliable it doesn't matter**

Conclusions/Concerns

- On a **LAN** with an ACP-ignorant switch
  - Such as a Top-of-Rack Switch full of 1U servers containing a BMC
- Every system sees every other system
  - RPL prunes things, **after** IPsec tunnel is created
- With 20 to 40 1U servers per cabinet...
  - 1560 IPsec tunnels
    - Worse if multiple cabinets are L2 connected
- Need to prune tunnels at GRASP level
  - GRASP DULL needs to offer priority so ACP daemons can pick 2-3 links, and avoid all piling on top of a single DODAG parent
  - RFC9032, (and draft-ietf-roll-enrollment-priority) do this for 6tisch/802.15.4, we need to repeat this in GRASP DULL M_FLOOD announcements

Conclusions/Concerns

- On a **LAN** with an ACP-ignorant switch
  - Such as a Top-of-Rack Switch full of 1U servers containing a BMC
- Every system sees every other system
  - RPL prunes things, **after** IPsec tunnel is created
- With 20 to 40 1U servers per cabinet...
  - 1560 IPsec tunnels
    - Worse if multiple cabinets are L2 connected
- Need to prune tunnels at GRASP level
  - GRASP DULL needs to offer priority so ACP daemons can pick 2-3 links, and avoid all piling on top of a single DODAG parent
  - RFC9032, (and draft-ietf-roll-enrollment-priority) do this for 6tisch/802.15.4, we need to repeat this in GRASP DULL M_FLOOD announcements