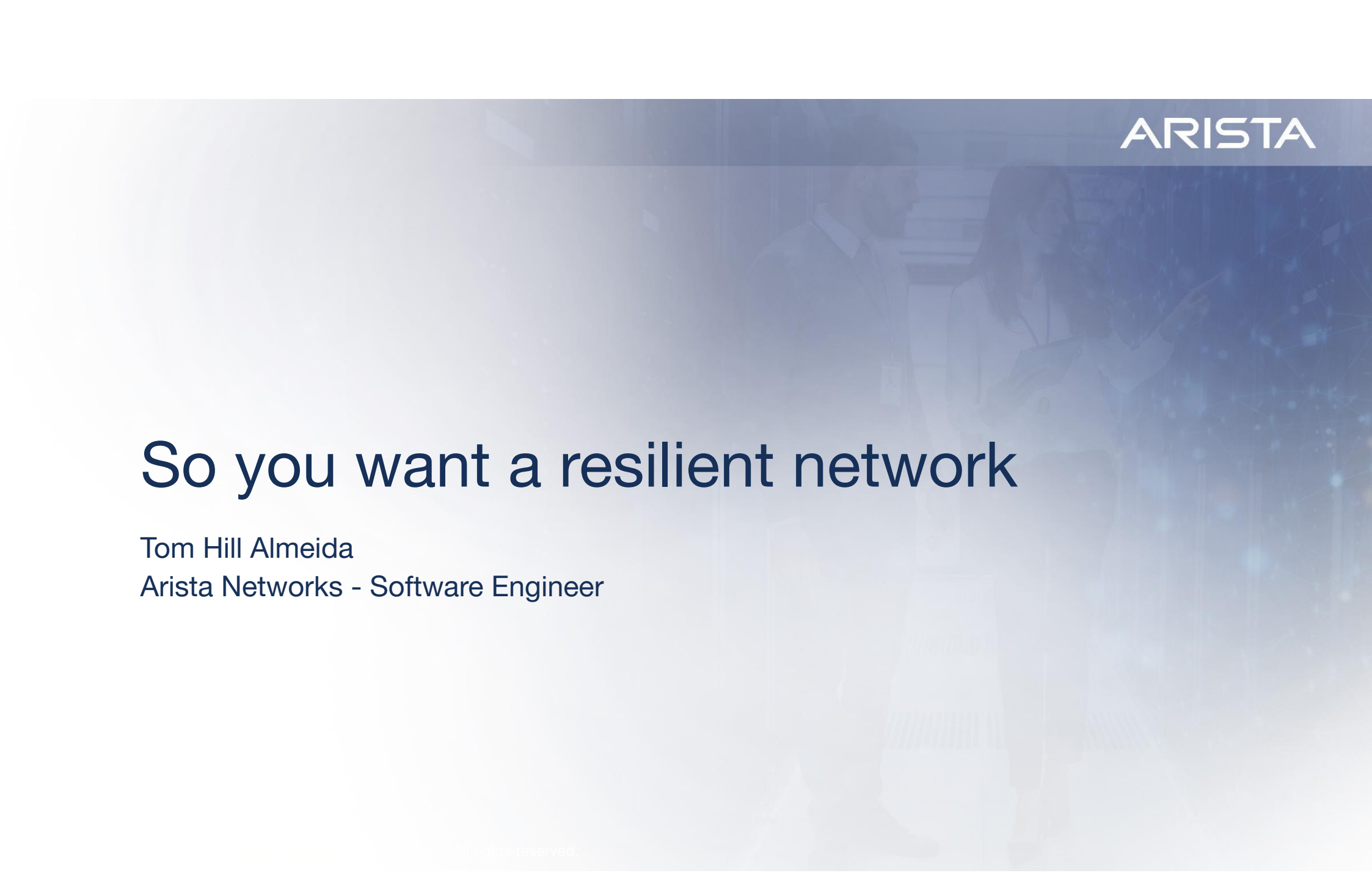
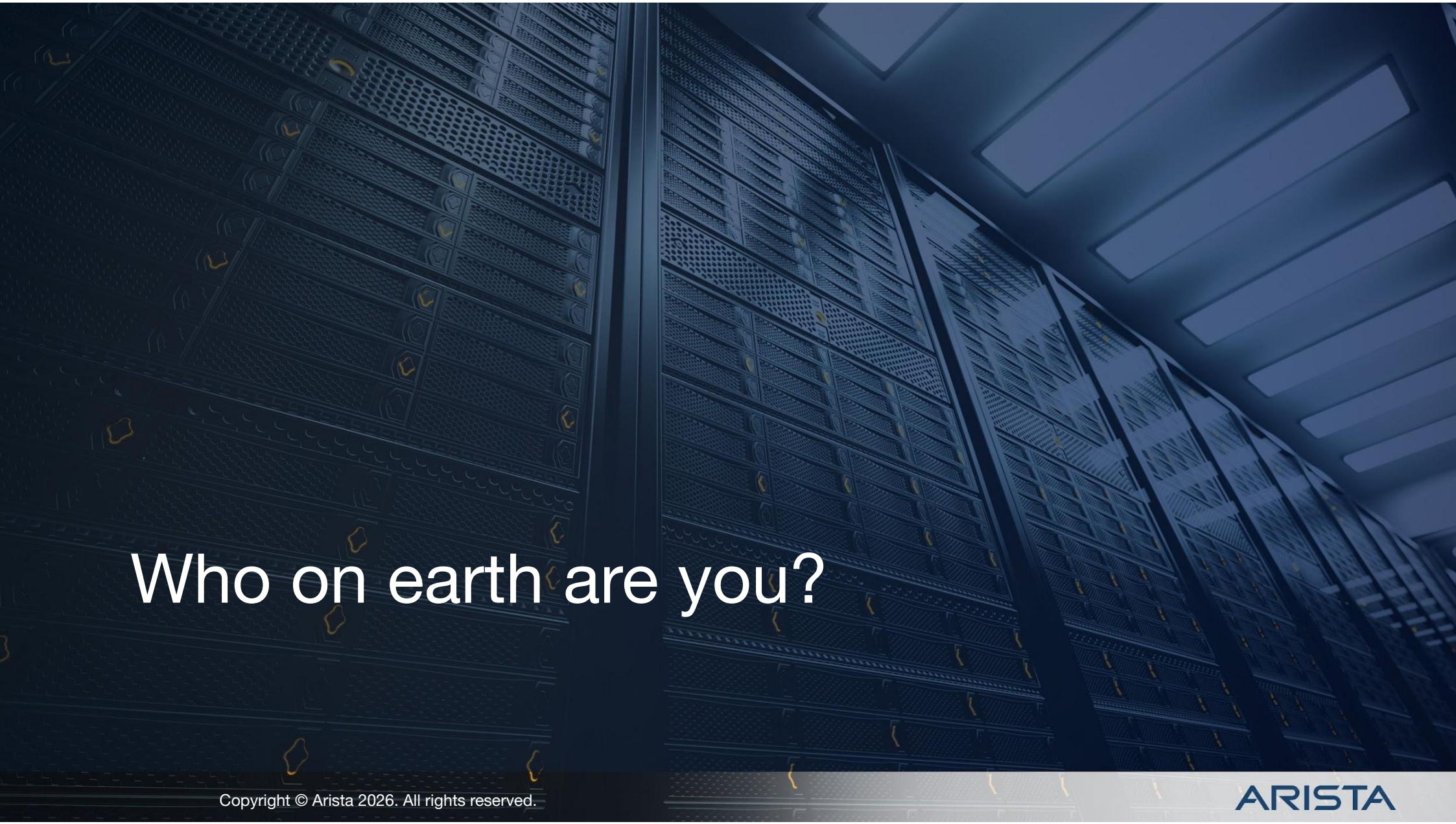


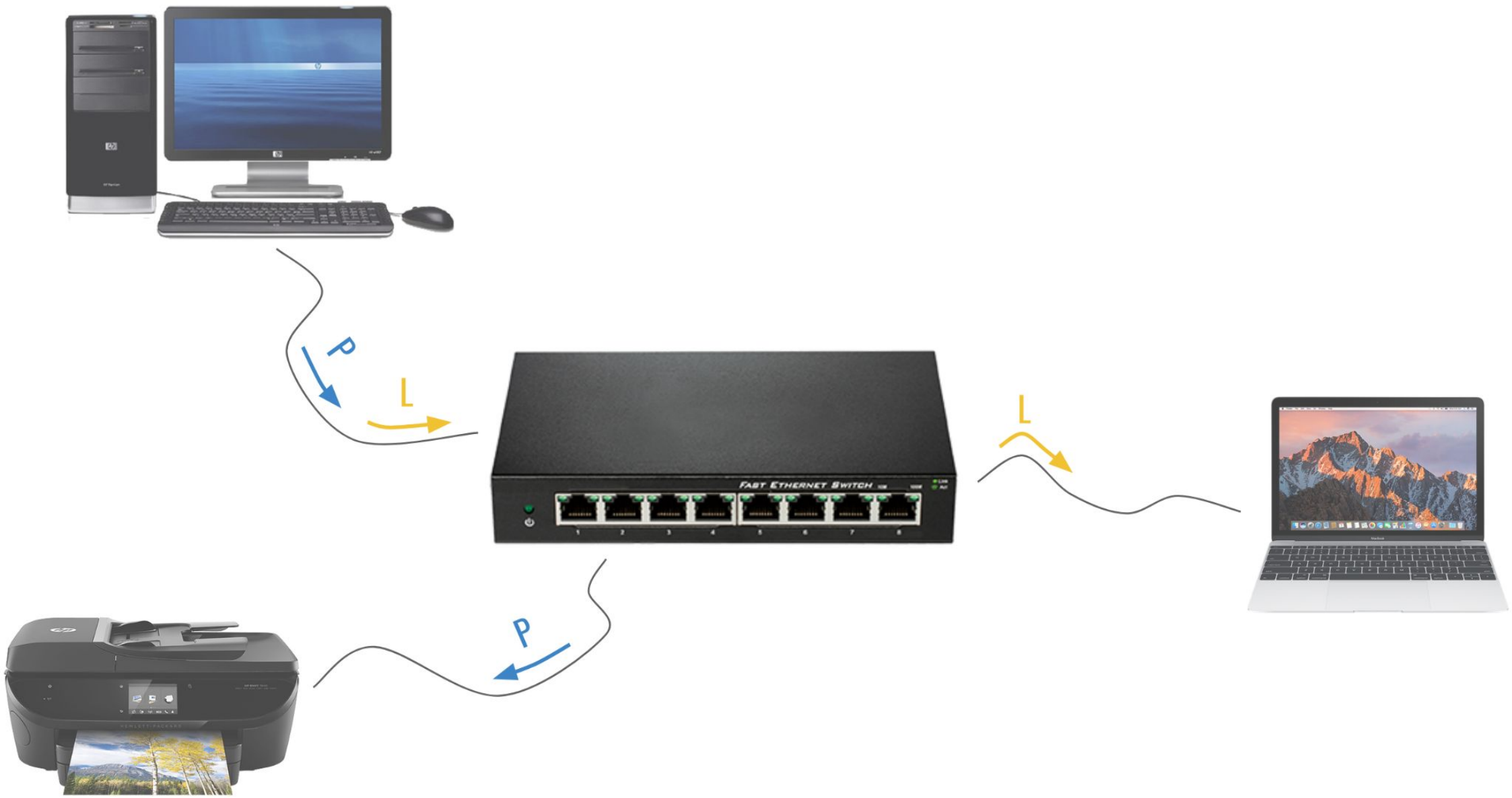

# So you want a resilient network

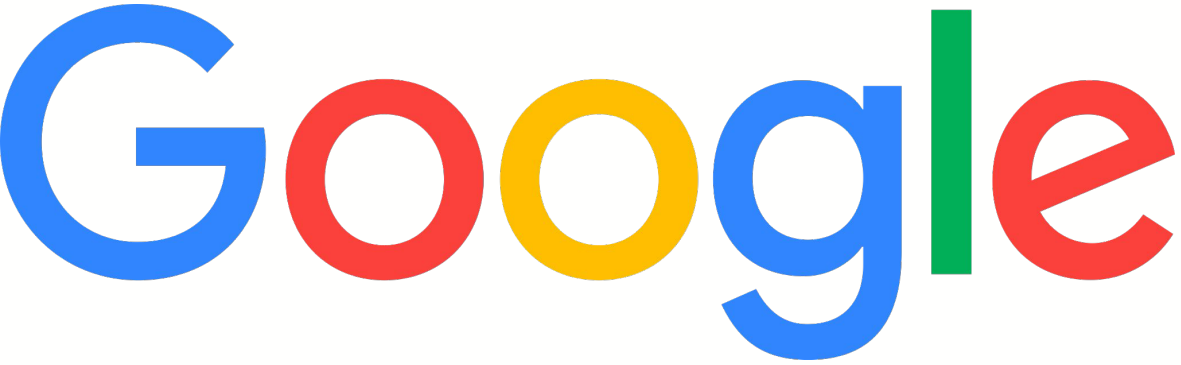
Tom Hill Almeida
Arista Networks - Software Engineer



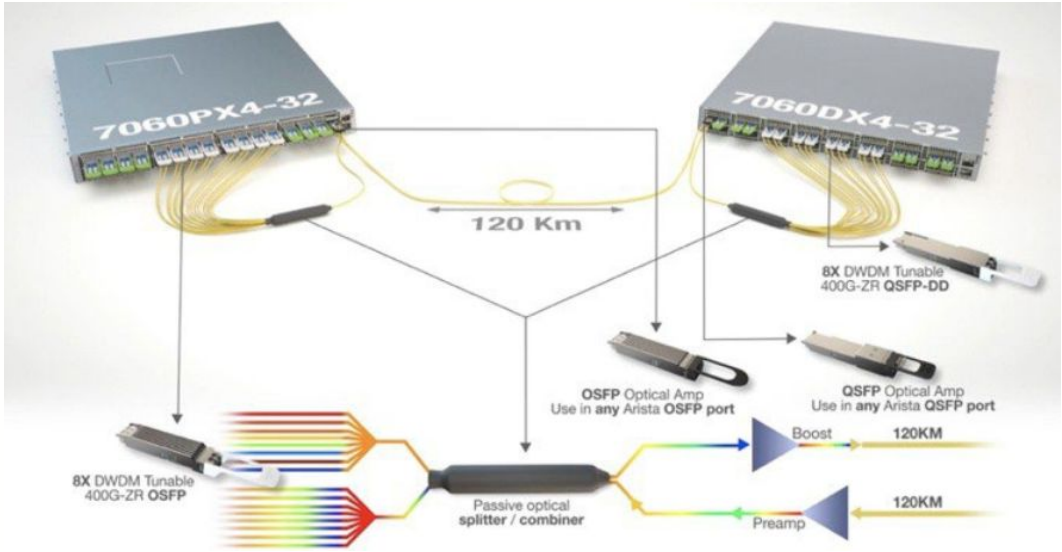



ARISTA


P
L
FAST ETHERNET SWITCH
L
P





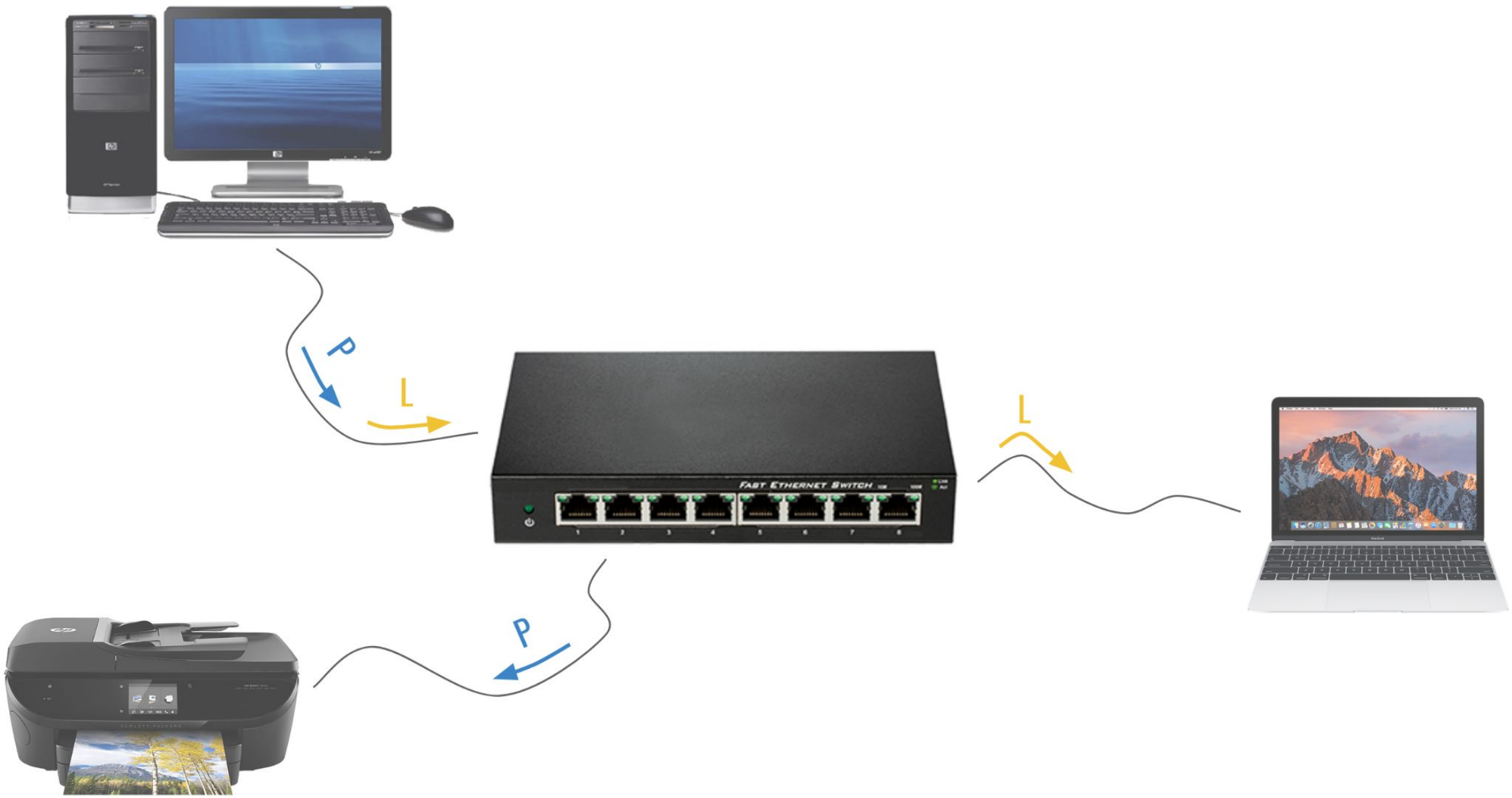



ARISTA

# How do they work?

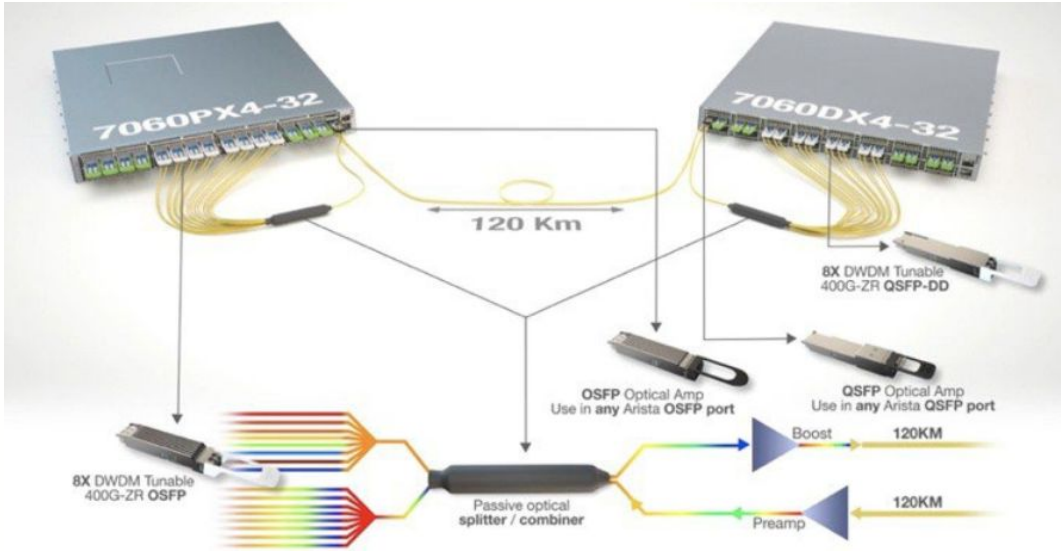


ARISTA


P
L
FAST ETHERNET SWITCH
L
P

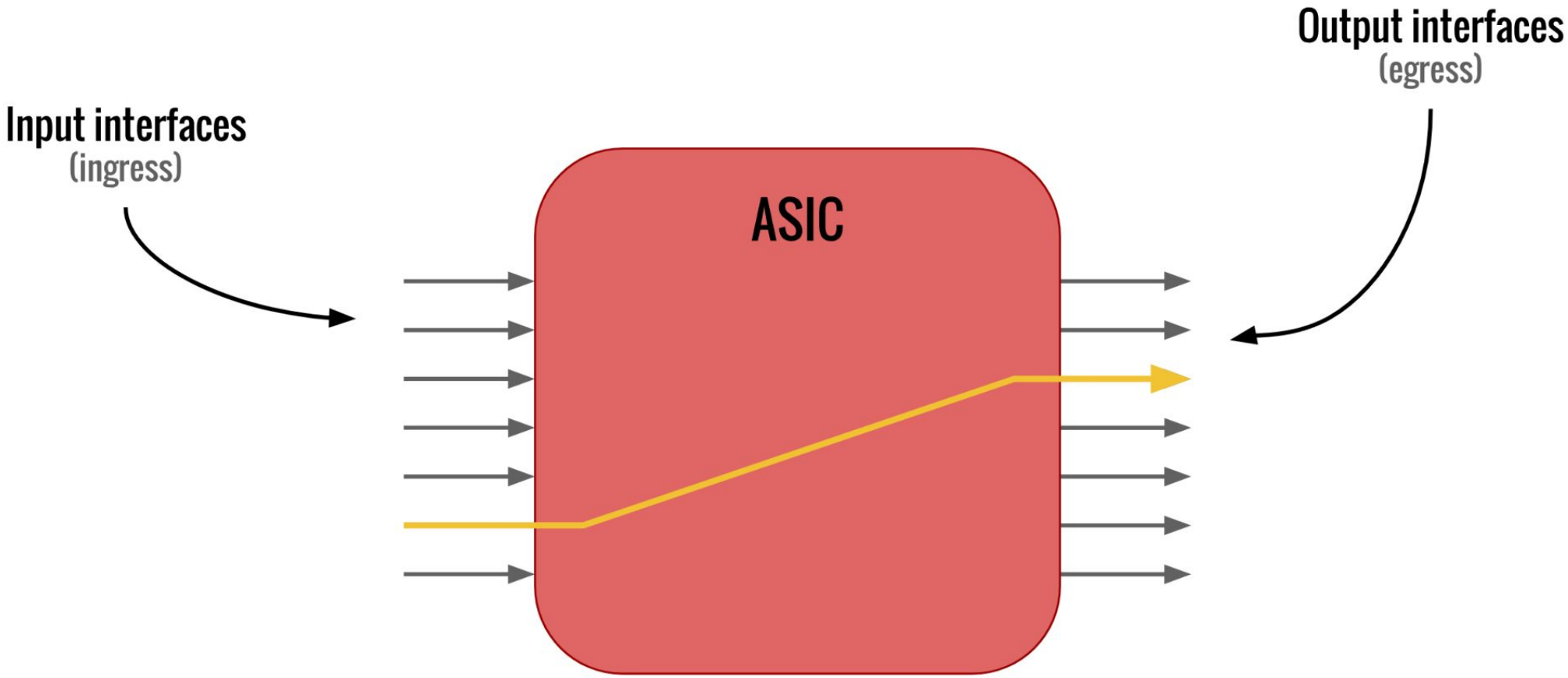
**802.3 Ethernet packet and frame structure**

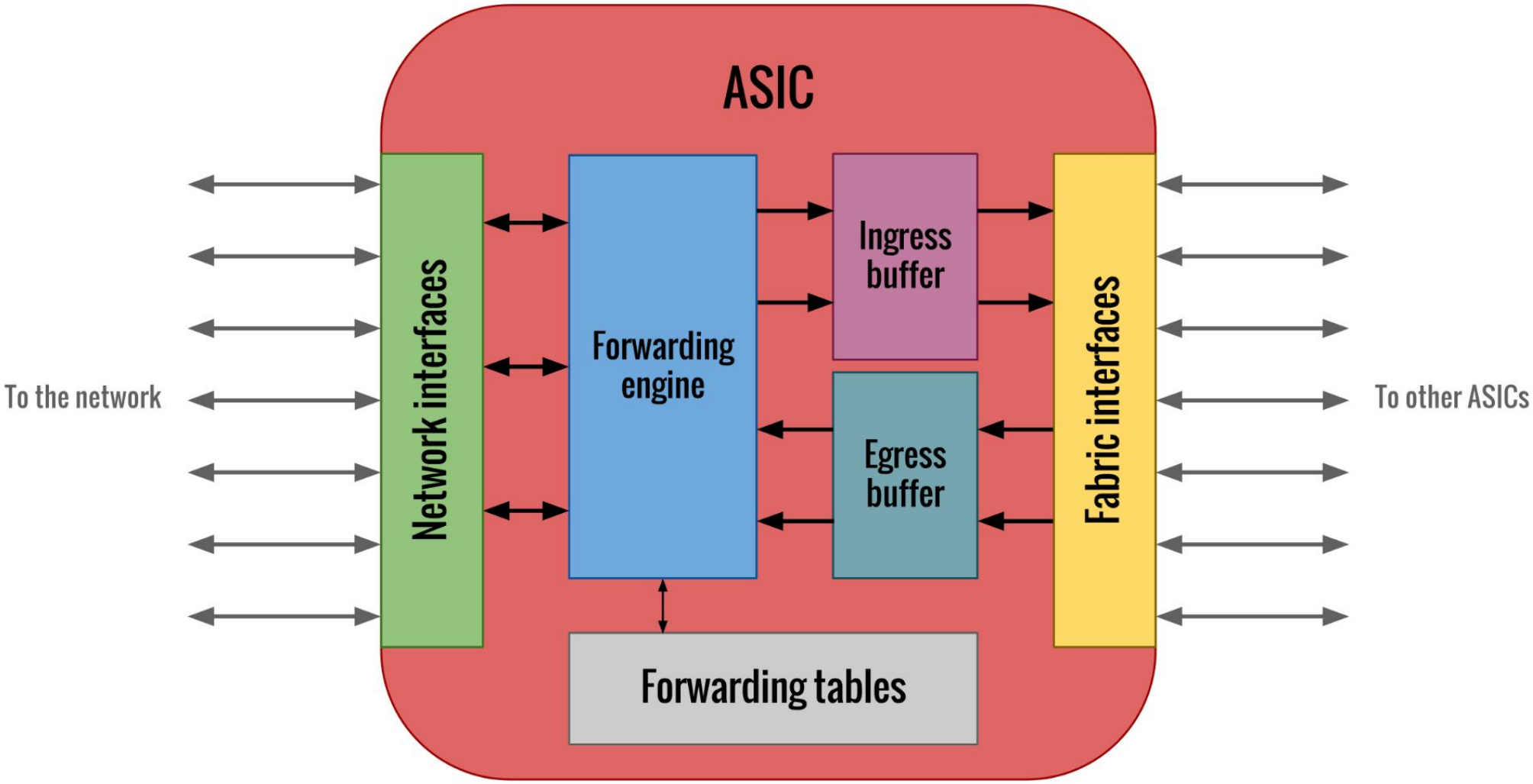
| Layer | Preamble | Start frame delimiter (SFD) | MAC destination | MAC source | 802.1Q tag (optional) | Ethertype (Ethernet II) or length (IEEE 802.3) | Payload | Frame check sequence (32-bit CRC) | Interpacket gap (IPG) |
|---|---|---|---|---|---|---|---|---|---|
| Length (octets) | 7 | 1 | 6 | 6 | (4) | 2 | 42–1500[c] | 4 | 12 |
| Layer 2 Ethernet frame | (not part of the frame) | | ← 64–1522 octets → | | | | | | (not part of the frame) |
| Layer 1 Ethernet packet & IPG | ← 72–1530 octets → | | | | | | | | ← 12 octets → |






ARISTA



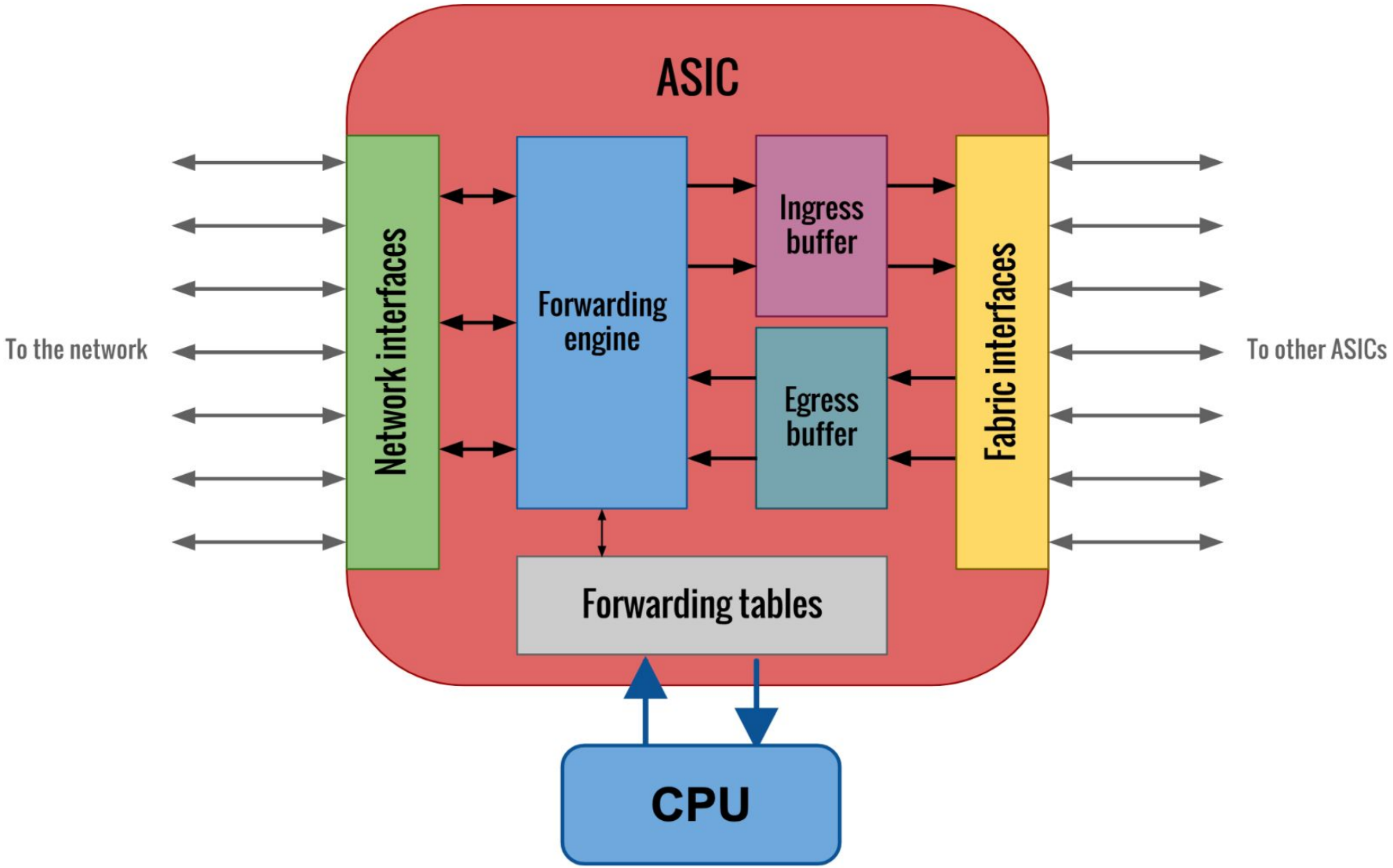


**802.3 Ethernet packet and frame structure**

| Layer | Preamble | Start frame delimiter (SFD) | MAC destination | MAC source | 802.1Q tag (optional) | Ethertype (Ethernet II) or length (IEEE 802.3) | Payload | Frame check sequence (32-bit CRC) | Interpacket gap (IPG) |
|---|---|---|---|---|---|---|---|---|---|
| Length (octets) | 7 | 1 | 6 | 6 | (4) | 2 | 42–1500[c] | 4 | 12 |
| Layer 2 Ethernet frame | (not part of the frame) | | ← 64–1522 octets → | | | | | | (not part of the frame) |
| Layer 1 Ethernet packet & IPG | ← 72–1530 octets → | | | | | | | | ← 12 octets → |

# Can an ASIC do everything?

- Which ports are allowed to talk to each other?
- Which links are used or blocked?
- Should some links be prioritised?
- How do we decide which packets to drop?



# What does the control plane handle?

- Operator configuration, policy, CLI and management APIs

# What does the control plane handle?

- Operator configuration, policy, CLI and management APIs
- Interface, transceiver and link-state changes

# What does the control plane handle?

- Operator configuration, policy, CLI and management APIs
- Interface, transceiver and link-state changes
- VLAN bridging, routing protocols and topology decisions

# What does the control plane handle?

- Operator configuration, policy, CLI and management APIs
- Interface, transceiver and link-state changes
- VLAN bridging, routing protocols and topology decisions
- Deriving and programming forwarding rules into the ASIC

# What does the control plane handle?

- Operator configuration, policy, CLI and management APIs
- Interface, transceiver and link-state changes
- VLAN bridging, routing protocols and topology decisions
- Deriving and programming forwarding rules into the ASIC
- Telemetry, logging, upgrades and hardware-specific control

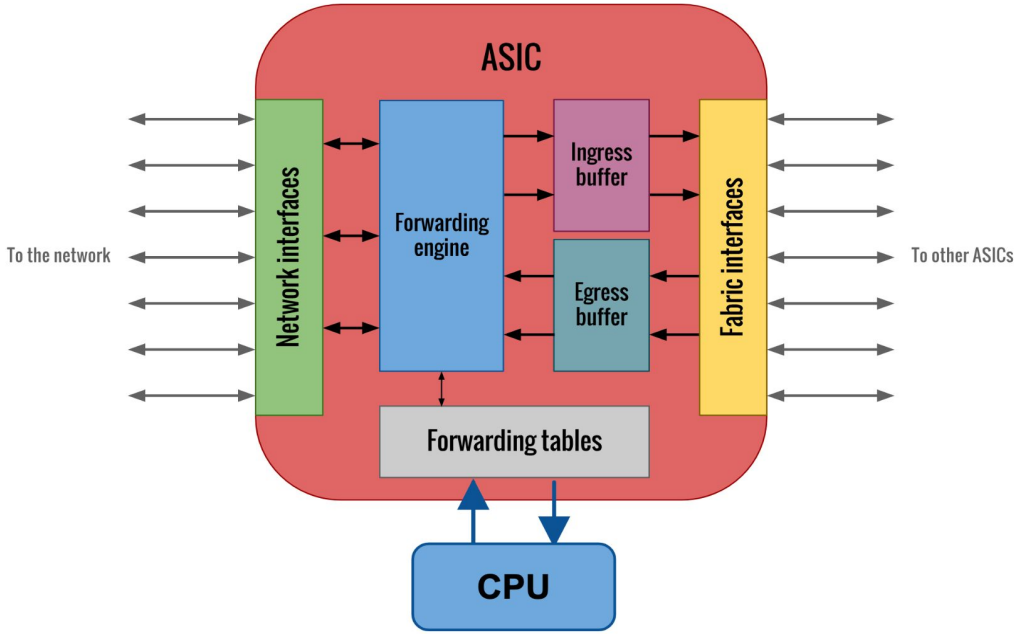
# What does the control plane handle?

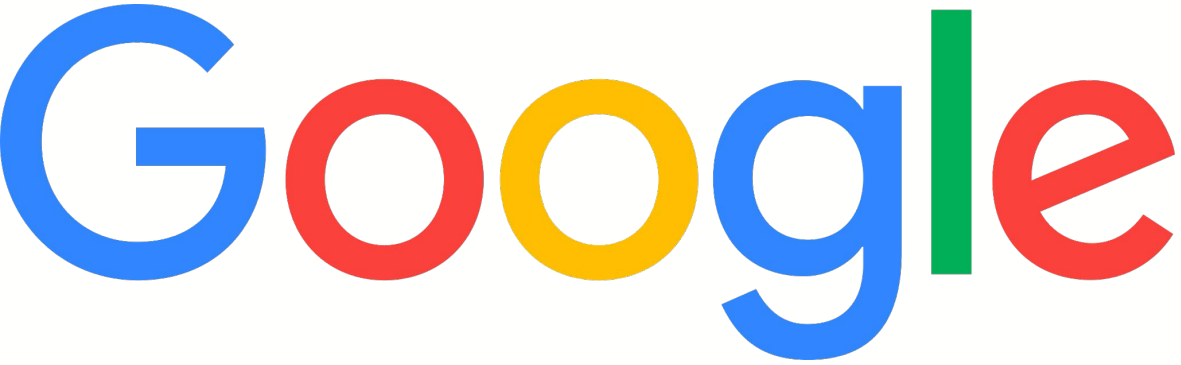
- Operator configuration, policy, CLI and management APIs
- Interface, transceiver and link-state changes
- VLAN bridging, routing protocols and topology decisions
- Deriving and programming forwarding rules into the ASIC
- Telemetry, logging, upgrades and hardware-specific control
- This is a non-exhaustive list

# But what happens if something goes wrong?

# What happens if your control plane goes down

- Existing traffic continues
- New configuration cannot be applied
- Learned state may become stale
- Traffic introspection becomes impossible








# So how do we do these things at the same time?

- Operator configuration, policy, CLI and management APIs
- Interface, transceiver and link-state changes
- VLAN bridging, routing protocols and topology decisions
- Deriving and programming forwarding rules into the ASIC
- Telemetry, logging, upgrades and hardware-specific control
- This is a non-exhaustive list

This is what our operating system, EOS, tries to do.

# Multi-threaded process

- Spin an event queue
- Wait for an event to come in from a driver
- Dispatch to a thread to do some work
- If the work generates another event, it can spawn other work

# Multi-threaded process

- Spin an event queue
- Wait for an event to come in from a driver
- Dispatch to a thread to do some work
- If the work generates another event, it can spawn other work

- Any errors may corrupt all threads' heaps

# Multi-threaded process

- Spin an event queue
- Wait for an event to come in from a driver
- Dispatch to a thread to do some work
- If the work generates another event, it can spawn other work

- Any errors may corrupt all threads' heaps
- Any errors may crash the whole process

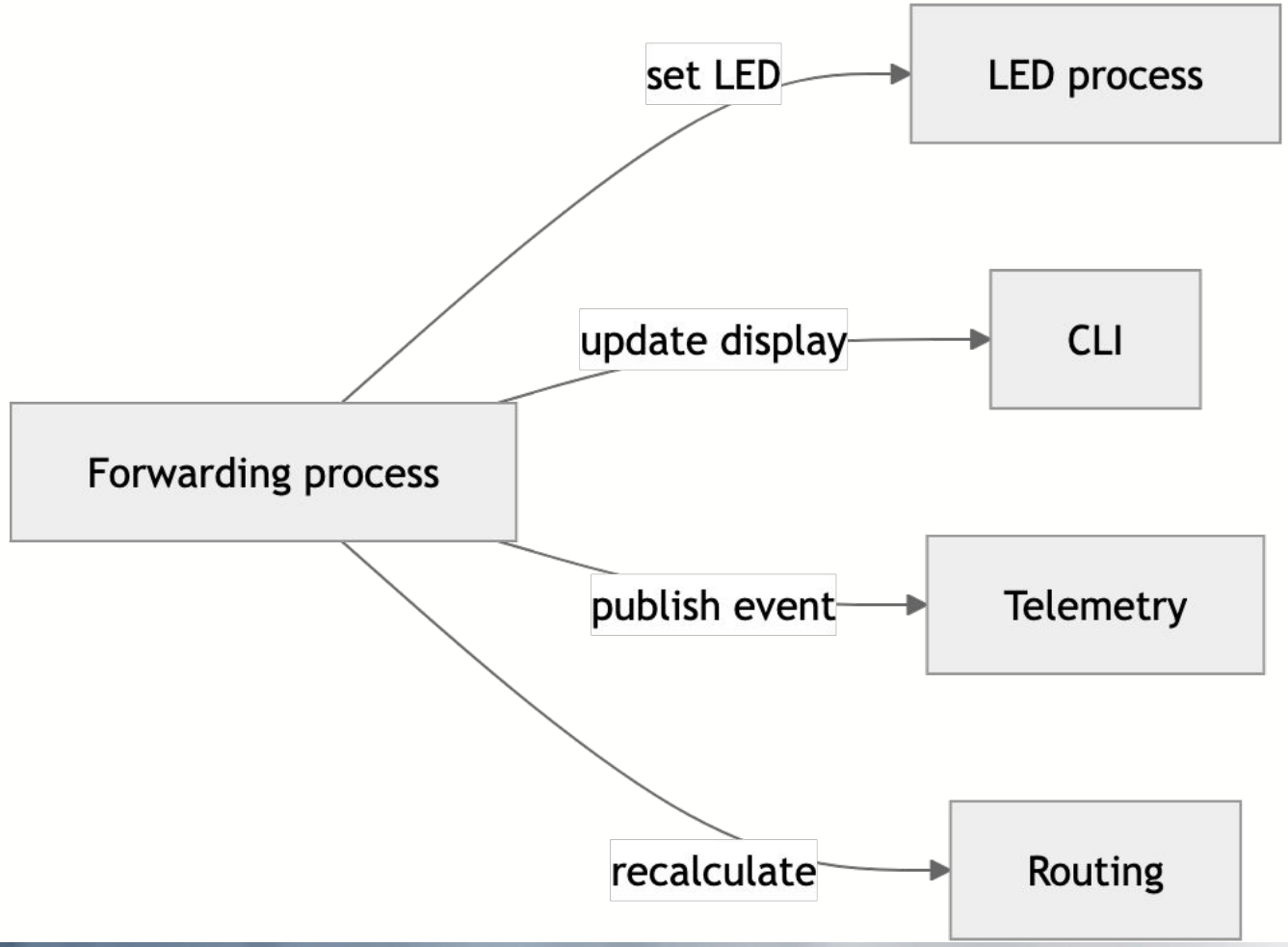
# Multi-process

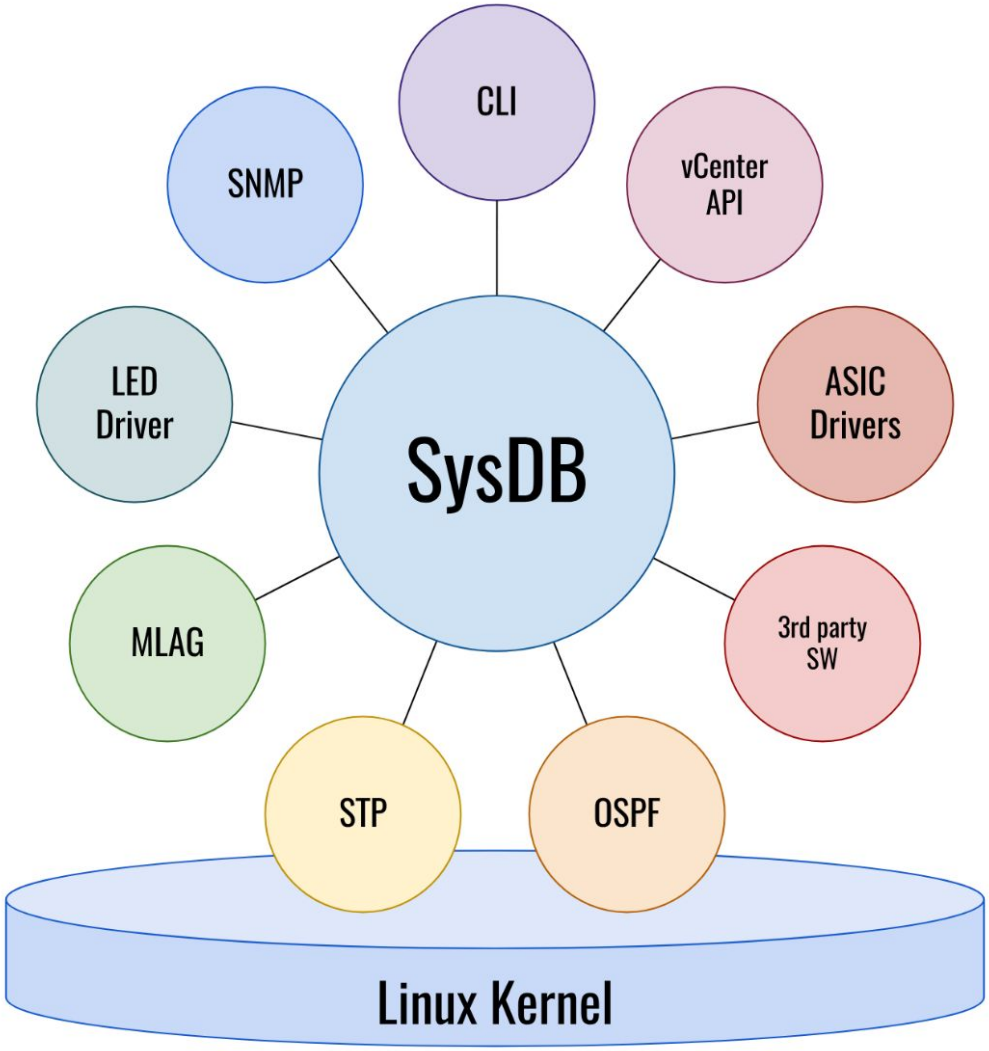
- Can’t poison another heap (without a serious bug in the kernel)

# Multi-process

- Can’t poison another heap (without a serious bug in the kernel)
- Can’t take down the whole system if you crash

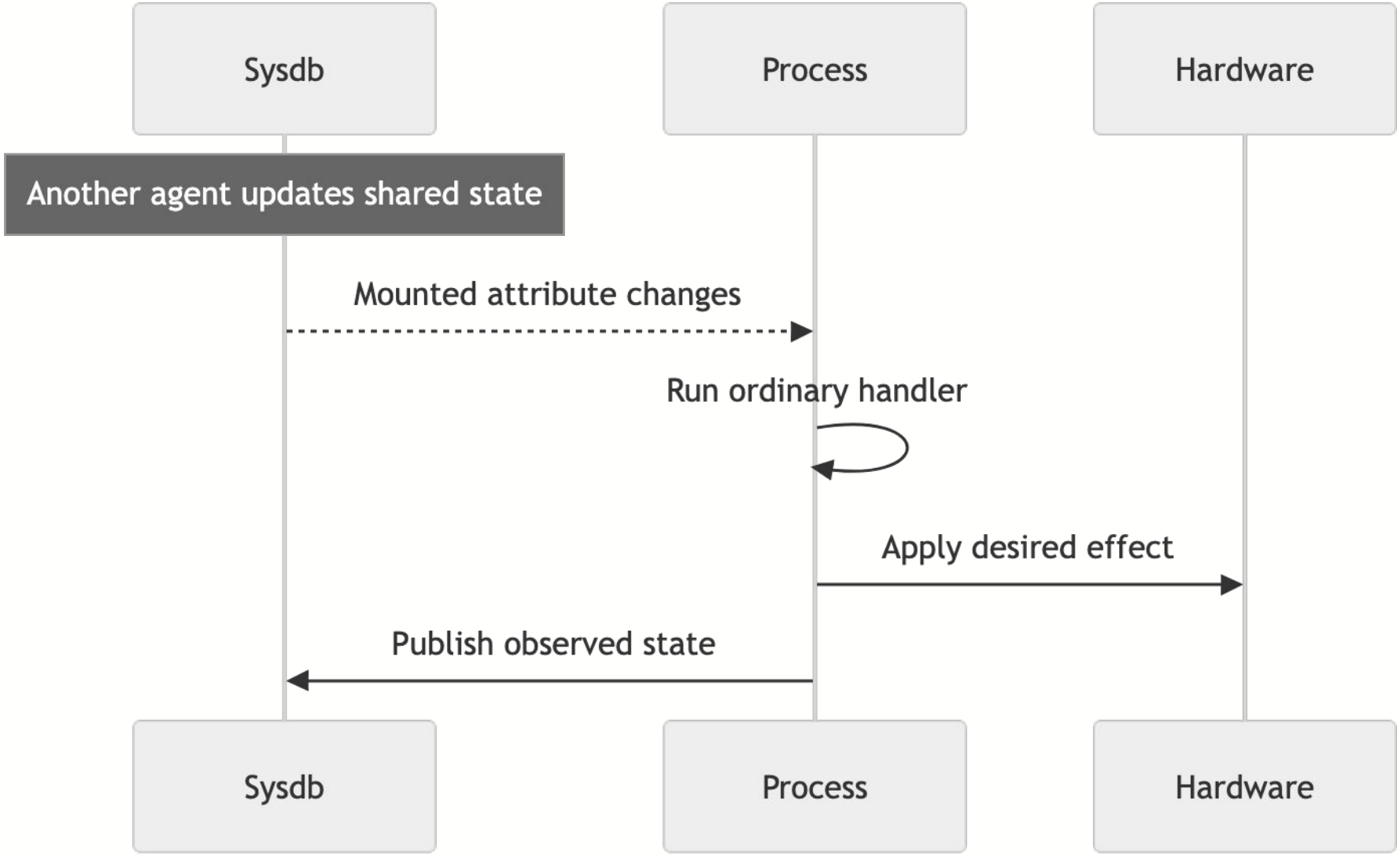
# What happens when a port goes down?



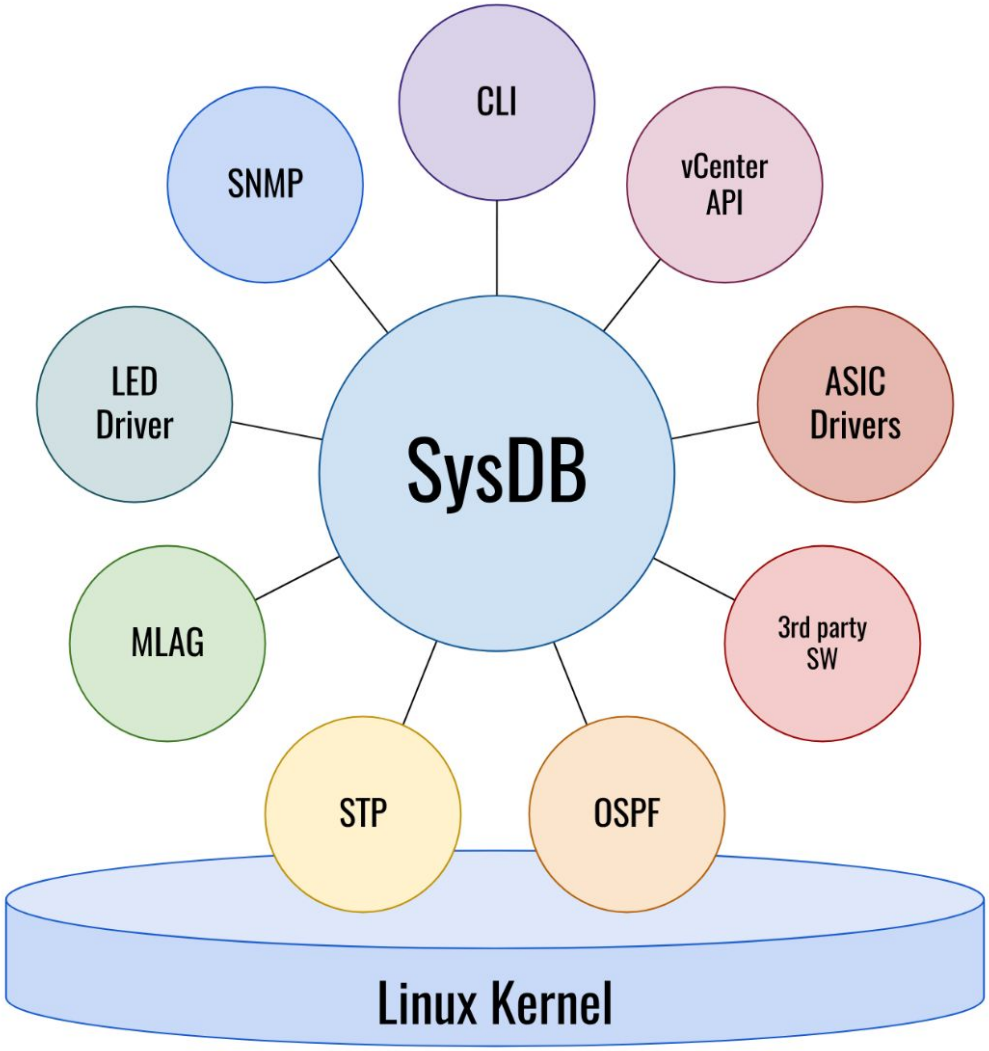





ARISTA

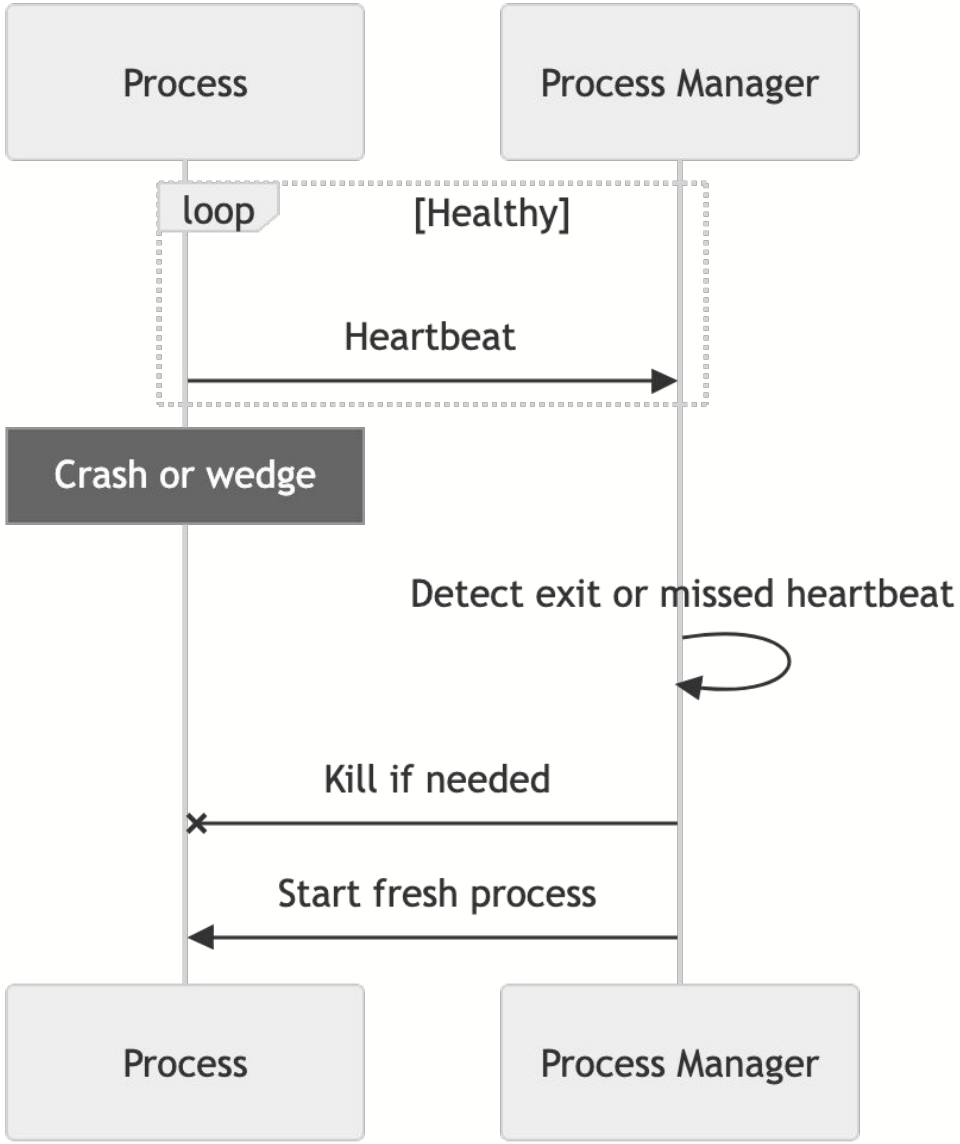


# What happens if a process crashes?



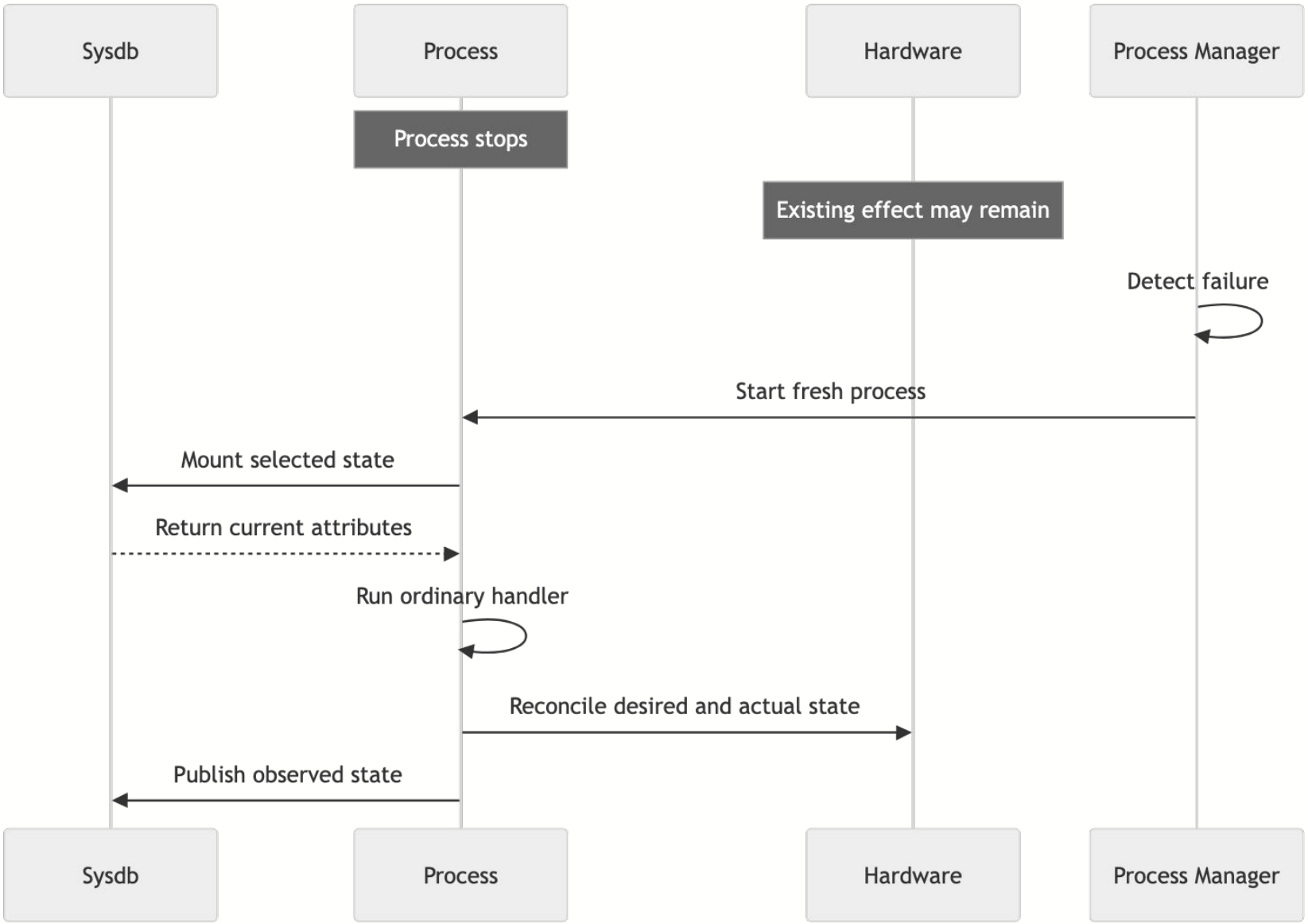



ARISTA



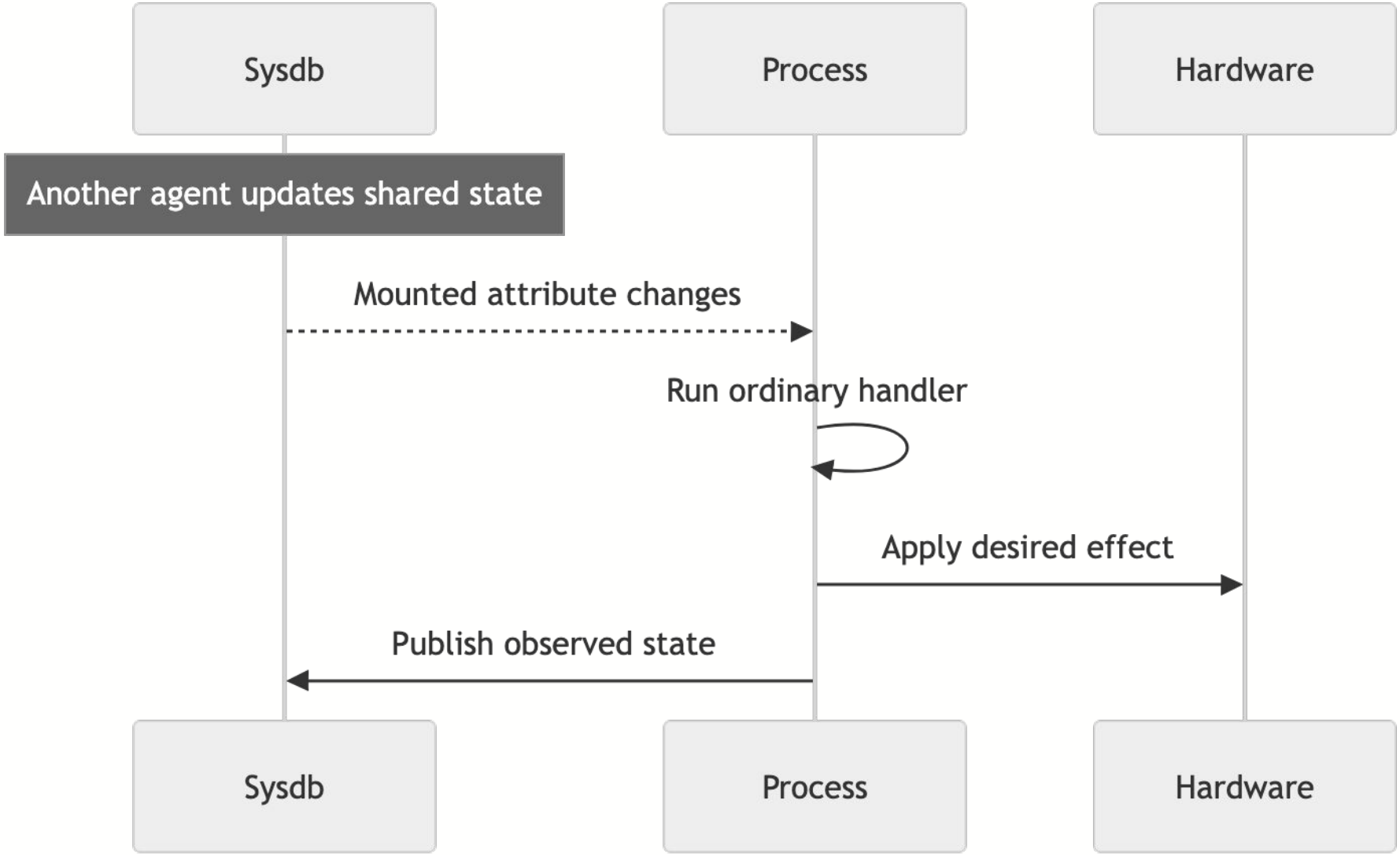


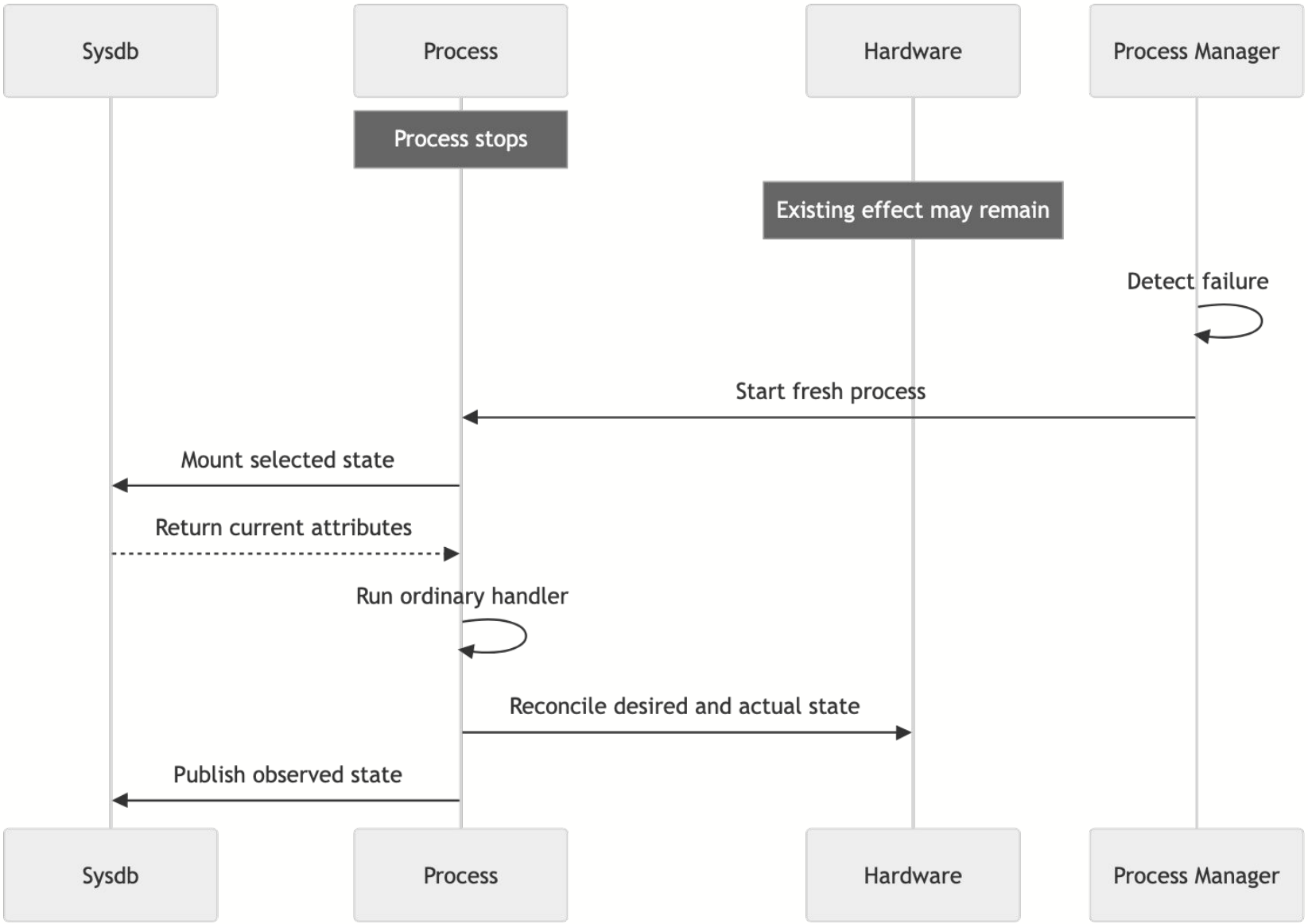

ARISTA






ARISTA

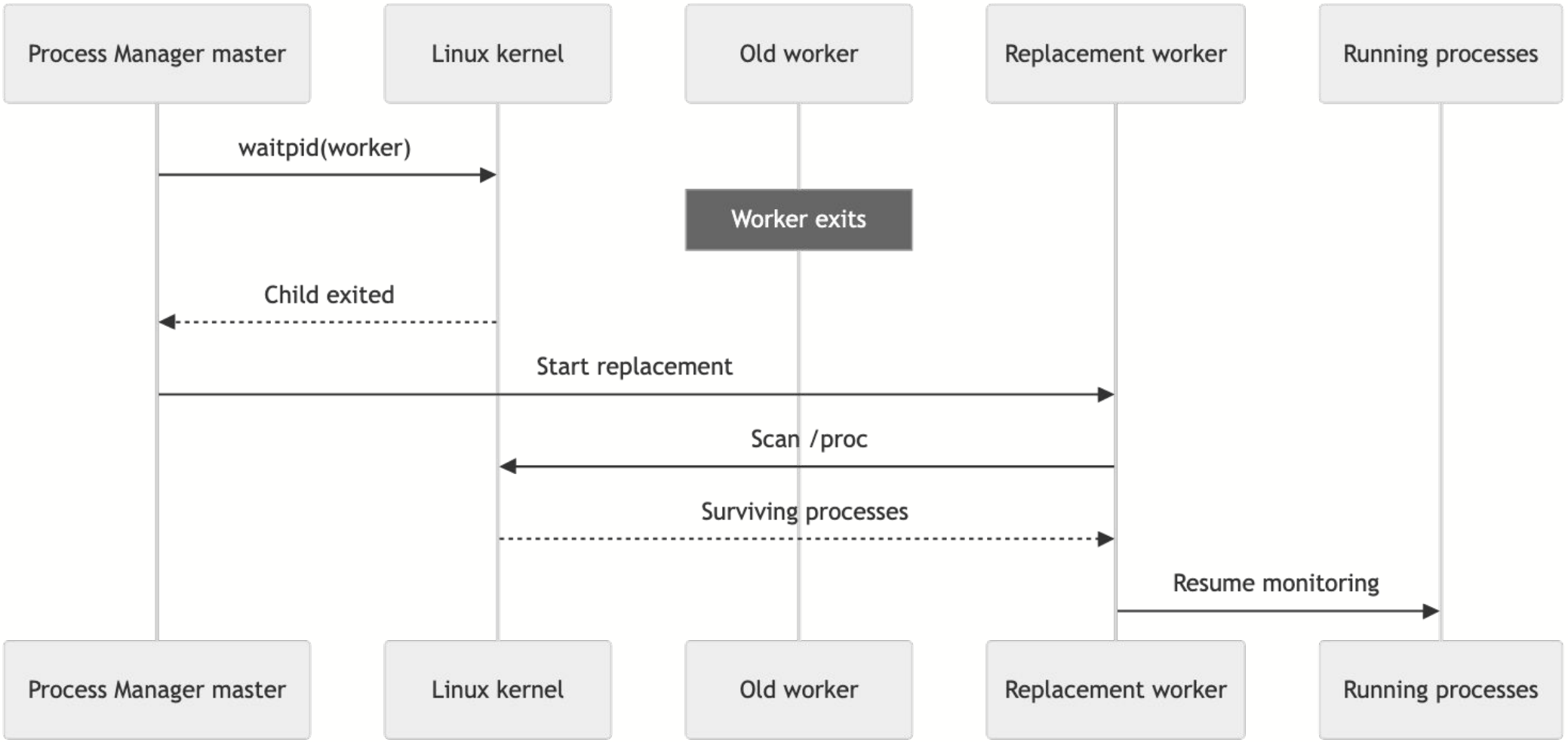







ARISTA

# What if the process manager crashes?



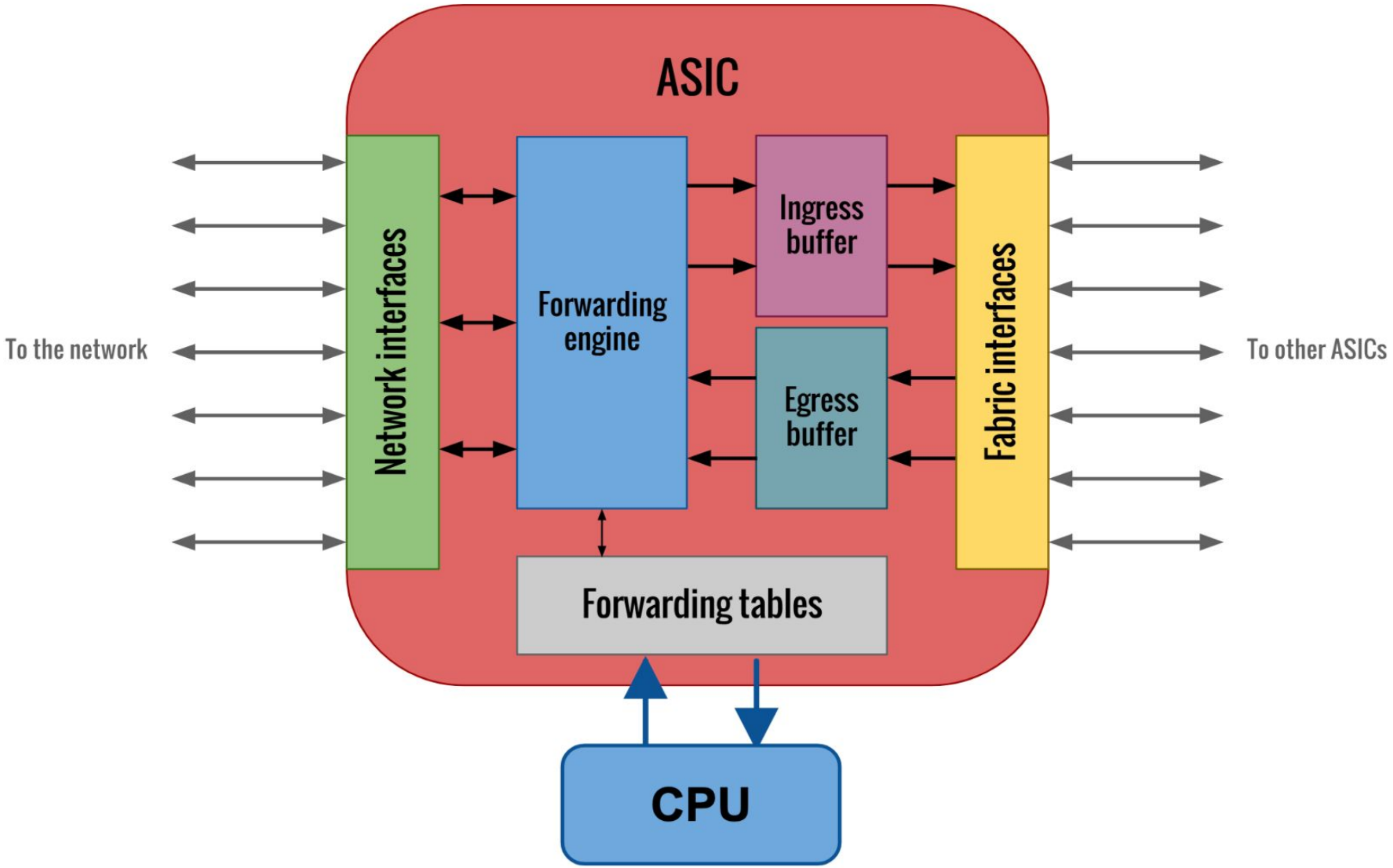


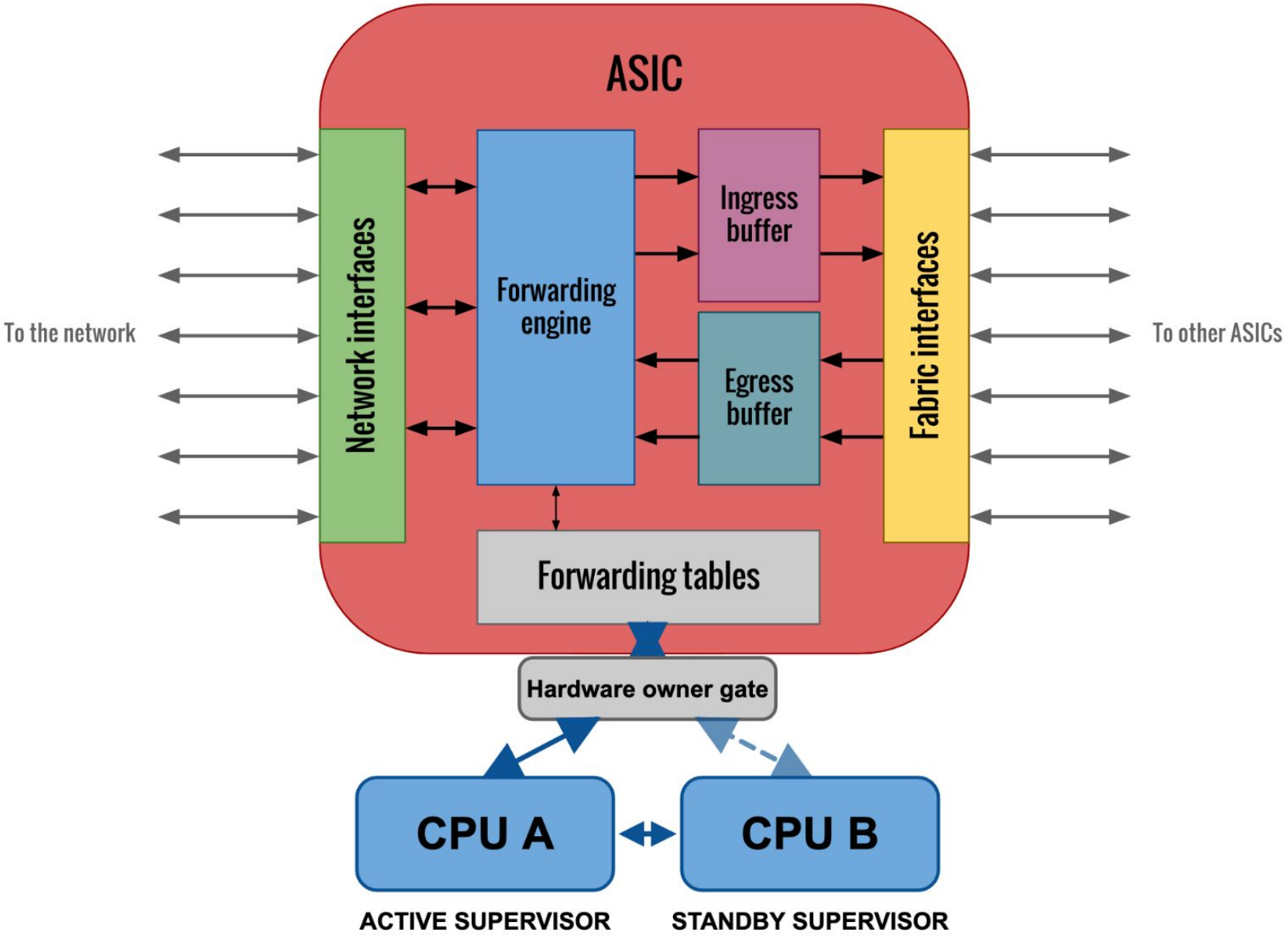

ARISTA

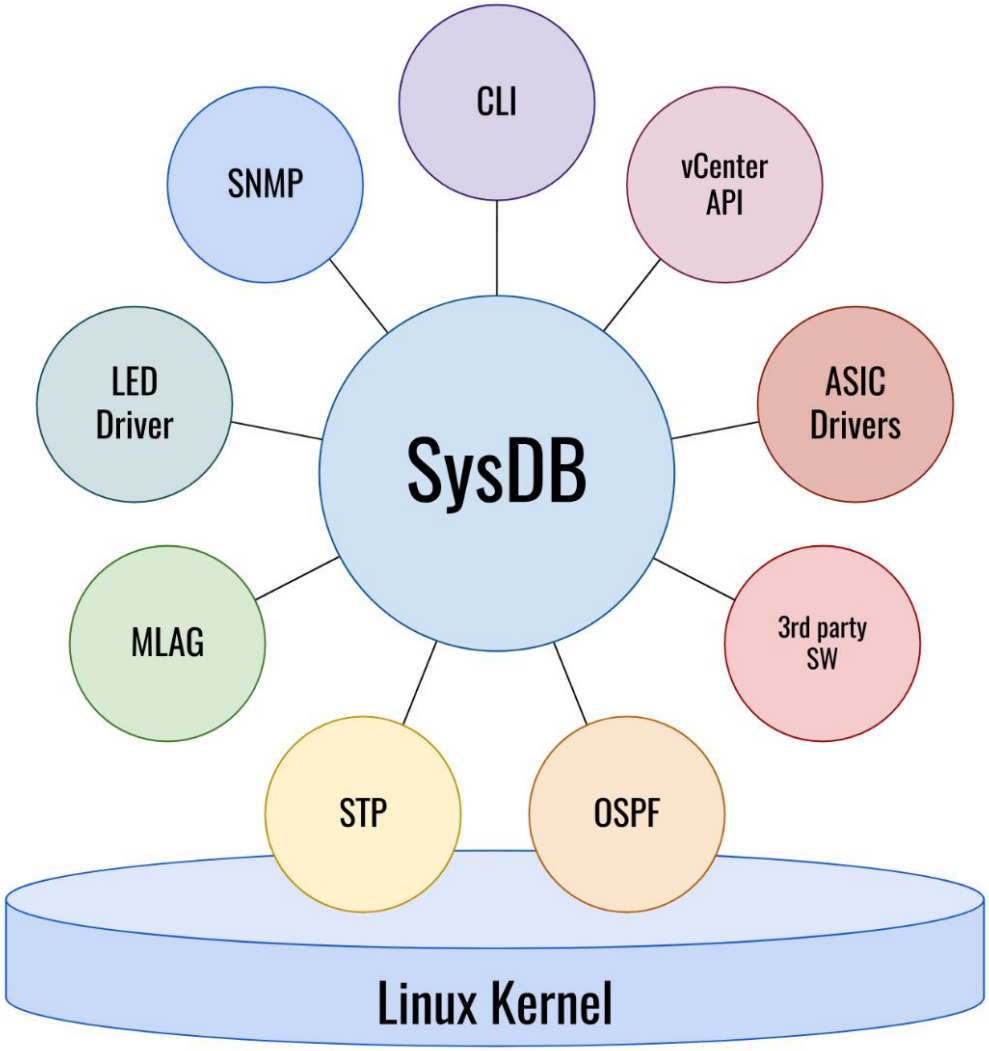
# But what if your CPU crashes?

# Watchdog code

```
# /etc/systemd/system.conf
RuntimeWatchdogSec=30s
```

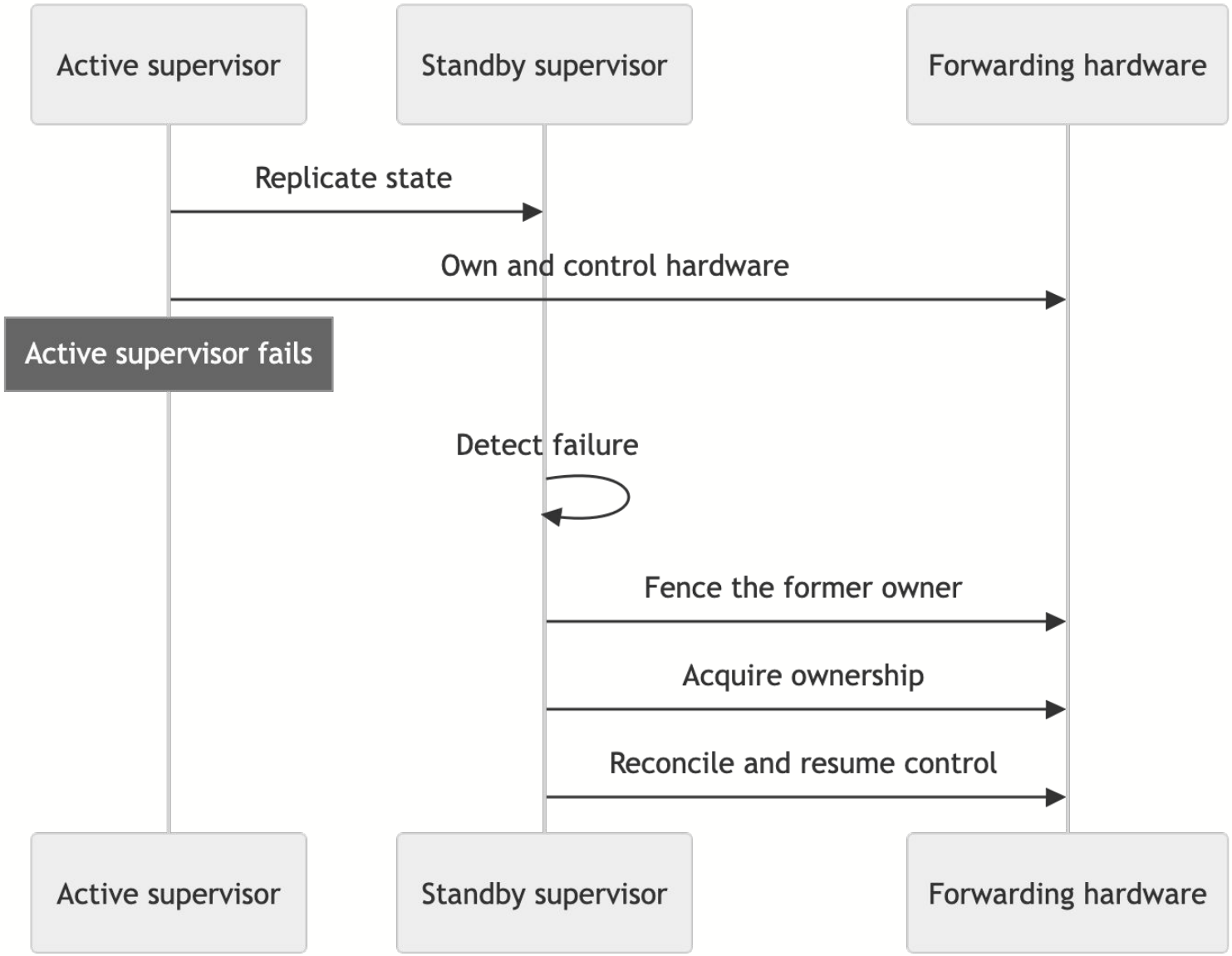



ASIC
To the network
Network interfaces
Forwarding engine
Ingress buffer
Egress buffer
Fabric interfaces
To other ASICs
Forwarding tables
Hardware owner gate
CPU A
CPU B
ACTIVE SUPERVISOR
STANDBY SUPERVISOR





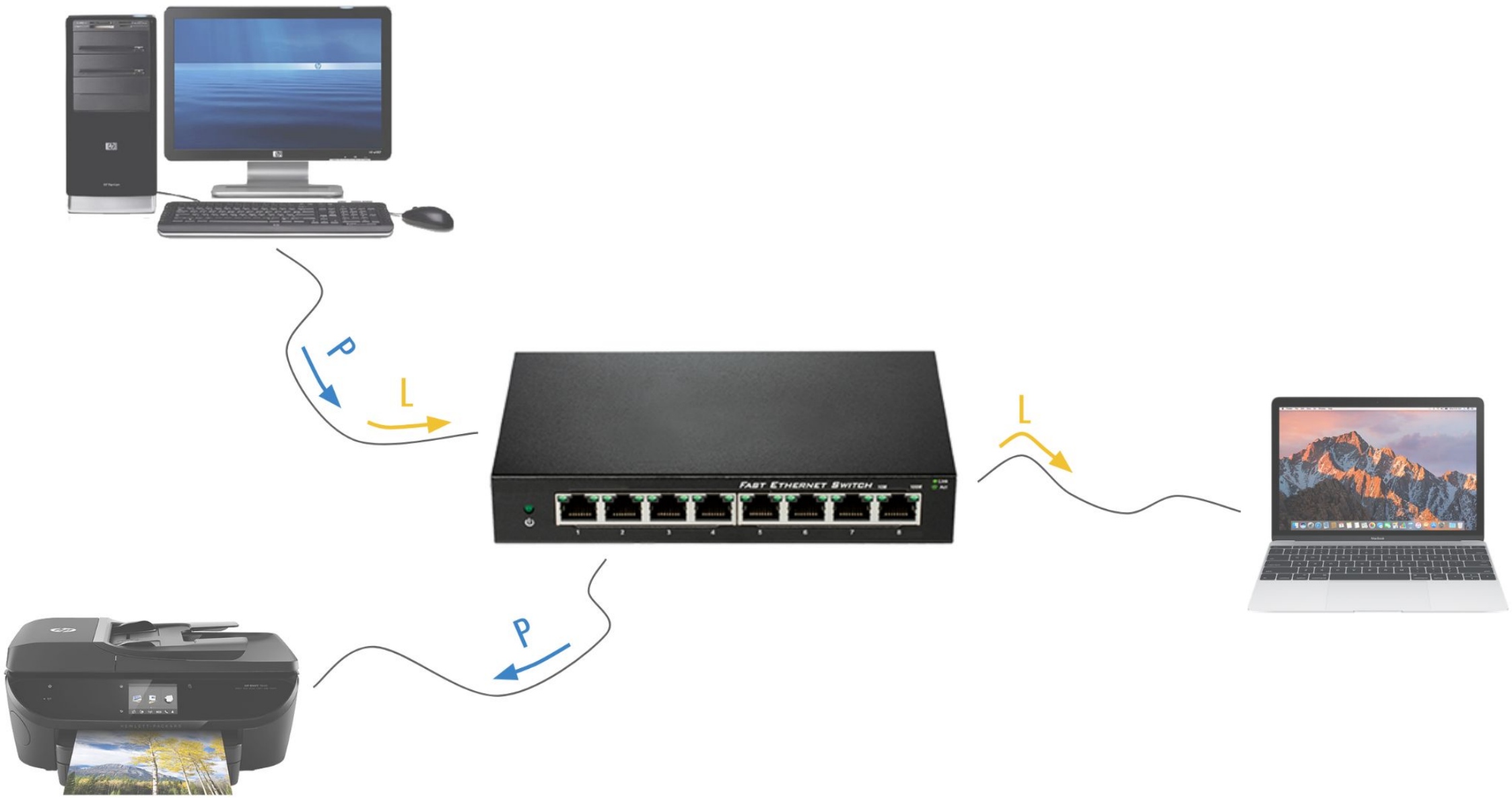

ARISTA





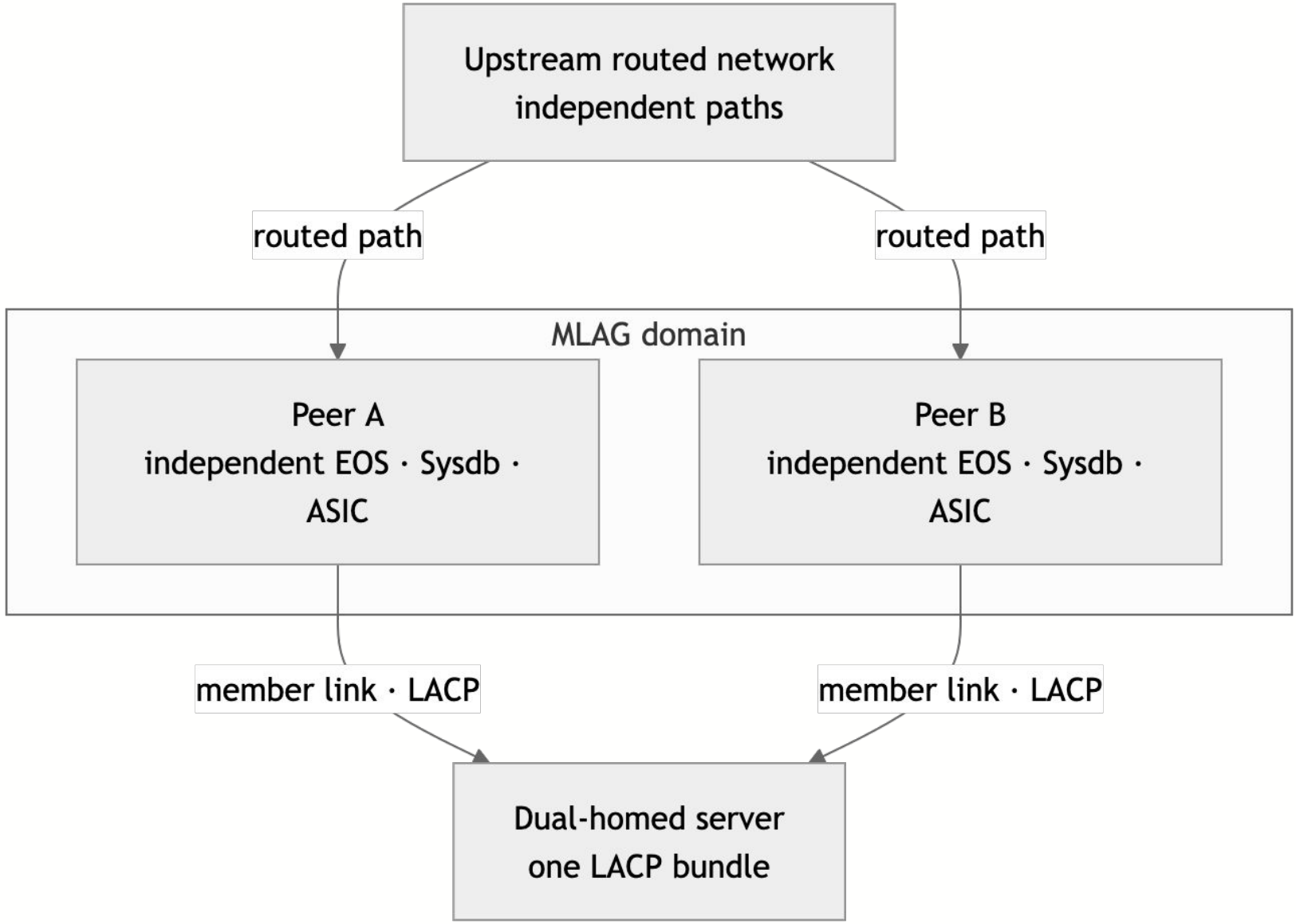

ARISTA

# But what if your power fails?


P
L
FAST ETHERNET SWITCH
L
P



ARISTA






ARISTA

# Notice a pattern?

- Contain the smallest replaceable component

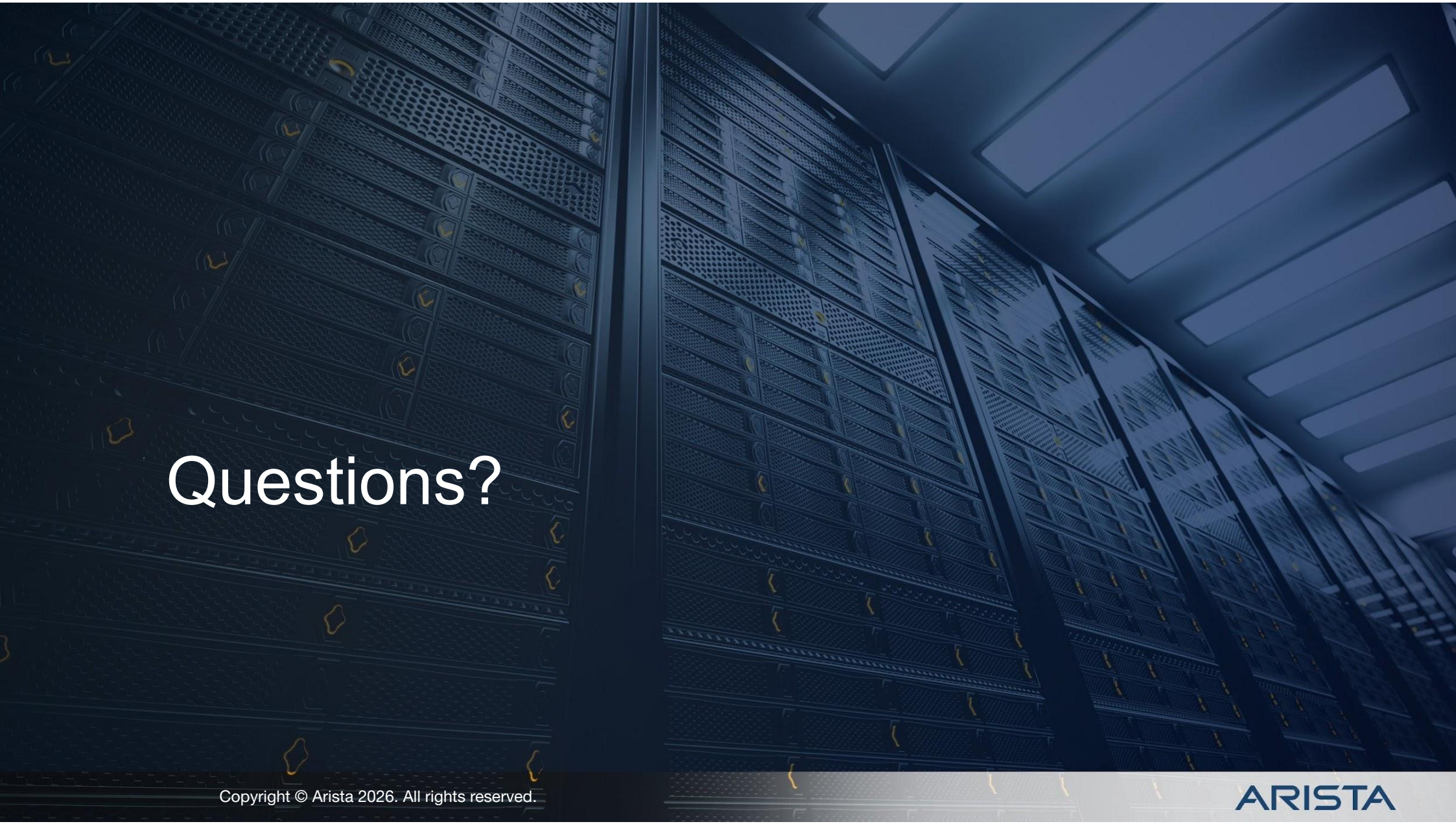
- Contain the smallest replaceable component
- Detect from outside the failed component

- Contain the smallest replaceable component
- Detect from outside the failed component
- Preserve the state and useful effects needed for recovery

- Contain the smallest replaceable component
- Detect from outside the failed component
- Preserve the state and useful effects needed for recovery
- Establish one owner before control changes hands

- Contain the smallest replaceable component
- Detect from outside the failed component
- Preserve the state and useful effects needed for recovery
- Establish one owner before control changes hands
- Re-enter normal handlers and reconcile

# So you want a resilient network?






ARISTA