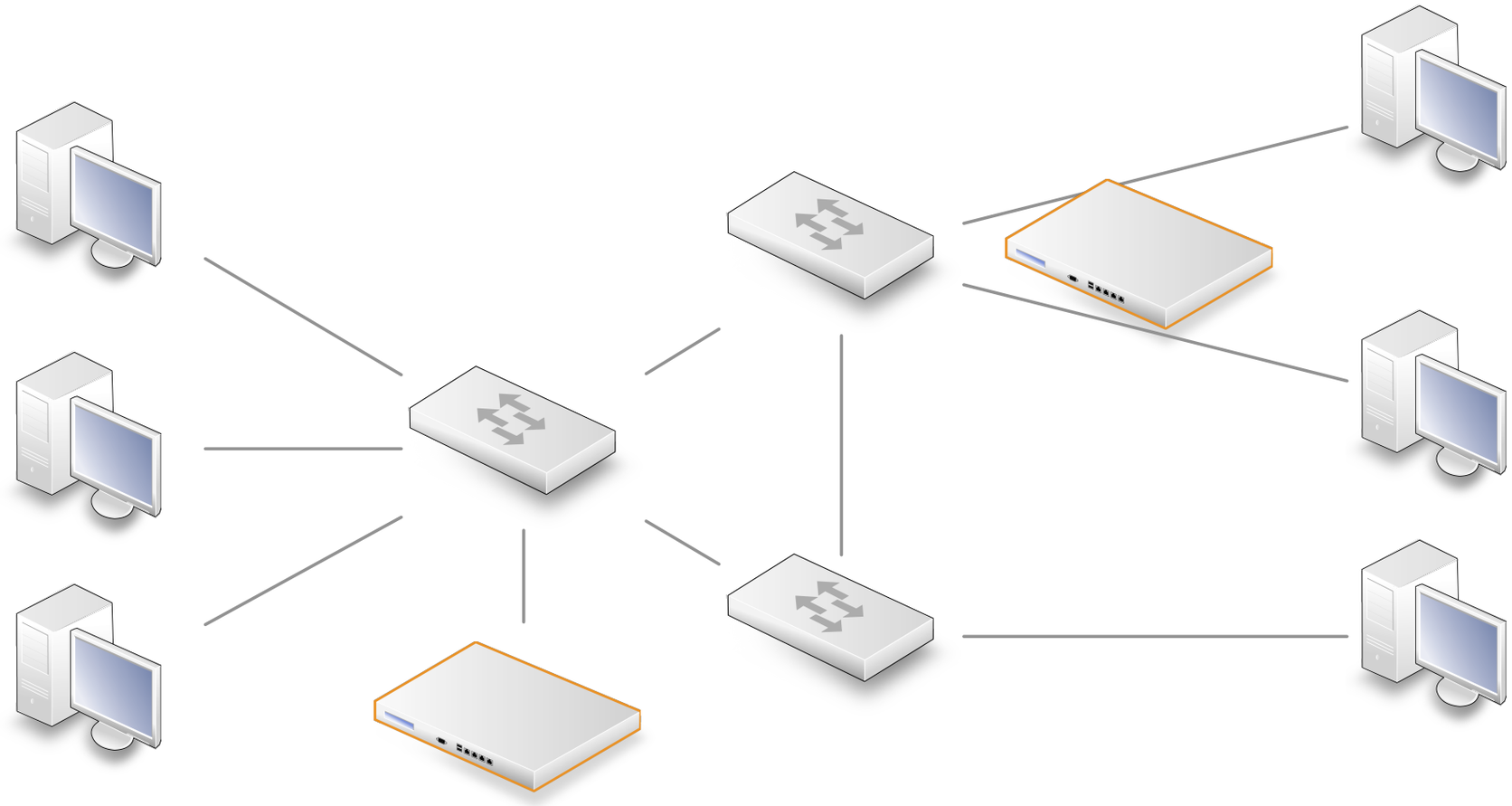
Policy Routing using Process-Level Identifiers

IEEE International Symposium on Software Defined Systems
April 4th, 2016, Berlin, Germany

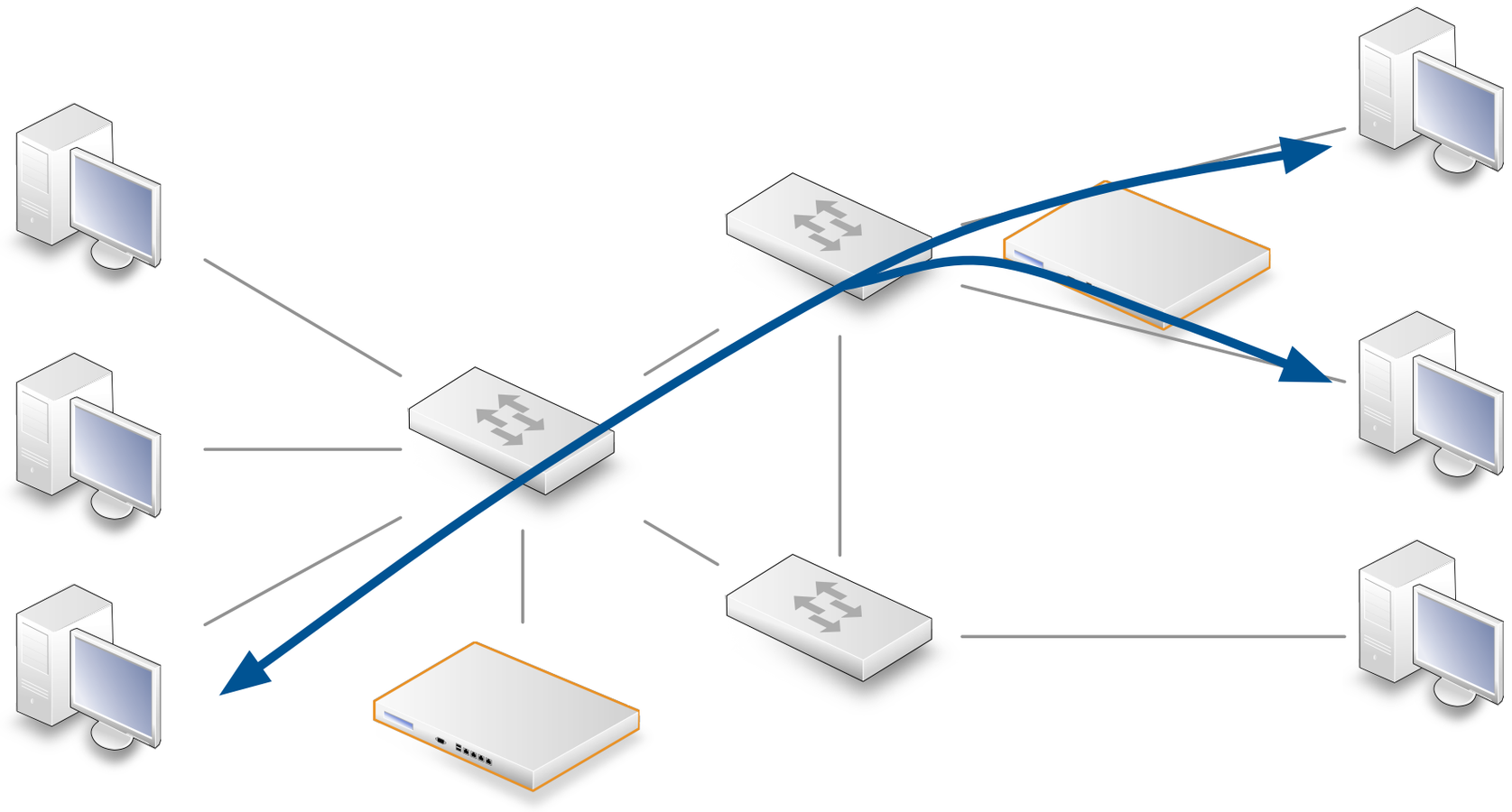
Oliver Michel, Eric Keller
Networking and Security Research Group

University of Colorado
Boulder

Network Policies

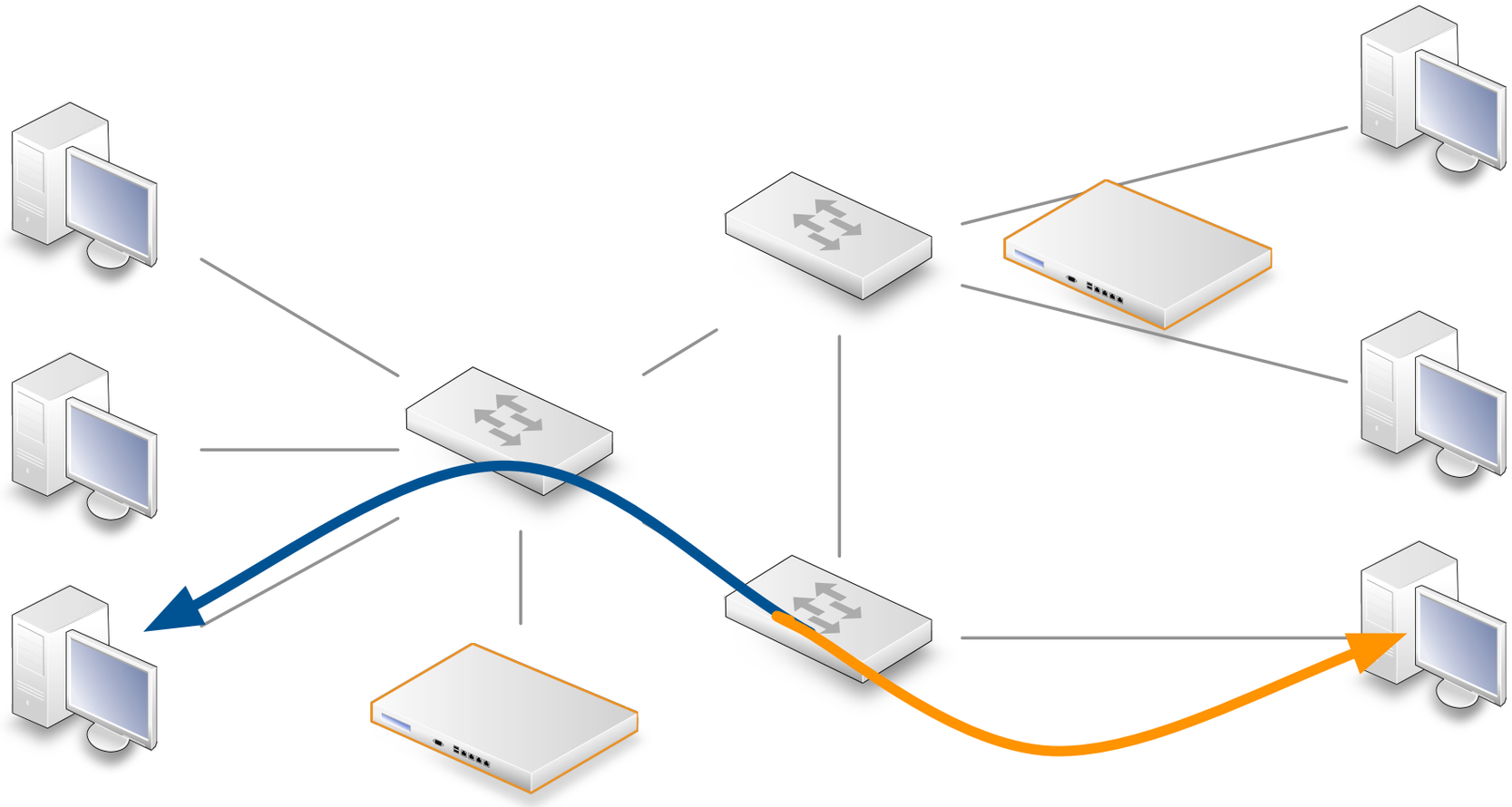


Network Policies



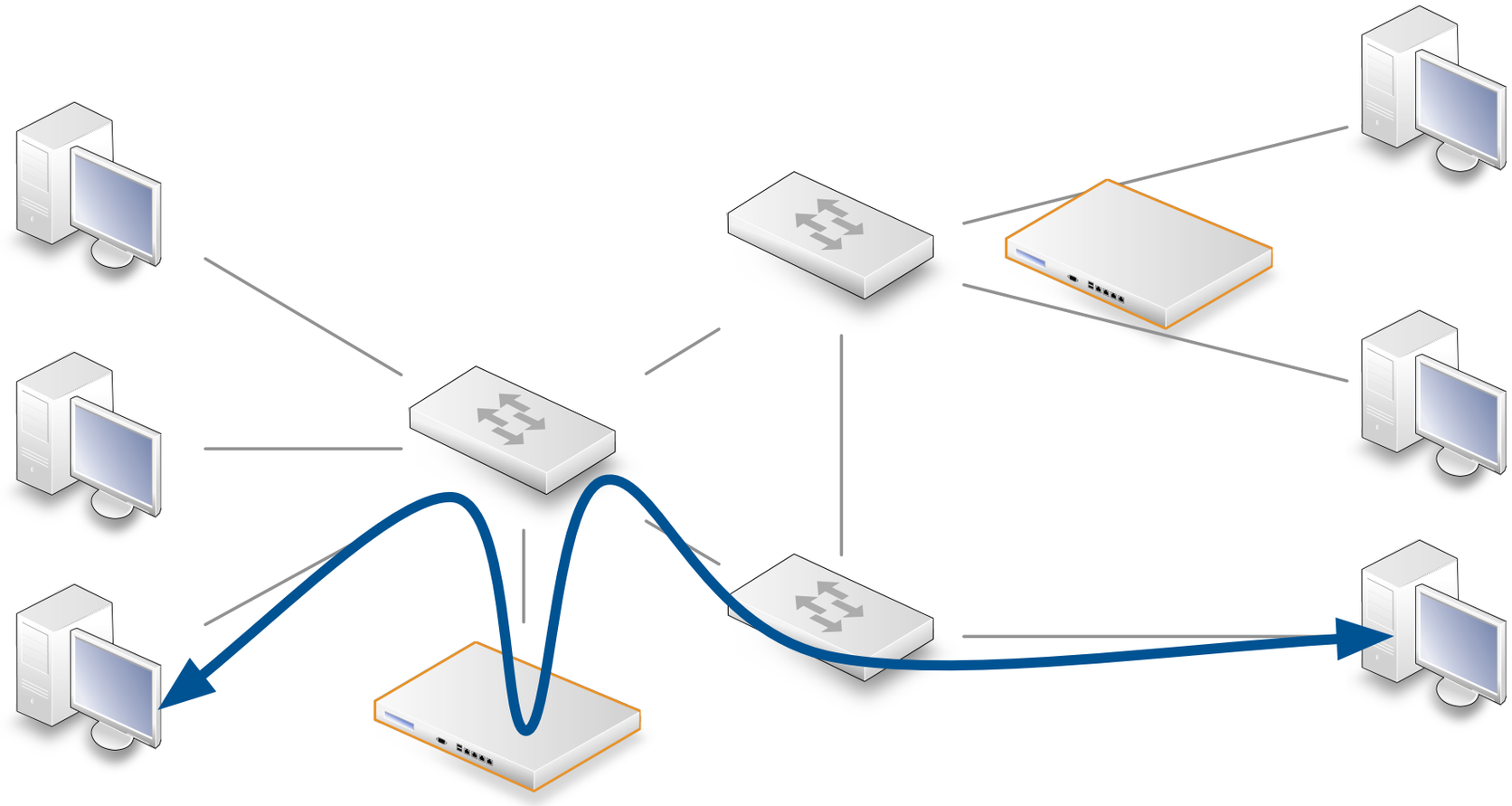
Load Balancing

Network Policies



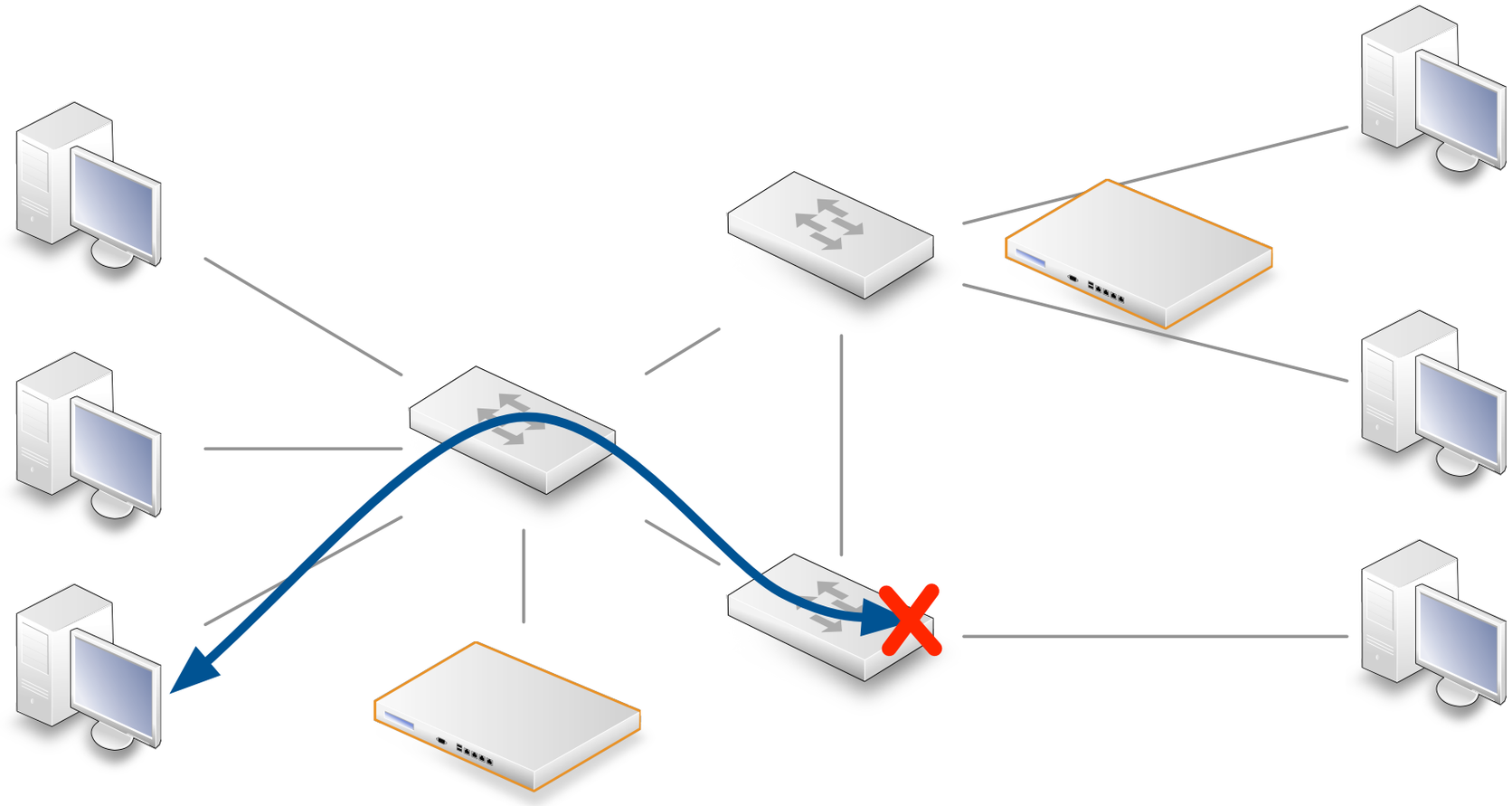
Address Translation

Network Policies



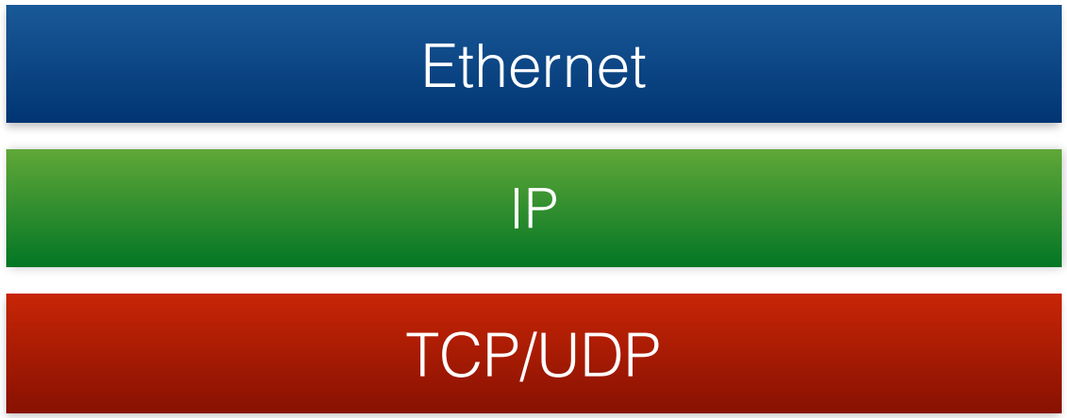
Intrusion Detection

Network Policies

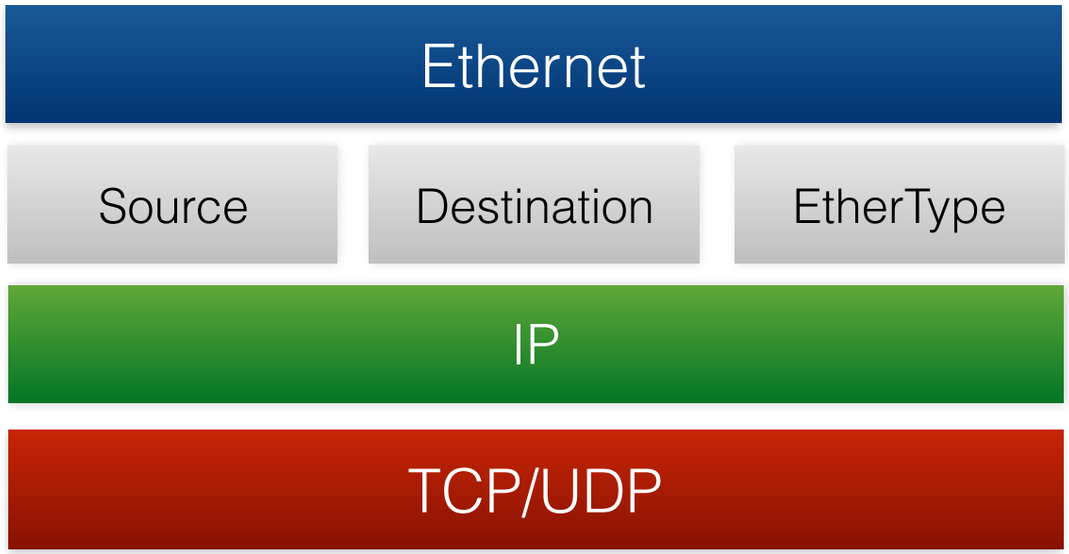


Firewalling

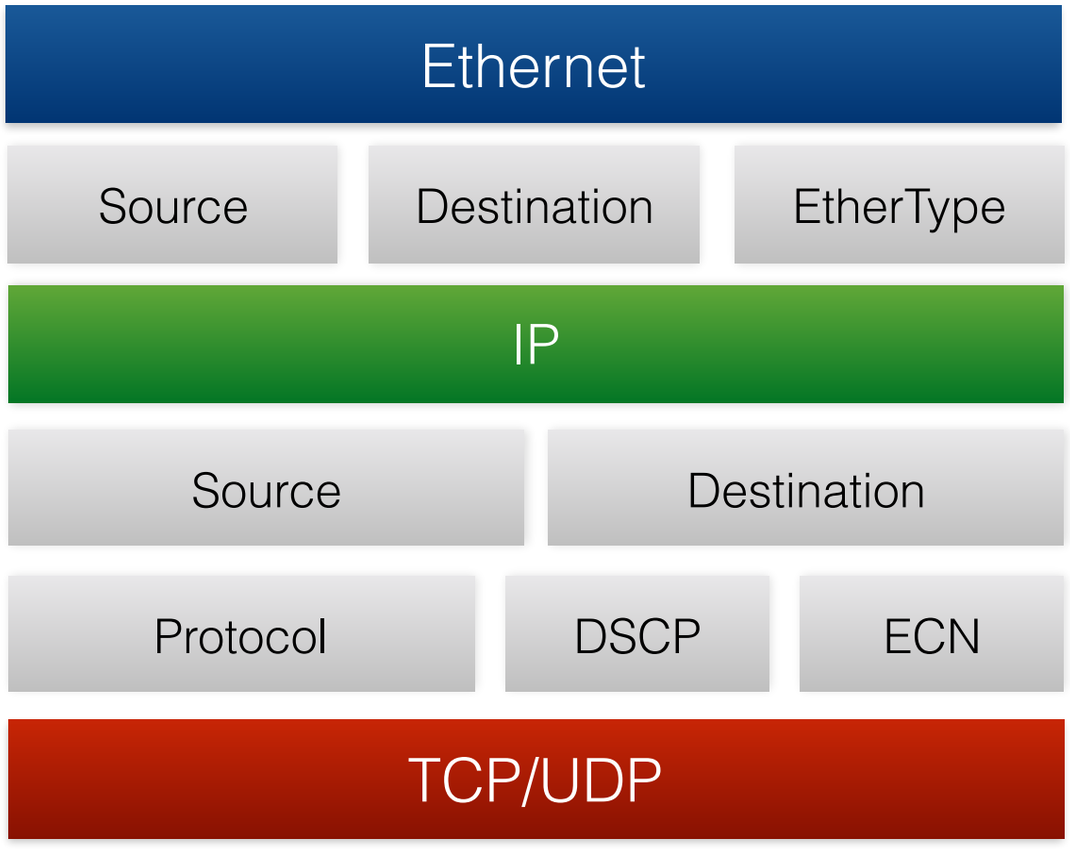
Limited Identifiers



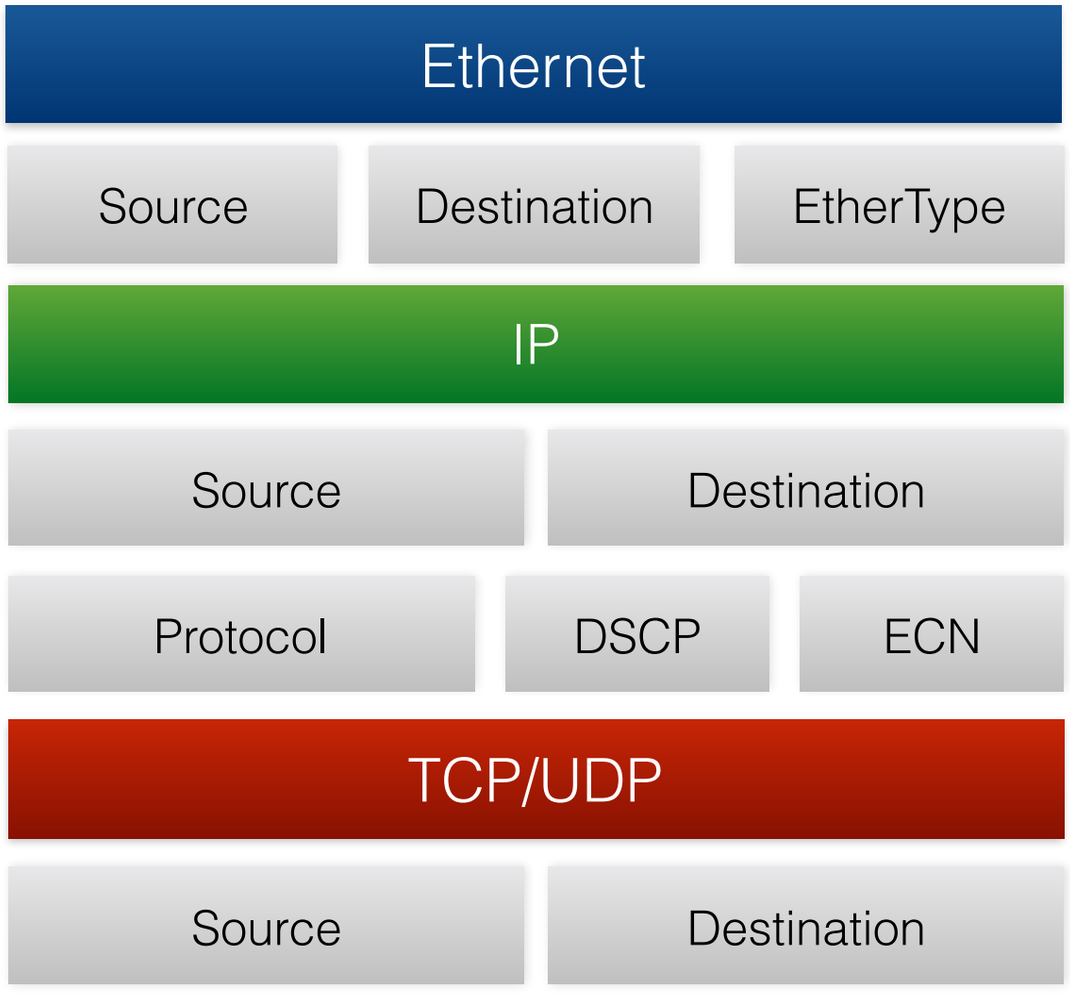
Limited Identifiers



Limited Identifiers

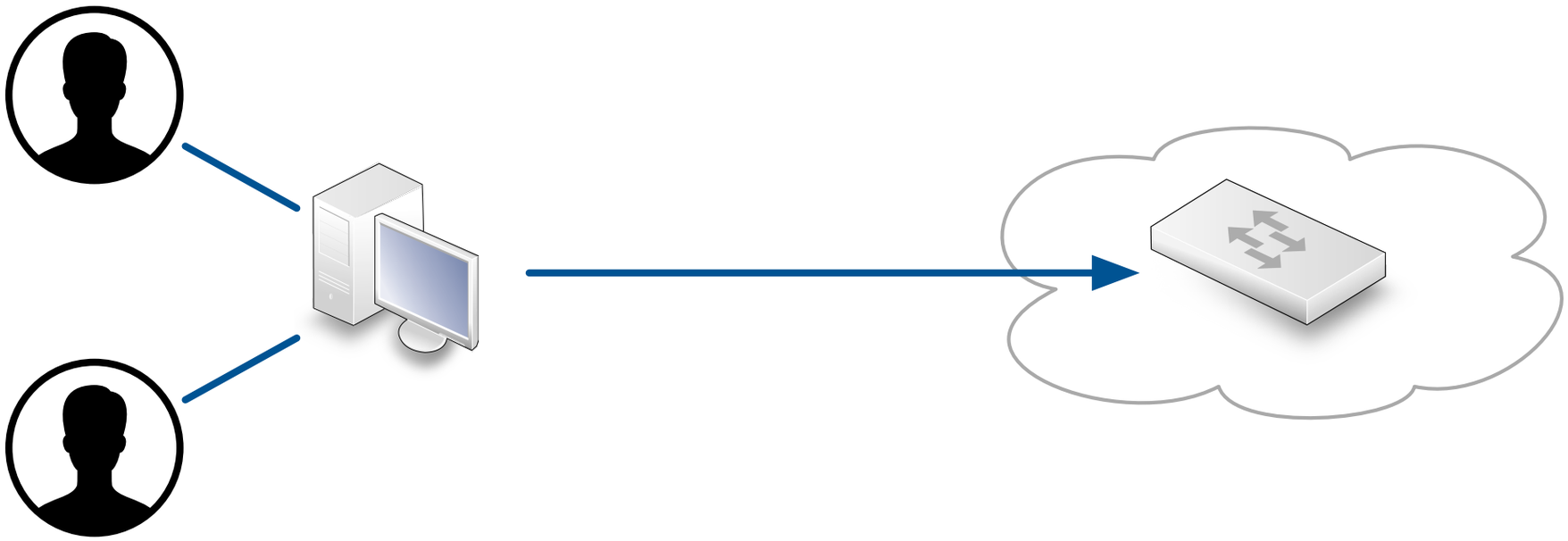


Limited Identifiers



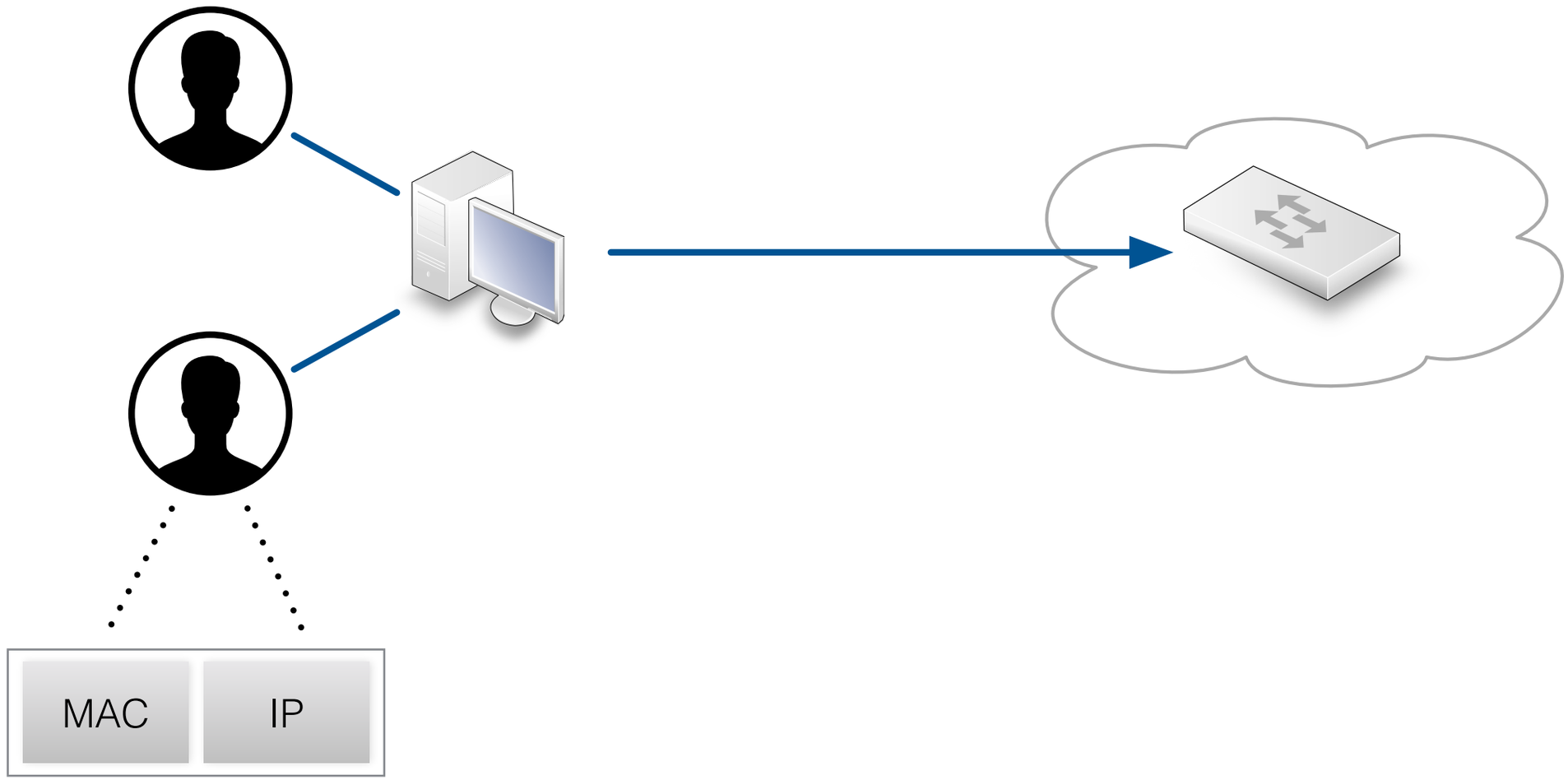
Why fine-grained identifiers?

- Uniquely identifying user sessions



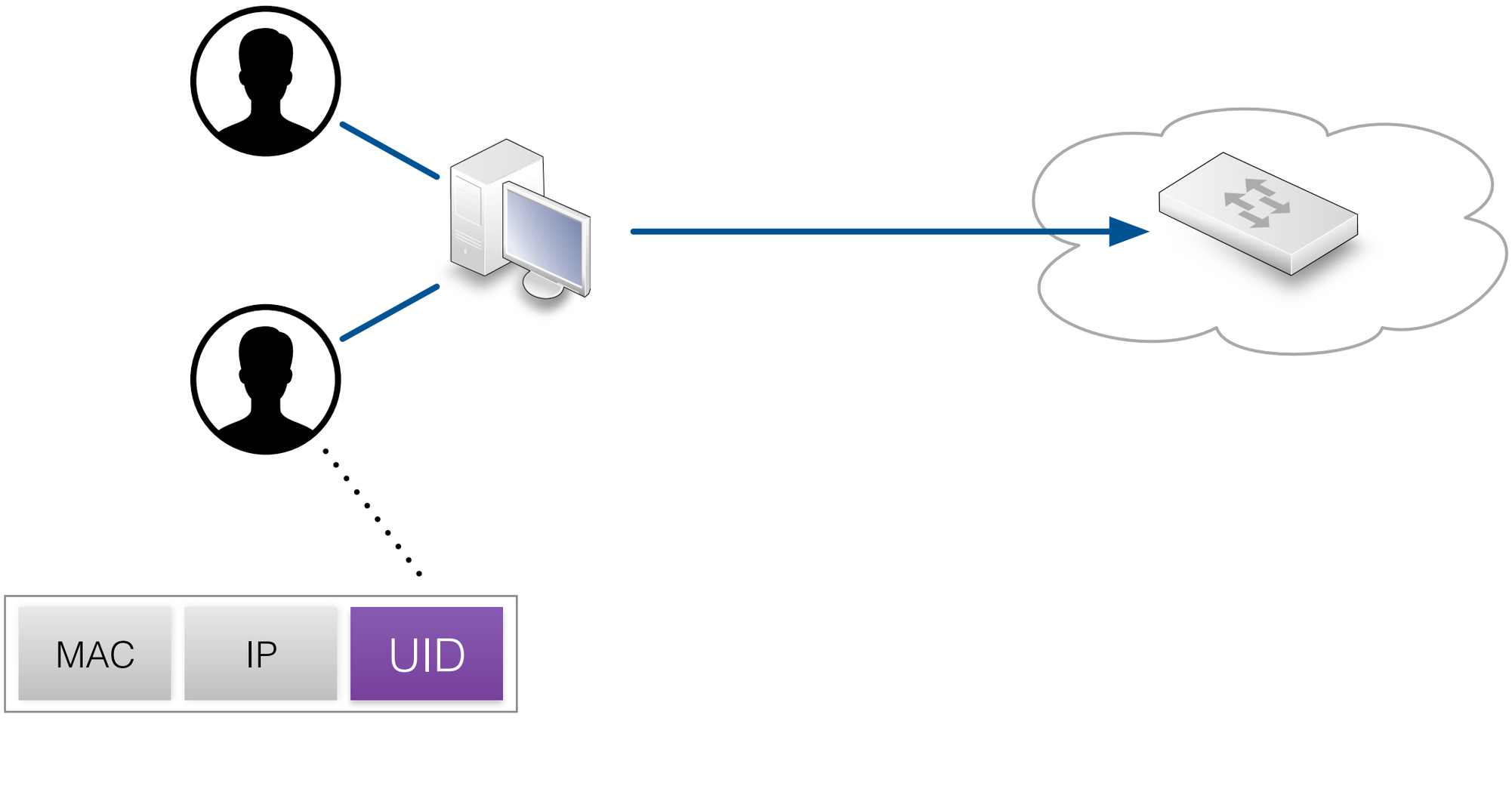
Why fine-grained identifiers?

- Uniquely identifying user sessions



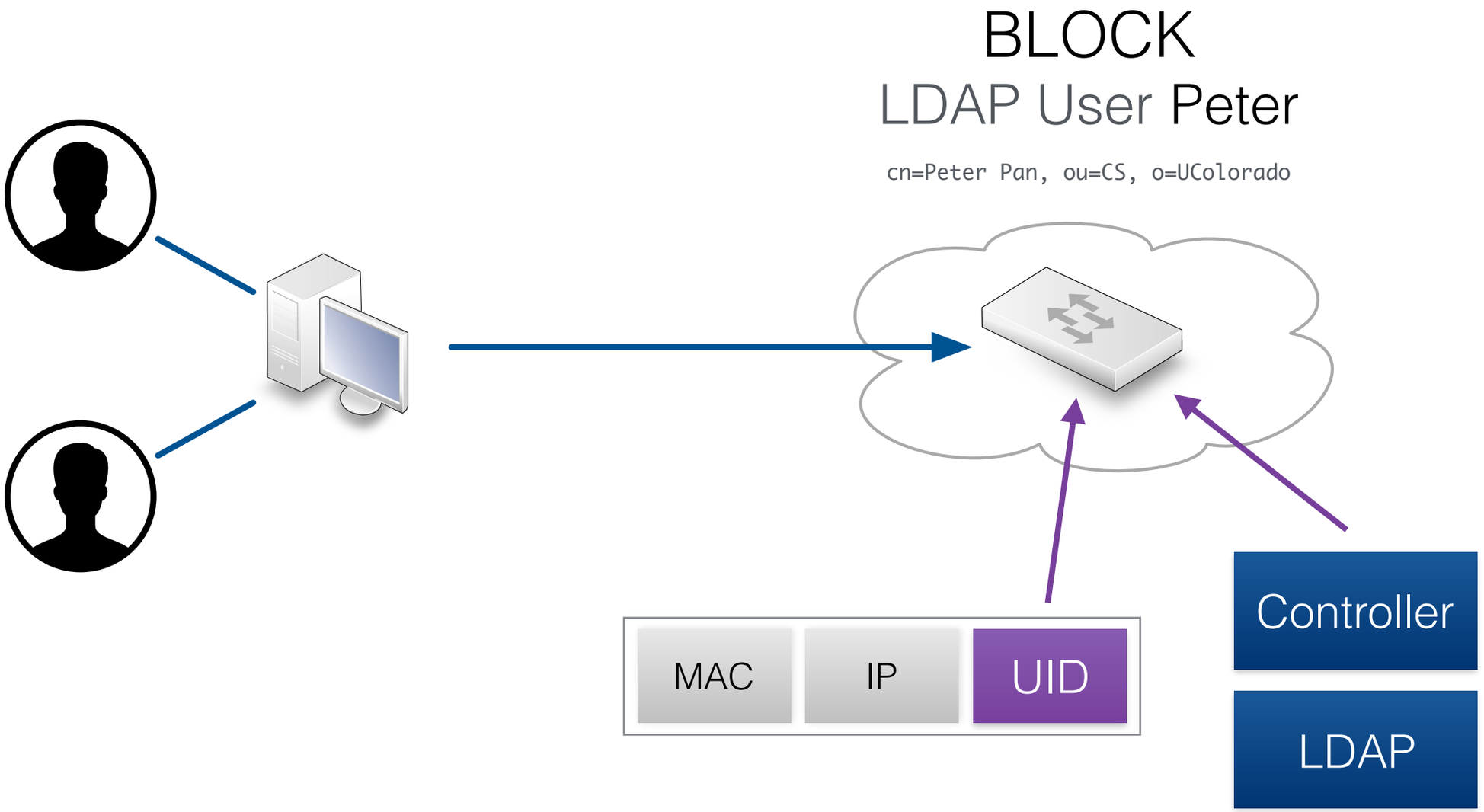
Why fine-grained identifiers?

- Uniquely identifying user sessions



Why fine-grained identifiers?

- Uniquely identifying user sessions



Why fine-grained identifiers?

- Isolating vulnerable software

Vulnerable source packages in the stable suite

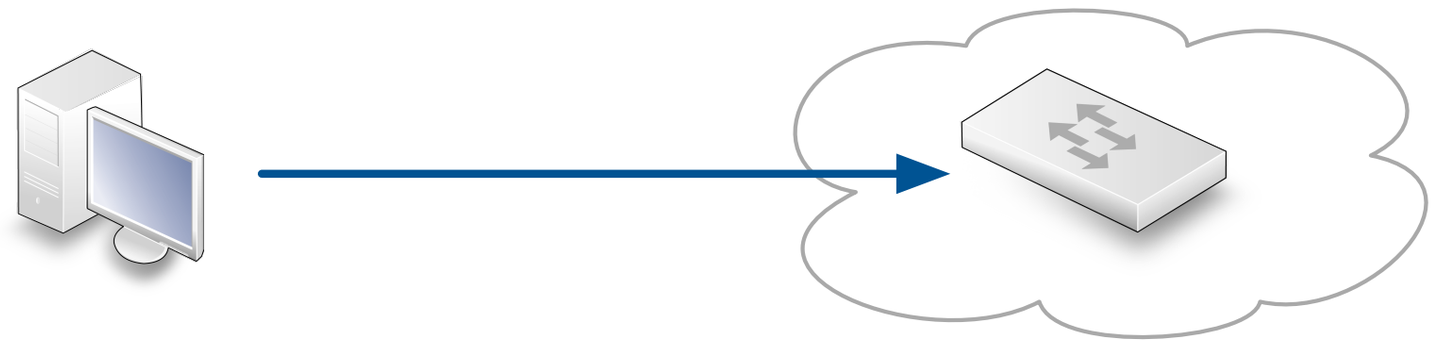
debian

✓ high ✓ medium ✓ low ☐ unimportant ✓ not yet assigned ☐ end-of-life ☐ hide remote scope ☐ hide local scope
☐ hide unclear scope ☐ include issues to be checked (shown in purple) ☐ include issues tagged <no-dsa>

Package	Bug	Urgency	Remote
389-ds-base	CVE-2015-1854	not yet assigned	?
	CVE-2015-3230	high**	yes
asterisk	CVE-2015-3008	medium**	yes
botan1.10	CVE-2015-5726	not yet assigned	?
	CVE-2015-5727	not yet assigned	?
	CVE-2015-7827	not yet assigned	?
	CVE-2016-2194	not yet assigned	?
	CVE-2016-2195	not yet assigned	?

Why fine-grained identifiers?

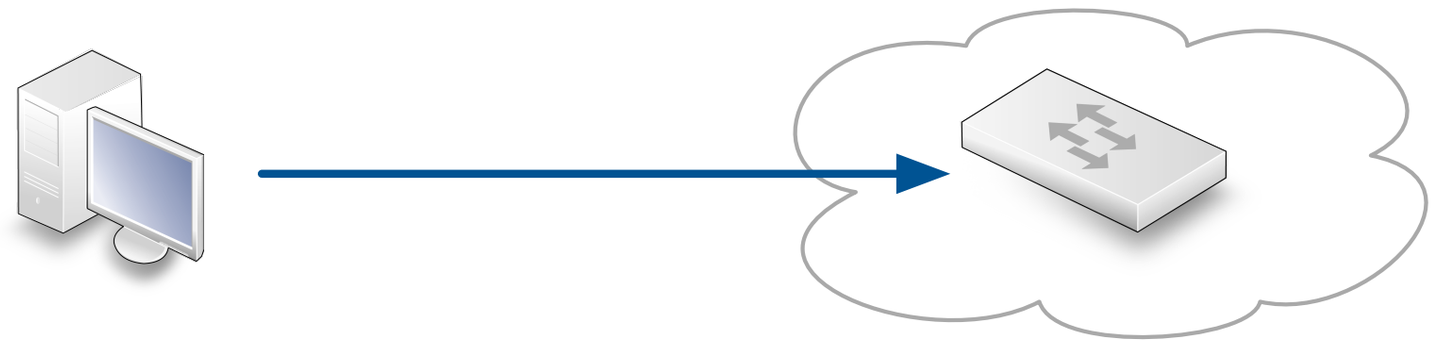
- Isolating vulnerable software



Why fine-grained identifiers?

- Isolating vulnerable software

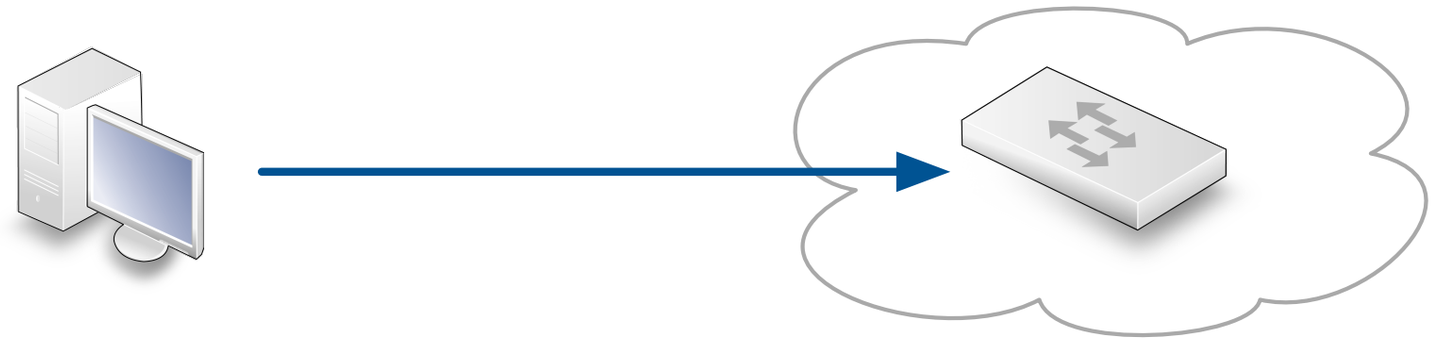
```
$ openssl sha1 /usr/sbin/httpd
SHA1(/usr/sbin/httpd)=5fdbdb587fce265656fd3e2960a6293262efedb7
```



Why fine-grained identifiers?

- Isolating vulnerable software

```
$ openssl sha1 /usr/sbin/httpd
SHA1(/usr/sbin/httpd)=5fdbdb587fce265656fd3e2960a6293262efedb7
```

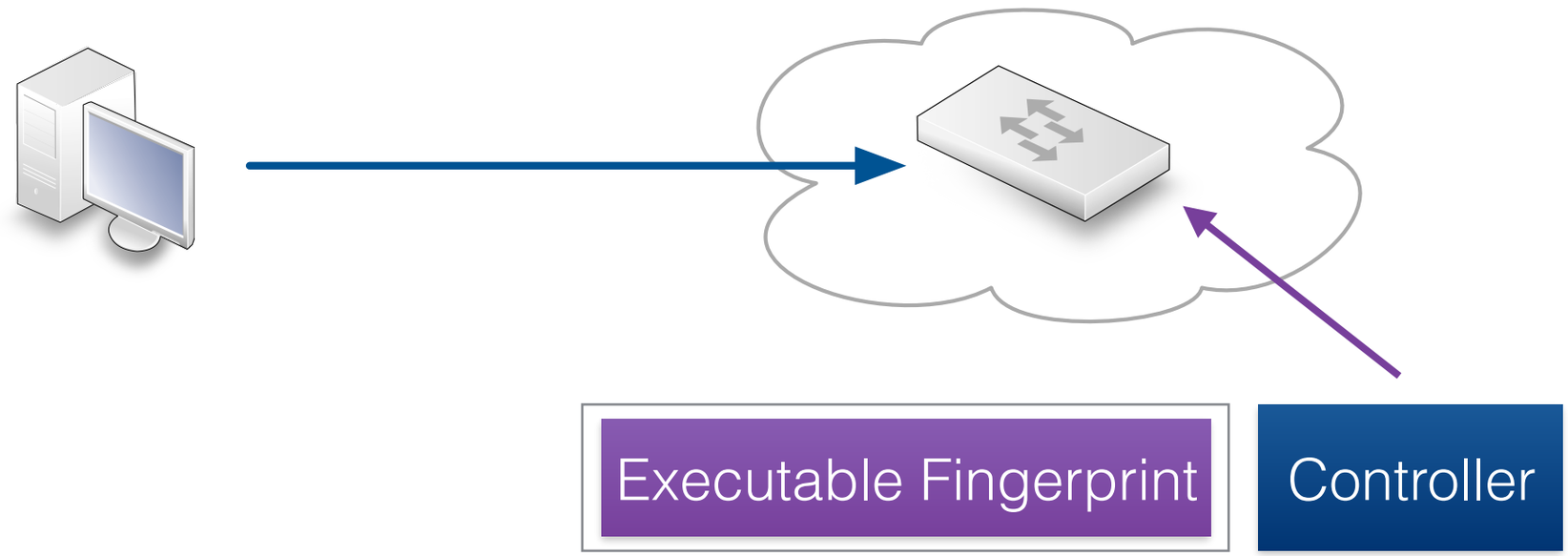


Executable Fingerprint

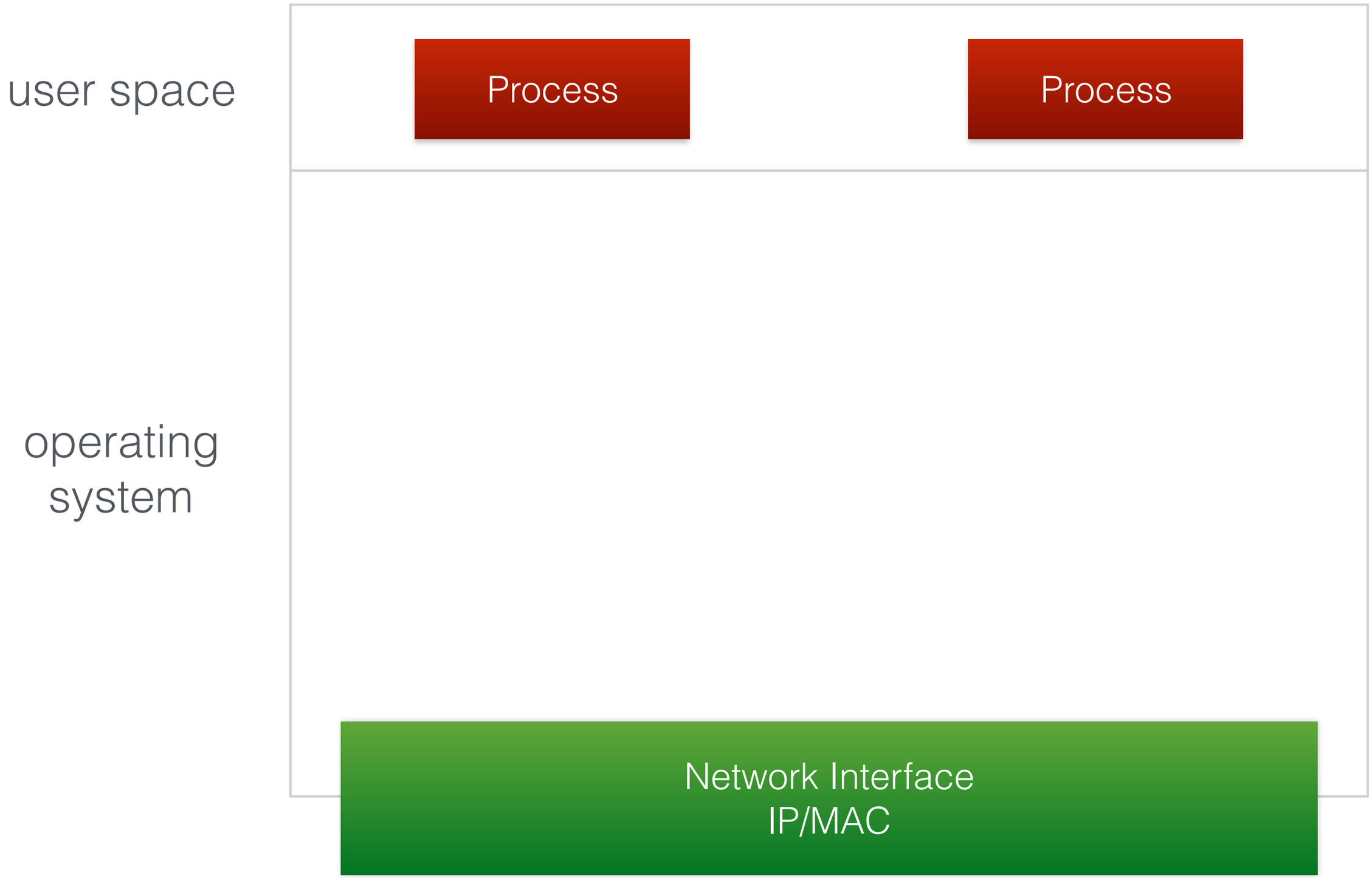
Why fine-grained identifiers?

- Isolating vulnerable software

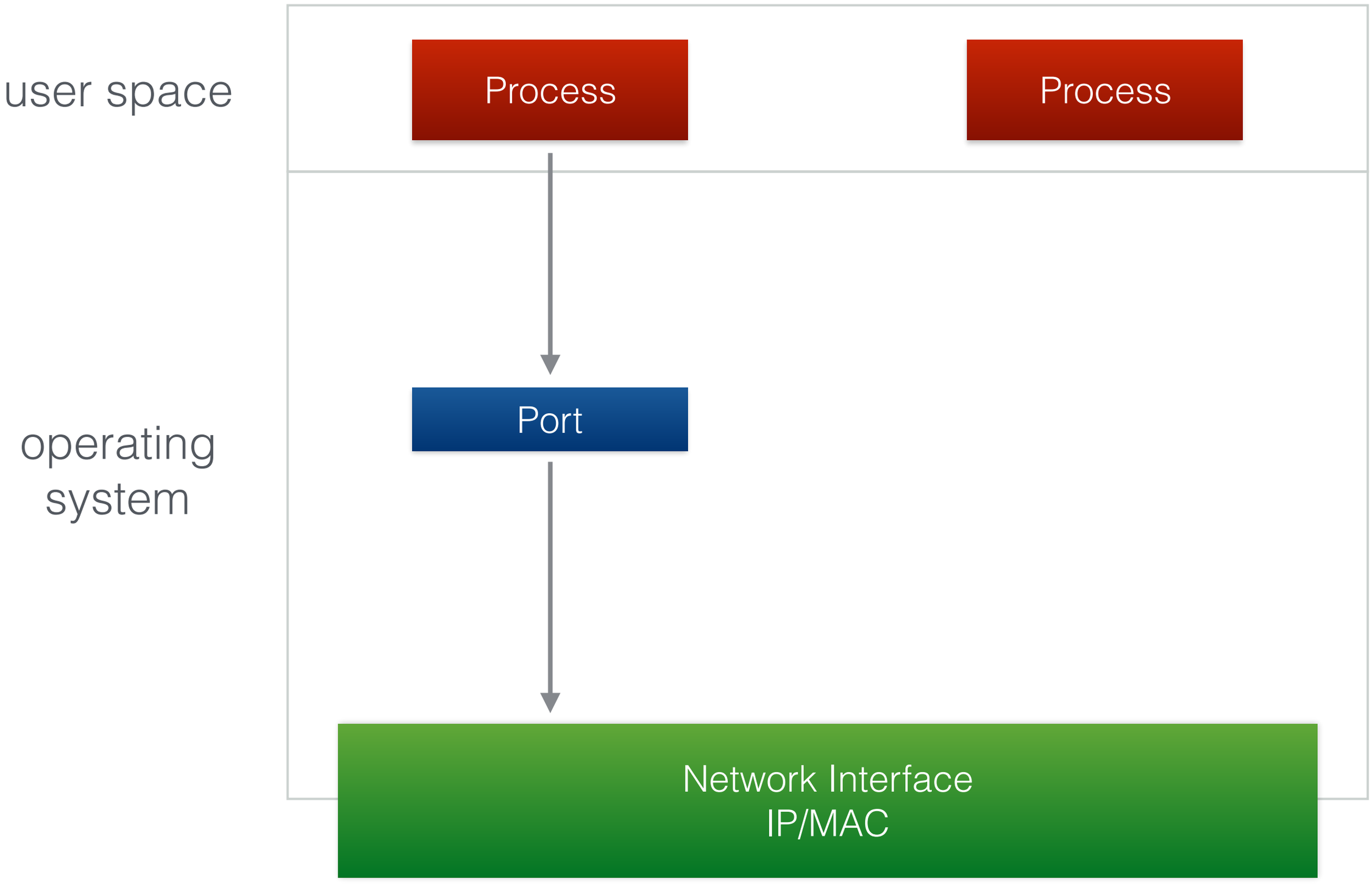
```
$ openssl sha1 /usr/sbin/httpd
SHA1(/usr/sbin/httpd)=5fdbdb587fce265656fd3e2960a6293262efedb7
```



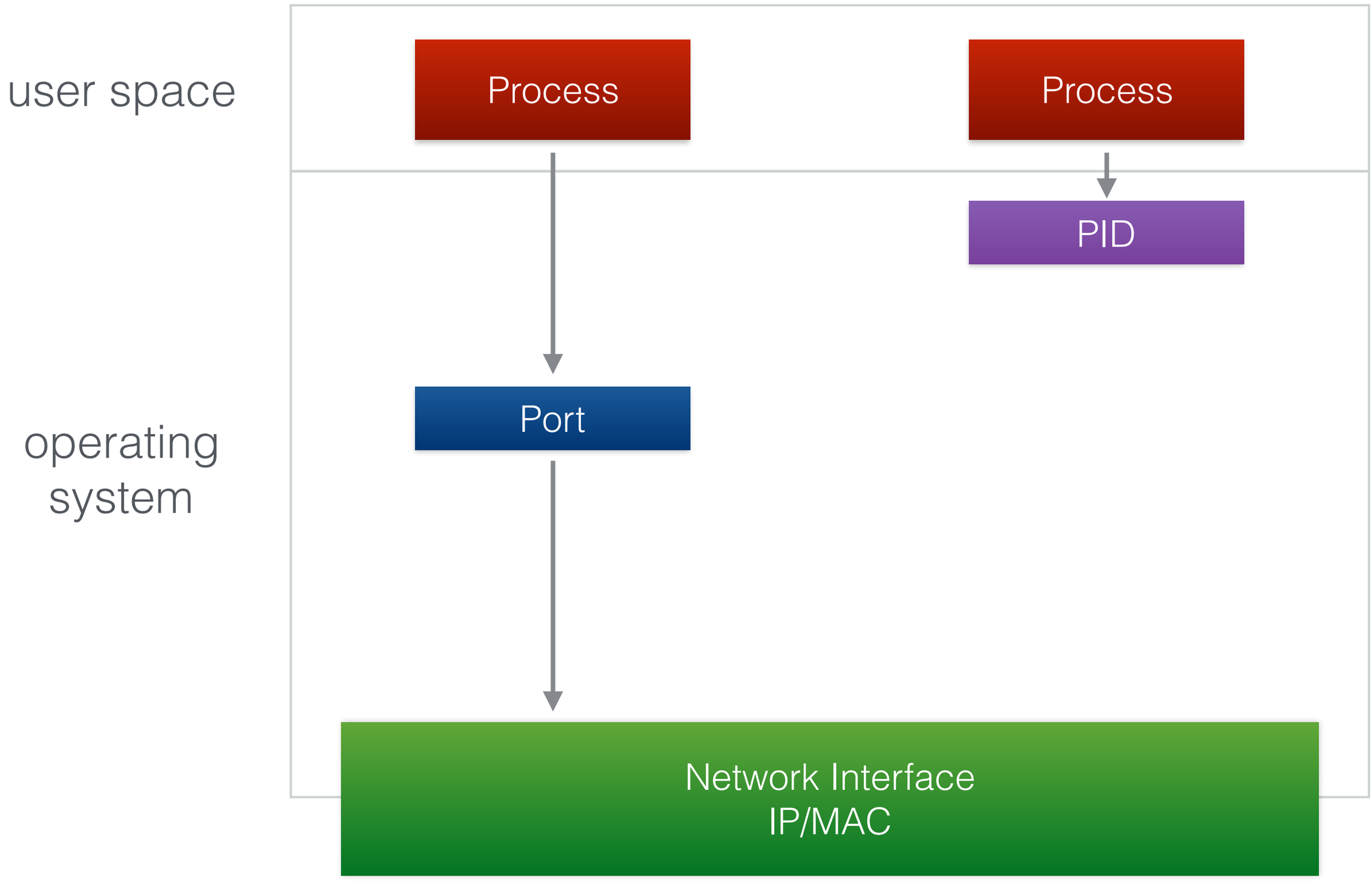
Fine-Grained Information



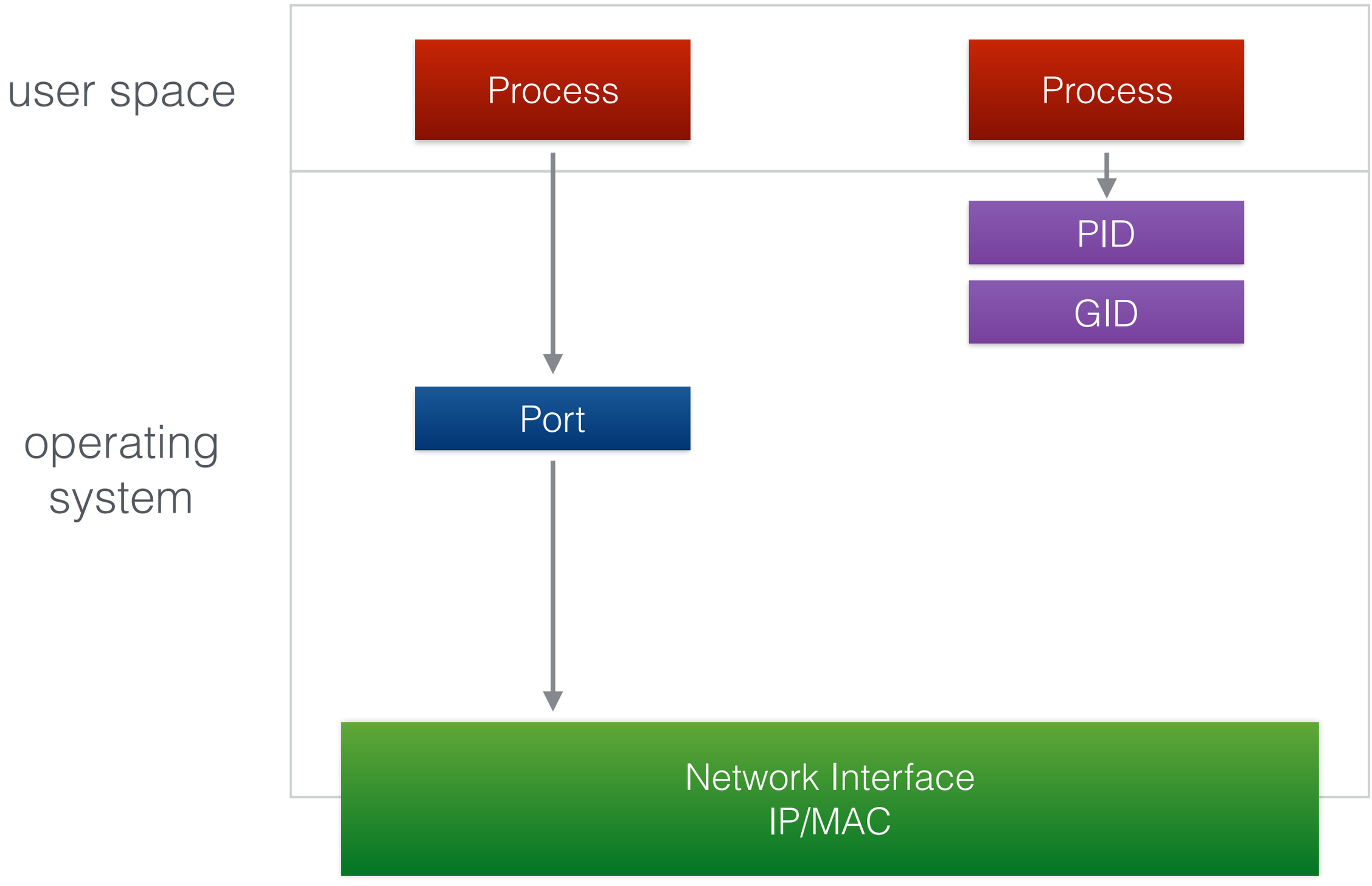
Fine-Grained Information



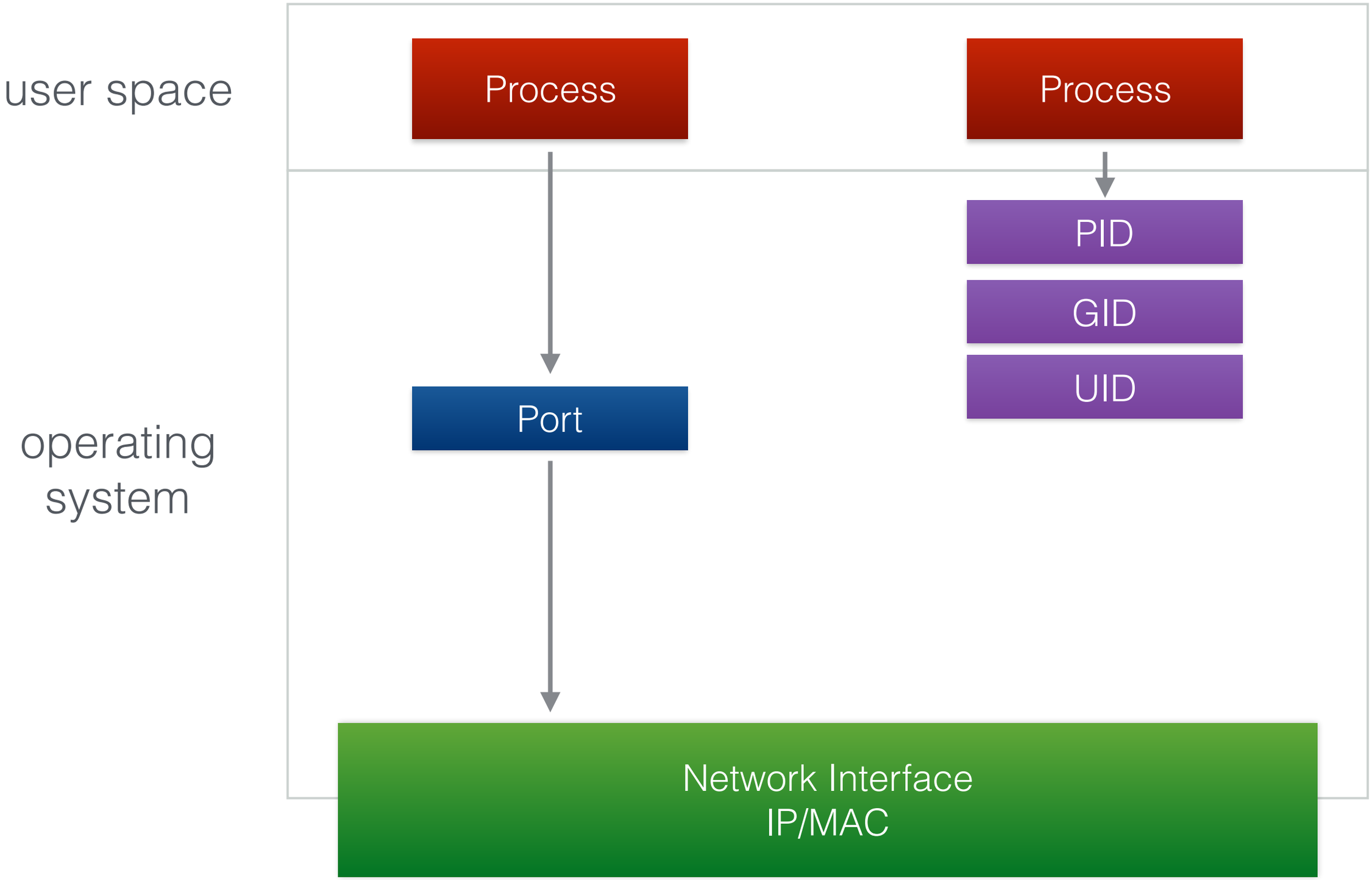
Fine-Grained Information



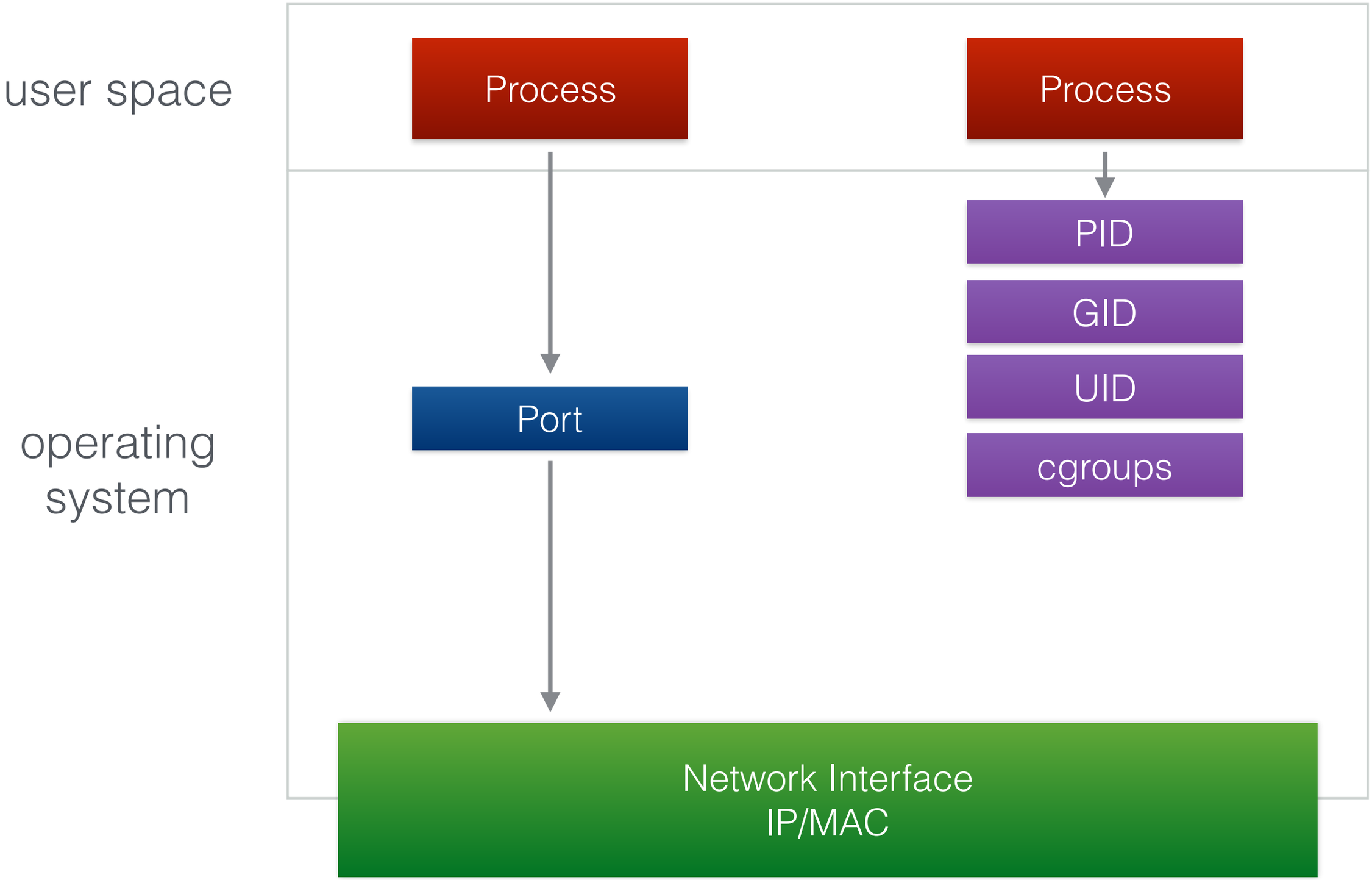
Fine-Grained Information



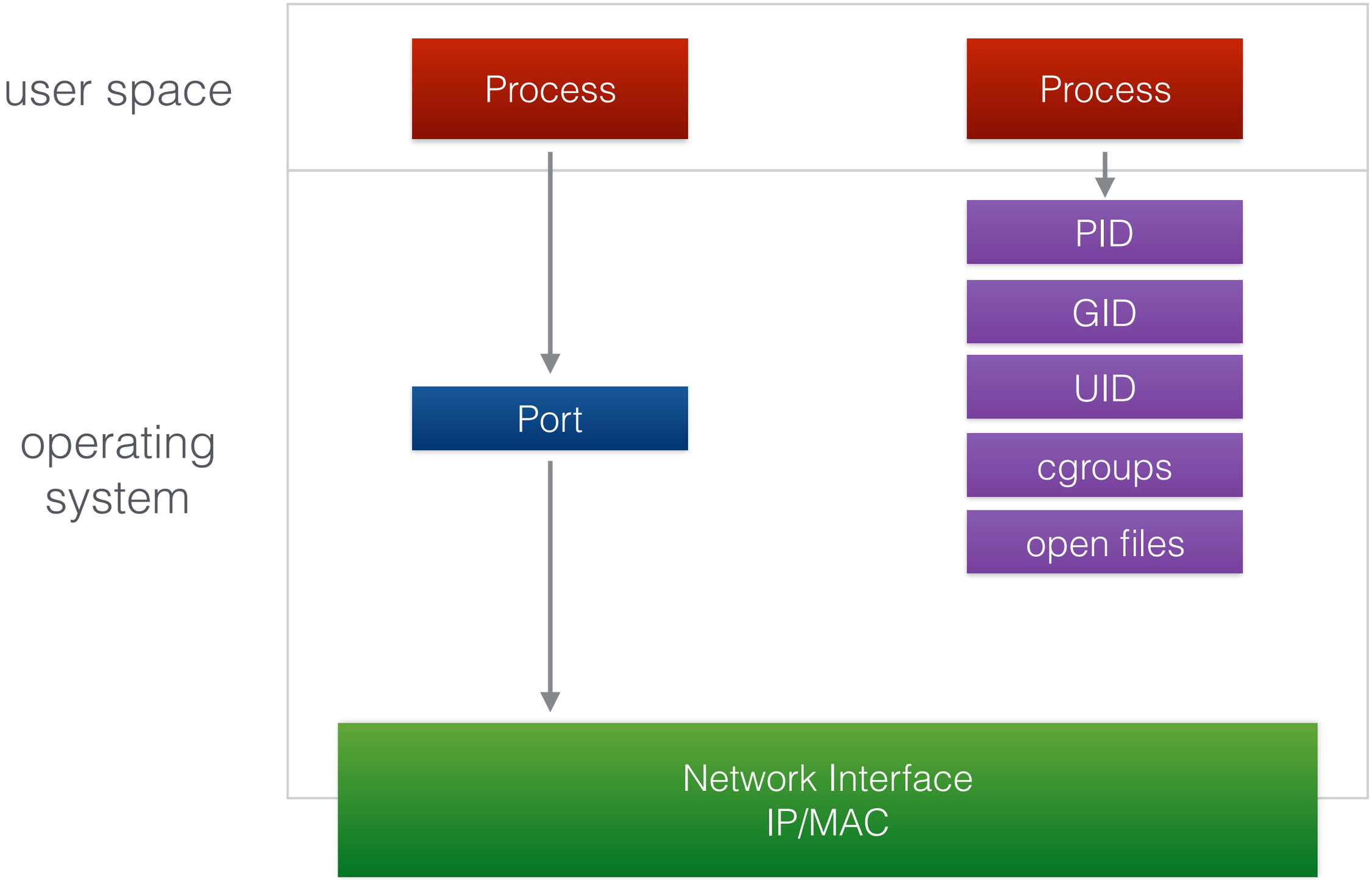
Fine-Grained Information



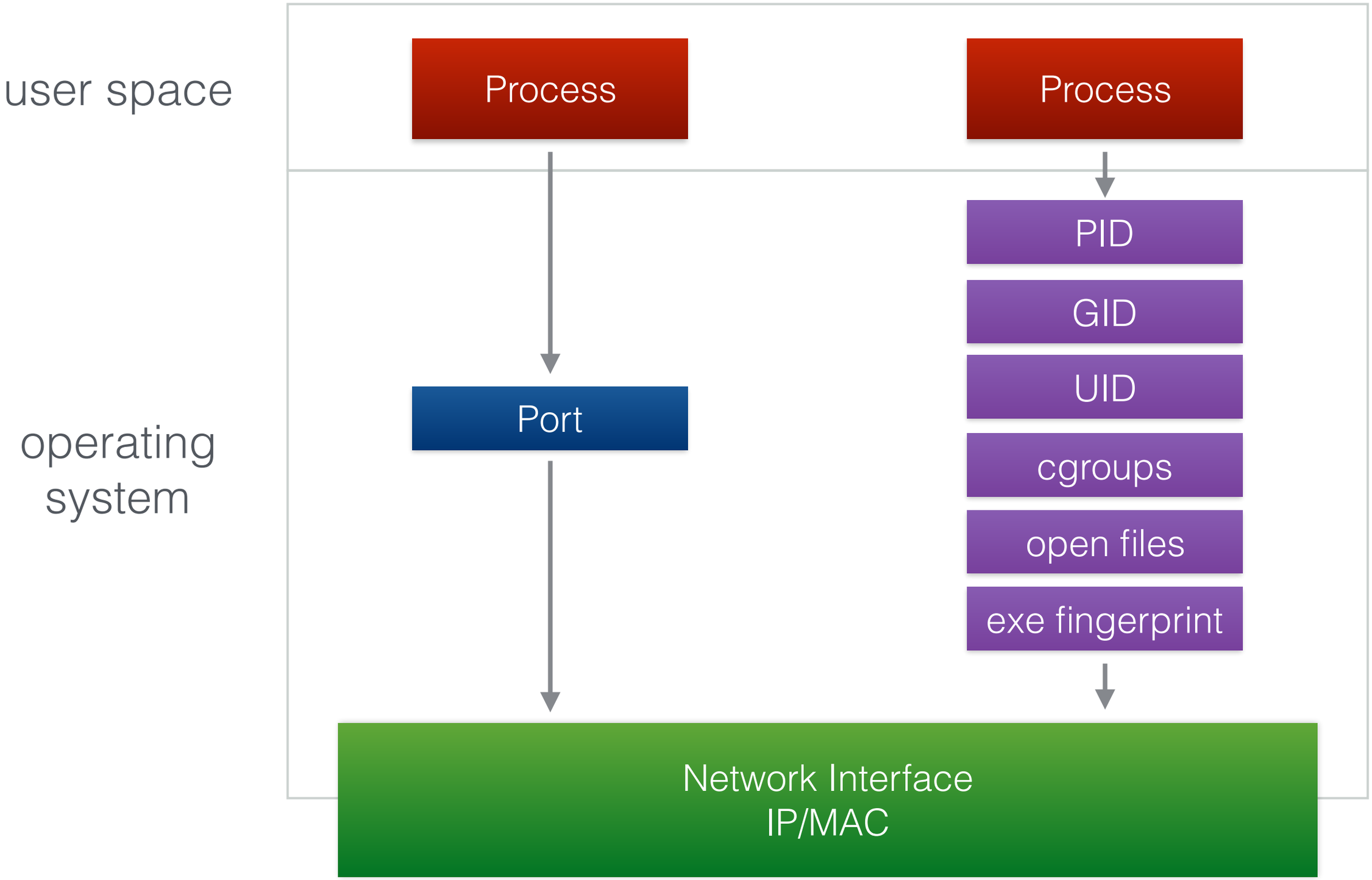
Fine-Grained Information



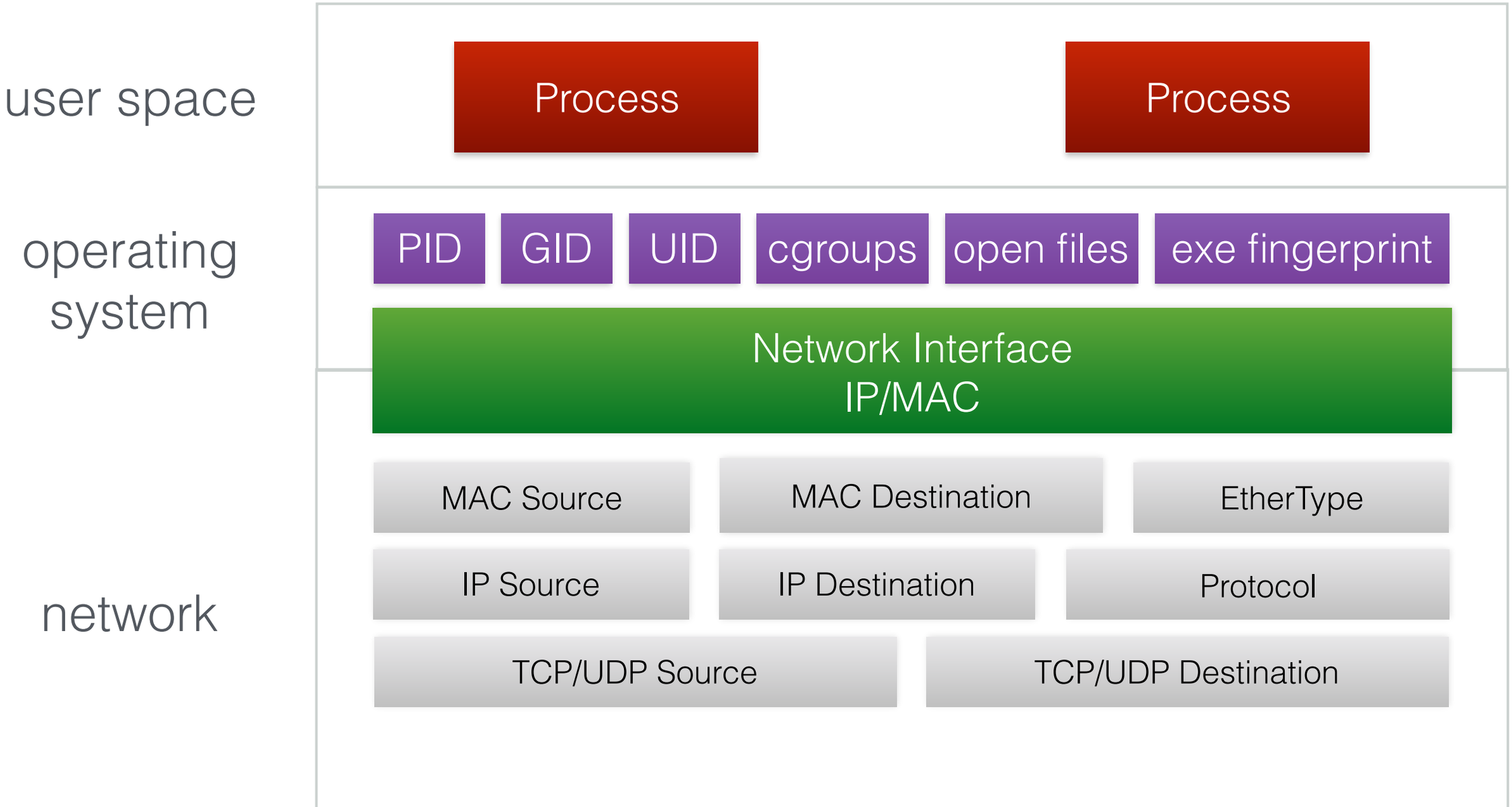
Fine-Grained Information



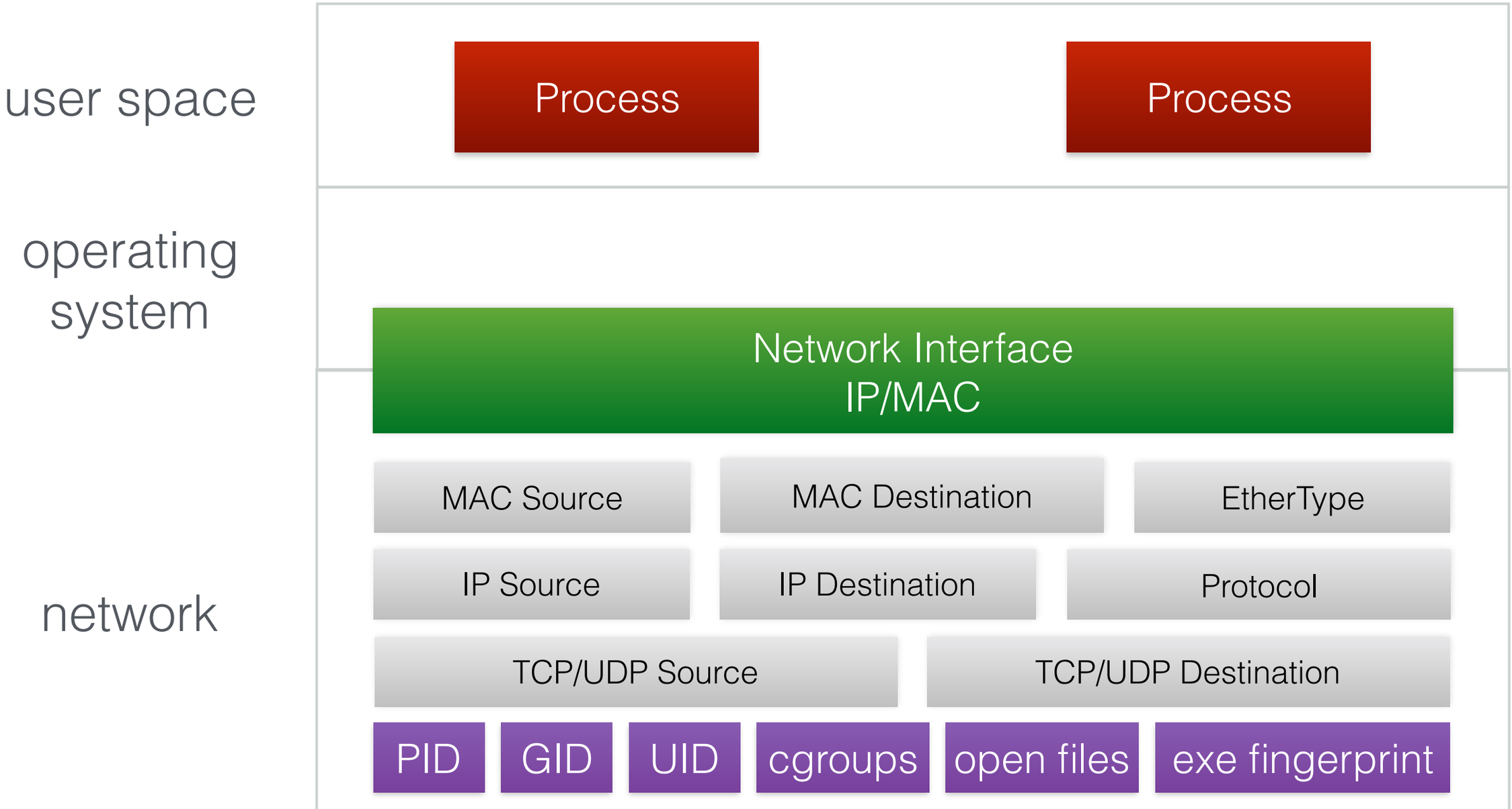
Fine-Grained Information



Fine-Grained Information



Fine-Grained Information



Benefiting Scenarios

- Uniquely identifying user sessions
- Isolating vulnerable software

Benefiting Scenarios

- Uniquely identifying user sessions
- Isolating vulnerable software
- Identifying services

Benefiting Scenarios

- Uniquely identifying user sessions
- Isolating vulnerable software
- Identifying services
- Quality of Service

Benefiting Scenarios

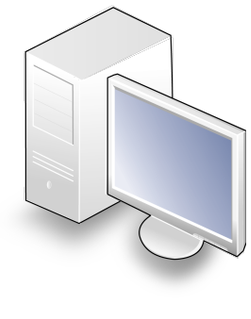
- Uniquely identifying user sessions
- Isolating vulnerable software
- Identifying services
- Quality of Service
- Forensic Analysis

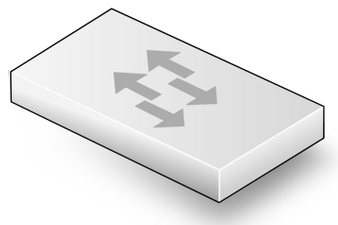
PRPL

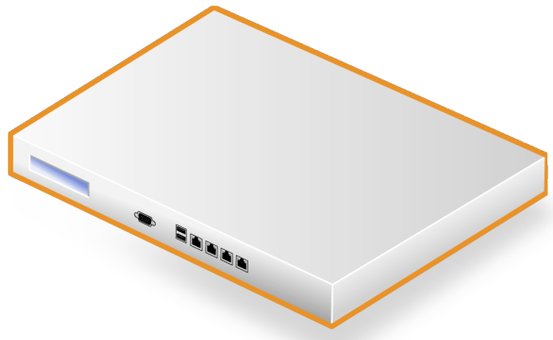
\ˈpər-pəl\

Policy Routing using Process-Level Identifiers

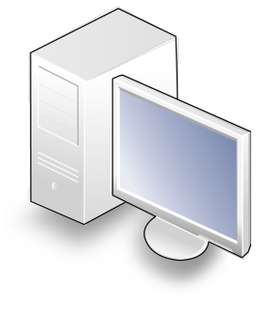
PRPL Overview

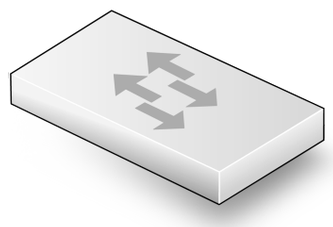




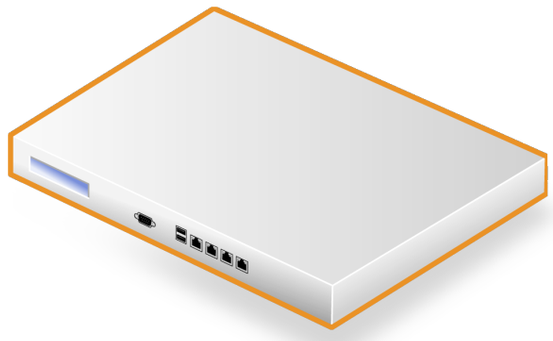


PRPL Overview

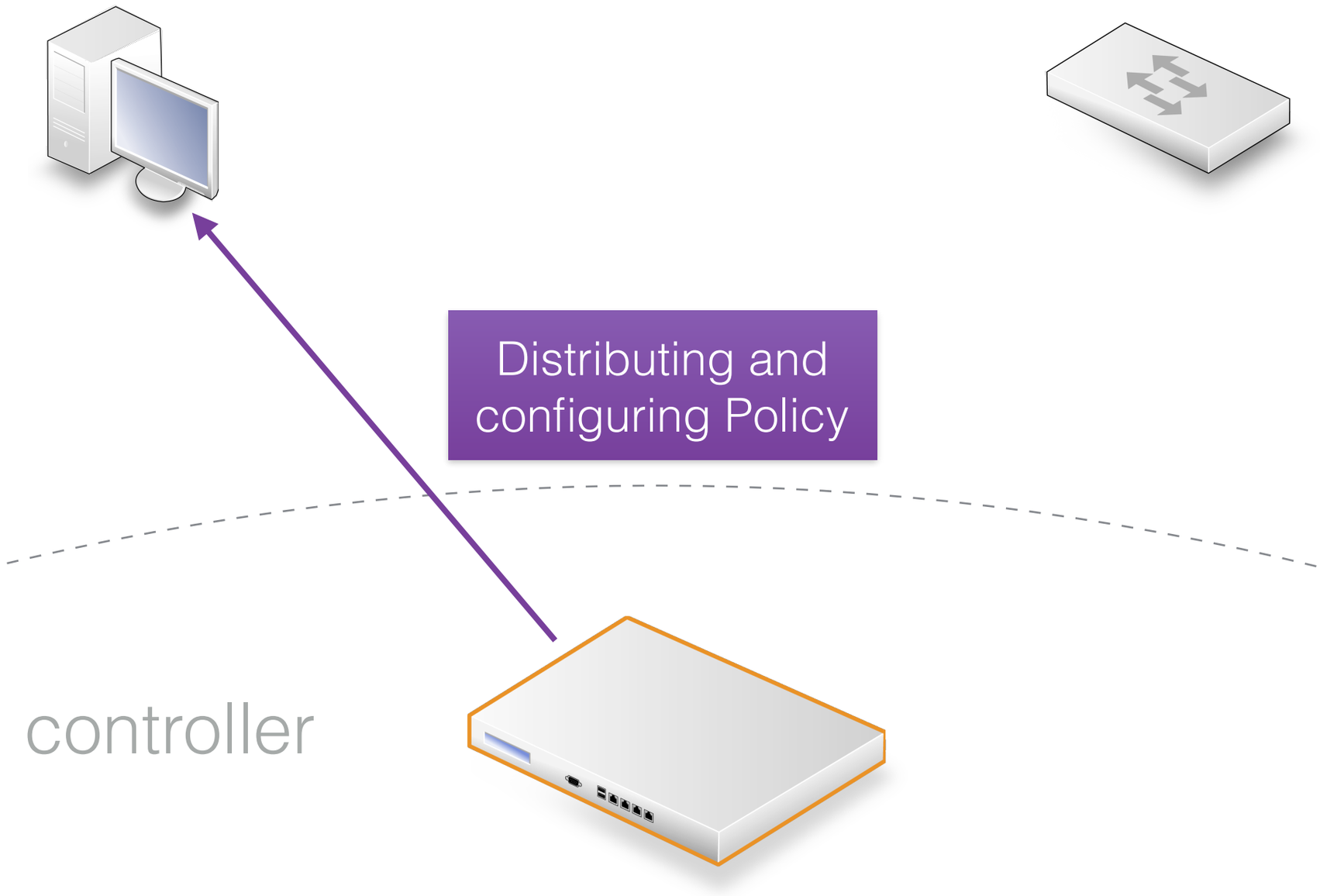




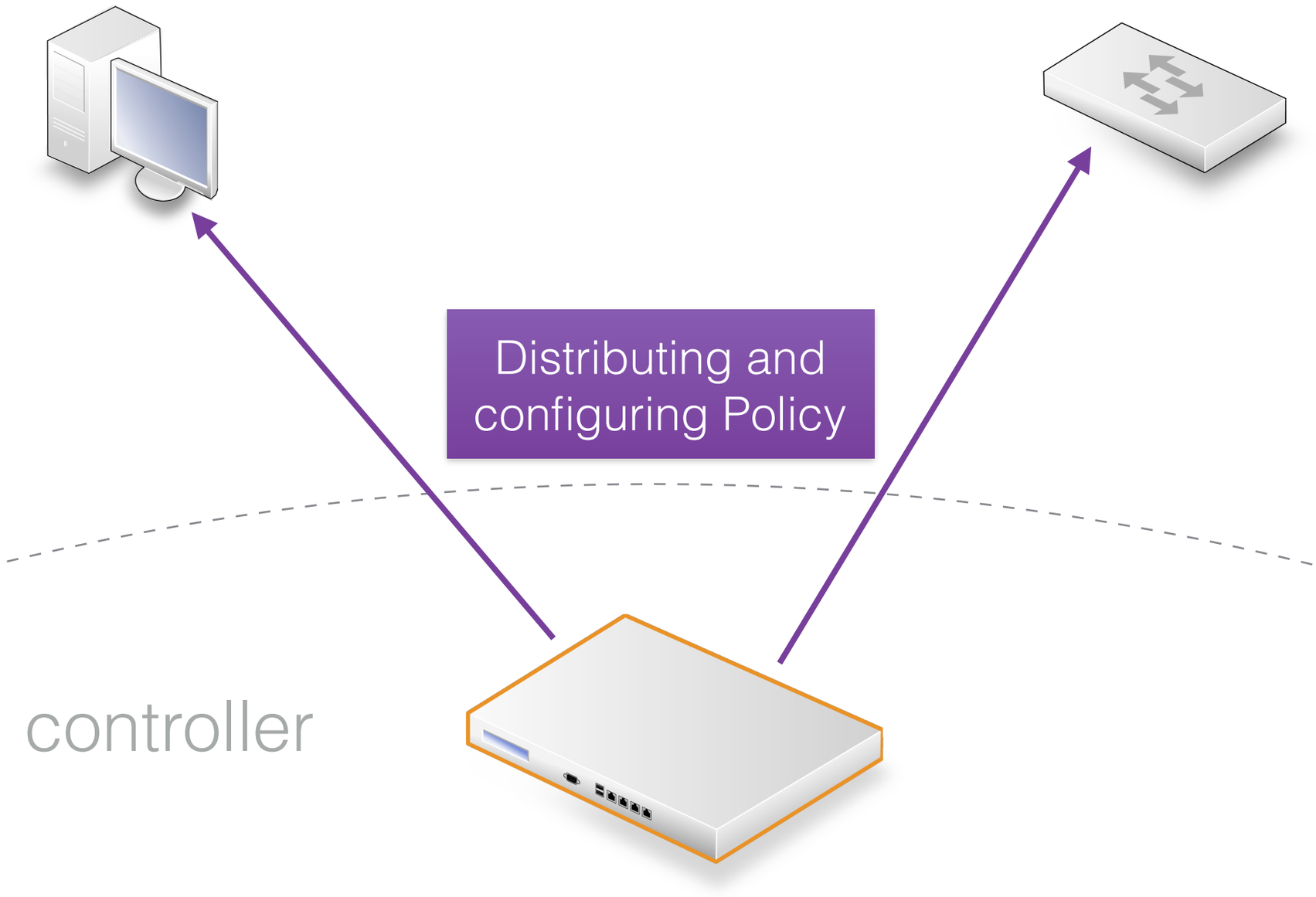
controller



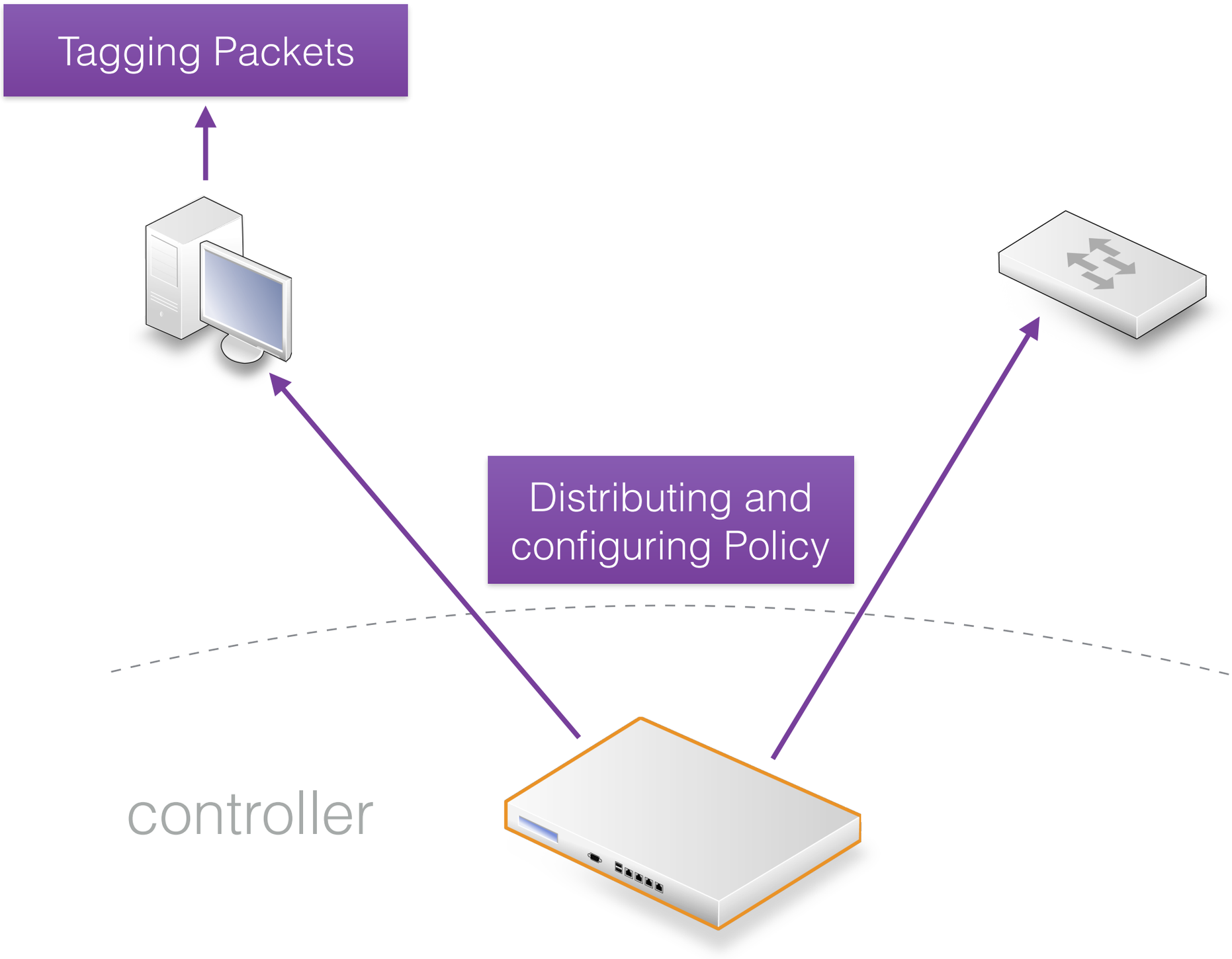
PRPL Overview



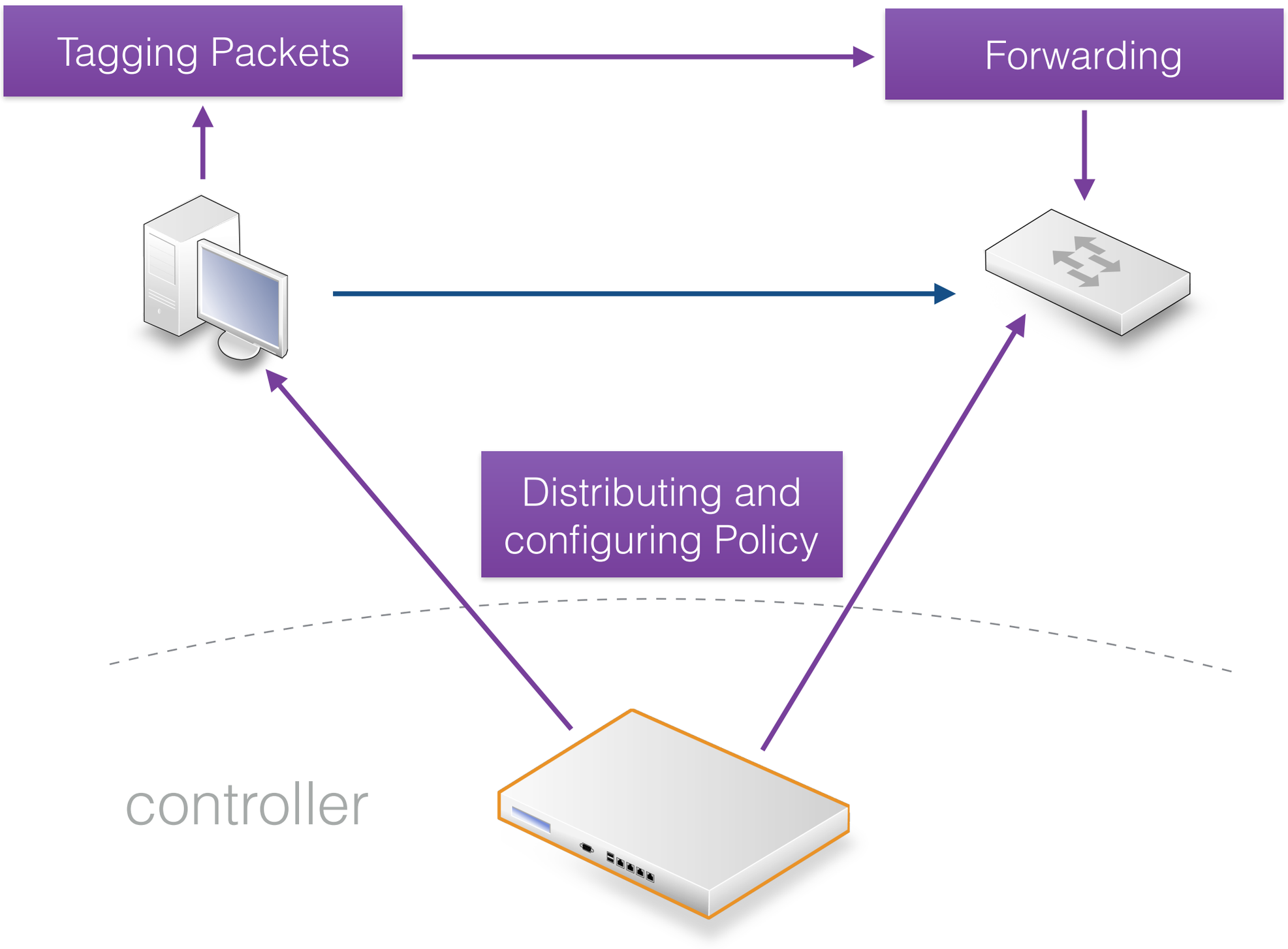
PRPL Overview



PRPL Overview

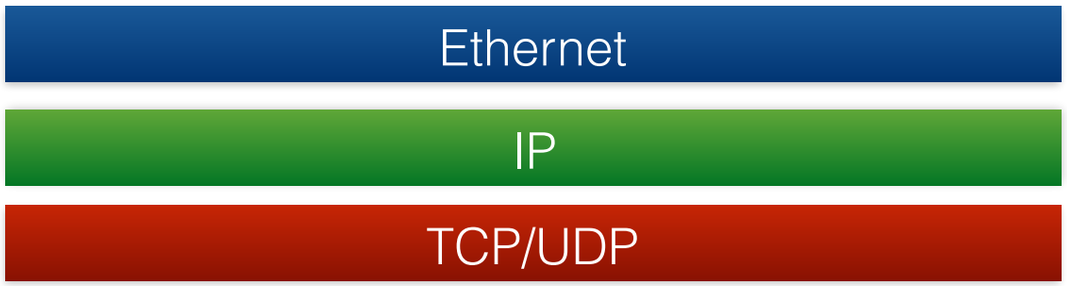


PRPL Overview



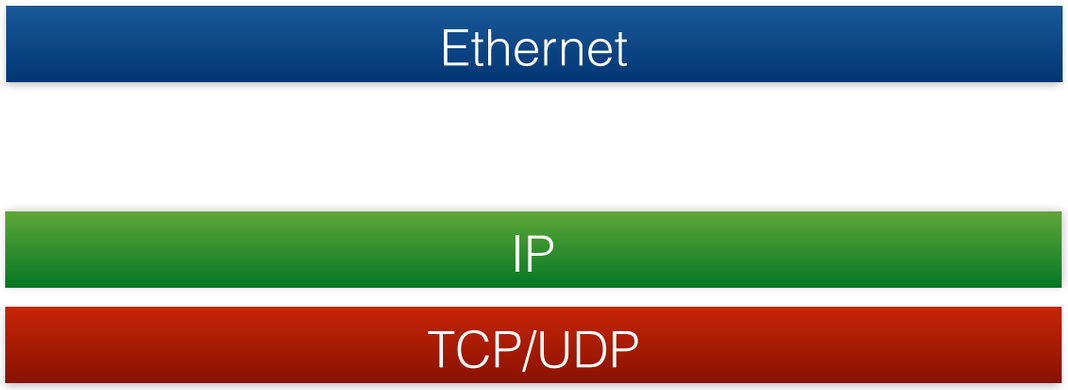
Tagging

- Insert a custom header containing a token associated with some policy



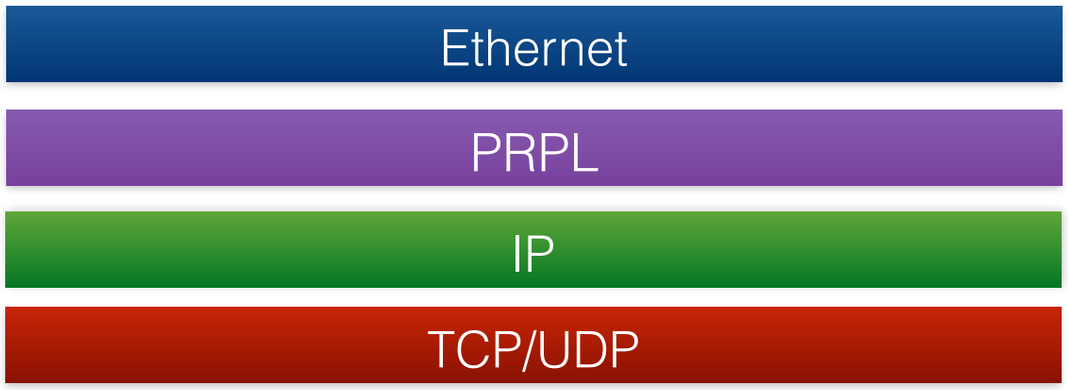
Tagging

- Insert a custom header containing a token associated with some policy



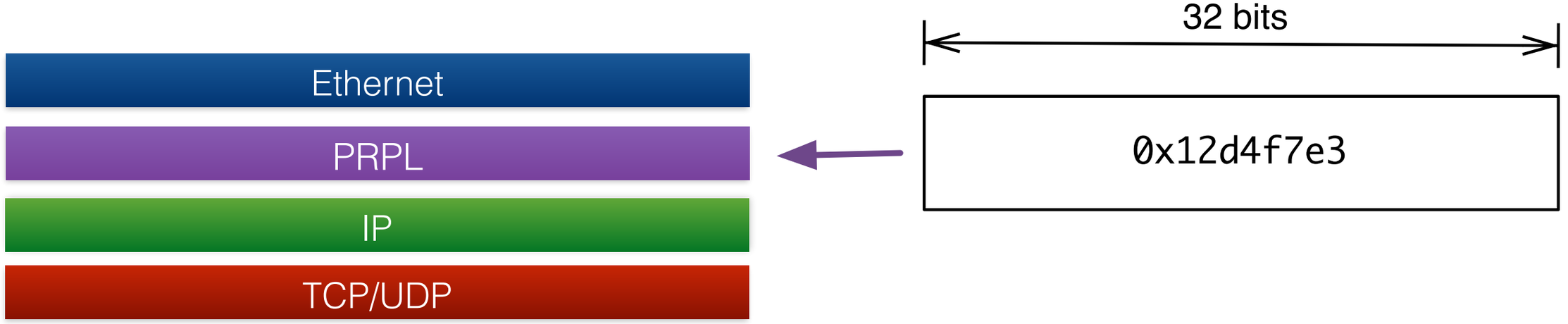
Tagging

- Insert a custom header containing a token associated with some policy

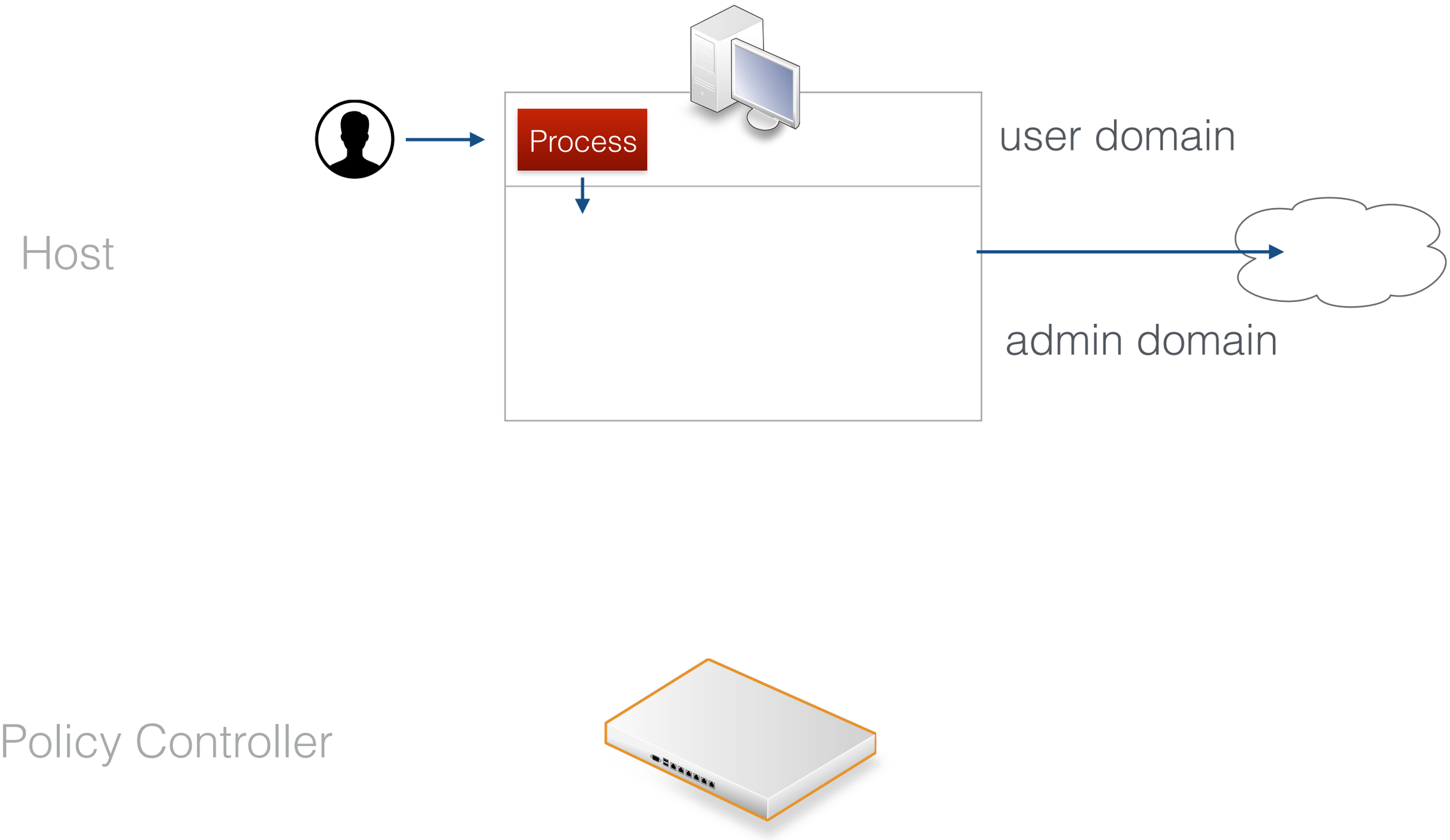


Tagging

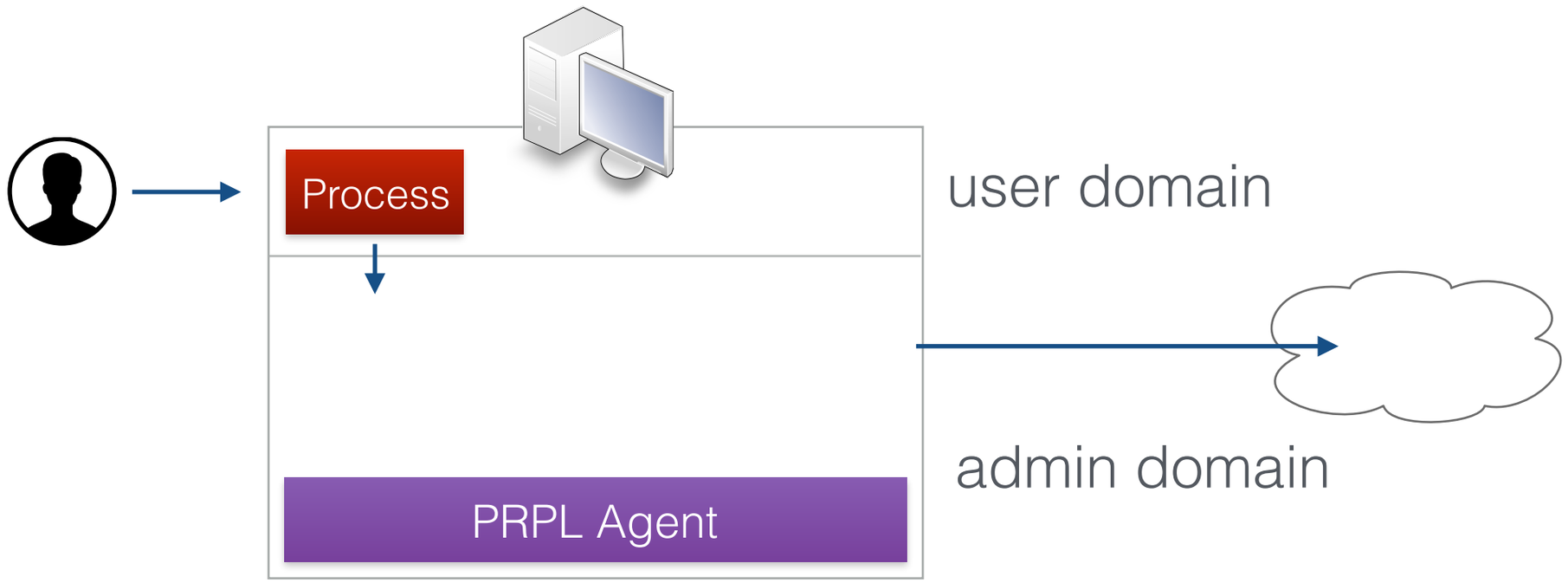
- Insert a custom header containing a token associated with some policy



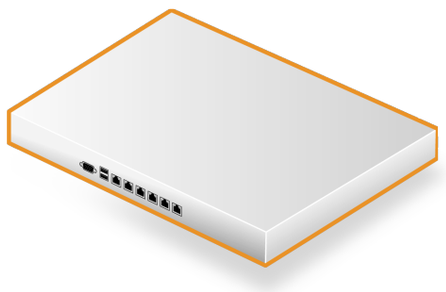
Tagging



Tagging

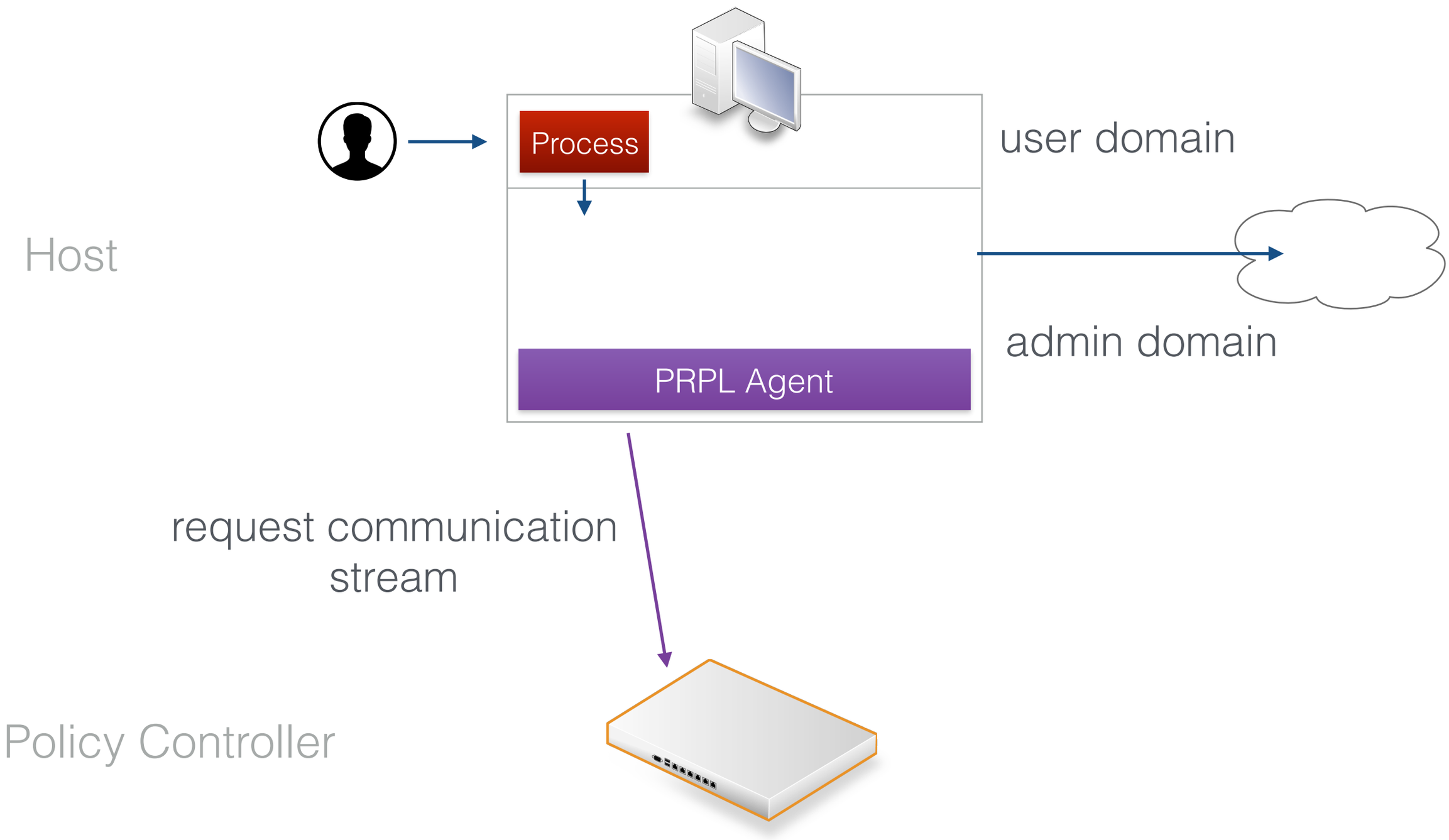


Host

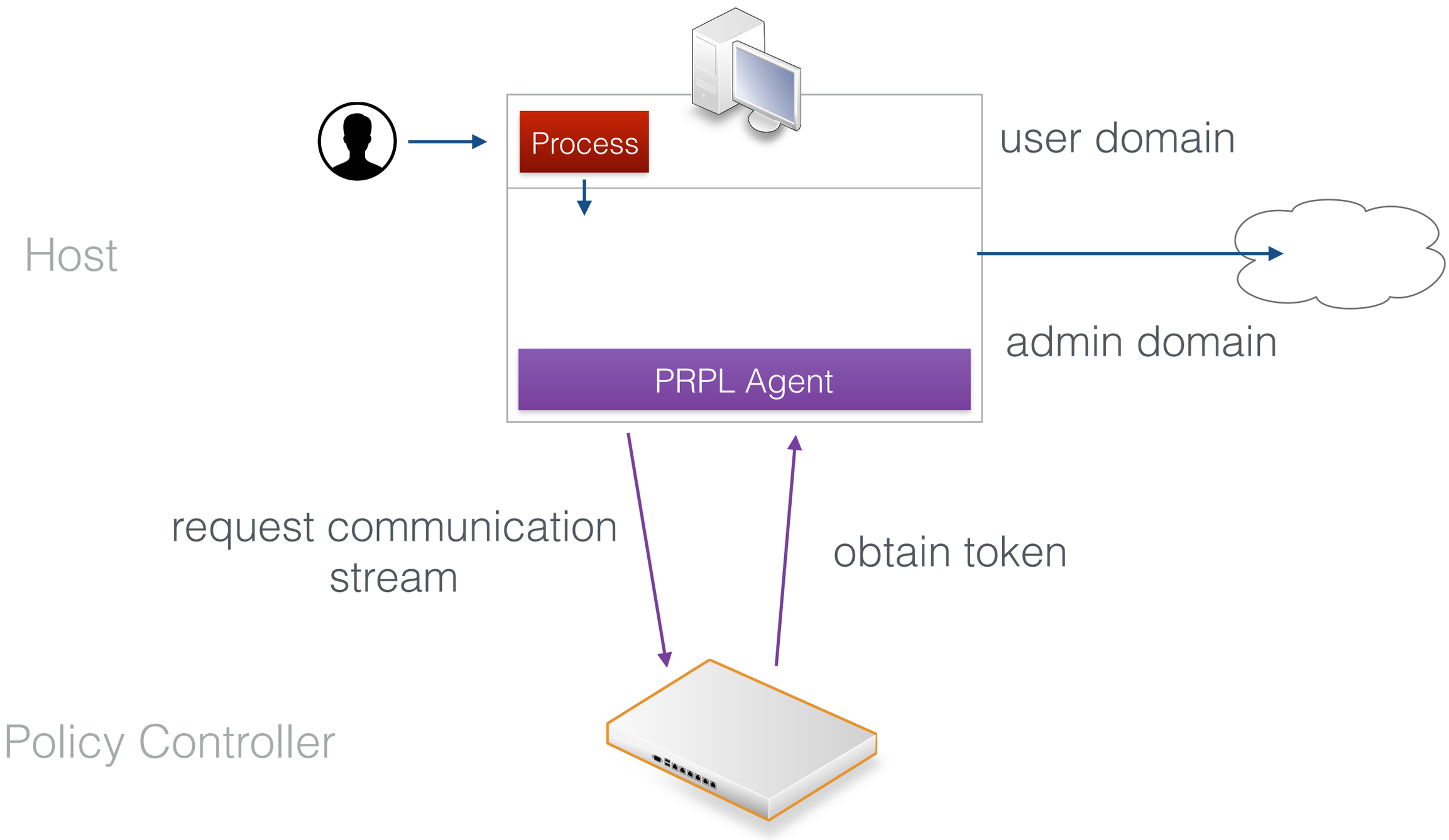


Policy Controller

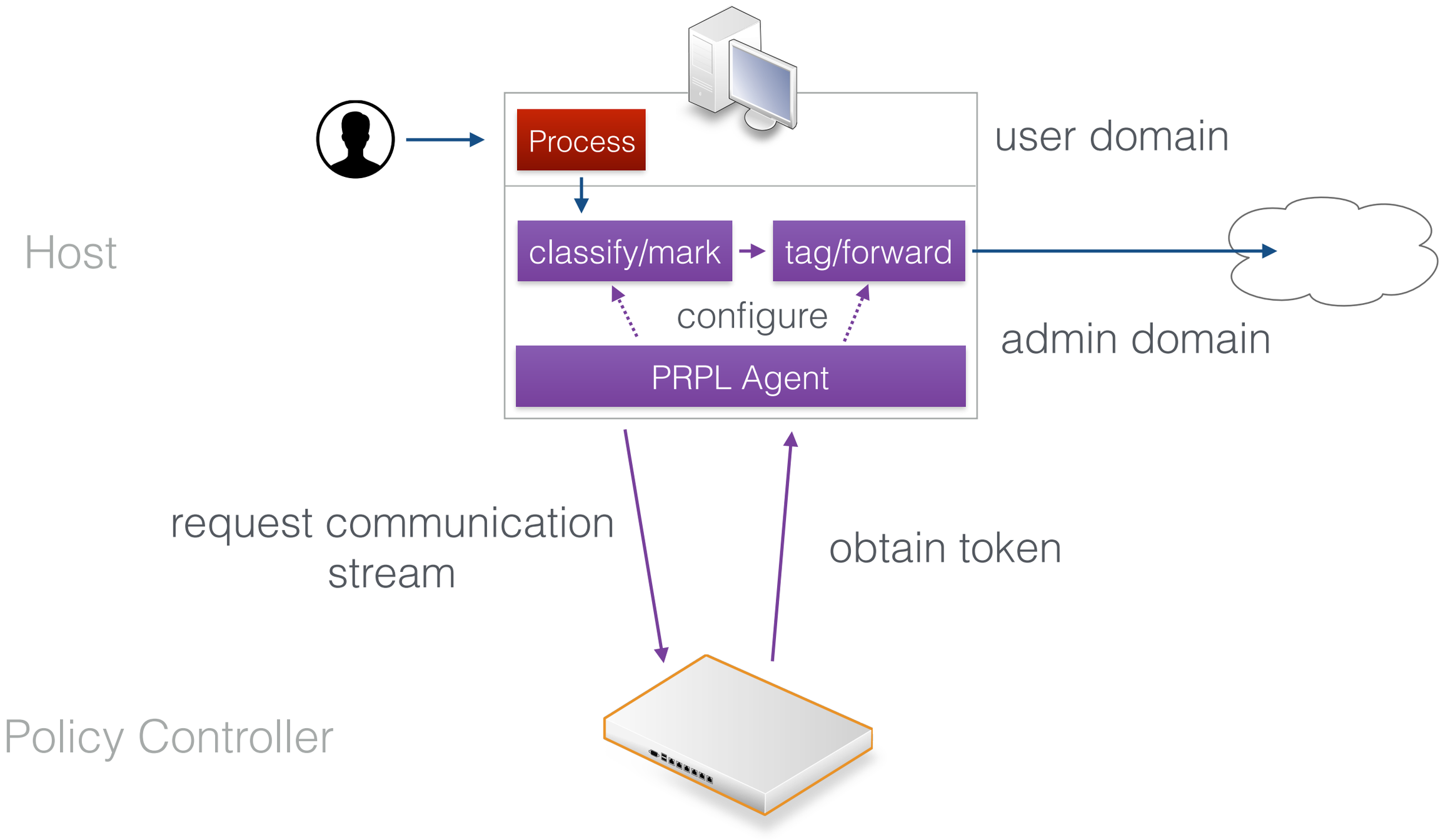
Tagging


Host
Process
PRPL Agent
user domain
admin domain
request communication stream
Policy Controller

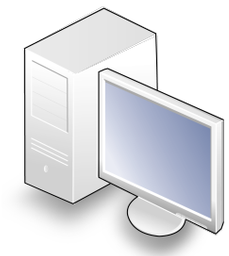
Tagging

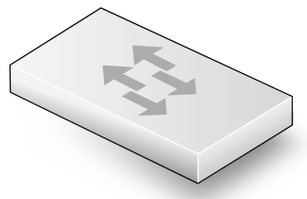

Host
Process
PRPL Agent
user domain
admin domain
request communication stream
obtain token
Policy Controller

Tagging


Host
Process
user domain
classify/mark
tag/forward
configure
admin domain
PRPL Agent
request communication stream
obtain token
Policy Controller

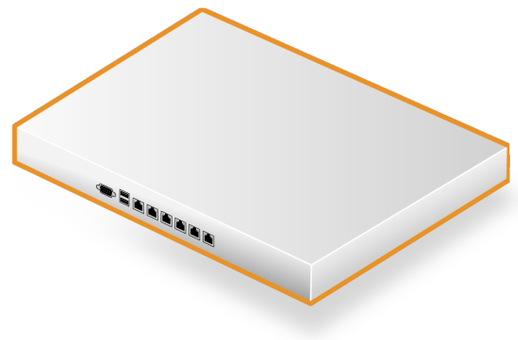
Forwarding





- Programmable Hardware
  [Kangaroo INFOCOM ’10, SDN Chip SIGCOMM ’13, Intel FM6000 switch silicon]

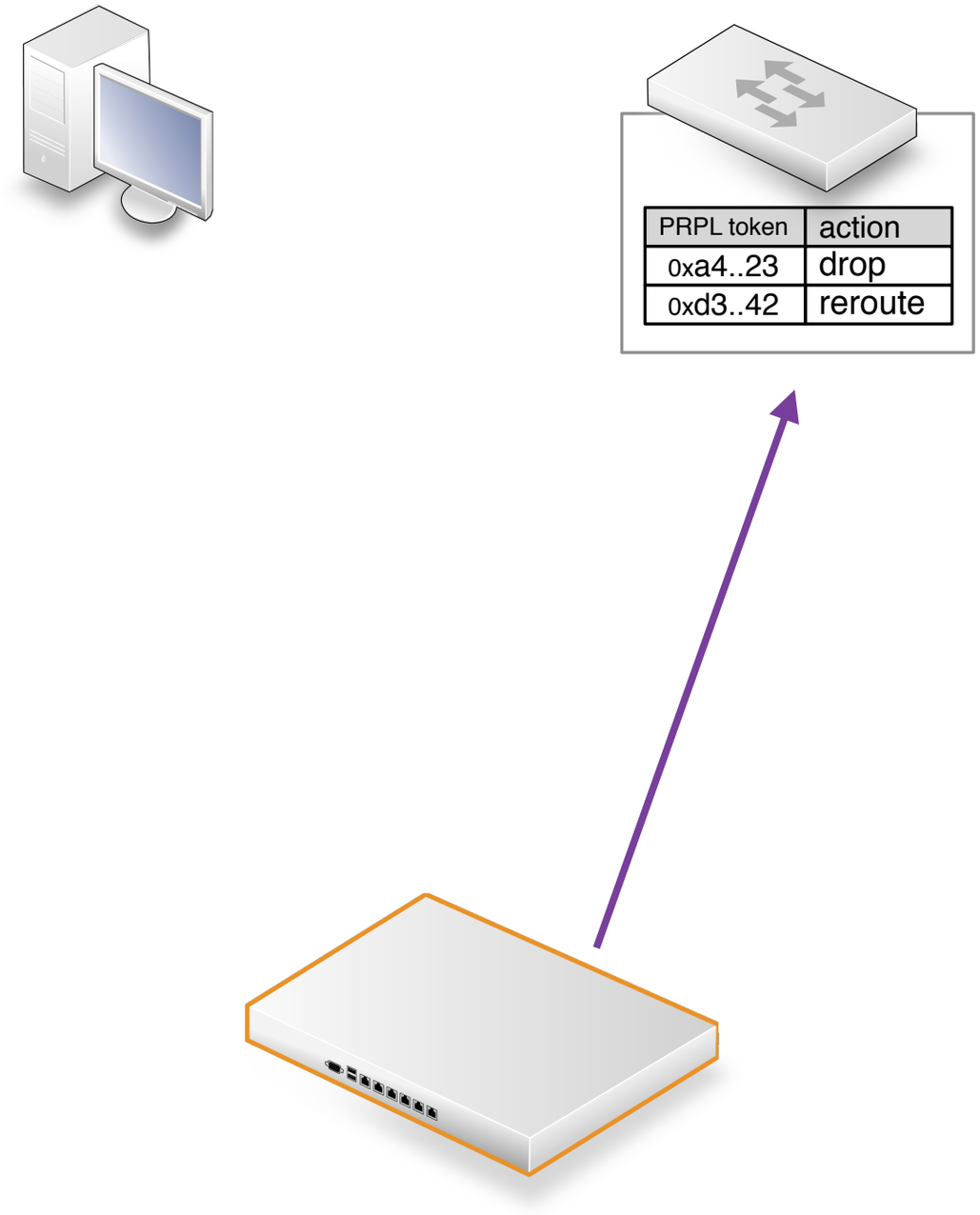
- Dataplane Forwarding Model in P4 [SIGCOMM CCR 2014]



Forwarding

- Programmable Hardware
  [Kangaroo INFOCOM ’10, SDN Chip SIGCOMM ’13, Intel FM6000 switch silicon]

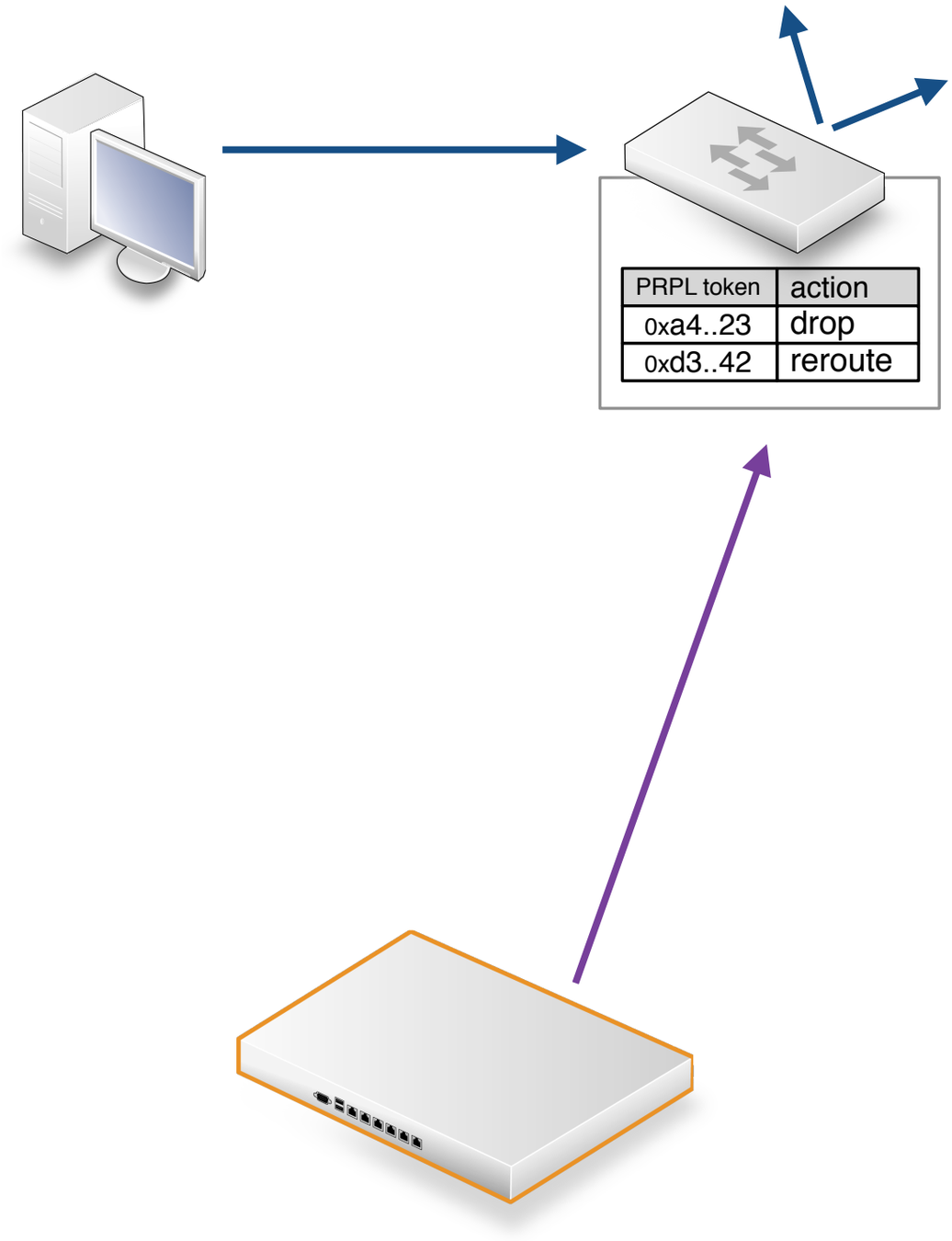
- Dataplane Forwarding Model in P4 [SIGCOMM CCR 2014]



Forwarding

- Programmable Hardware
  [Kangaroo INFOCOM ’10, SDN Chip SIGCOMM ’13, Intel FM6000 switch silicon]

- Dataplane Forwarding Model in P4 [SIGCOMM CCR 2014]



Implementation

Implementation

- Linux on-board tools: iptables, custom routing, tunnel devices

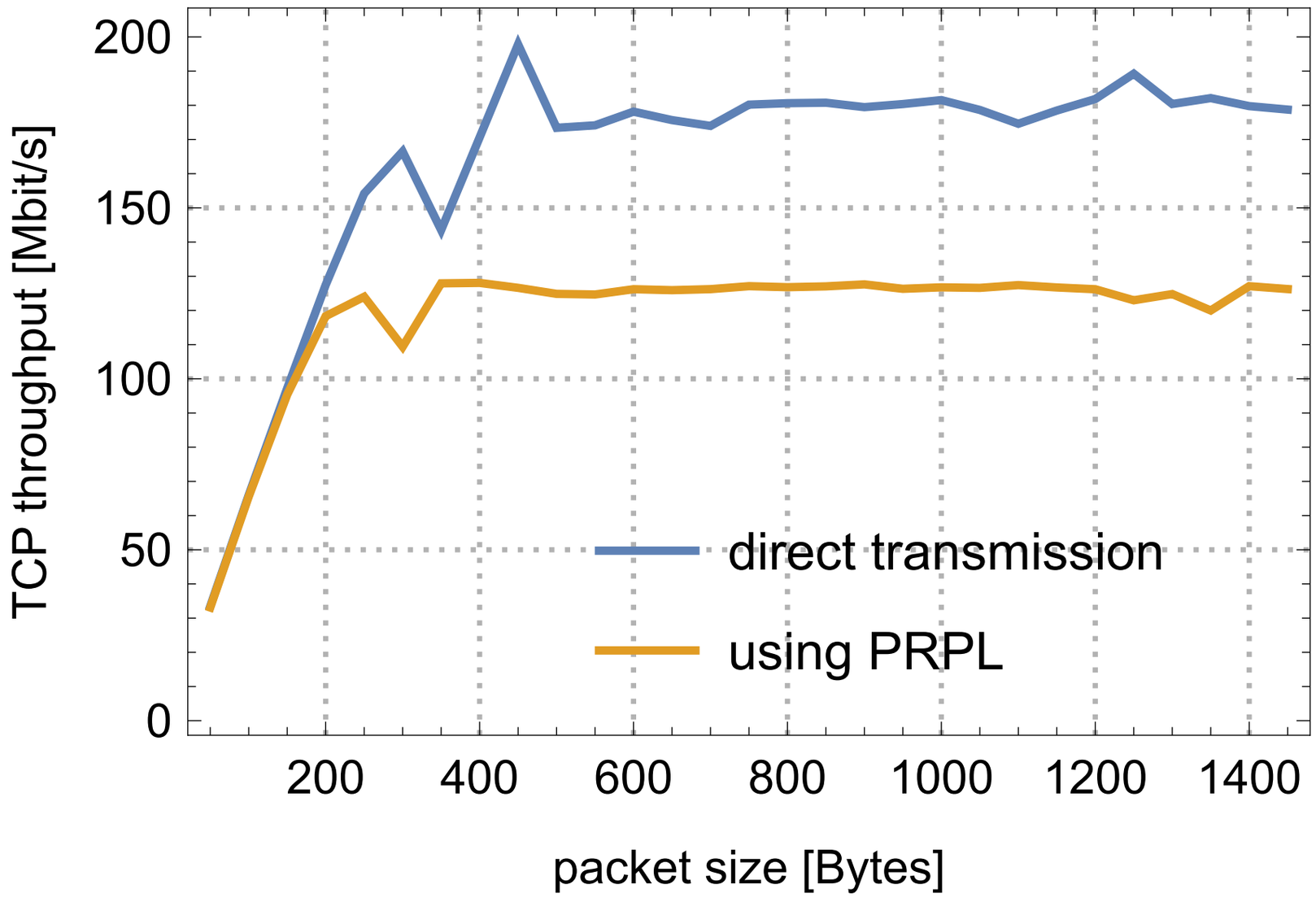
Implementation

- Linux on-board tools: iptables, custom routing, tunnel devices
- P4: Matching on token

Implementation

- Linux on-board tools: iptables, custom routing, tunnel devices
- P4: Matching on token
- Prototype
  - P4 behavioral model
  - tag based on uid
  - forward or drop

Implementation



- No performance penalty for packets < 200 Bytes

Future Work and Conclusion

Future Work and Conclusion

- Network Management can greatly benefit from fine-grained process-level information

Future Work and Conclusion

- Network Management can greatly benefit from fine-grained process-level information
- System Architecture and Prototype enabling packet processing based on such information

Future Work and Conclusion

- Network Management can greatly benefit from fine-grained process-level information
- System Architecture and Prototype enabling packet processing based on such information
- Future work: expansion beyond current examples, more complex policies

Source Code

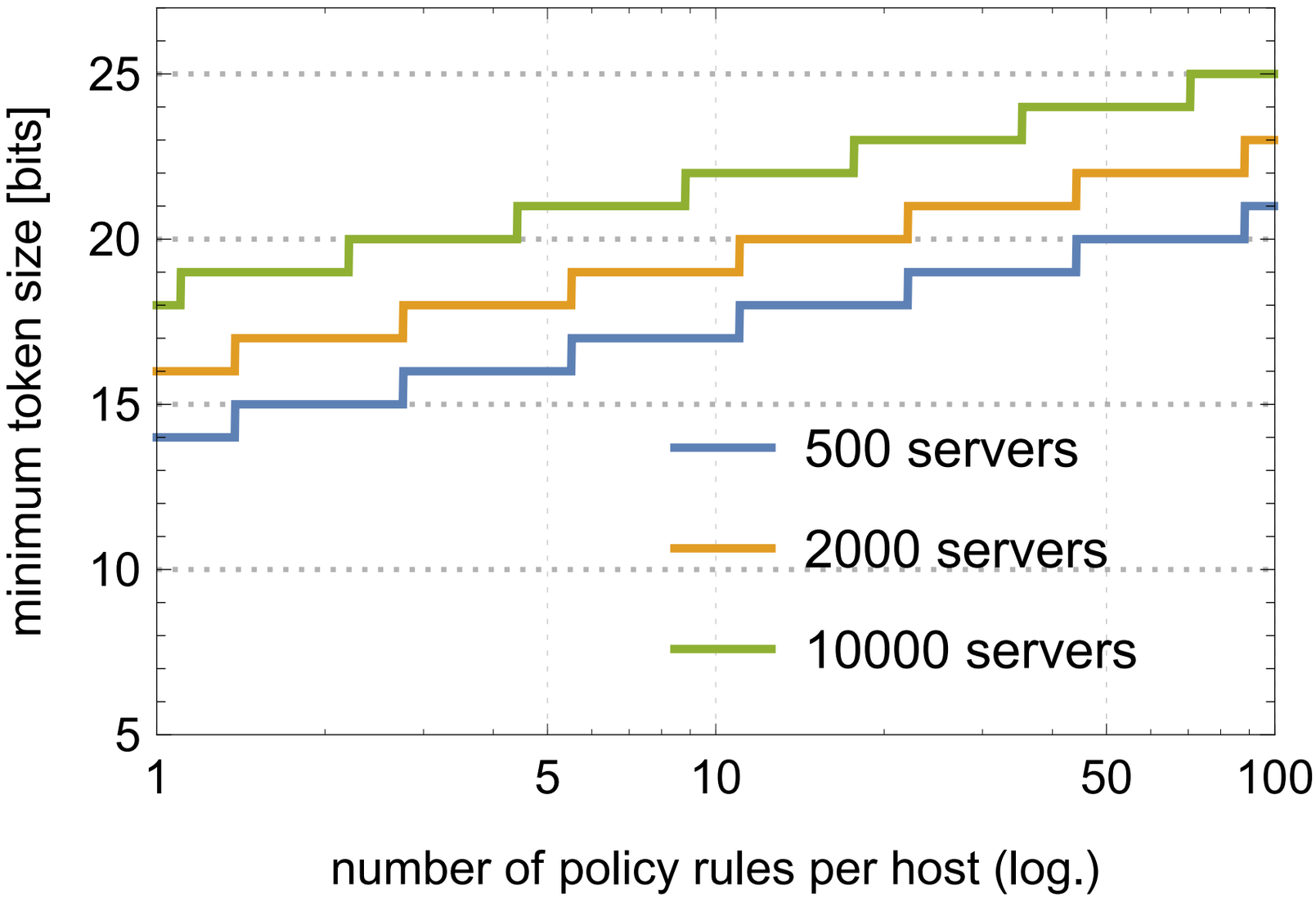
https://github.com/nsr-colorado/prpl

Backup Slides

Future Work

- Study feasibility of more complex policy scenarios
- Granularity of Tokens
- Controller - Agent Interface
- Proactive vs. reactive configuration
- Trust in tagging process

Tagging



- Token sizes between 16 bits and 32 bits sufficient even for large networks

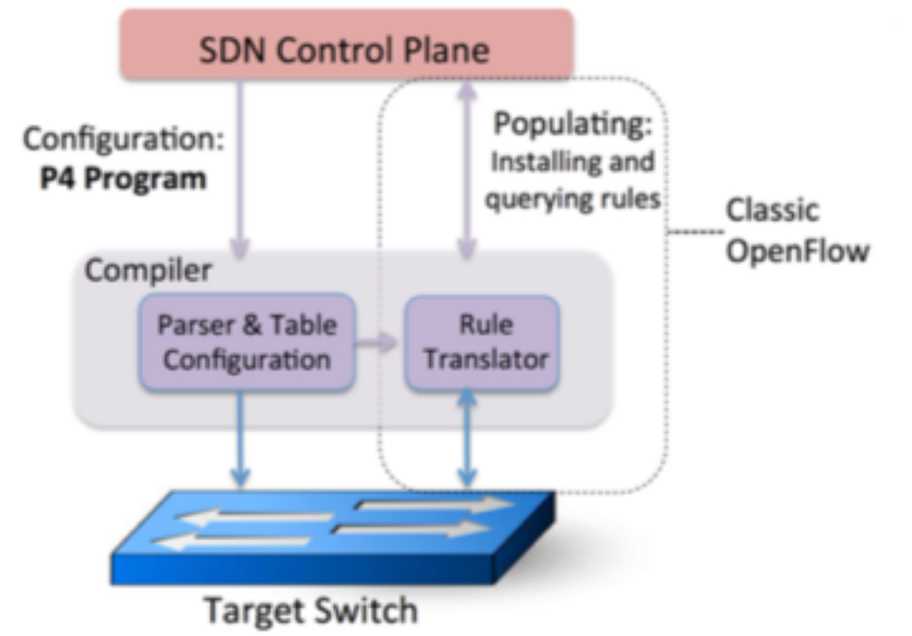
Programmable Hardware

- new custom ASICs can achieve such flexibility at terabit speeds [Kangaroo INFOCOM ’10, SDN Chip SIGCOMM ’13, Intel FM6000 switch silicon]

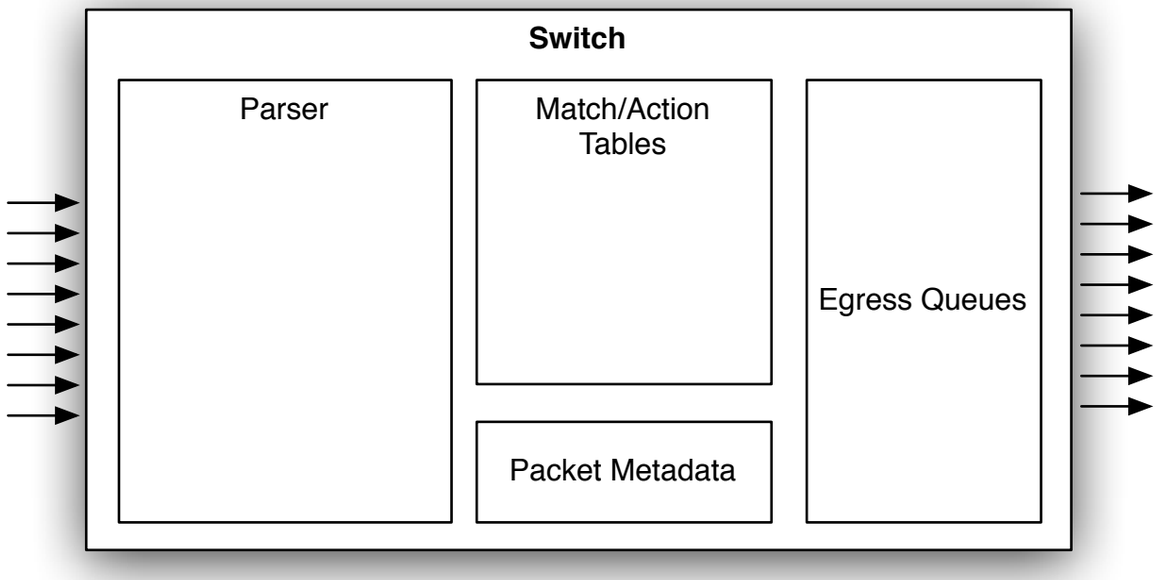
- some switches are more programmable than others:
  - FPGA (Xilinx, Altera, Corsa)
  - NPU (Ezchip, Netronome)
  - CPU (OVS, …)

P4 Language

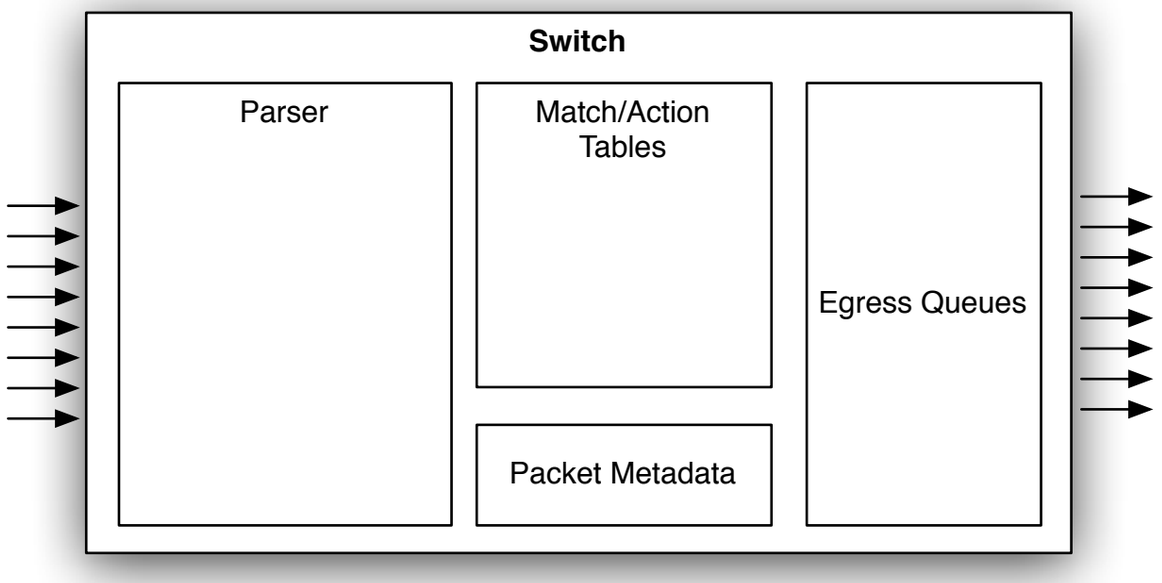
- P4 program configures forwarding behavior (abstract forwarding model)
- express serial dependencies (e.g. ARP/L3 Routing)
- P4 compiler translates into a target-specific representation
- OF can still be used to install and query rules once forwarding model is defined



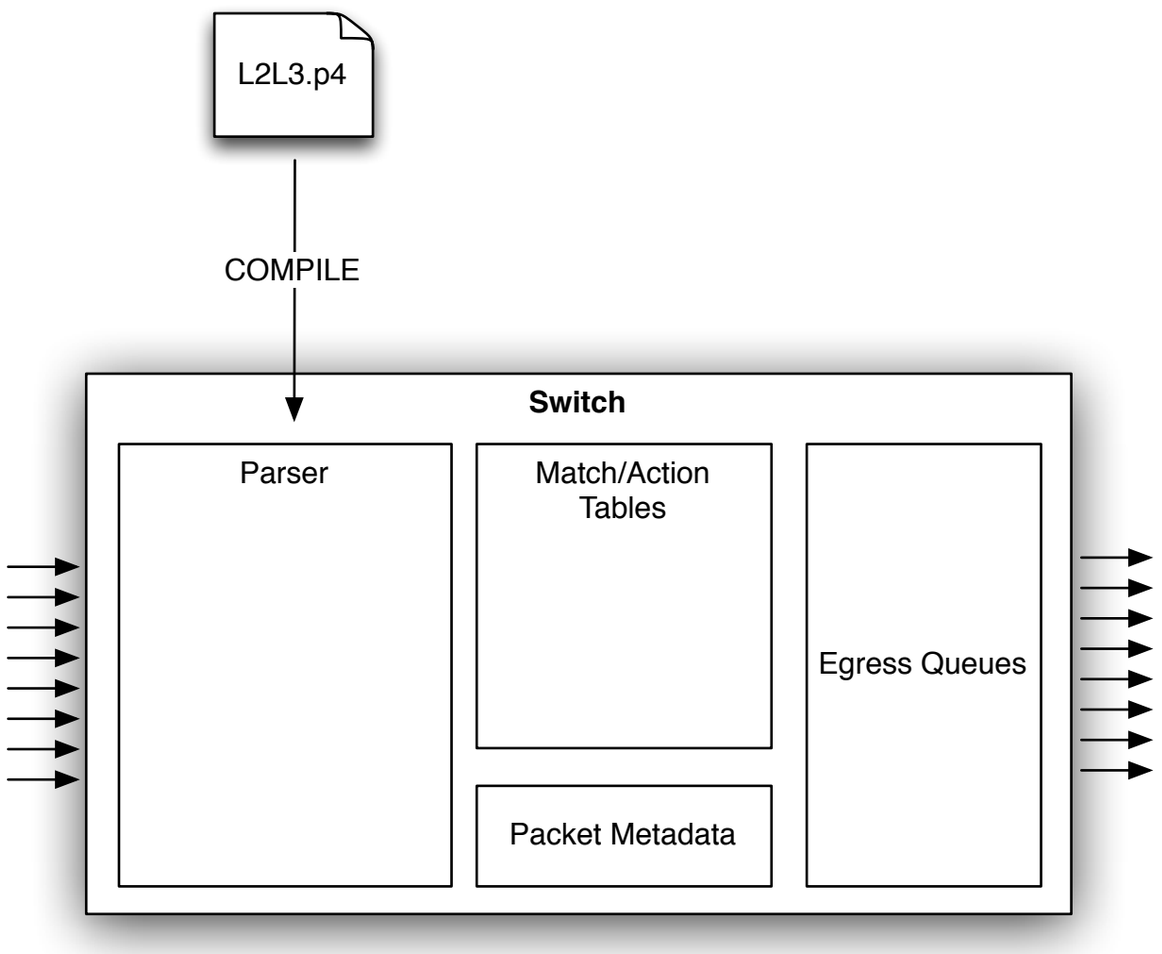
P4 Forwarding Model / Runtime



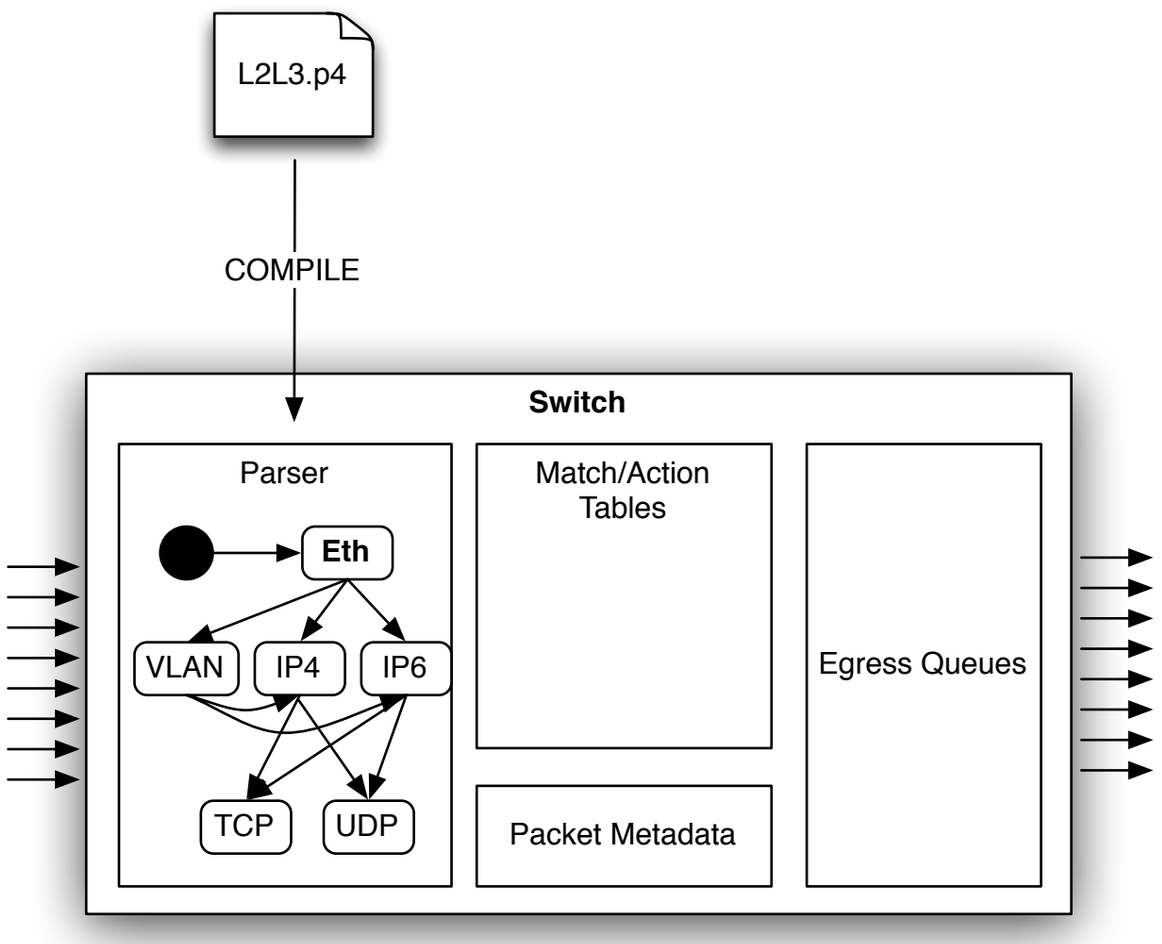
P4 Forwarding Model / Runtime



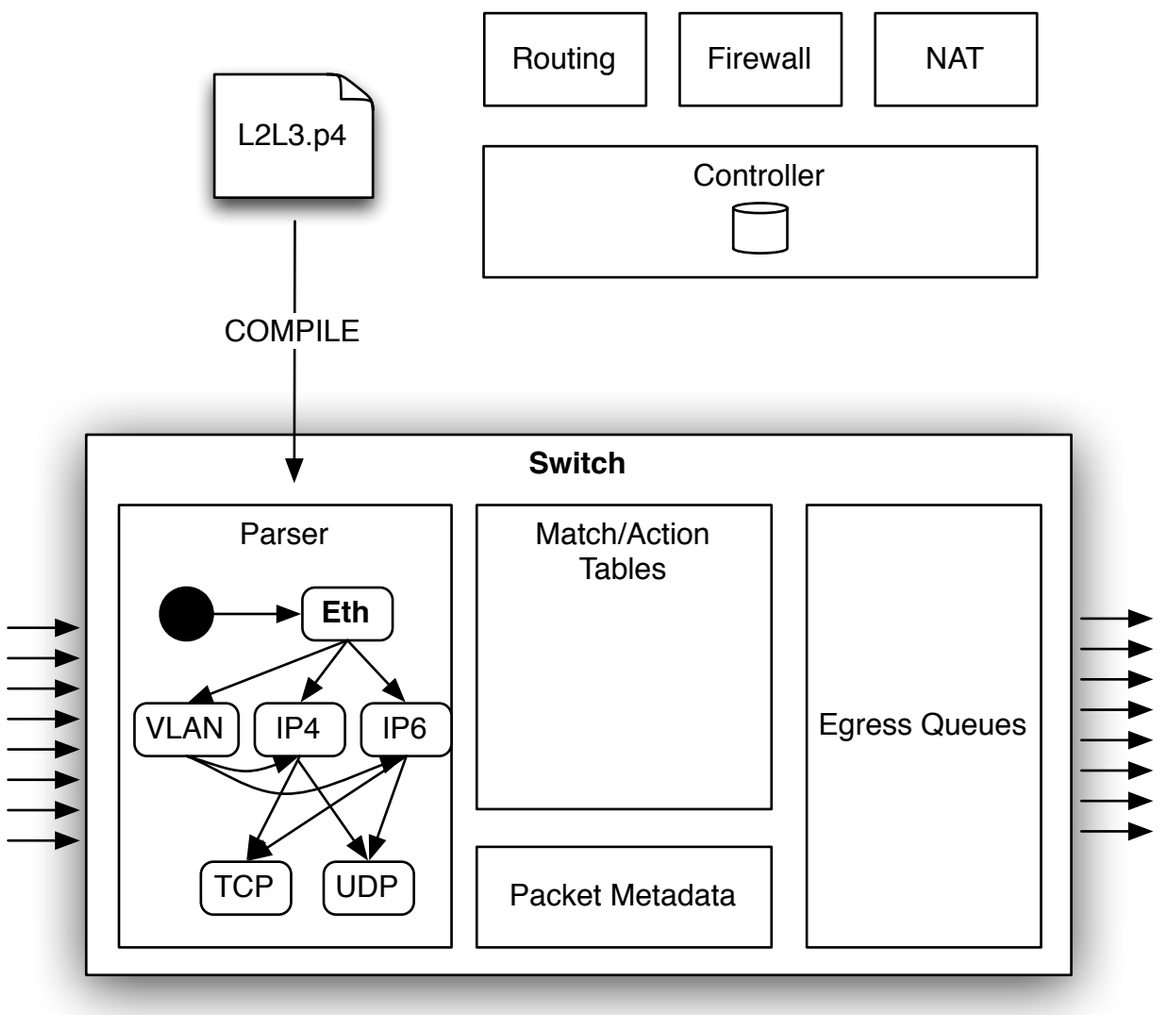
P4 Forwarding Model / Runtime



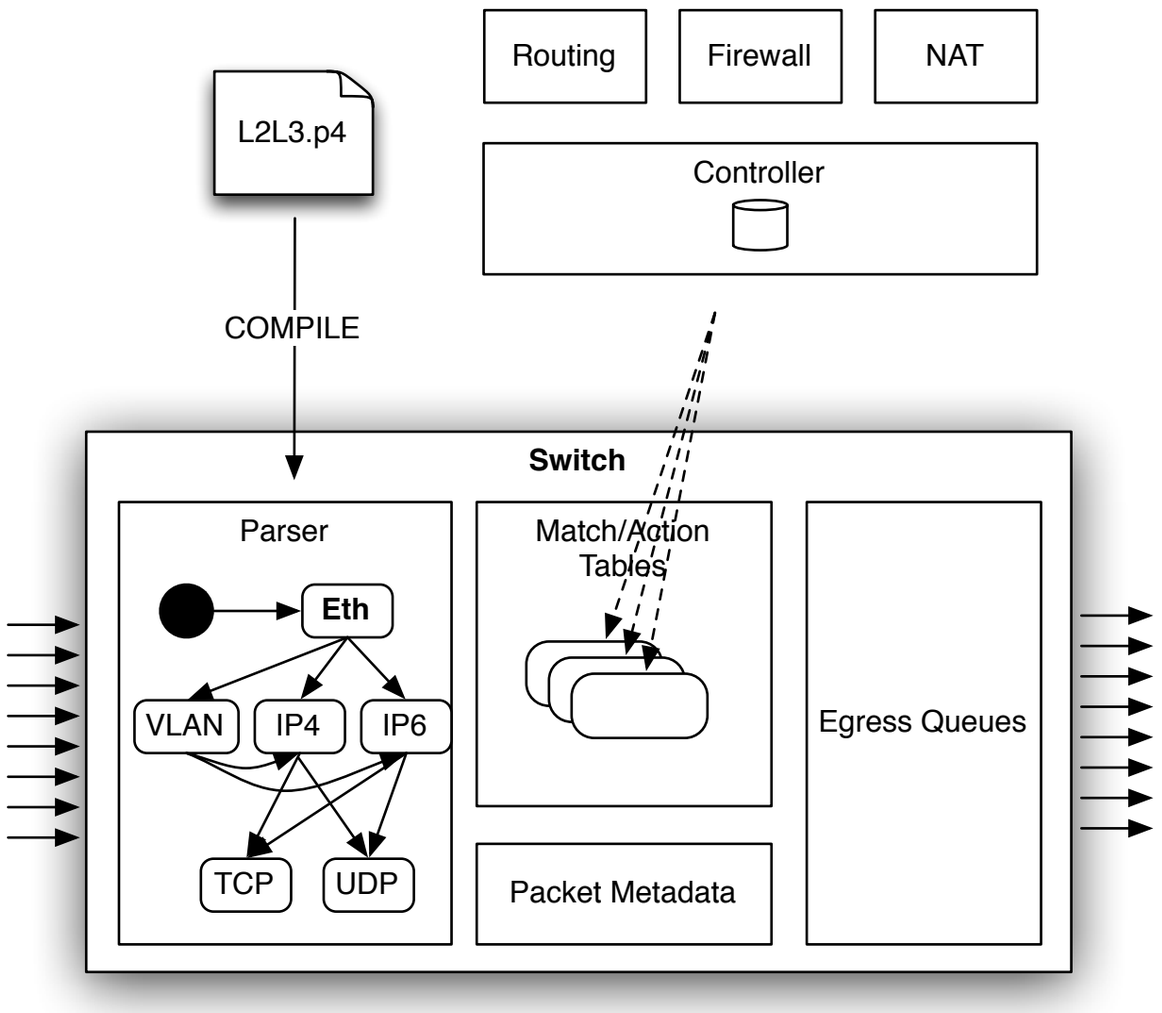
P4 Forwarding Model / Runtime



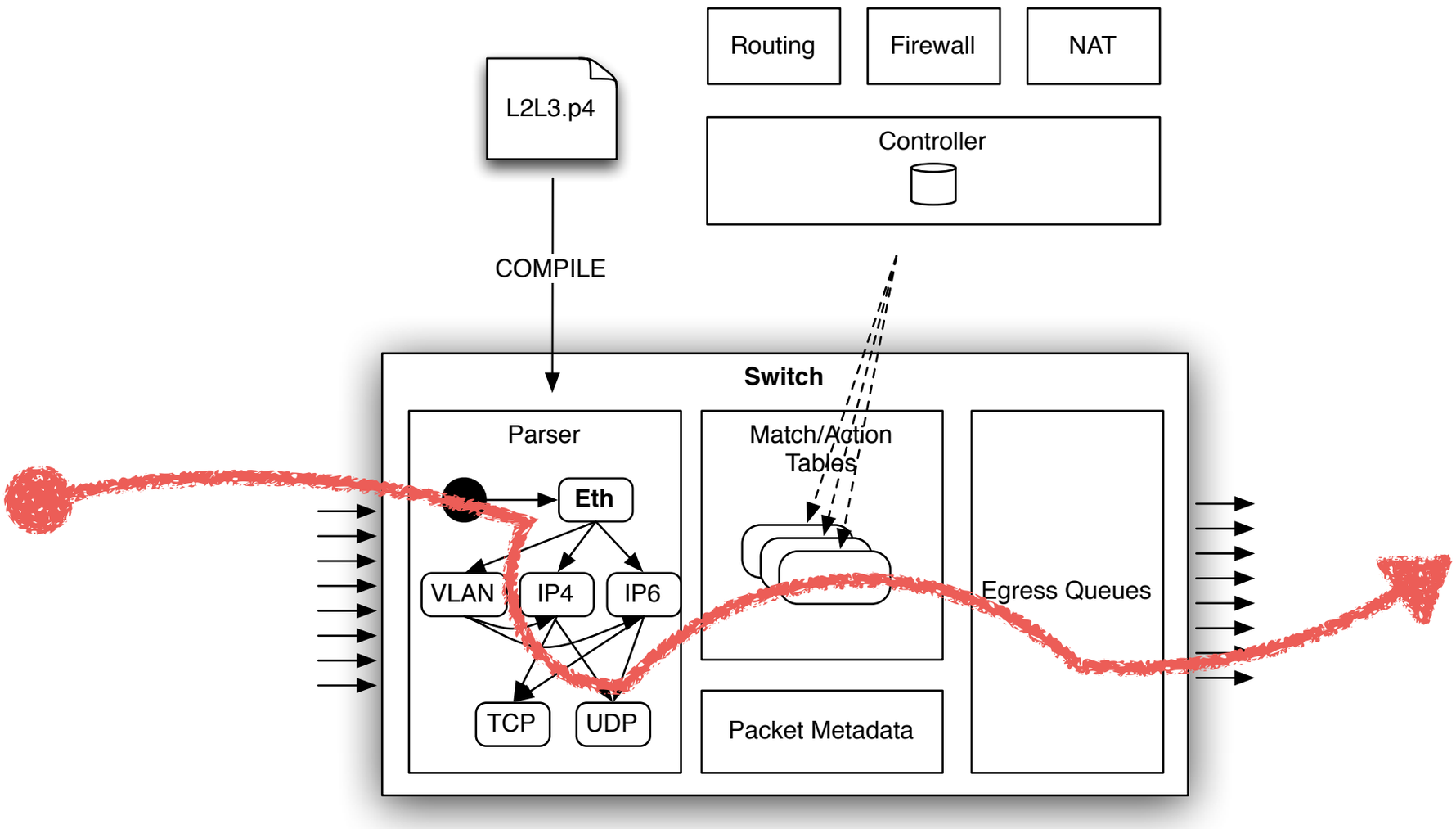
P4 Forwarding Model / Runtime



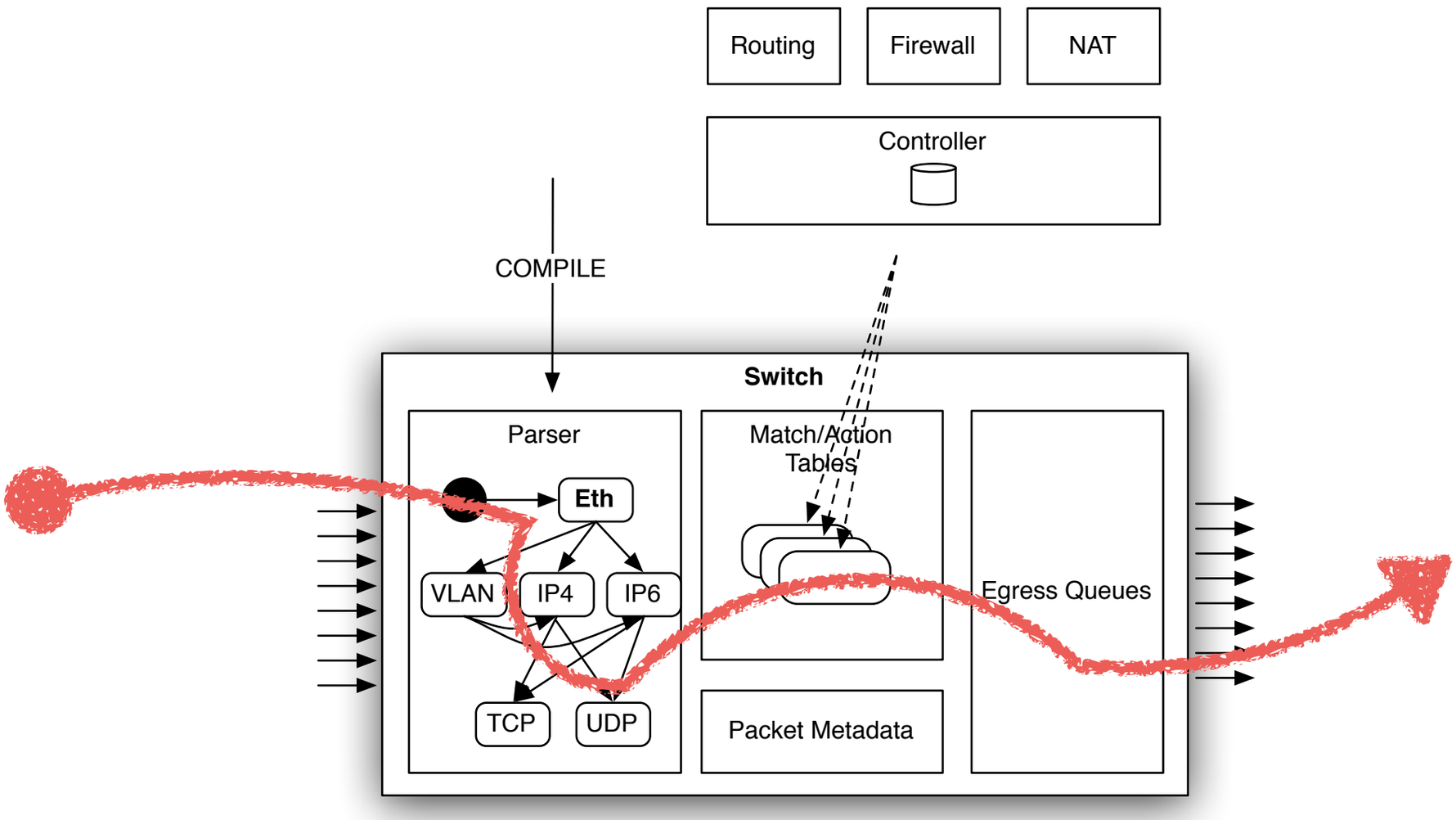
P4 Forwarding Model / Runtime



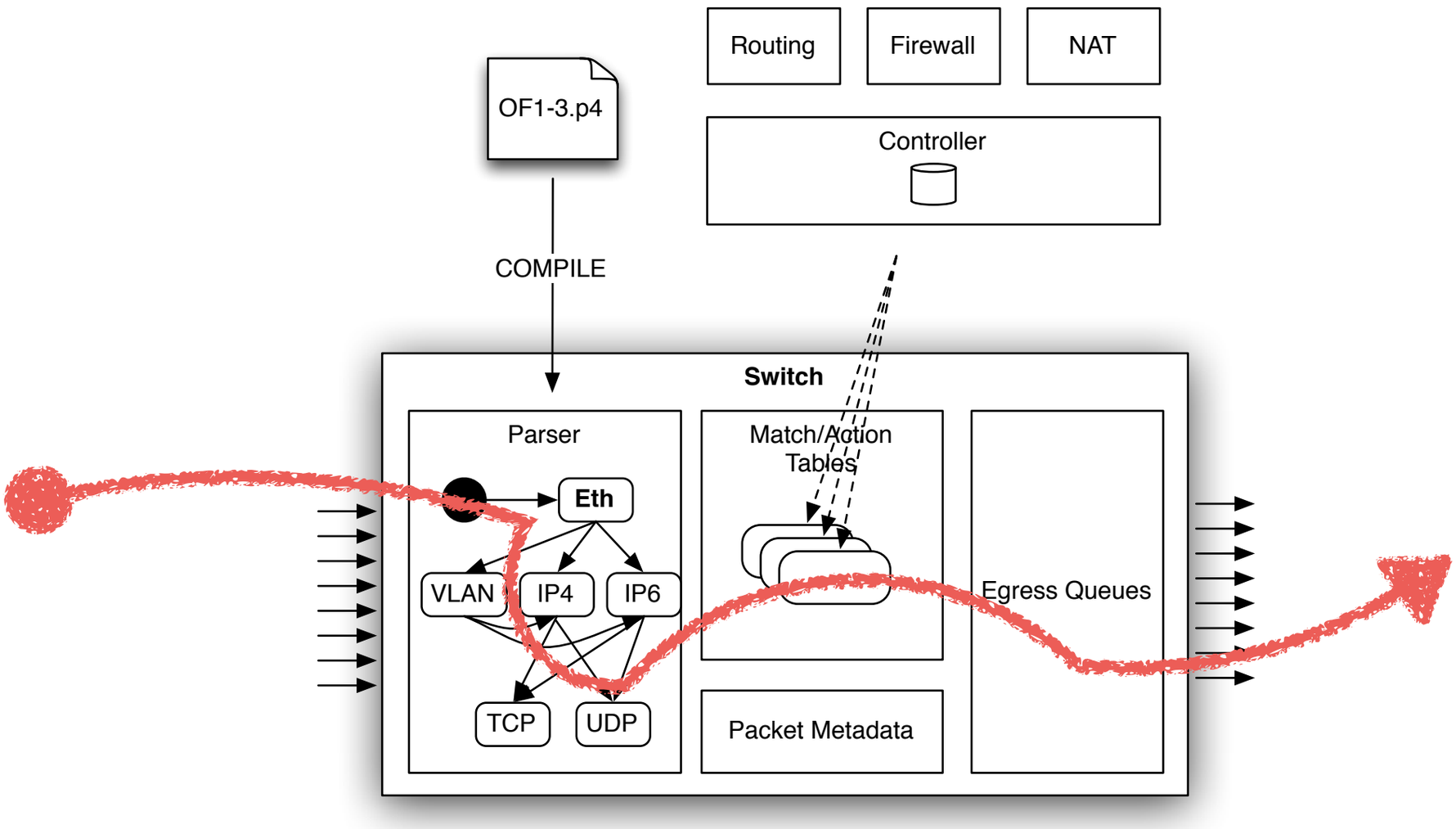
P4 Forwarding Model / Runtime



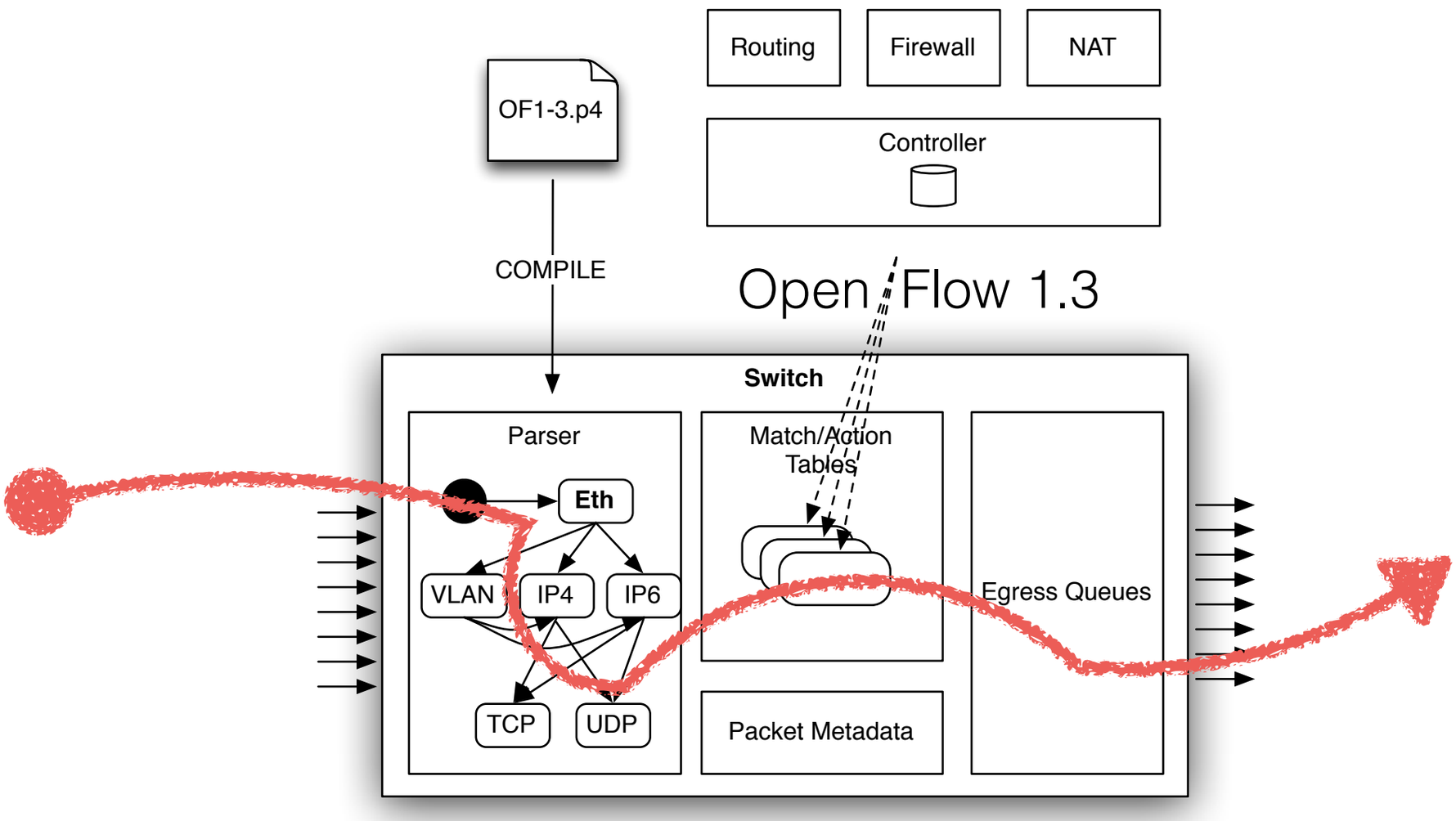
P4 Forwarding Model / Runtime



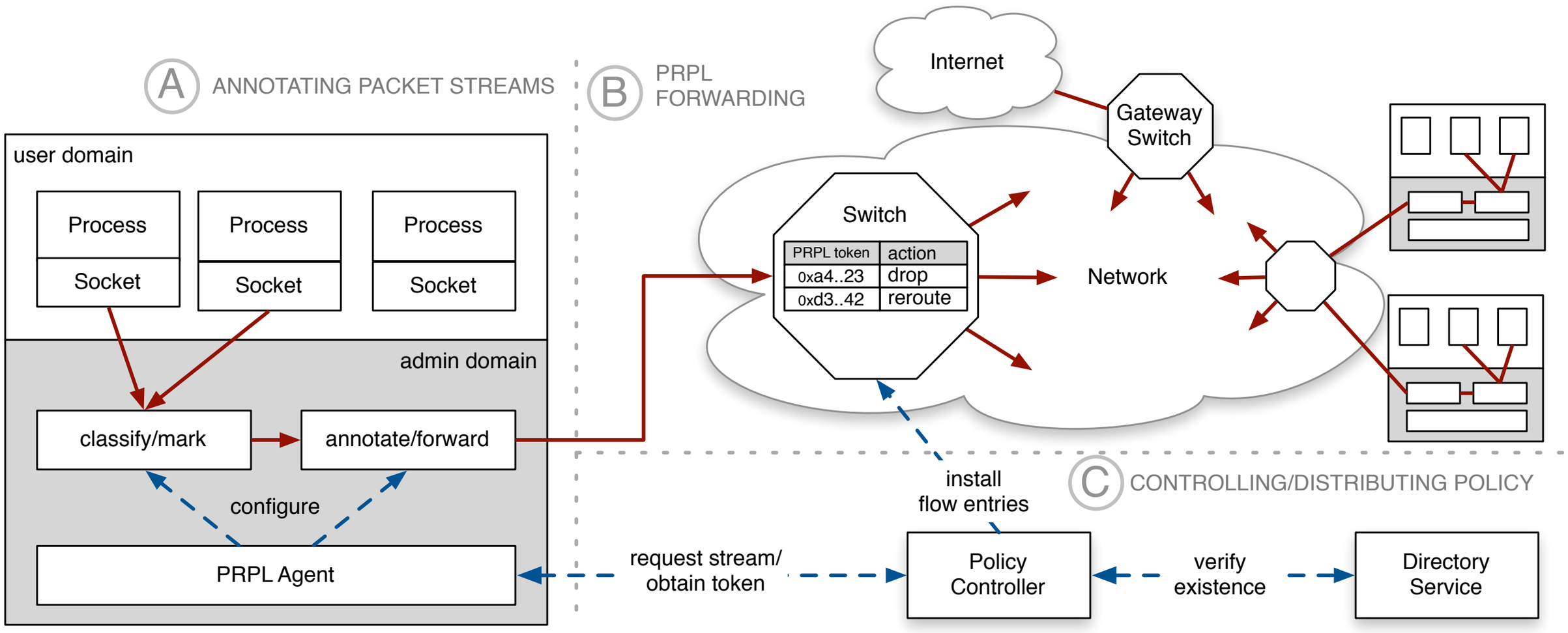
P4 Forwarding Model / Runtime



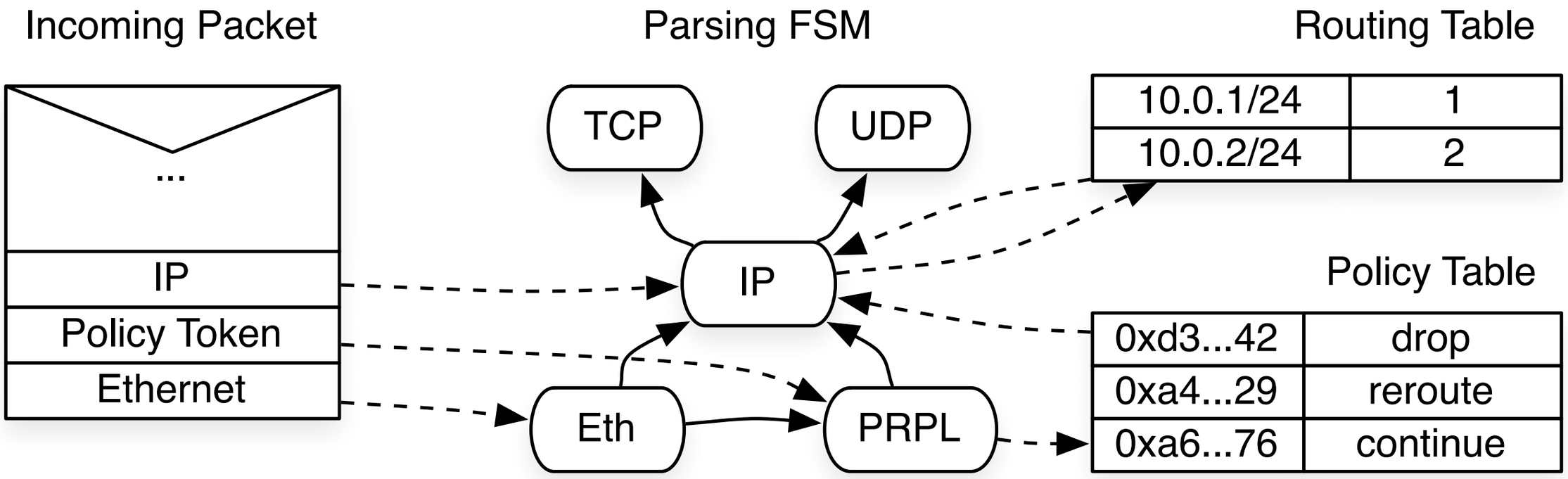
P4 Forwarding Model / Runtime



Architecture



P4 Parsing


Incoming Packet
...
...
IP
Policy Token
Ethernet
Parsing FSM
TCP
UDP
IP
Eth
PRPL
Routing Table
10.0.1/24
1
10.0.2/24
2
Policy Table
0xd3...42
drop
0xa4...29
reroute
0xa6...76
continue

P4 Implementation

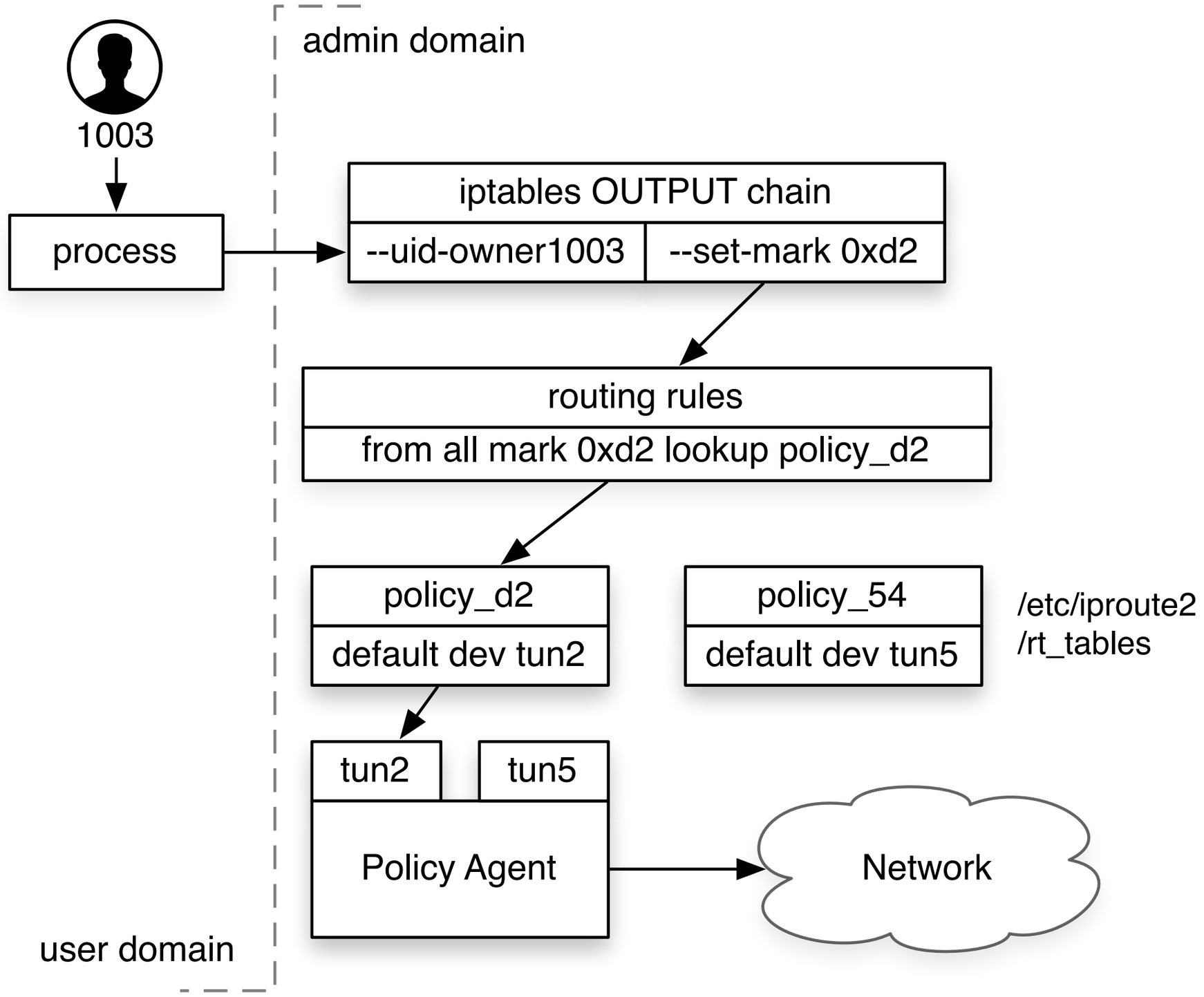
```
header_type prpl_t {
  fields {
    token : 8;
  }
}

header prpl_t prpl;
```

```
parser start {
  return parse_ethernet;
}
parser parse_ethernet {
  extract(ethernet);
  return parse_prpl;
}
parser parse_prpl {
  extract(prpl);
  return ingress;
}
```

```
table prpl {
  reads {
    prpl.token : exact;
  }
  actions { _nop; _drop; forward; }
  size : 128;
}
```

```
table_set_default prpl _nop
table_add prpl _nop     0x00000001 =>
table_add prpl _drop    0x00000002 =>
table_add prpl forward 0x00000001 => 4
```

iptables


admin domain
1003
process
iptables OUTPUT chain
--uid-owner1003
--set-mark 0xd2
routing rules
from all mark 0xd2 lookup policy_d2
policy_d2
default dev tun2
policy_54
default dev tun5
/etc/iproute2
/rt_tables
tun2
tun5
Policy Agent
Network
user domain