P4 User Plane Function (P4-UPF)

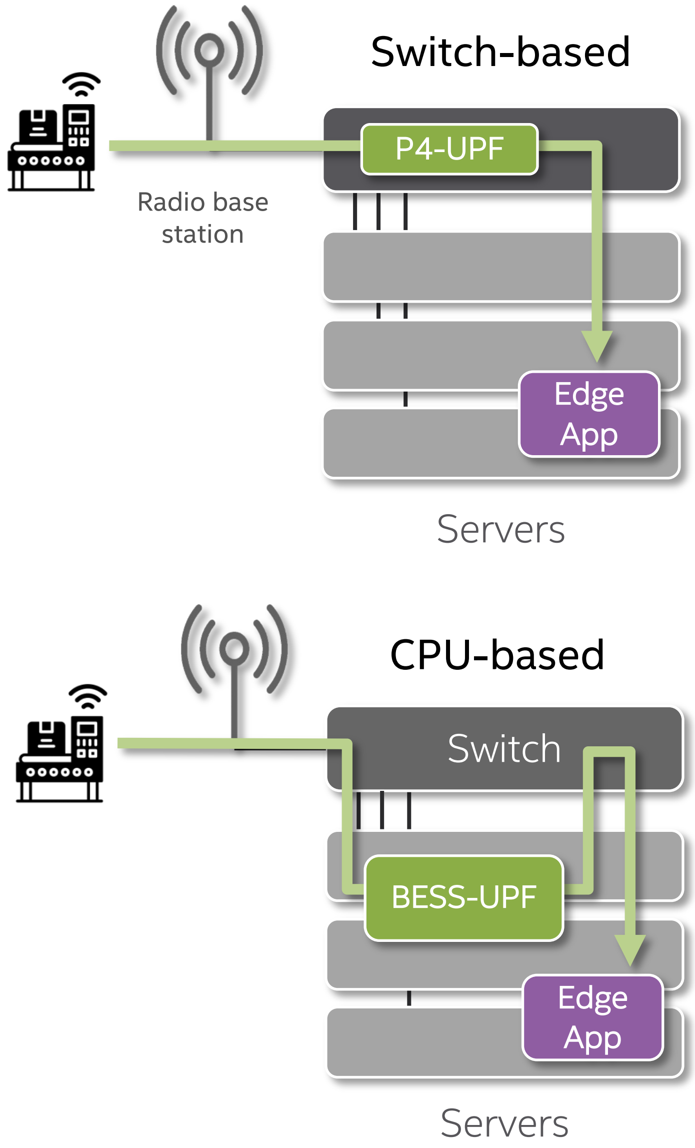
SD-Fabric Tutorial – Part 3

Part 3 Agenda

- P4-UPF architecture and pipeline design
- Hands-on lab
  - Configure P4-UPF
  - Generate traffic
  - Observe GTP-U termination performed by switches

Switch-Based P4-UPF

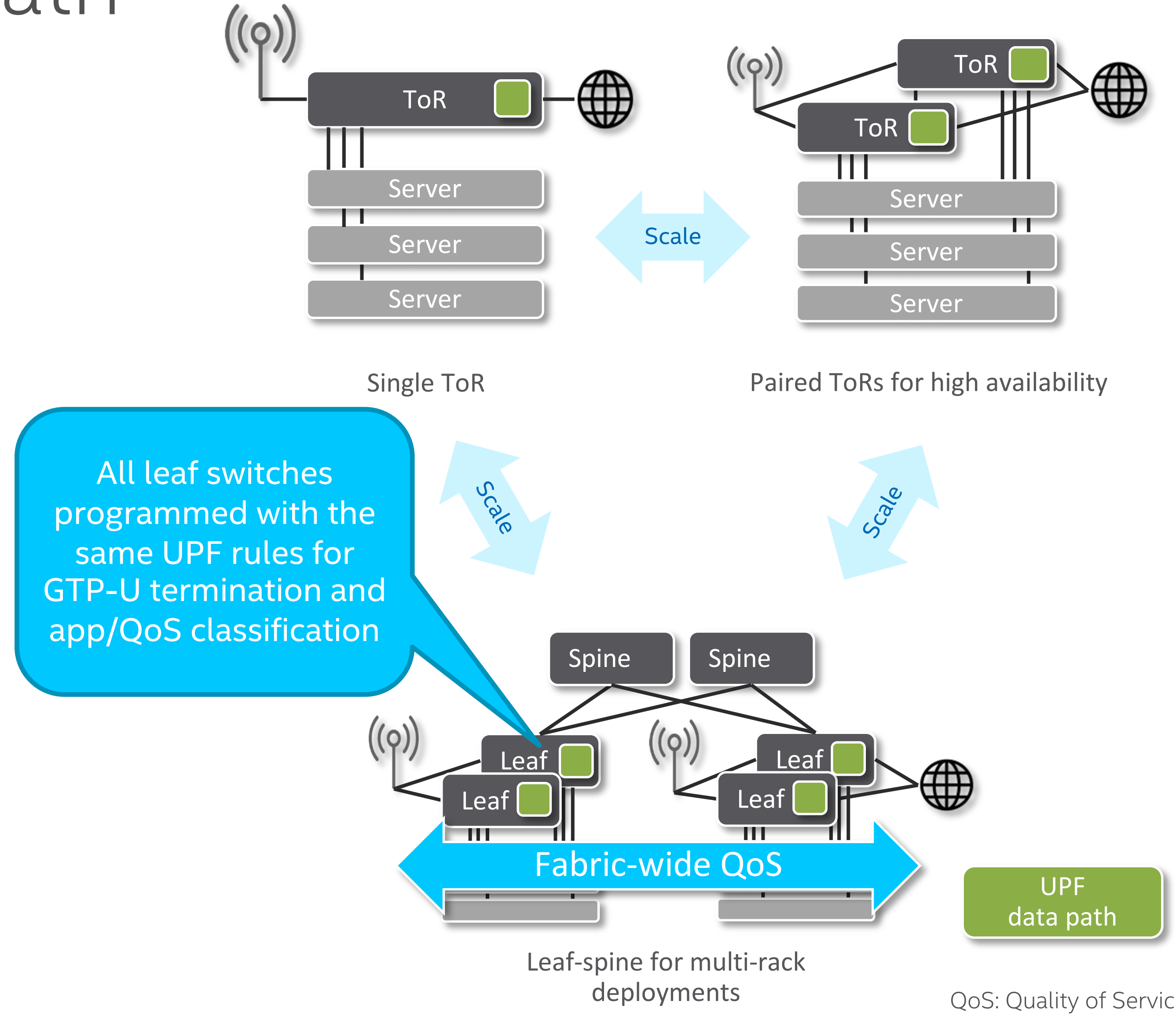
- **Frees up CPU resources**
  - To be used by edge applications
  - UPF data path fully offloaded to switches
- **Addresses Industry 4.0 requirements**
  - Ultra low latency (<1.5µs) and jitter (<4ns)
  - Tbps throughput
- **Tailored for enterprise and IoT use cases**
  - GTP-U termination (incl. 5G extensions)
  - Application filtering (ACL)
  - Slicing & QoS
  - Usage reporting
  - Idle-mode buffering (cloud-native service)
- **INT visibility for SLA validation**
  - Monitor flows inside GTP-U tunnels
  - Support UPF-specific drop reasons



BESS: Berkeley Extensible Software Switch

Distributed UPF Data Path

- **Minimum latency**
  - Tunnels terminated at the ingress leaf, without detouring through additional devices
- **Fast failover**
  - With paired-ToRs, if one switch fails, the other can take over as it is already programmed with the same rules.
- **Fabric-wide QoS**
  - packets are classified as soon as they hit the first leaf. We then use a custom DSCP-based marking to enforce the same QoS at each hop.

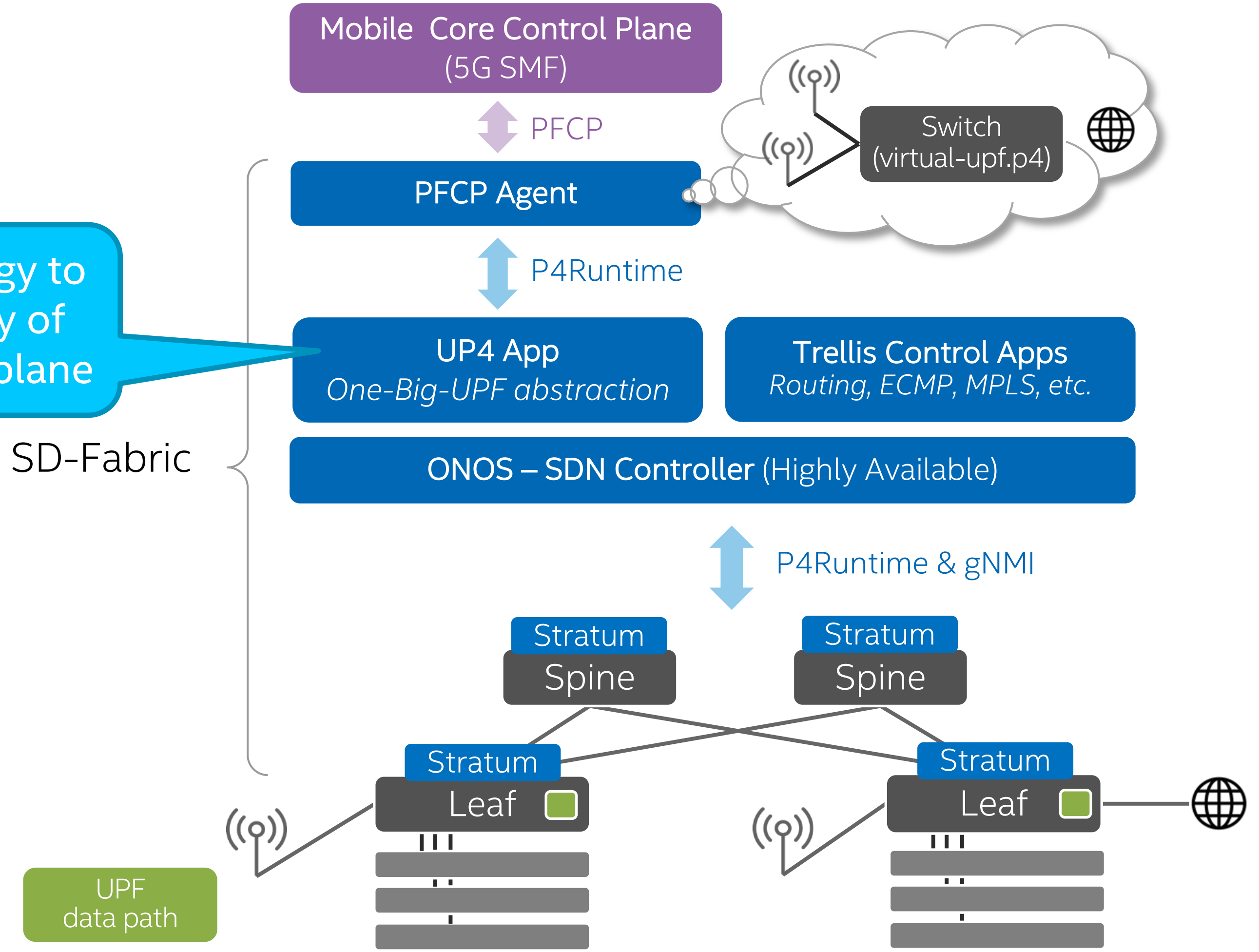

ToR
Server
Server
Server
Single ToR
Scale
ToR
ToR
Server
Server
Server
Paired ToRs for high availability
All leaf switches programmed with the same UPF rules for GTP-U termination and app/QoS classification
Scale
Scale
Spine
Spine
Leaf
Leaf
Leaf
Leaf
Fabric-wide QoS
UPF data path
Leaf-spine for multi-rack deployments

QoS: Quality of Service

Integration with Mobile Core

Via One-Big-UPF Abstraction

Allows fabric topology to scale independently of mobile core control plane

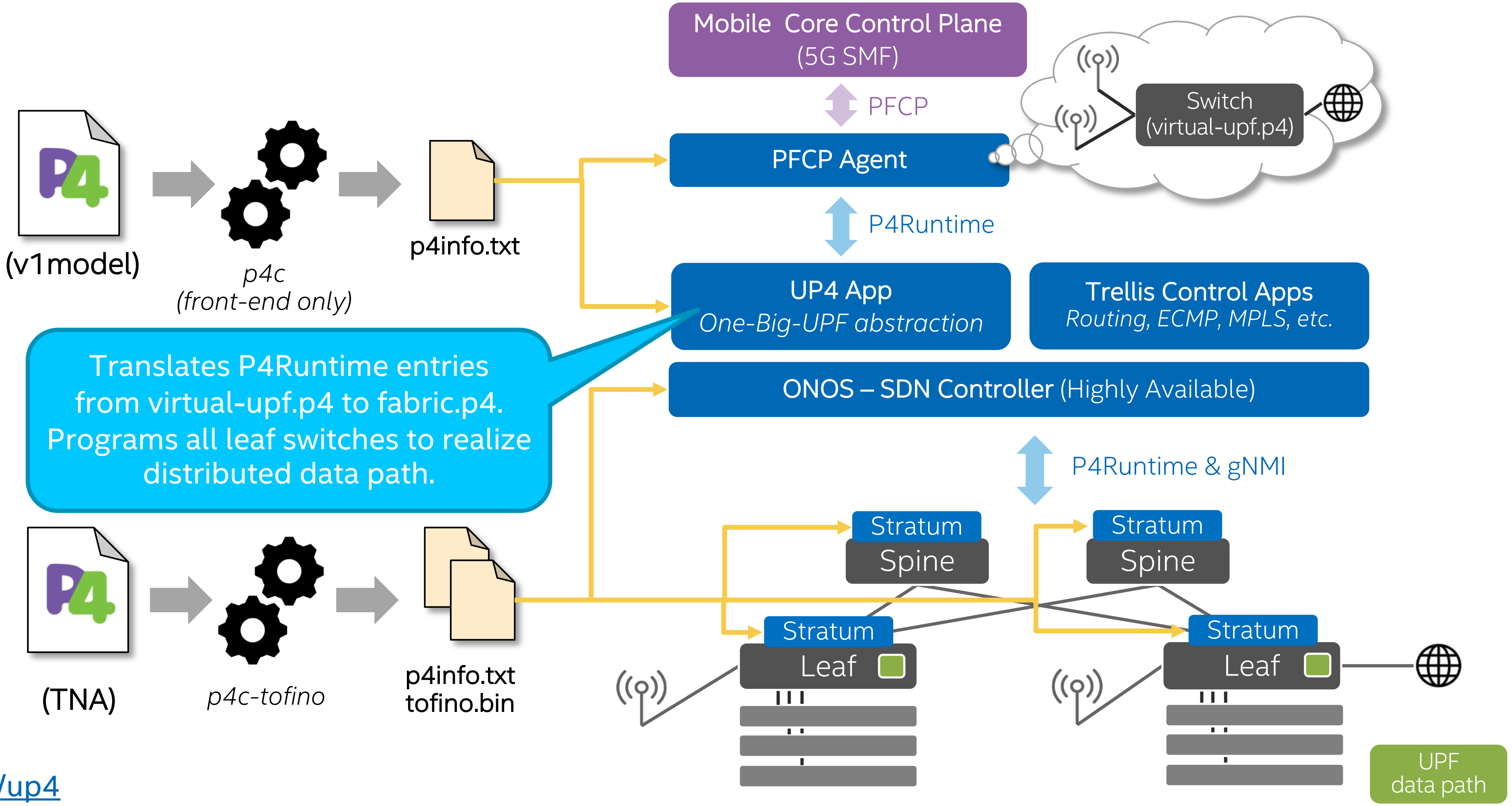


PFCP: Packet Forwarding Control Protocol
(3GPP standard interface)

Role of UP4 App

virtual-upf.p4
Defines only UPF tables, not optimized for any HW target

fabric.p4
Optimized for Tofino. Defines tables for UPF, routing, ECMP, MPLS, INT, etc.

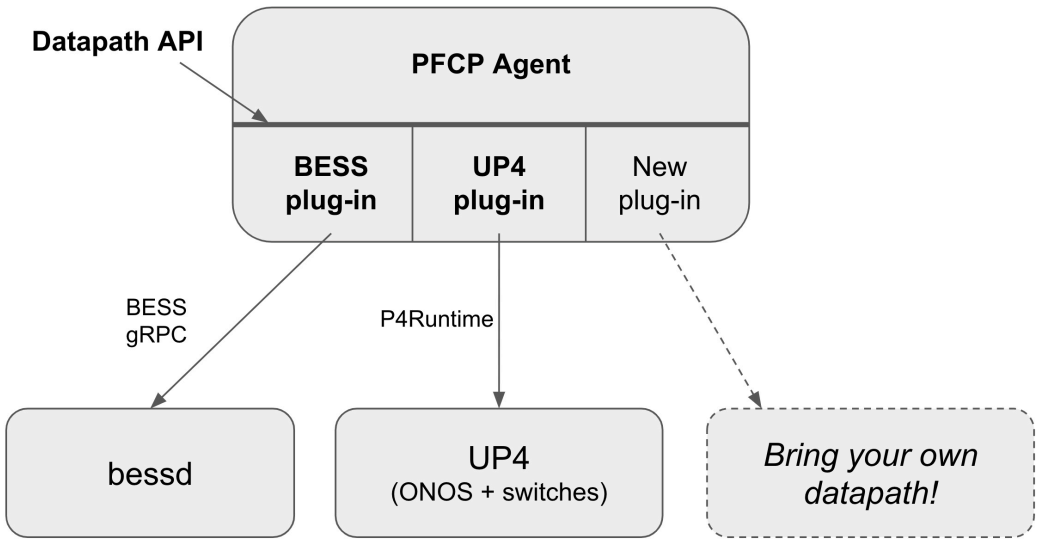


Translates P4Runtime entries from virtual-upf.p4 to fabric.p4. Programs all leaf switches to realize distributed data path.

https://github.com/omec-project/up4

Role of PFCP Agent

- Go-based micro-service
- Implement complex PFCP protocol once, for many data paths
- Main functions:
  - PFCP session handling
  - UE IP address allocation
  - Volume/time-based triggers for Usage Reporting Rules (URR)
  - Etc.
- Support multiple southbound protocols via plug-in mechanism

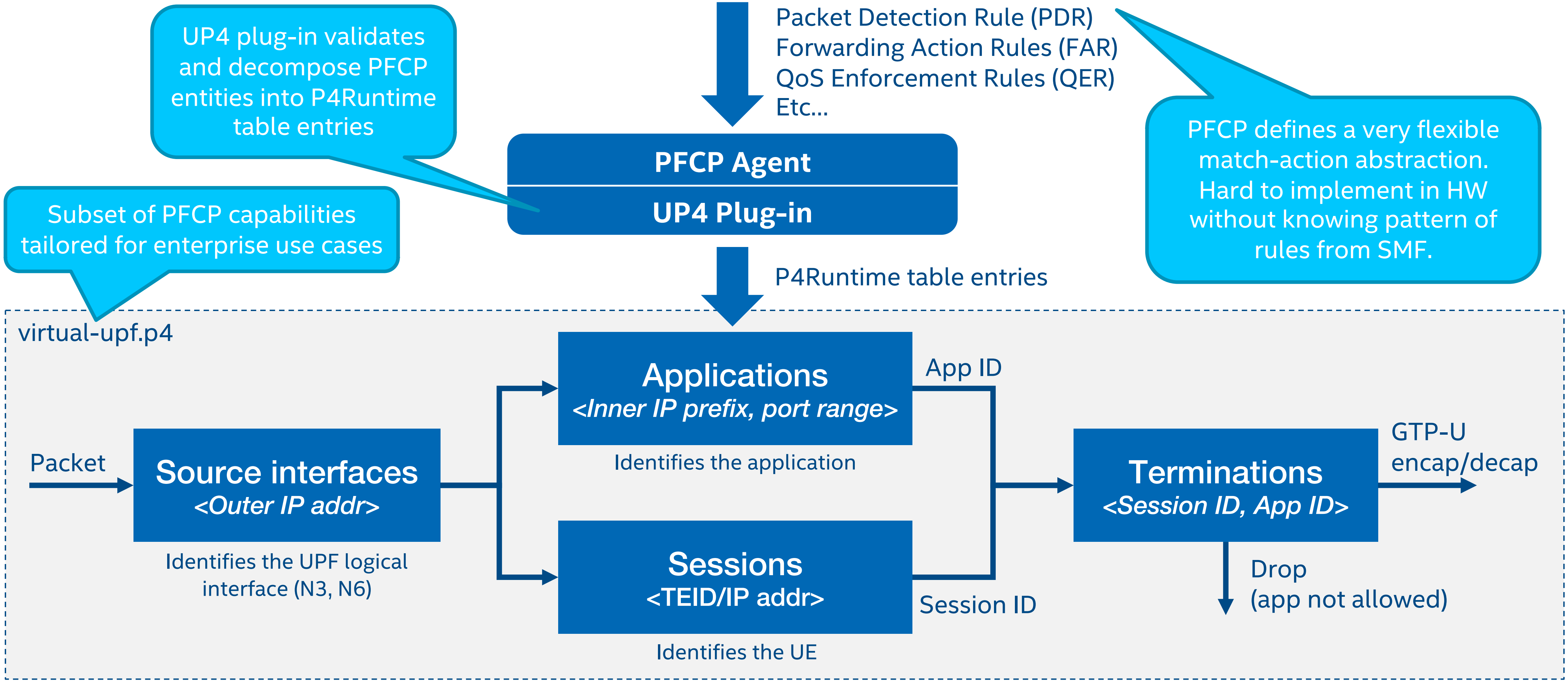


https://github.com/omec-project/upf

UPF P4 Pipeline Design

With an aside on fabric.p4

UP4 Logical Pipeline

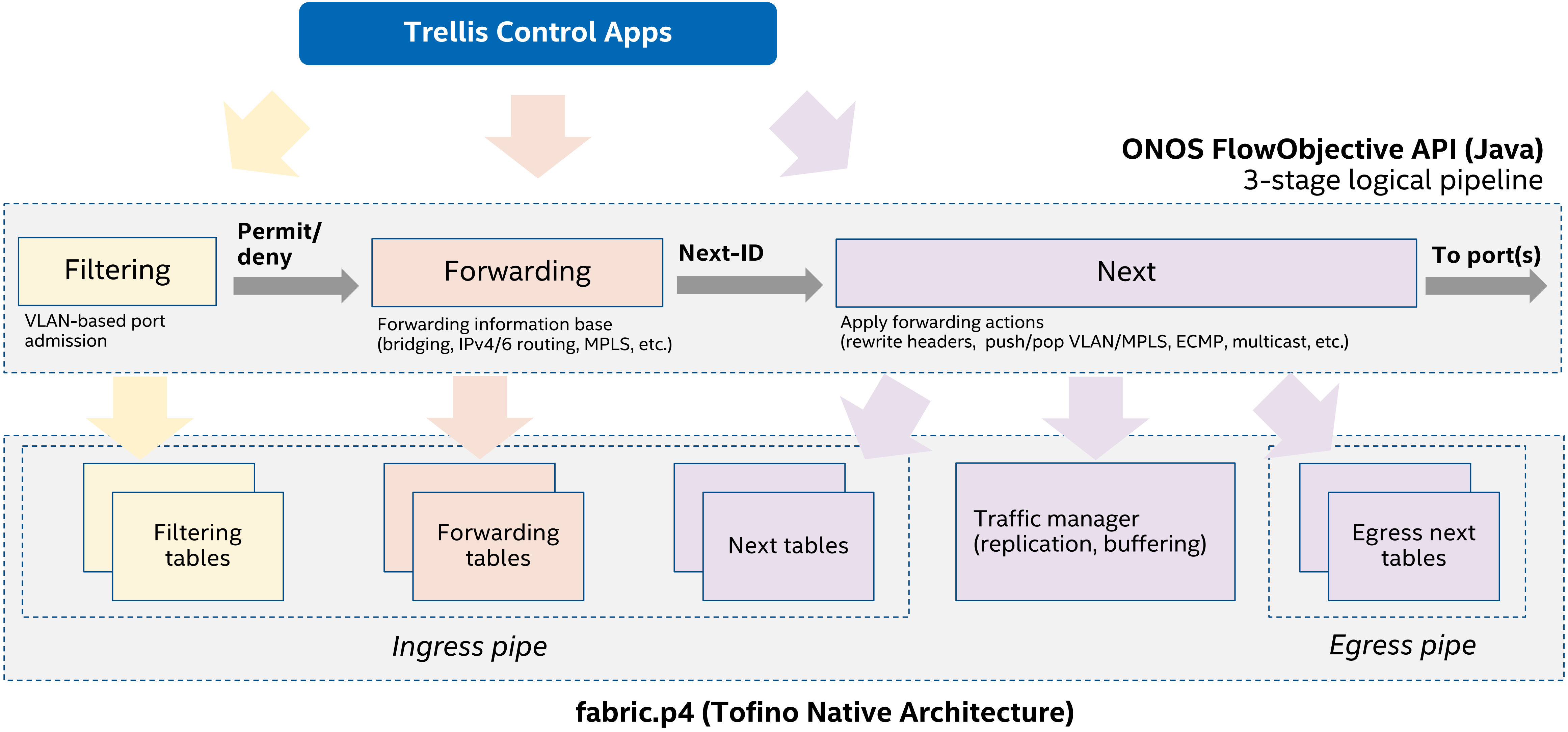


UP4 plug-in validates and decompose PFCP entities into P4Runtime table entries

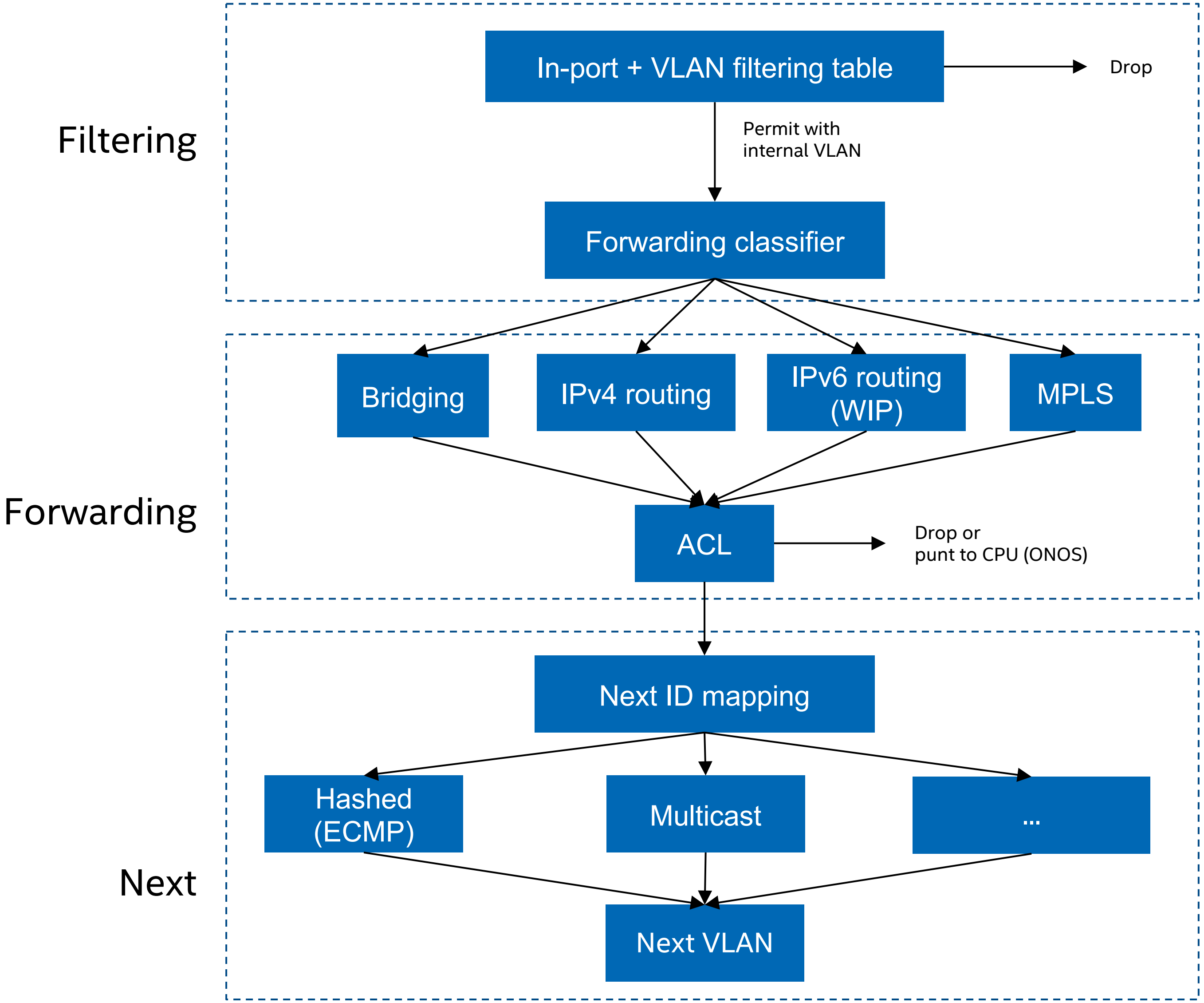
Subset of PFCP capabilities tailored for enterprise use cases

PFCP defines a very flexible match-action abstraction. Hard to implement in HW without knowing pattern of rules from SMF.

An Aside: Fabric.p4 Design Rationale



Fabric.p4 Tables (Simplified)



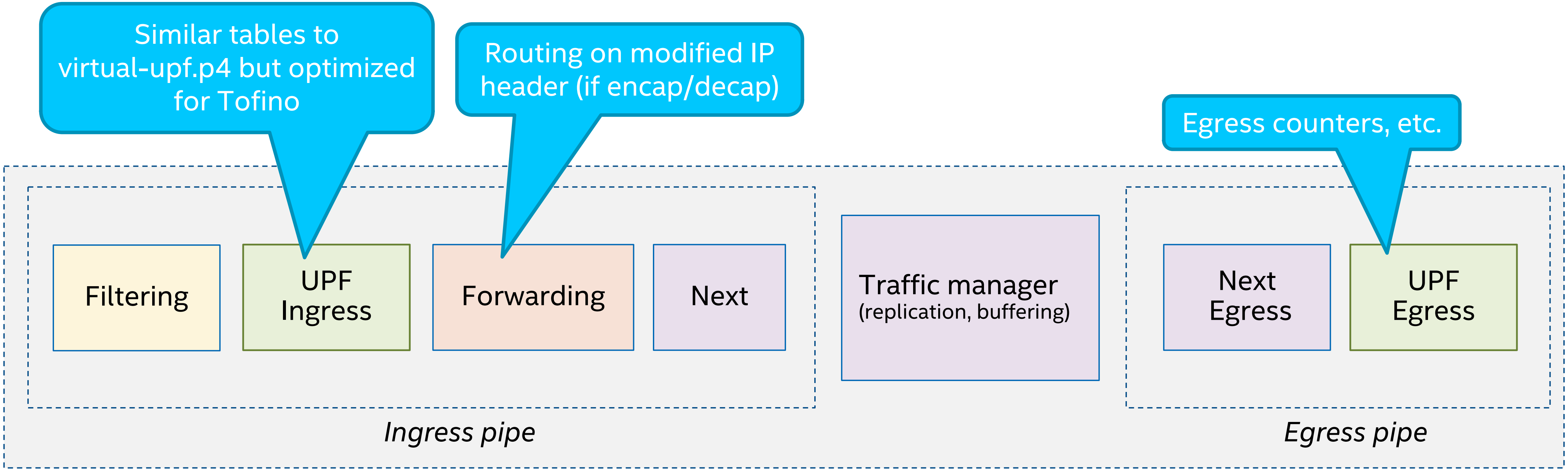
Compile-Time Profiles

- Same P4 program, multiple profiles
- Choose which capabilities to include via p4c preprocessor flags

Profile name	p4c preprocessor flags	Description
fabric	*None*	Basic fabric profile
fabric-upf	-DWITH_UPF	With UPF tables
fabric-int	-DWITH_INT	With Inband-Network Telemetry (INT) spec v0.5
fabric-upf-int	-DWITH_UPF -DWITH_INT	With both UPF and INT functions

https://github.com/stratumproject/fabric-tna

UPF Integration with Fabric.p4


Similar tables to virtual-upf.p4 but optimized for Tofino
Routing on modified IP header (if encap/decap)
Egress counters, etc.
Filtering
UPF Ingress
Forwarding
Next
Traffic manager (replication, buffering)
Next Egress
UPF Egress
Ingress pipe
Egress pipe

fabric.p4 (fabric-upf profile)

P4-UPF Summary

What we talked about

- Distributed UPF data path
- Integration with 5G mobile core via:
  - PFCP-Agent: multiple southbound plug-ins
  - UP4 ONOS app: One-Big-UPF abstraction
- Two P4 programs:
  - Virtual-upf.p4: logical, API data model for UP4
  - Fabric.p4: runs on Tofino

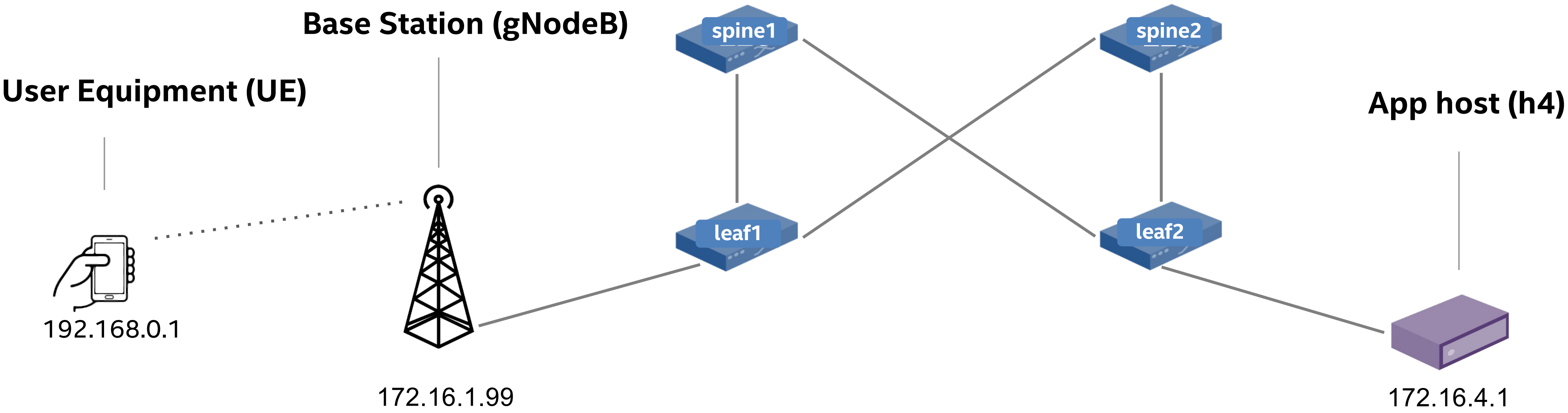
What we didn’t talk about

- Idle-mode buffering
- Slicing & QoS
  - Dedicated tutorial session soon
- INT integration
  - Dedicated tutorial sessions soon
- Further reading:
  - docs.sd-fabric.org/master/advanced/p4-upf.html
  - R. MacDavid et al. A P4-based 5G User Plane Function, SOSR 2021

Exercise 2

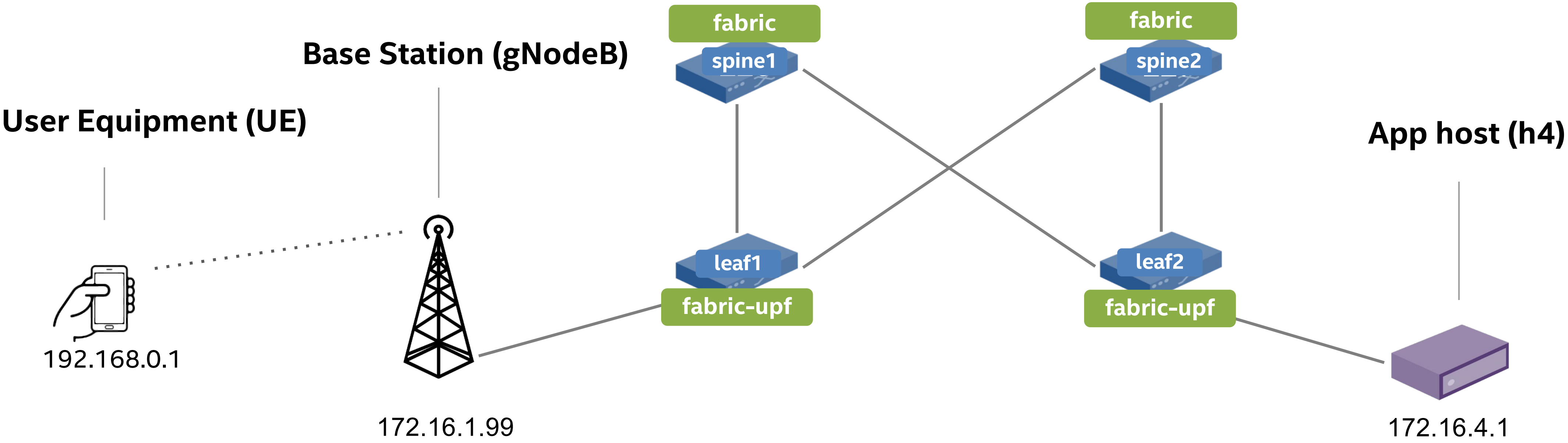
GTP-U Tunnel Termination with P4-UPF

Exercise 2 Overview



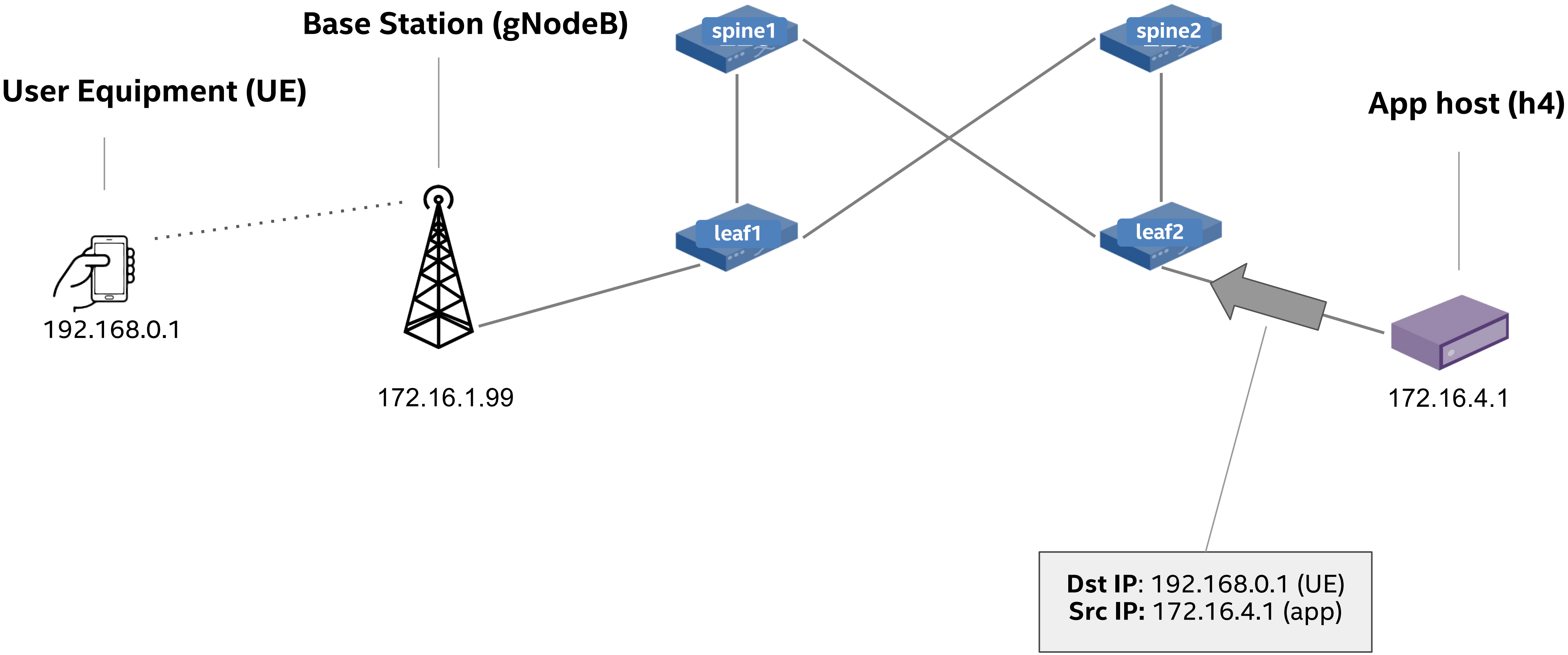
Same 2x2 leaf-spine fabric as in Exercise 1. We will use only two hosts: gNodeB (emulated) and app host

Exercise 2 Overview



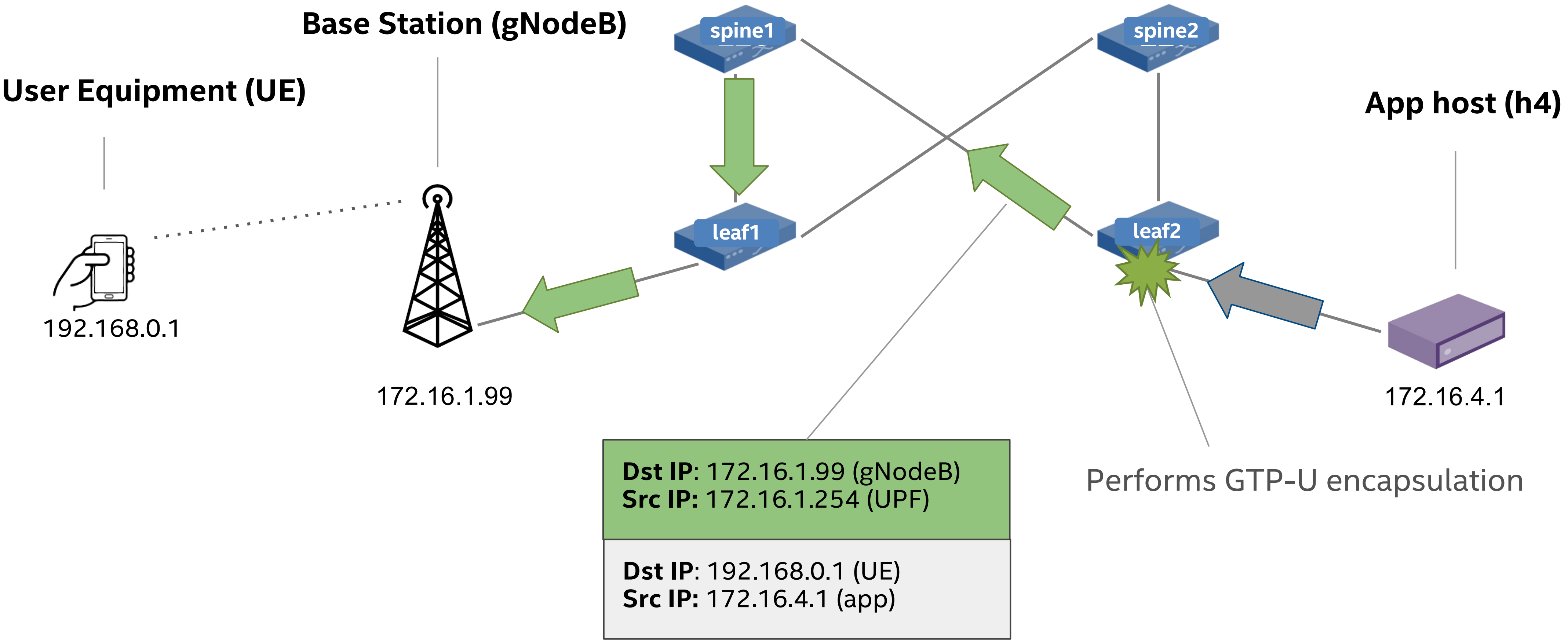
UPF function distributed on leaf1 and leaf2
(using fabric-upf pipeconf)

Exercise 2 Overview



Dst IP: 192.168.0.1 (UE)
Src IP: 172.16.4.1 (app)

Exercise 2 Overview



Dst IP: 172.16.1.99 (gNodeB)
Src IP: 172.16.1.254 (UPF)

Dst IP: 192.168.0.1 (UE)
Src IP: 172.16.4.1 (app)

Performs GTP-U encapsulation

P4-UPF Workflow

3

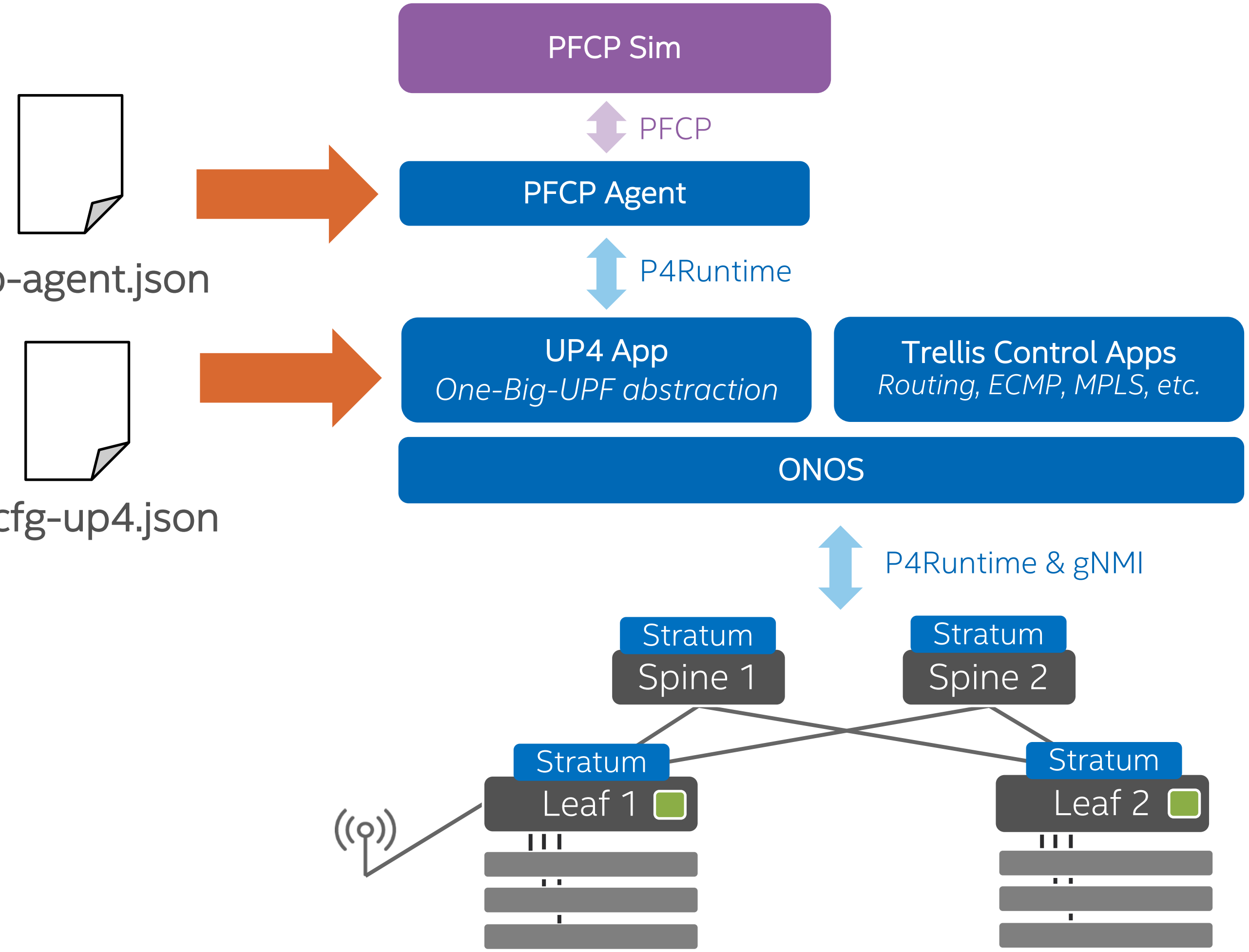
- PFCP Association
- PFCP Session Establishment/Modification/Deletion

2

- Set UE subnet
- Set UPF IP address (N3)
- Set UP4 P4Runtime server

1

- Set UPF switches



PFCP: Packet Forwarding Control Protocol (3GPP standard interface)

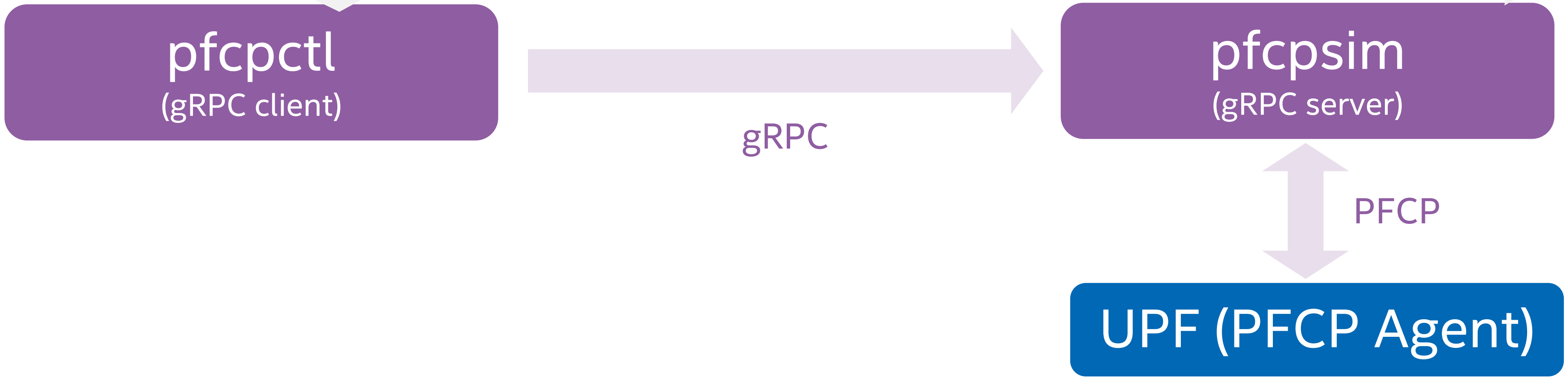
github.com/opennetworkinglab/sdfabric-tutorial

PFCP Sim

- Emulates 5G SMF
- CLI interface to manually set up UE sessions

```
pfcpctl service associate
pfcpctl session create --ue-pool 192.168.0.0/16 --gnb-addr 172.16.1.99
pfcpctl session modify --ue-pool 192.168.0.0/16 --gnb-addr 172.16.1.99
…
```

Handles keepalives, session bookkeeping, etc.

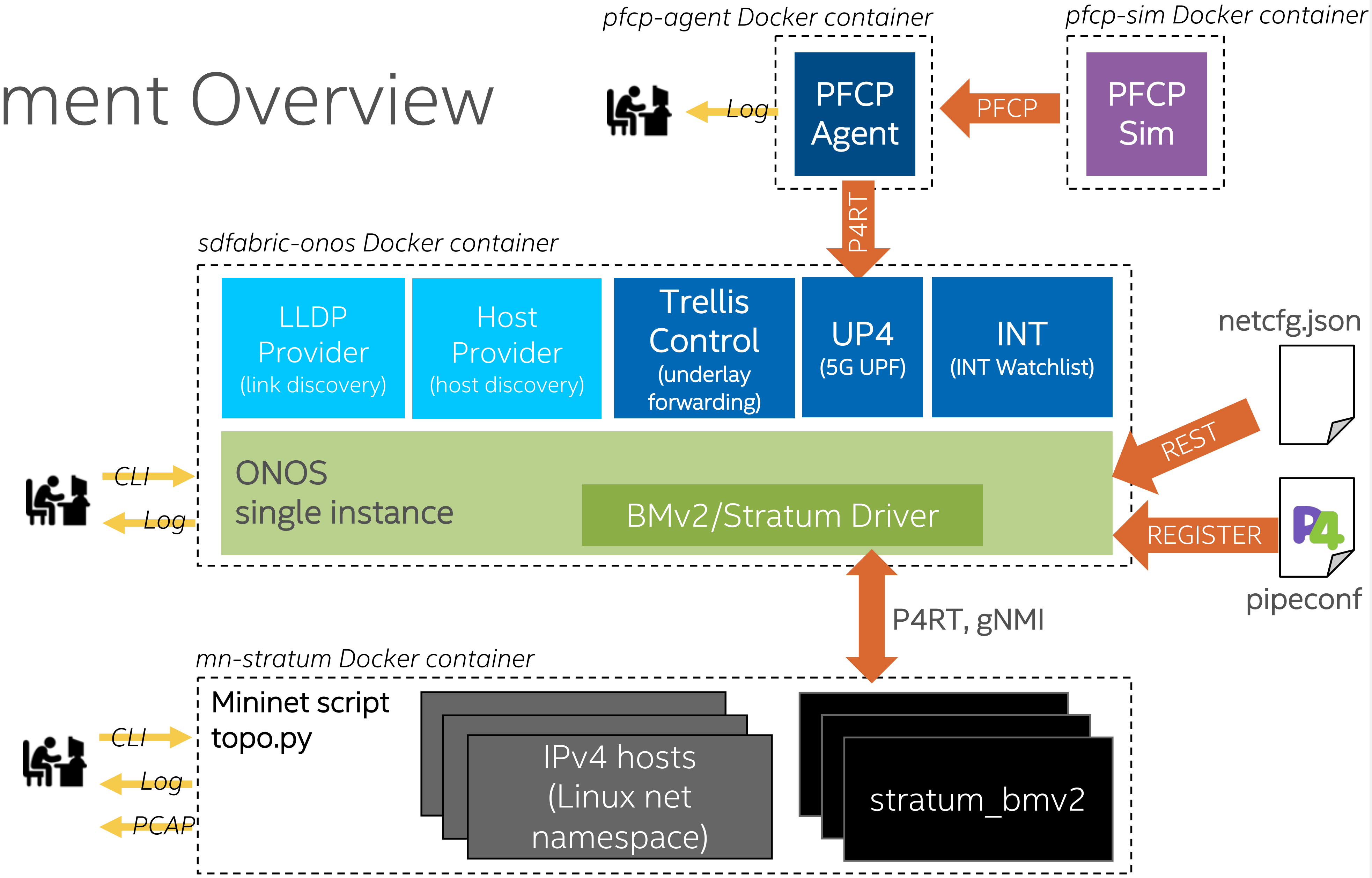


https://github.com/omec-project/pfcpsim

Environment Overview

Useful Commands

```
make deps
make start
make start-upf
make netcfg
make onos-cli
make onos-log
make mn-cli
make mn-log
make mn-pcap
make pfcp-log
```



pfcp-agent Docker container
pfcp-sim Docker container
Log
PFCP Agent
PFCP
PFCP Sim
P4RT
sdfabric-onos Docker container
LLDP Provider (link discovery)
Host Provider (host discovery)
Trellis Control (underlay forwarding)
UP4 (5G UPF)
INT (INT Watchlist)
netcfg.json
REST
CLI
Log
ONOS single instance
BMv2/Stratum Driver
REGISTER
pipeconf
P4RT, gNMI
mn-stratum Docker container
Mininet script topo.py
IPv4 hosts (Linux net namespace)
stratum_bmv2
CLI
Log
PCAP

github.com/opennetworkinglab/sdfabric-tutorial

Exercise 2 Steps

- Modify configuration files
- Start PFCP Agent
- Use pfcpctl to set up UE session
- Use Python scripts to generate and sniff traffic
- Verify that switch is performing GTP-U encapsulation as expected

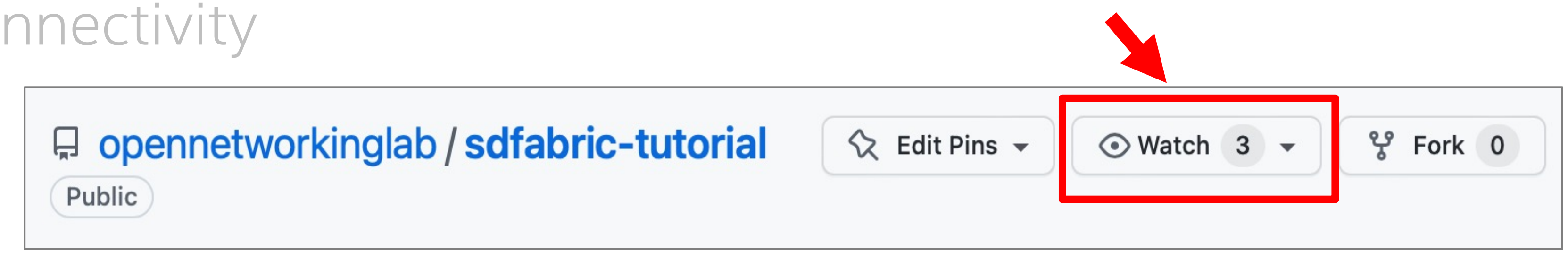
Exercise 2: Get Started

- Open lab README on GitHub
  - http://github.com/opennetworkinglab/sdfabric-tutorial
- Or open in text editor
  - sdfabric-tutorial/README.md
  - sdfabric-tutorial/EXERCISE-2.md
- Solution
  - sdfabric-tutorial/solution

That’s All For Now!

- Part 1 – Introduction to SD-Fabric: motivation, architecture, use cases
- Part 2 – Basics & Configuration + hands-on lab
- Part 3 – P4 User Plane Function (UPF) + hands-on lab
- Part 4 – In-band Network Telemetry (INT)
- Part 5 – Extending SD-Fabric
- Part 6 – Slicing & QoS
- Part 7 – Advanced Connectivity
- And more…

More sessions and labs on the way!
Make sure to watch the GitHub repo
github.com/opennetworkinglab/sdfabric-tutorial



Notices & Disclaimers

- Intel technologies may require enabled hardware, software or service activation.
- No product or component can be absolutely secure.
- Your costs and results may vary.
-