Benchmarks: Pipelines

Benchmarks: Goals

- Typical telco pipelines
- Can be coded into most of our switches/data-planes (with documented restrictions)
- Allow realistic dynamic workloads (add/del entries to/from flow table)
- Performance illustrative in general
- Currently BESS and OVS are to be supported (with prefab the pipeline config), Lagopus and ERFS should be added (Vpp would be very difficult)

Benchmarks: Pipelines

- Macro-benchmarks:
  - Mobile gateway (5G?) – more or less ready
  - Broadband Network Gateway (BNG) – work-in-progress
- Micro-benchmarks: what is a micro-benchmark after all?
- Possible (not so) micro-benchmarks:
  - port forward
  - L2
  - L3
  - encap/decap
  - ????

Mobile Gateway

- Put users behind the base stations (bsts) to the Internet

- Mobile Gateway, different UL/DL
  - UL (from UE): GTP encapsulated packets
    1. L2, L3 and L4 check (GW, UDP, 2152)
    2. GTP decaps, save TEID
    3. rate limit per bearer (TEID)
    4. L3 routing towards Internet + L2 fwd
  - DL (to UE): normal IP packets
    1. L2 and L3 check (DstIP is in UE range)
    2. DstIP -> rate limit + GTP encaps + set ext DstIP
    3. L3 routing towards BSTs + L2 fwd

- Traces:
  - Uplink: 10k GTP packets from 100 Base Stations (BSTs)
    - external: DstIP = GW, DMAC = GW, SrcIP = BST(i)
    - internal: DstIP = server (1k), SrcIP = UE (10k)
  - Downlink: 10k IP packets, SrcIP = server, DstIP = UE
  - packet sizes:
    - 64, 128, 256, 512, 1024, 1280
  - filenames:
    - TBD

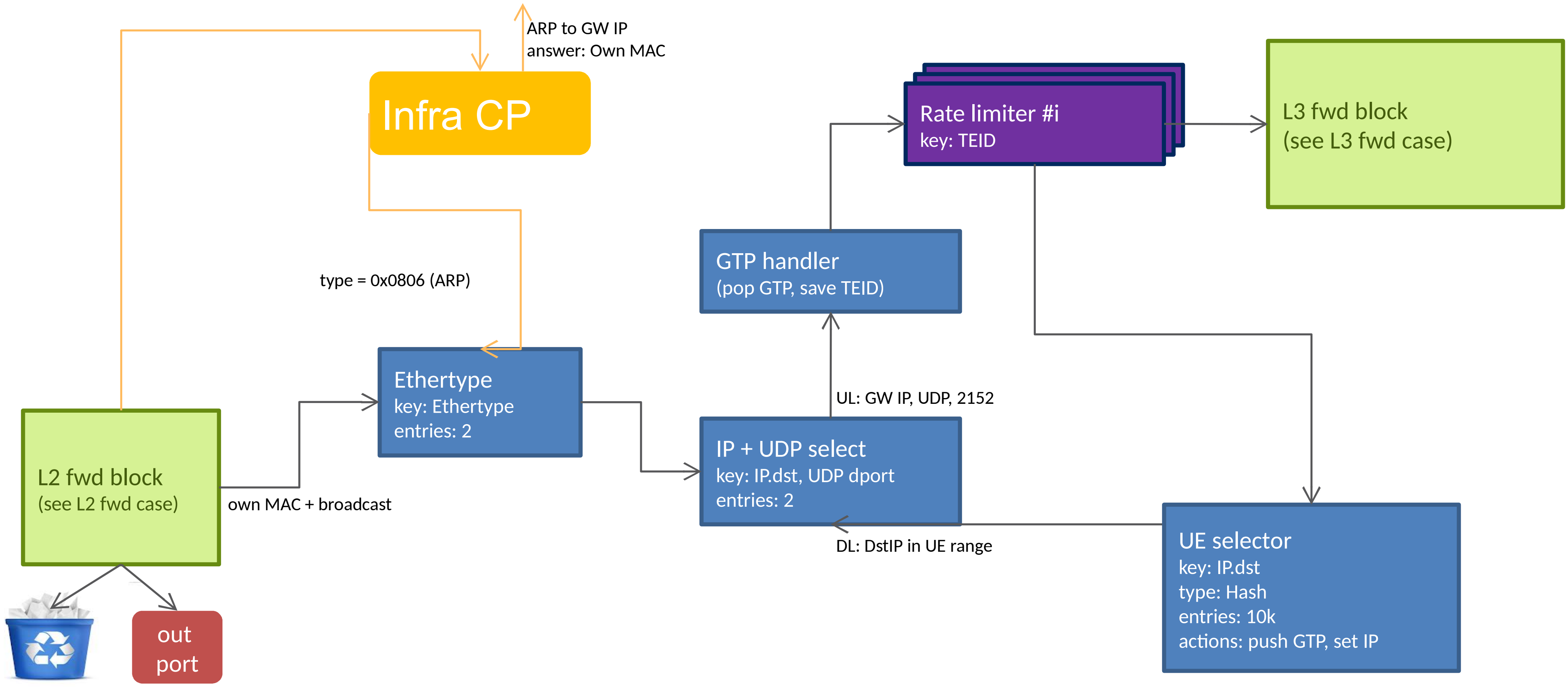
- Configuration
  - before test
  - use traces + next slide
  - set S/DMAC
  - set IP addresses + TEIDs
    - use same TEIDs UL/DL
  - do not do fragmentation (limit MTU)
  - routes should match traffic

- Tests:
  - 2 x uni + bi-directional
  - 1-2-4-8-… cores
  - 64-1280 byte packets
    - to avoid fragmentation
    - (if needed we can test it later)

- In: physical 10/40/100G
- Out: physical 10/40/100G

Mobile Gateway: Static config

- Uplink: GTP decap → rate-limit → L3 lookup → group
- Downlink: id user → GTP encap → L3 lookup → bst



BESS: Considerations

- **Egress/ingress traffic mapped** to a single physical port; **drop port** configurable, either to default port (for rate measurements), or real drop port
- **No GTP in BESS:** substituted with VxLAN
- **Rate limiters in BESS:** no rate limiter *per se*, emulated with a Queue + Scheduler
- **Uplink/downlink go through a single per-user rate limiter:** currently downlink rate limiting is for packets with VXLAN header, downlink for packets without VXLAN header

BESS: Considerations

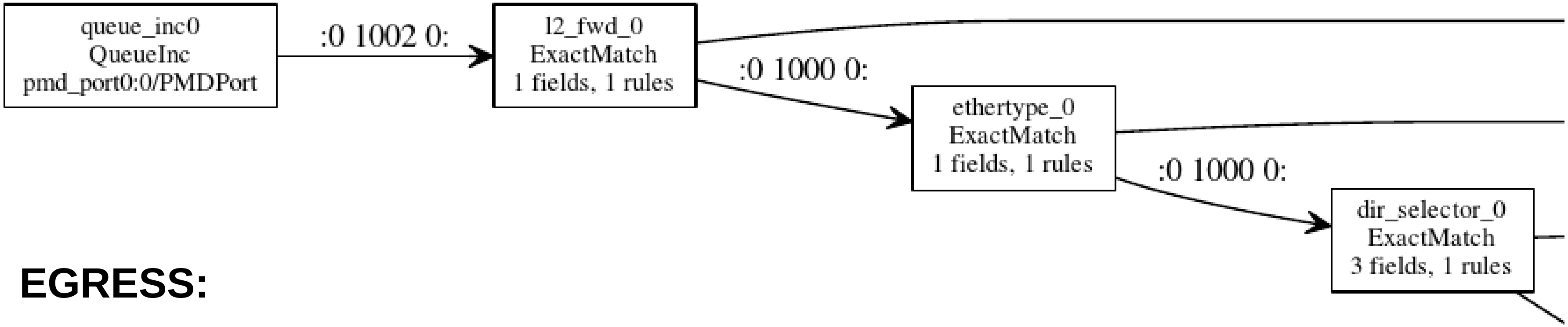
- **Scheduling:** the pipeline is scheduled from the input port and runs in run-to-completion mode, with users' queues separately

```
$ show tc
<worker 0>
  +-- !default_rr_0            round_robin
      +-- t_0_0                rate_limit          100000.000 Mbps
      |   +-- !leaf_rl_0_0:0   leaf
      +-- t_0_1                rate_limit          100000.000 Mbps
      |   +-- !leaf_rl_0_1:0   leaf
      +-- t_0_2                rate_limit          100000.000 Mbps
      |   +-- !leaf_rl_0_2:0   leaf
      +-- !leaf_queue_inc0:0   leaf
```

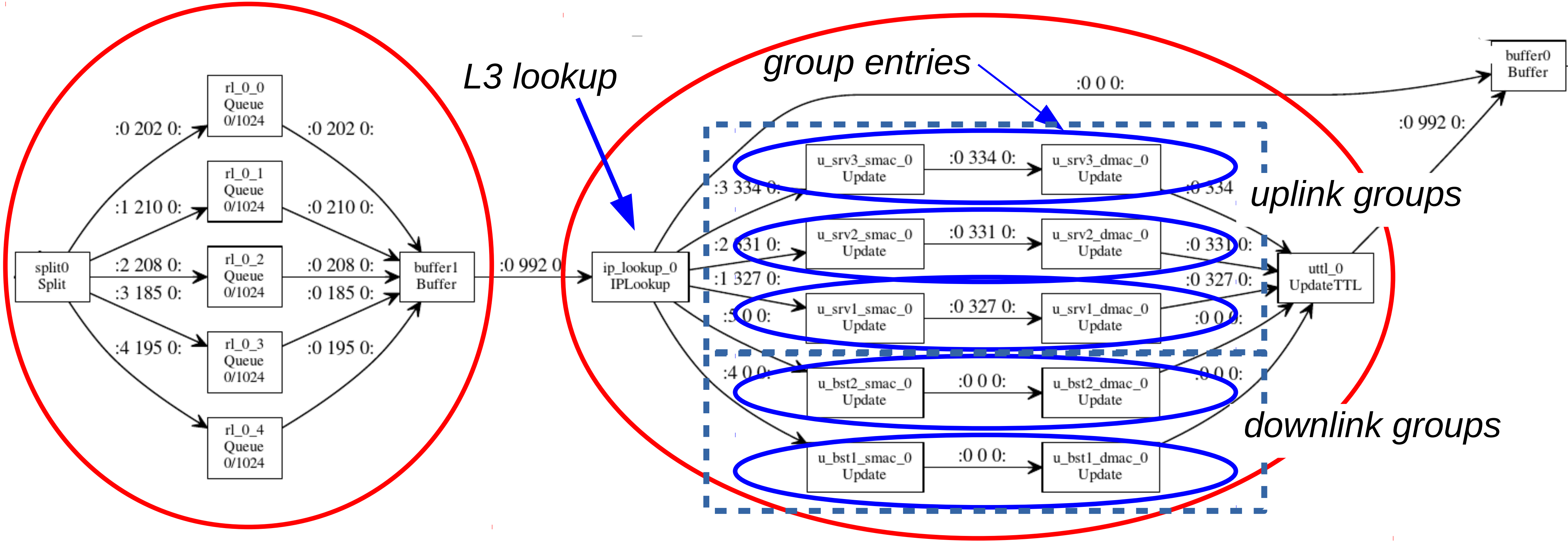
- *3 users, users' queues (t_0_?) are rate limited*
- *scheduled together with the input queue in round-robin*

BESS pipeline

INGRESS:



EGRESS:



L3 lookup

group entries

uplink groups

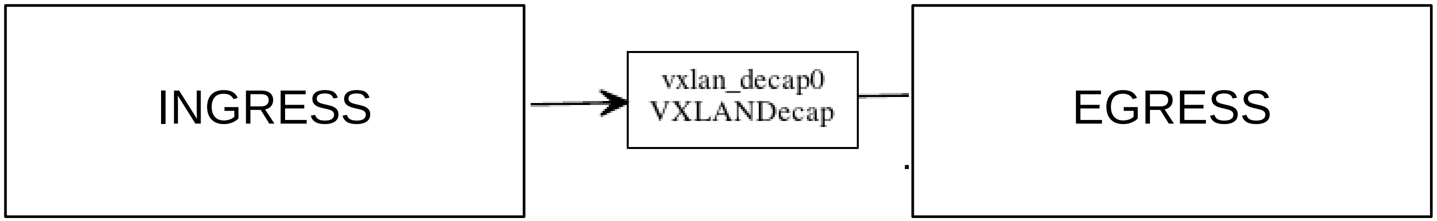
downlink groups

Rate limiters (per user)

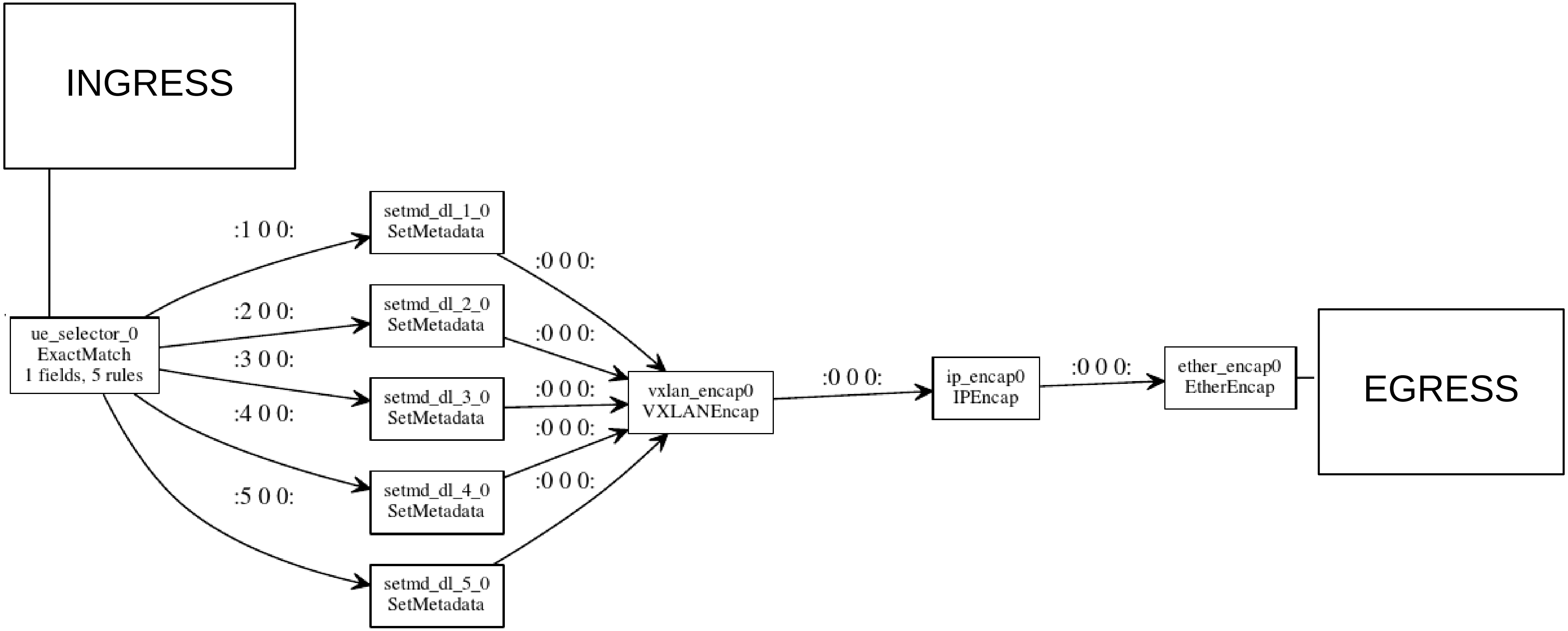
L3 fwd block

BESS pipeline

UPLINK:



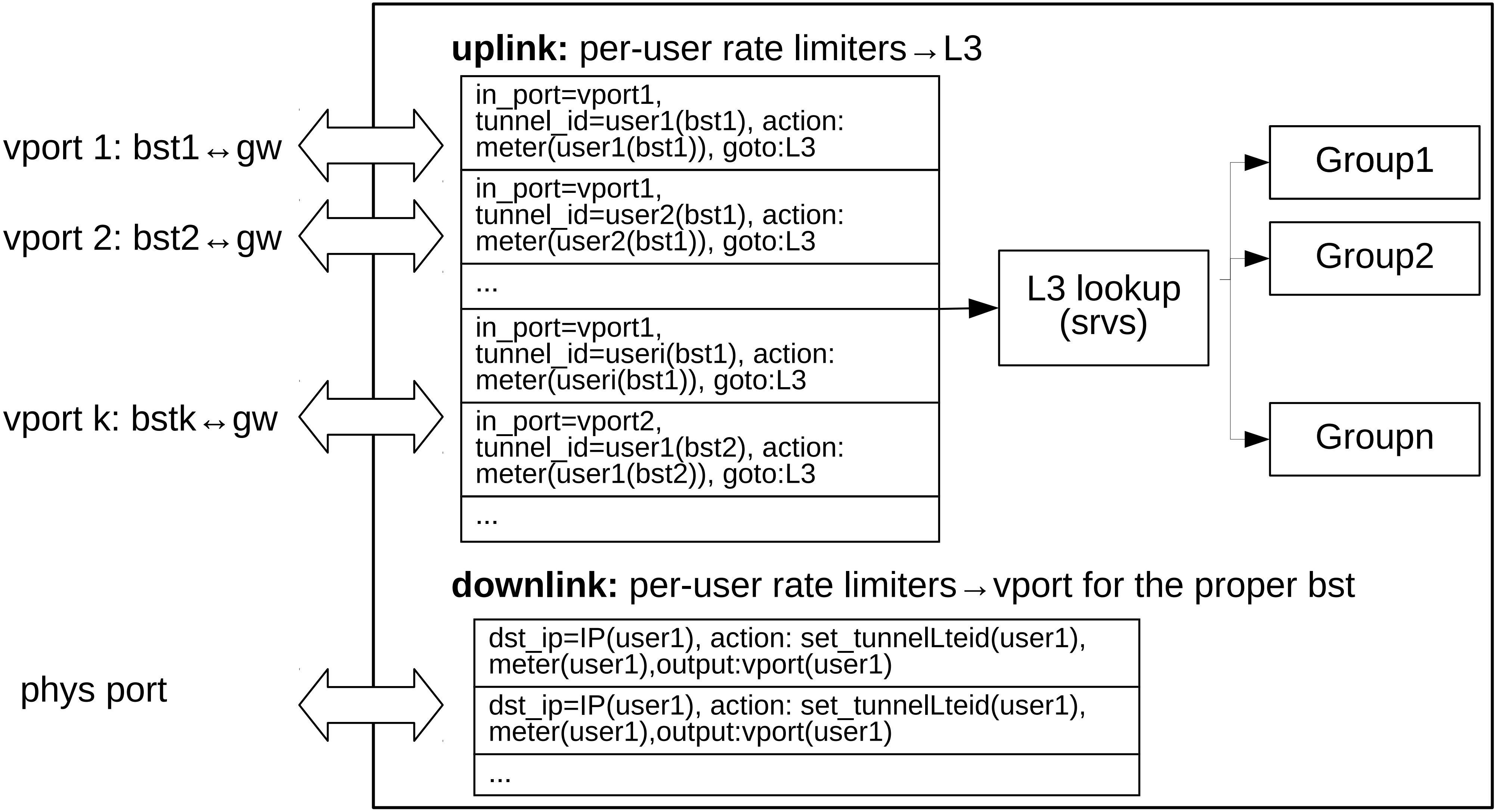
DOWNLINK:



OVS: Considerations

- **Egress/ingress traffic mapped** to a single physical port; **drop port** configurable, either to default port (for rate measurements), or real drop port
- **OpenFlow 1.3:** should be supported by OVS, Lagopus, ESwitch/ERFS
- **GTP → VXLAN:** to be on par with the BESS pipeline
- VXLAN via standard **OpenFlow "virtual port" abstraction** (OXM_OF_TUNNEL_ID): no need to explicit encap/decap but we lose OpenFlow compatibility (Lagopus/ERFS??)
- **Rate limiters:** per user OpenFlow "meters"

OVS pipeline



vport 1: bst1↔gw

vport 2: bst2↔gw

vport k: bstk↔gw

phys port

uplink: per-user rate limiters→L3

in_port=vport1,
tunnel_id=user1(bst1), action:
meter(user1(bst1)), goto:L3

in_port=vport1,
tunnel_id=user2(bst1), action:
meter(user2(bst1)), goto:L3

...

in_port=vport1,
tunnel_id=useri(bst1), action:
meter(useri(bst1)), goto:L3

in_port=vport2,
tunnel_id=user1(bst2), action:
meter(user1(bst2)), goto:L3

...

L3 lookup
(srvs)

Group1

Group2

Groupn

downlink: per-user rate limiters→vport for the proper bst

dst_ip=IP(user1), action: set_tunnelLteid(user1),
meter(user1),output:vport(user1)

dst_ip=IP(user1), action: set_tunnelLteid(user1),
meter(user1),output:vport(user1)

...

Dynamic workloads

- Goal: measure impact of flow table changes to performance
- Scenarios:
  - handover (user changes base station)
  - user add/delete: users disappear/appear
  - server changes (L3 destination entries change)
- Considerations:
  - change rate: perform the entire change in one shot every second (in one batch)
  - dynamism: add/delete the same entry (e.g., user comes, same user leaves)
- BESS: from bess config script, OVS: from Ryu

How to use

- Clone git tree: TODO

1) **Generate config** (10 users, 5 servers, 5 bsts, 4 handovers/sec, 5 usermods/sec)

```
$ gen_scripts/mgw-gen-conf.py -u 10 -s 5 -b 5
--handovers 4 --fluctusers 5 -o mgw-updates.json
```

2) **Generate pcaps** (200 thousand packets, each of 100 bytes, uplink+downlink, ASCII dump)

```
$ gen_scripts/mgw-gen-pcap.py -c mgw-updates.json
--pkt-num=200000 --pkt-size 100 -a --dir b
```

3) (Optionally) **start flow table modification scripts**

4) **Start measurement** (manually)

BNG

- TODO