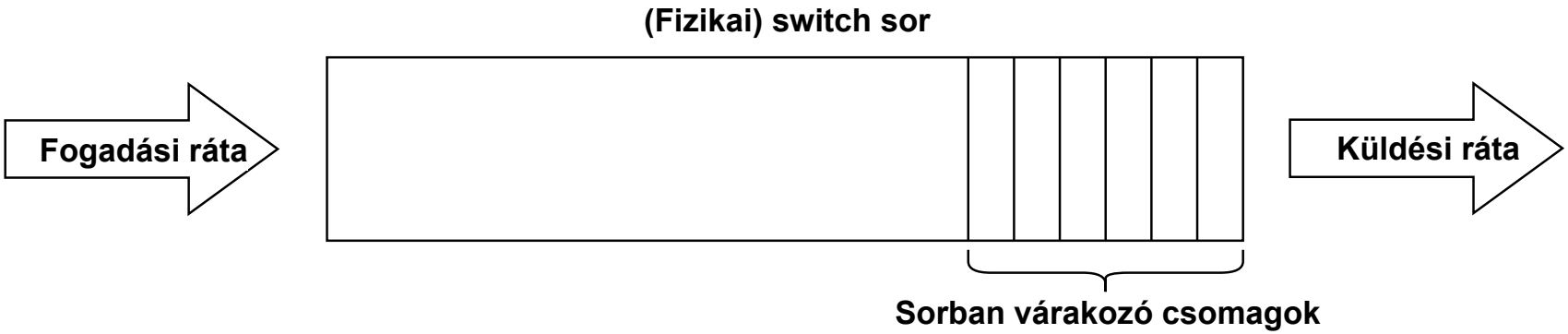
Per-flow virtual queues

Kámán Rebeka Sárközi Gergely

2024. 05. 13.

Fizikai sor



- Börsztös forgalom miatt van rá szükség
- Egy bizonyos rátával ürül
- Megtelik, ha a fogadási ráta tartósan meghaladja a küldési rátát
- Egy hosszú sor késleltetést okoz, ezt szeretnénk minimalizálni

Virtuális sor

- Emulál egy sort, ami a fizikai sornál kicsit lassabban ürül
- A sávszélesség kicsit kevesebb (például $\sim 98\%$)
- Nem telítődik meg, nem lesz túl hosszú:
  - TCP folyamok $\sim 98\%$-os sávszélességet fogják csak kihasználni
  - Börsztös forgalom miatt néha pár csomag várakozik
  - Átlagosan legalább $\sim 2\%$-kal gyorsabban ürül, mint telítődik

Feladatleírás

- Minden (aktív) folyam számára egy virtuális sor
  - Virtuális és fizikai küldési ráták aránya: $0 < \alpha < 1$
  - Folyam azonosítás: hasítással vagy táblával
- Virtuális sor teli $\implies$ csomag eldobása
- Cikk[1]: How to Build a Virtual Queue from Two Leaky Buckets
  - Threshold-marking[2] alkalmazását ajánlja
  - Beépített P4 `meter` megfelelő
- Cikk[3]: The Native AQM for L4S Traffic
  - L4S (Low Latency, Low Loss, Scalable Throughput) AQM fejlesztése
  - Kimeneti portonként $\sim$ 98%-os virtuális sorokat javasol
- Per-flow virtual queue gyakorlatilag per-flow rate limiter?

[1] https://www.bobbriscoe.net/projects/ipe2eqos/pcn/vq2lb/vq2lb_tr.pdf
[2] https://datatracker.ietf.org/doc/html/rfc5670
[3] https://www.bobbriscoe.net/projects/latency/l4saqm_tr.pdf

Adatsík

- L3 forwarding a kiindulási alap:

```
table l3_forward { ... } // Filled by control plane
if (l3_forward.apply().miss) { drop(); return; }
```

- Folyam virtuális sorhoz rendelése:

```
hash(meta.flow_id, HashAlgorithm.crc32, 0,
    { /* 5-tuple */ }, (1 << FLOW_ID_T_WIDTH));
meta.vq_id = egress_port ++ meta.flow_id;
```

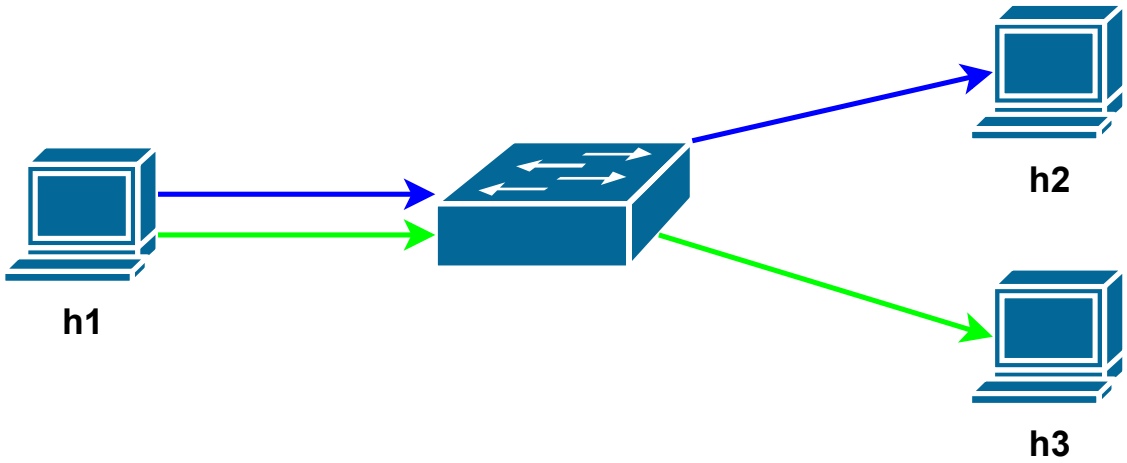
- Sor telítettség megállapítása és kezelése:

```
meter((1 << VQ_ID_T_WIDTH), MeterType.packets) vq;
vq.execute_meter(meta.vq_id, color);
if (color == METER_YELLOW) { hdr.ipv4.ecn = 0x11; }
else if (color == METER_RED) { drop(); }
```

Vezérlősík

- Python script
- L3 forwarding tábla feltöltése `topology.json` alapján
- "Fizikai" és virtuális sor konfigurálása (ráta, méret)
- Konstans alfa helyett dinamikus alfa érték?
  - $\alpha * \text{count(flows)} > 1 \implies$ fizikai sor megtelik
  - Folyamok száma *bloom filter*-rel becsülhető
  - Dinamikus alfa: $\alpha := max(\alpha_{\text{min}}, \frac{0.98}{\text{count(flows)}})$
  - Kevés flow esetén sokat javít $\alpha := \alpha_{\text{min}}$-hez képest

Hálózat



- L3 címkiosztási stratégia (`net.l3()`)
- Vezérlősík automatikus elindítása (`net.execScript(...)`)
- Automatikus traffik generálás Mininet task-ok és iperf3 segítségével
  - Kis- és nagyobb méretű folyamok vegyesen
  - Minden folyam adott ideig fut párhuzamosan
- Terv: DCTCP használata ECN támogatás érdekében

Szimulációs eredmények kiértékelése (terv)

- Adatsíkban log-olunk adatokat: 5-tuple, méret, queue delay, stb.
- Különböző grafikonok készítése
  - Kicsi és nagy folyamokhoz külön
  - Átküldött adatmennyiség (boxplot)
  - Queue delay kumulatív eloszlásfüggvénye
- Összehasonlítás:
  - Különböző α értékek
  - Virtuális sor folyamonként
  - Virtuális sor portonként
  - Virtuális sor nélkül