The Latency Characteristics of GTP-U and SRv6 Stateless Translation on VPP Software Router

Chunghan Lee¹), Naoyuki Mori²), Yasuhiro Ohara³), Tetsuya Murakami⁴), Shogo Asaba⁵), Satoru Matsushima⁶)

IEEE ADMNET 2021
July.12th.2021

Toyota Motor Corporation¹), Intel Corporation²), NTT Communications Corporation³)
Arrcus, Inc.⁴), NEC corporation⁵), SoftBank Corp.⁶)
Introduction

• Segment routing IPv6 (SRv6)
  • Encode the path in each packet
  • The SRv6 based on source routing has many advantages: stateless traffic steering, state reduction, network programming, and so on

The SRv6 is promising for the reduction of complexity and dependency on mobile network
Actual use case
GTP-U/SRv6 translation for mobile network

- Multiple stacking headers on the data plane of mobile network
  - Mobile devices are only communicated with UPF (GTP-U) on LTE network
  - After network failure, the efficient network path would not be selected due to the state of stacking headers

- Segment routing gateway (SRGW)
  - The stacking headers can be embedded to IPv6 address space and Segment Routing Header (SRH) and the efficient path can be selected
What is GTP-U and SRv6 Stateless Translation?

- All identifiers of a GTP-U tunnel (IPv4 addresses and TEID) can be embedded into the IPv6 address space
  - We can also use the SRH for multiple GTP-U headers
- The translation method (SRv6 for Mobile UserPlane) has been proposed in IETF
Our previous work¹)…

• We implemented the translation functions on the H/W P4 switch and measured the latency on the H/W P4 switch

However, the software-based SRv6 Mobile UserPlane would be better to deploy Edge/MEC or local 5G network (We consider other platforms, such as an DPDK-based software router)

¹) "Performance Evaluation of GTP-U and SRv6 Stateless Translation", 2nd Workshop on Segment Routing and Service Function Chaining (SR+SFC 2019)
DPDK-based software router (VPP)

**FD.io VPP**

- It is based on Vector Packet Processing (VPP) and an DPDK-based software router
  - Multiple packets are processed at one batch processing
  - The batch processing is called as *vectors/call*
- It is an open source software ([https://fd.io/](https://fd.io/))
  - It is easy to extend additional network functions (e.g., SRv6 functions)
- The VPP router is integrated to ODL, OPNFV, OpenStack, and K8s
  - It is also used as LB in production (Yahoo! Japan)
- **There are no evaluation results on the latency of the SRv6 for Mobile UserPlane**

We select the VPP router as our measurement platform for the translation latency
Research goal

• Evaluate the quantitative performance of SRv6 Mobile User Plane on the VPP software router

• Observe the latency characteristics on the VPP software router
How to measure the accurate latency on the VPP software router?

1. How can we accurately measure the latency when a receiving packet type is changed from the corresponding packet type sent?
   → Use the nano scale timestamp injection in the P4 switch with a pair of unique inner headers

2. What kind of latency characteristics do we observe on the software router?
   → Analyze evaluation results in detail
Overview

Experiments on VPP software router

• SRv6 translation functions are used for the latency measurement
  • **DUT**: the VPP software router
  • **Traffic generator (IXIA)**: a hardware-based commercial traffic generator (40Gbps)
  • **P4 switch**: write a packet timestamp to source mac address at mirrored packets
  • **Packet capture**: capture the mirrored packets without packet loss
Accurate latency measurement (P4 switch)

Nano scale timestamp using P4 switch

- During the packet mirroring, the nano scale timestamp is stored in the source MAC address field
  - The P4 switch overwrites and abandons the original source MAC address
VPP software router

GTP-U/SRv6 translation functions

- VPP graph nodes for GTP-U/SRv6 stateless translation
  - There are two IP address lookups from IPv4 to IPv6 (vice versa)
    - **GTP-U → SRv6** \((gtp4.d-plugin (4))\) : translates a GTP-U over IPv4 to an SRv6
    - **SRv6 → GTP-U** \((gtp4.e-plugin (10))\) : translates an SRv6 to a GTP-U over IPv4
  - The translation functions are already merged to the VPP master branch**

** https://github.com/FDio/vpp
Measurement scenarios

- We prepared the following conditions for the accurate latency measurement
  - One CPU core for packet processing thread with one NIC port
  - Input traffic loads: 1Mpps ~ 5Mpps (NO drop condition)
  - Two stateless translation functions
    - GTP-U → SRv6 (T.M.GTP4.D)
    - SRv6 → GTP-U (End.M.GTP4.E)
  - Three types of packet sizes
    - The short size: no payload
    - The middle size: intermediate size with the payload
    - The long size: the throughput (5 Mpps) is equal to the link capacity (40 Gbps)

<table>
<thead>
<tr>
<th></th>
<th>Short [bytes]</th>
<th>Middle [bytes]</th>
<th>Long [bytes]</th>
</tr>
</thead>
<tbody>
<tr>
<td>GTP-U (IPv4)</td>
<td>94</td>
<td>508</td>
<td>976</td>
</tr>
<tr>
<td>SRv6 (IPv6)</td>
<td>98</td>
<td>512</td>
<td>980</td>
</tr>
</tbody>
</table>

<The packet sizes for the translation latency measurement>
Latency measurement results on VPP router

The translation latency

- The translation latency is in the feasible range (roughly 3–30 μs)
- As the packet size is increased from the short to the long, the latency tends to increase slightly
- We also observe that the input load (Mpps) impacts the latency

<table>
<thead>
<tr>
<th></th>
<th>Input load [1Mpps]</th>
<th>Input load [5Mpps]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Short</td>
<td>3.3</td>
<td>3.9</td>
</tr>
<tr>
<td>Middle</td>
<td>3.5</td>
<td>4.1</td>
</tr>
<tr>
<td>Long</td>
<td>3.8</td>
<td>4.3</td>
</tr>
</tbody>
</table>

<The statistics of translation latency(SRv6→GTP-U)>

<The CDF of translation latency(SRv6→GTP-U)>
Latency measurement results on VPP router

Outgoing packet interval of translation latency

- The packet interval is a time period between packets on the VPP router
- When the input traffic load is increased to 5 Mpps, multiple packets are simultaneously queued and they are processed in a bulk manner
  - The approximately 80% packet intervals are close to zero (a few nanoseconds)

[Graph: The CDF of packet interval (GTP-U → SRv6)]

[Graph: The CDF of packet interval (SRv6 → GTP-U)]
Latency characteristics on VPP router

Batch processing model

• **Vectors/Call**: how many packets are processed in one batch processing cycle

• The jitter on the batch model is defined as follow

\[ Jitter = \text{Latency}_{\text{max}} - \text{Latency}_{\text{min}} \]

• To eliminate the unexpected latency fluctuation, we determine 95%tile latency \( (p_{95}) \) as \( \text{Latency}_{\text{max}} \) and 5%tile latency \( (p_{5}) \) as \( \text{Latency}_{\text{min}} \)

• The jitter depends on the Vectors/Call

![Input/output latency model on batch processing](image-url)
Latency characteristics on VPP router

Contribution factors of Vectors/Call – (1)

• We calculated the correlation coefficient (r) between the Vectors/Call and the following contribution factors
  • We mainly present the contribution factor (1)

<table>
<thead>
<tr>
<th>A list of contribution factor</th>
<th>Relationship using (r)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Input traffic load</td>
<td>Strong</td>
</tr>
<tr>
<td>2. CPU frequency</td>
<td>Strong</td>
</tr>
<tr>
<td>3. Translation function type</td>
<td>Strong</td>
</tr>
<tr>
<td>4. Packet sizes</td>
<td>No relationship</td>
</tr>
</tbody>
</table>

Refer to the contribution factors in the paper 😊
Latency characteristics on VPP router

Contribution factors of Vectors/Call – (2)

• Relation between the Vectors/call and the input traffic load
  • As the input traffic load linearly increases (e.g., 1 - 5 Mpps), the Vectors/Call and the jitter seem to increase exponentially

< GTP-U→SRv6 (short packet) >

< SRv6→GTP-U (short packet) >
Experiences on VPP router

The effect of hash space

• For flow load balancing, we implement the payload hashing

  - The results of hash space
    - We encountered the significant performance degradation on the middle and long sizes
    - We fixed the hashing space as 40 bytes (The inner IPv4 and TCP headers only)
      - The CPU time of hash function is maintained as approximately 15%
      - The source code is already contributed to the main official VPP (v20.05.1)

<table>
<thead>
<tr>
<th></th>
<th>Short packet</th>
<th>Middle packet</th>
<th>Long packet</th>
<th>Fixed (40 bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traffic load [Mpps]</td>
<td>5.00</td>
<td>2.72</td>
<td>1.69</td>
<td>5.00</td>
</tr>
<tr>
<td>CPU time ratio of hash function using Intel VTune [%]</td>
<td>15.1</td>
<td>55.7</td>
<td>72.9</td>
<td>15</td>
</tr>
</tbody>
</table>
Conclusion and future work

• Conclusion
  • Evaluate the latency of translation functions on the VPP router using nano-scale measurement with P4 switch
  • Latency characteristics
    • The translation latency is in the feasible range (roughly 3–30 μs)
    • The batch processing impacts the latency characteristics and the Vectors/Call is a key factor
    • The three major contribution factors of Vectors/Call:
      Input traffic load, CPU frequency, and translation function types
  • Our findings can help to improve the performance of software-based network system and to design beyond-5G mobile network systems

• Future work
  • Measure the latency on more complicated cases, such as multiple routing entries and multiple CPU cores at maximum performance
  • Evaluate the translation latency on other software routers
Backup slide
L3 forwarding vs Translation

Comparison of maximum throughput [Mpps]

- Measure the maximum throughput (L3 forwarding [IPv4] vs Translation)
  - There is **NO** packet loss and the single CPU core is used
  - L3 forwarding [IPv4]: 6.7 Mpps (100%)
  - SRv6/GTP-U Translation : 5.4 Mpps (79.1%)
  - GTP-U/SRv6 Translation : 5.3 Mpps (80.5%)
The DPDK-based software router (VPP)

Performance measurement

• Although there are extensive results on throughput, there seem to be little latency evaluation on the VPP software router

• Moreover, there are no evaluation results on latency for the SRv6 for Mobile UserPlane

<table>
<thead>
<tr>
<th>Benchmarked Workload</th>
<th>Throughput [Mpps]</th>
<th>Throughput [Mpps]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dedicated 1 physical core with =&gt;</td>
<td>noHT</td>
<td>HT</td>
</tr>
<tr>
<td>CoreMark [Relative to CMPS ref*]</td>
<td>0.99</td>
<td>1.35</td>
</tr>
<tr>
<td>DPDK-Testpmd L2 Loop</td>
<td>54.6</td>
<td>59.5</td>
</tr>
<tr>
<td>DPDK-L3Fwd IPv4 Forwarding</td>
<td>32.3</td>
<td>38.4</td>
</tr>
<tr>
<td>VPP L2 Patch Cross-Connect</td>
<td>23.0</td>
<td>28.1</td>
</tr>
<tr>
<td>VPP L2 MAC Switching</td>
<td>8.3</td>
<td>9.5</td>
</tr>
<tr>
<td>OVS-DPDK L2 Cross-Connect</td>
<td>7.2</td>
<td>10.9</td>
</tr>
<tr>
<td>VPP IPv4 Routing</td>
<td>12.8</td>
<td>14.8</td>
</tr>
</tbody>
</table>

< Throughput^2> of network applications>

2) Benchmarking Software Data Planes Intel® Xeon® Skylake vs. Broadwell
Why the translation is required?

- Multiple stacking headers on the data plane of mobile network
  - Stacking multiple small IDs to fulfill the requirements of reliability
  - After network failure, the efficient network path would not be selected due to the state of stacking headers

IPv6 as user PDN protocol
GTPv1U as mobile user-plane protocol
L3 VPN for mobile core and back-haul
L2 VPN for virtual networks
LSP for high quality and reliability

Consolidate all layer headers using SRH* into one single IPv6 layer!

*Segment Routing Extension Header (SRH)
Latency measurement setup (server)

- **H/W specification**
  - One CPU core is used for latency measurement

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPU</strong></td>
<td>Intel Xeon Gold 6126 (2.6 GHz) [19.25 MB L3 cache]</td>
</tr>
<tr>
<td><strong>Memory</strong></td>
<td>384 GB</td>
</tr>
<tr>
<td><strong>NIC</strong></td>
<td>Mellanox ConnectX-4</td>
</tr>
<tr>
<td><strong>Bandwidth</strong></td>
<td>40 Gbps</td>
</tr>
</tbody>
</table>

- **S/W configuration**
  - C state/P state are disable
  - CPU frequency is fixed (2.6 GHz)
    - Intel TurboBoost is also disable

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OS</strong></td>
<td>Ubuntu 18.04.2 LTS</td>
</tr>
<tr>
<td><strong>Kernel</strong></td>
<td>5.3.0-28</td>
</tr>
<tr>
<td><strong>VPP</strong></td>
<td>v20.05.1 (stable)</td>
</tr>
</tbody>
</table>
How to generate unique packets for latency measurement?

- We fixed outer headers and changed unique inner packet headers randomly and sequentially.

<table>
<thead>
<tr>
<th>Packet header</th>
<th>Header</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>GTP-U over IPv4</td>
<td>SIP</td>
<td>172.20.0.2 (fixed)</td>
</tr>
<tr>
<td>(Outer)</td>
<td>DIP</td>
<td>172.20.0.1 (fixed)</td>
</tr>
<tr>
<td>SRv6 over IPv6</td>
<td>SIP</td>
<td>c1::ac14:2:0:0 (fixed)</td>
</tr>
<tr>
<td>(Outer)</td>
<td>DIP</td>
<td>d4:0:ac14:1::c800:0 (fixed)</td>
</tr>
<tr>
<td>GTP-U (Outer)</td>
<td>TEID</td>
<td>10 (fixed)</td>
</tr>
<tr>
<td>IPv4 (Inner)</td>
<td>SIP</td>
<td>1.0.0.1 - 100.0.0.254 (incremental)</td>
</tr>
<tr>
<td>DIP</td>
<td>DIP</td>
<td>101.0.0.1 - 200.0.0.254 (incremental)</td>
</tr>
<tr>
<td>TCP (Inner)</td>
<td>SPort</td>
<td>1 - 65535 (random)</td>
</tr>
<tr>
<td></td>
<td>DPort</td>
<td>1 - 65535 (random)</td>
</tr>
</tbody>
</table>