Recommendations for using Multiple IP Addresses in Benchmarking Tests

draft-lencse-bmwg-multiple-ip-addresses

Gábor LENCSE lencse@sze.hu (Széchenyi István University) – presenter

Keiichi SHIMA keiichi.shima@g.softbank.co.jp (SoftBank)

IETF 121, BMWG, November 7, 2024.

Outline

- The progress of the draft
- Reminder
  - Problem description: why testing with multiple IP addresses is needed?
  - Recommended Solution: Usage of multiple, pseudorandom IP addresses
- Measurements: How many IP addresses should be used?
  - Description of the measurements, results, and conclusion
  - Recommendation: new test setups using gateways

Progress of the Draft

- “00” Presented at IETF 118 (in person)
  - Problem statement and recommendation of the usage of multiple IP addresses
- “01” Presented at IETF 119 (online)
  - Measurement results to prove the effectiveness of the solution
- “02” Minor update
- “03” current version at IETF 121 (in person)
  - New measurement setups using gateways
  - Investigation of the number of IP addresses to be used
  - Recommendation for testing according to the setup using gateways

Reminder: Problem Description: Conditions

- RFC 2544 has defined a test frame format with fixed IP addresses and fixed port numbers.
- RFC 4814 introduced pseudorandom port numbers, but it kept the usage of a single source and destination IP address pair when a single destination network is used.
- Receive Side Scaling (RSS) supports the receiving of multi-million packets per second by distributing the load among the CPU cores
  - Depending on implementation, the hash function includes:

    1st type: source IP address, destination IP address

    2nd type: source IP address, destination IP address, source port, destination port

Reminder: Problem Description: Unfairness

- RFC 4814 pseudorandom port numbers + 2nd RSS implementation
  - Works perfectly (port numbers ensure entropy)
  - All CPU cores are used, load is distributed approximately evenly
- RFC 4814 pseudorandom port numbers + 1st RSS implementation
  - Gives poor results (no entropy is ensured as IP addresses are fixed)
  - Thus only two CPU cores are used (one core per direction)
- However, network interconnect devices using the 1st RSS implementation work perfectly, when they forward Internet traffic (IP addresses ensure entropy)

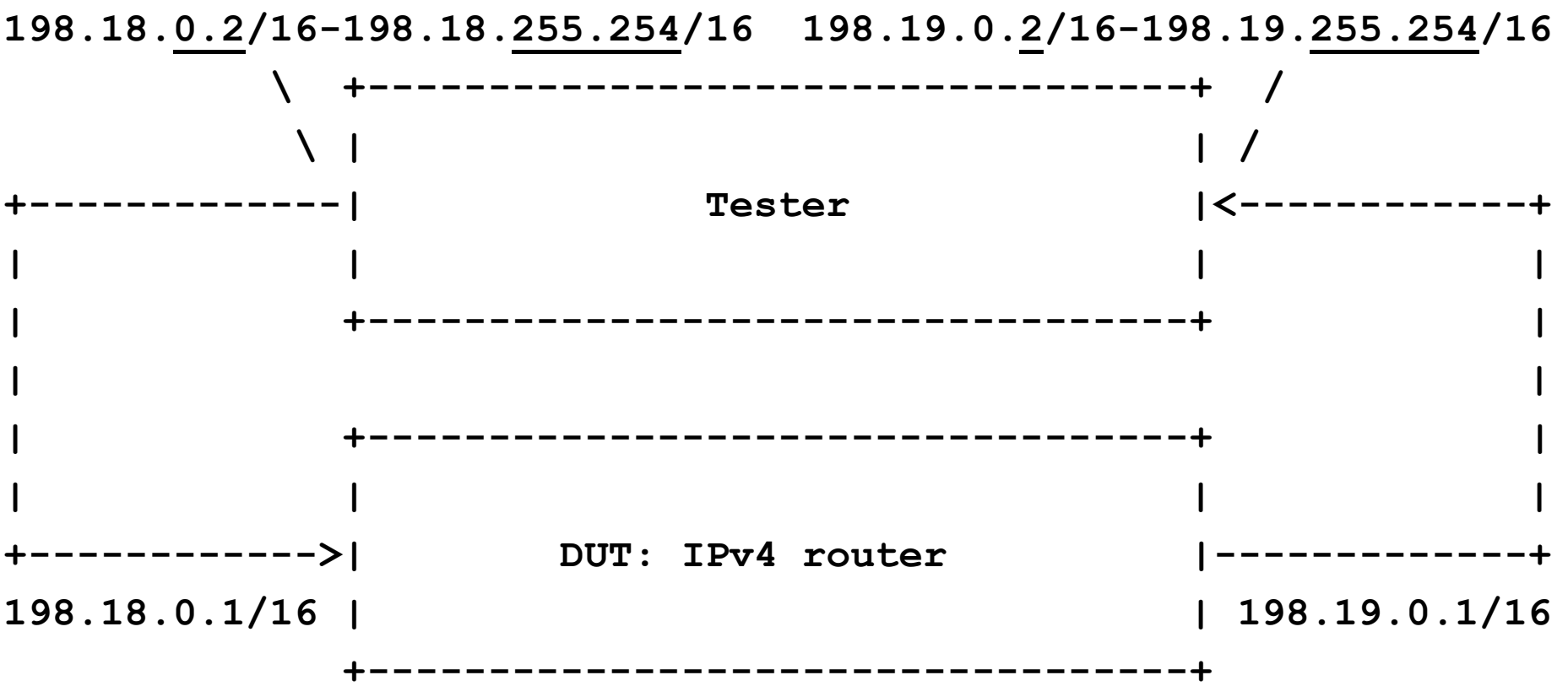
→ Conditions for the laboratory tests should be improved!

Reminder: Recommended Solution

- Basic idea: Let us use pseudorandom IP addresses!
  - This is the spirit of RFC 4814 applied to the IP addresses ☺
- Problems to solve:
  - What ranges can be used?
    - There is scarcity in IPv4 addresses reserved for benchmarking
      - 198.18.0.0/15 was reserved for benchmarking
    - There is abundance in IPv6 addresses reserved for benchmarking
      - 2001:2::/48 was reserved for benchmarking
  - What ranges should be used?
    - A trade-off is pointed out
    - And a better solution is proposed ☺

Potential Ranges When No Gateway Is Used (IPv4)

- .0.1 is for the tester; .0.2 to .255.254 can be used (64k-3)

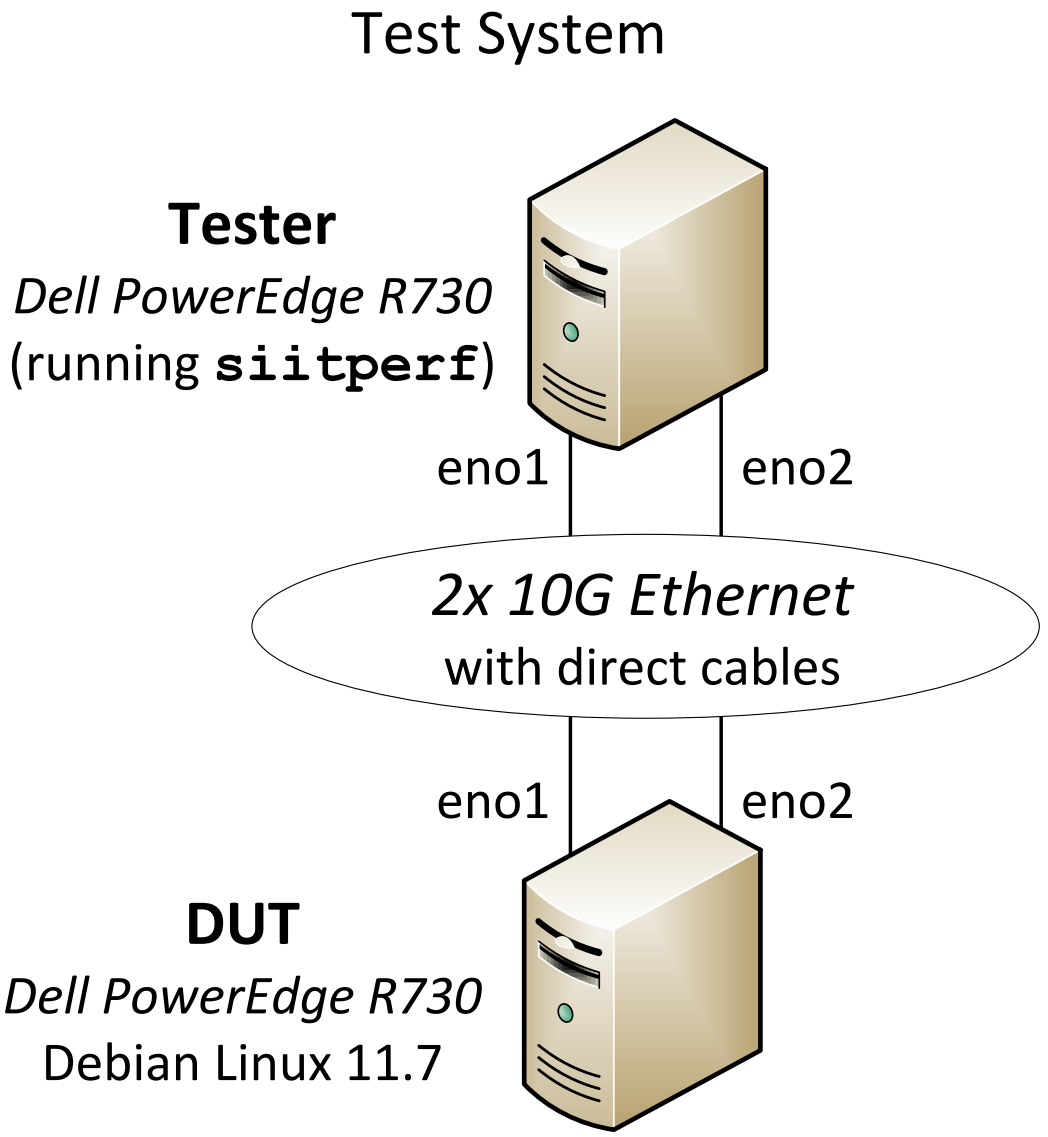


Tested Ranges When No Gateway Is Used (IPv4)

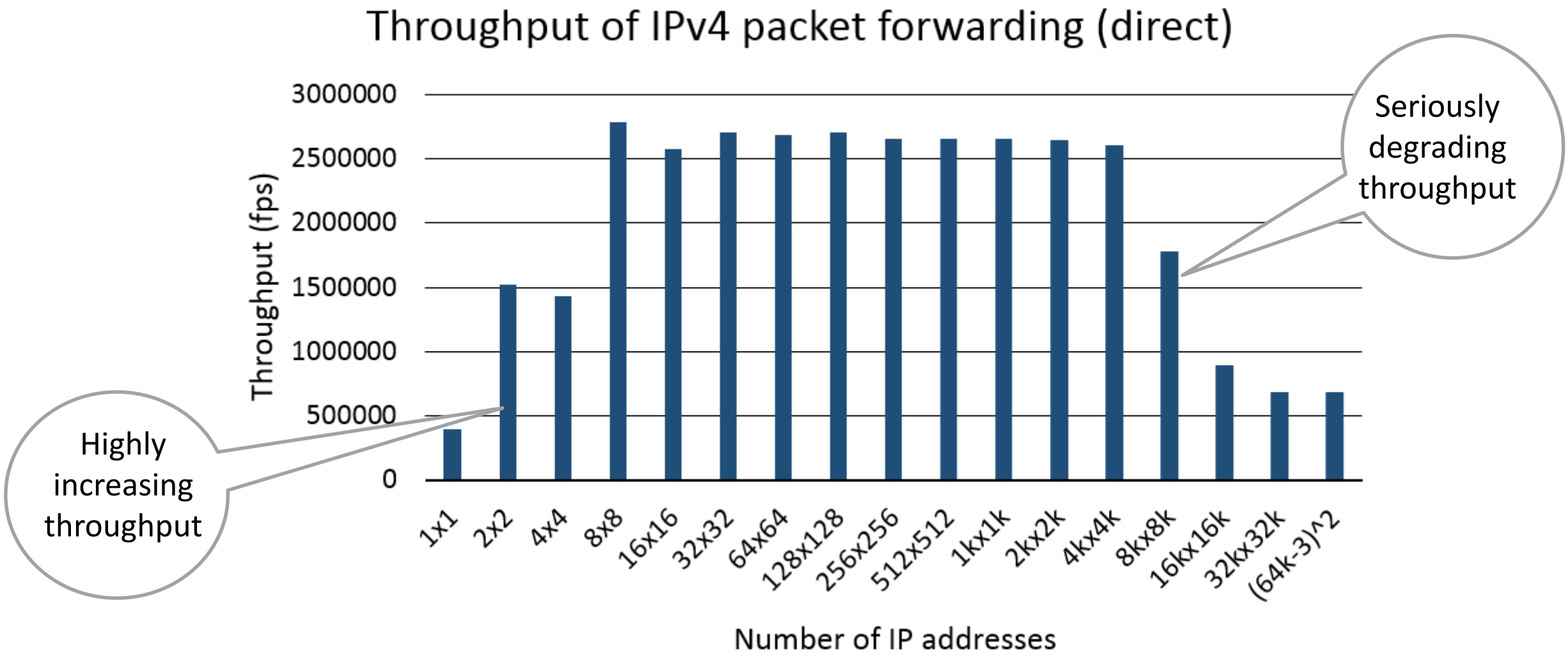
- On each side, the number of IPv4 addresses were
  - 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1k, 2k, 4k, 8k, 16k, 32k, 64k-3
- Thus the number of IP address combinations were
  - 1x1, 2x2, 4x4, 8x8, 16x16, 32x32, … , 16kx16k, 32kx32k, (64k-3)x(64k-3)
- Note: The maximum potential hash values were the double due to bidirectional traffic.
  - 198.18.0.2 --> 198.19.0.2 , that is: src: 198.18.0.2, dst: 198.19.0.2
  - 198.18.0.2 <-- 198.19.0.2 , that is: src: 198.19.0.2, dst: 198.18.0.2

Measurement Environment

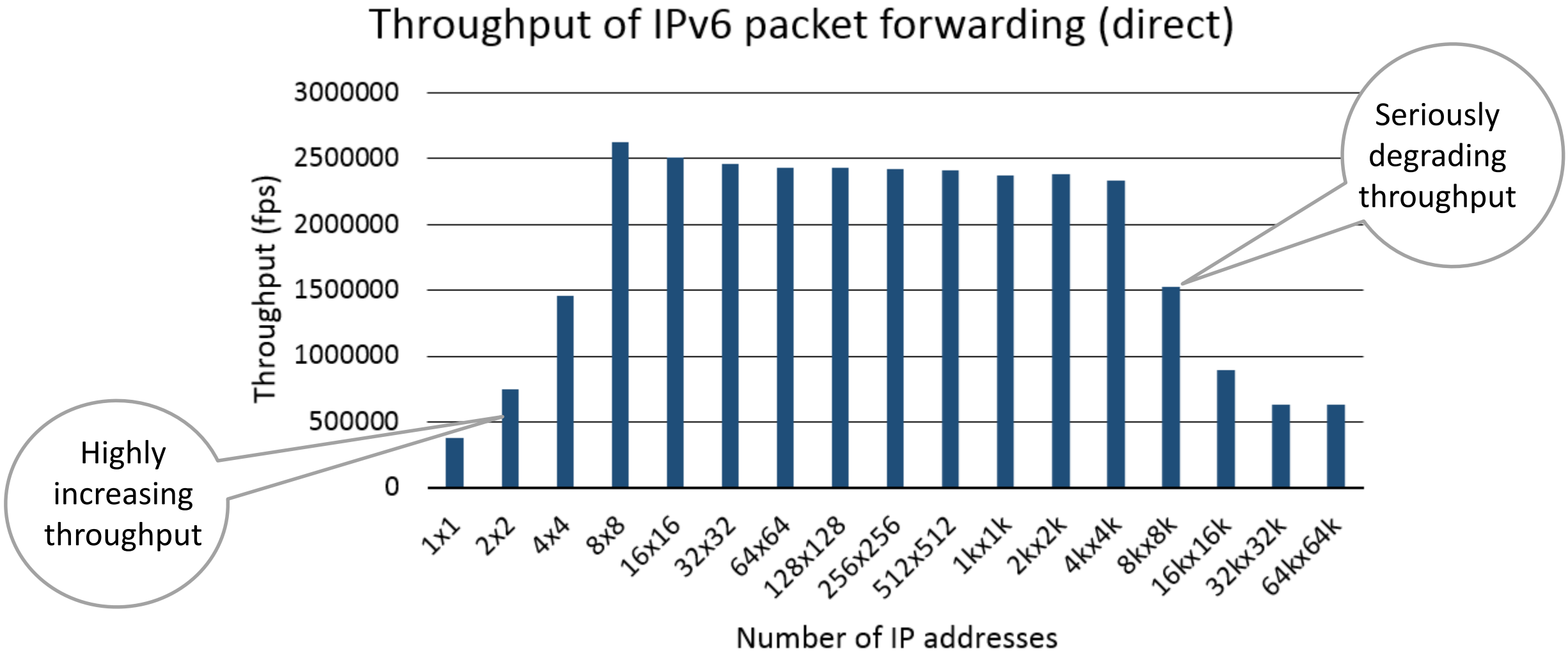
- Dell PowerEdge R730 servers
  - Intel Xeon E5-2667v3 CPUs
  - 2x8=16 CPU cores per server
- Tester: Debian 9.13
  - Running **`siitperf`**
- DUT: Debian 11.7 with 5.10 kernel
  - CPU clock frequency set to **1.2GHz**
  - IPv4 and IPv6 packet forwarding
  - **1st type of RSS implementation**



Results When No Gateway Is Used (IPv4)



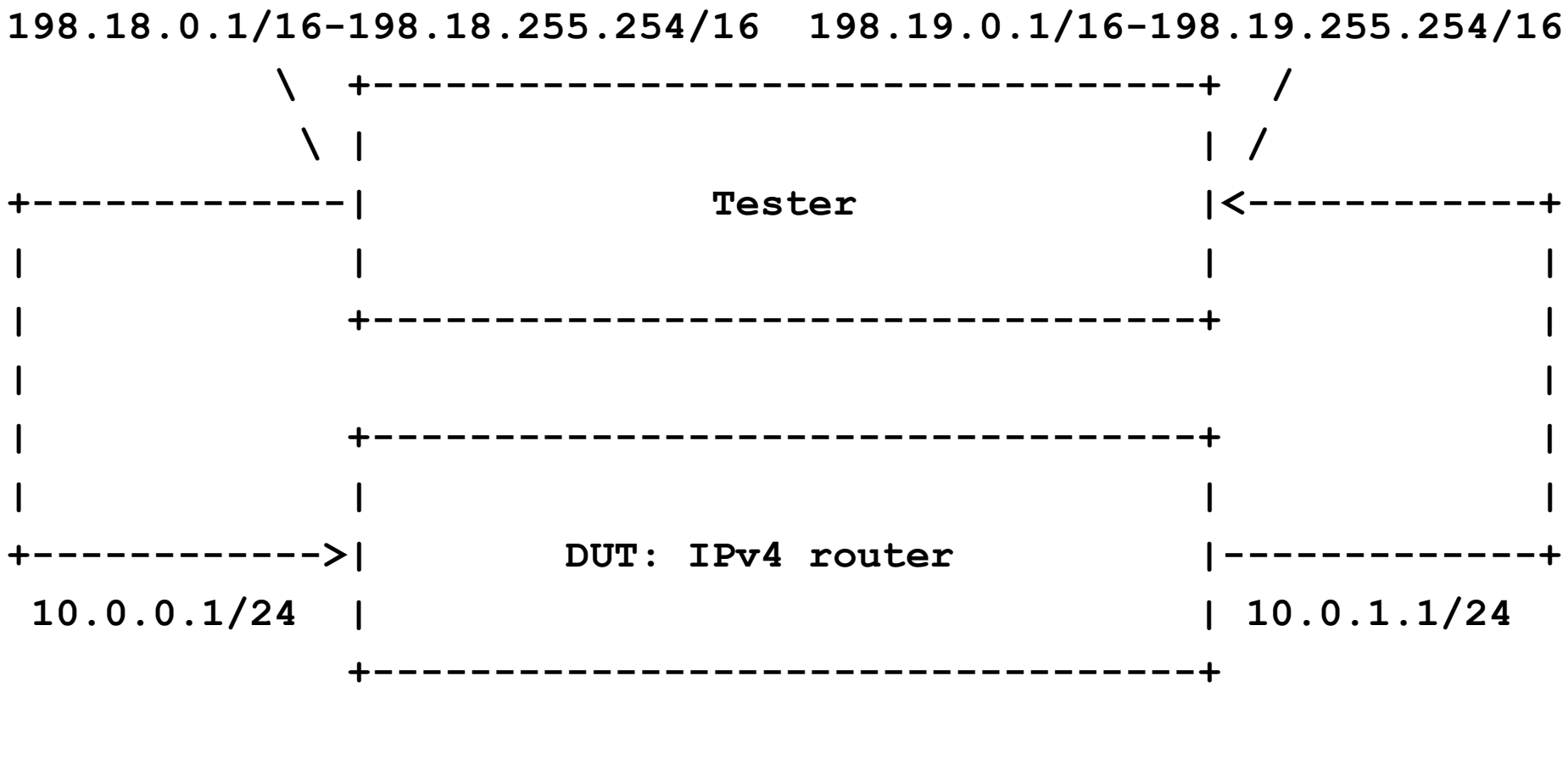
Results When No Gateway Is Used (IPv4)



When No Gateway Is Used

- What is the “right” or “optimal” number of IP addresses?
  - Lower limit may depend on the number of CPU cores of the DUT!
  - Higher limit may depend on the ARP / NDP table implementation, and the CPU cache size of the DUT!
- Let us eliminate the “higher limit”!

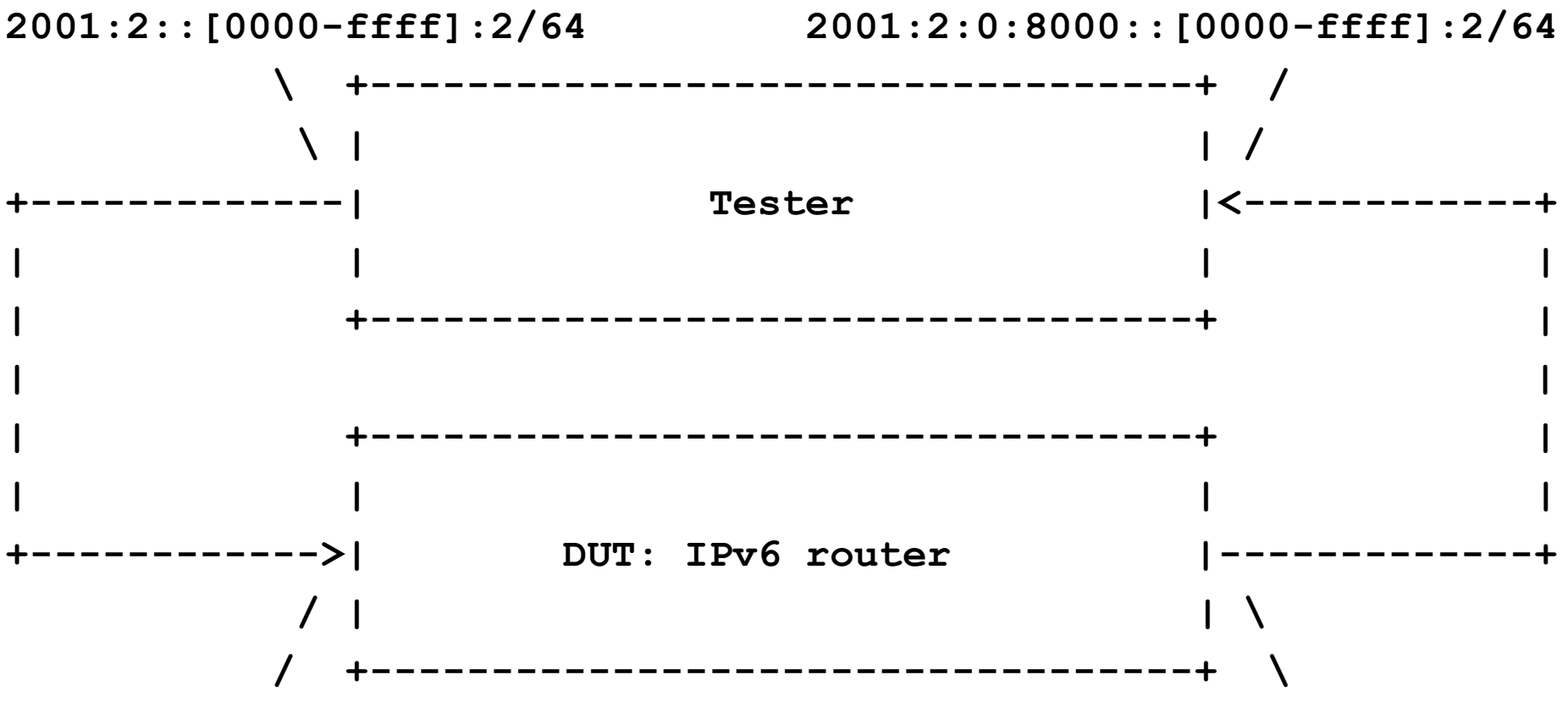
When Gateways Are Used (IPv4)



routes in the DUT:

destination network	next hop
198.18.0.0/16	10.0.0.2
198.19.0.0/16	10.0.1.2

When Gateways Are Used (IPv6)



routes in the DUT:

destination network	next hop
2001:2::/64	2001:2:0:1000::2/64
2001:2:0:8000::/64	2001:2:0:9000::2/64

When Gateways Are Used

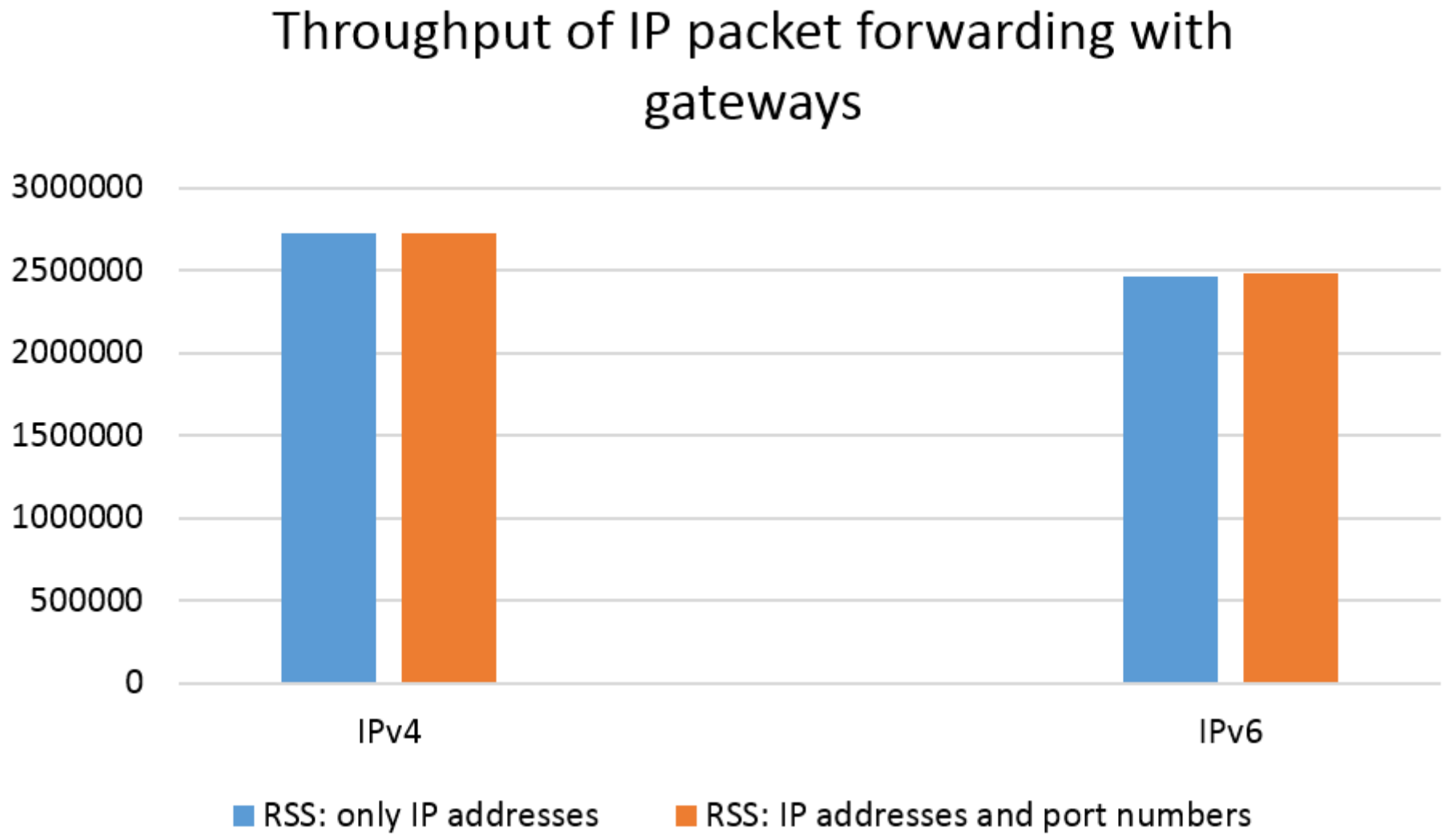
- In the DUT, the ARP / NDP tables are not overloaded
- Only the mappings of the IPv4 / IPv6 addresses of the gateways are stored (only two ARP / NDP table entries)
- And this is typical in the case of Internet routers
  - (no direct connections to the hosts)
- No trade-off about the “right number” of IP addresses

  →All potential IP addresses may be used!

  →In IPv4: (64k-2)x(64k-2)

  →In IPv6: 64kx64k

Results When Gateways Are Used



We propose

- The usage of multiple IP addresses
  - To support the 1st type of RSS
- The usage of a gateway
  - To avoid the overhead caused by the high number of ARP / ND table entries
- The usage of all available IP addresses (in their ranges)
  - 64k-2 for IPv4 and 64k for IPv6

We believe that our method is usable in practice and it significantly improves the conditions of laboratory tests by removing their bias against the 1st type of RSS implementation.

We would like to ask for feedback

- Do you think that the proposed solution is appropriate?
- Do you have any idea what to change, add, etc.?
- All your comments and suggestions are welcome!

- Would you support the adoption of the draft as a WG item?