Pros and Cons of IPv6 Transition Technologies for IPv4aaS

draft-ietf-v6ops-transition-comparison

Gábor LENCSE lencse@hit.bme.hu (BUTE) – presenter

Jordi PALET MARTINEZ jordi.palet@theipv6company.com (The IPv6 Company)

Lee HOWARD lee@asgard.org (Retevia)

Richard PATTERSON richard.patterson@sky.uk (Sky UK)

Ian FARRER ian.farrer@telekom.de (Deutsche Telekom AG)

IETF 111, v6ops, July 29, 2021.

Considerations before WGLC

- On the one hand, this draft should be published ASAP to assist network operators with a stable document
- On the other hand, yet there are two incomplete parts
  - – Benchmarking different implementations of all five IPv4aaS technologies
    - It would last about 2-3 more years, because most RFC 8219 Testers are yet to be implemented. – This is way too long!
      - – It MUST be left out, we recommend that a pointer to a new draft be included into **Section 5**
  - – The issue of the scalability of the stateful technologies
    - Meaningful results can be produced before IETF 112
      - – We recommended for the WG chairs that they should be included into **Section 4.2**
      - – We need a decision of the entire WG about this question (see last slide)
      - – We need feedback about the method and its parameters (if they are acceptable)

What should / can be measured?

- Which technologies are concerned?
  - MAP-T, MAP-E, lw4o6: stateless in the core network
  - 464XLAT, DS-Lite: stateful in the core network
    - 464XLAT is a combination of stateless NAT46 and **stateful NAT64**
    - We have a methodology and an RFC 8219 compliant Tester for benchmarking of **stateful NAT64** implementations
    - We do not have an RFC 8219 compliant Tester for benchmarking DS-Lite implementations
      - But we *hope* that stateful NAT64 represents well the entire stateful class

Measurement Methodology

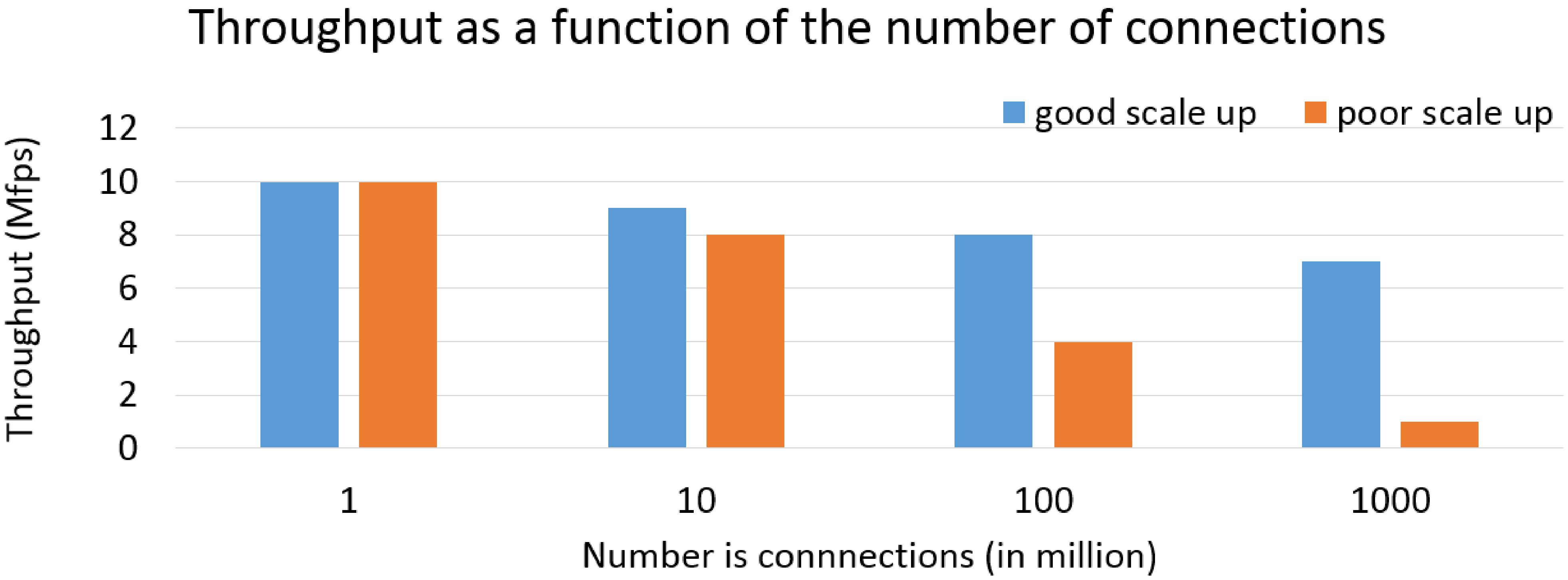
- It has been proposed in:
  - G. Lencse, K. Shima, "Benchmarking Methodology for Stateful NATxy Gateways using RFC 4814 Pseudorandom Port Numbers", Internet Draft, May 17, 2021, draft-lencse-bmwg-benchmarking-stateful-00
    - https://datatracker.ietf.org/doc/html/draft-lencse-bmwg-benchmarking-stateful
    - Presented in BMWG on Monday
- One partial implementation exists:
  - https://github.com/lencsegabor/siitperf/tree/stateful , documented in:
    - Lencse, G., "Design and Implementation of a Software Tester for Benchmarking Stateful NAT64 Gateways: Theory and Practice of Extending Siitperf for Stateful Tests", under review in *Computer Communications*, may be revised or removed without notice, 2021, http://www.hit.bme.hu/~lencse/publications/SFNAT64-tester-for-review.pdf

Method for Scalability Testing

- What can be easily measured?
  - How the number of connections influence the throughput of the stateful NAT64 gateway?
- What parameters can be used?
  - Siitperf supports the specification of
    - Source port range (e.g. from 1024 to 10,023 is 10,000 different values)
    - Destination port range (e.g. from 80 to 1079 is 1,000 different values)
    - E.g. 10,000*1,000*1=10,000,000 combinations (the potential number of connections stored in the stateful NAT64 gateway)
      - Siitperf was tested up to 400,000,000 combinations (in its own state table)

Method for Scalability Testing

- We plan to determine the throughput as a function of the number of connections, looking like the following *example*:



Throughput as a function of the number of connections

Method for Scalability Testing

- Challenge: What parameters should we use to provide meaningful results for Network Operators?
  - Number of connections? e.g. from 1 million to 1 billion?
    - Its detailed break down to source port range, destination port range
  - Frame size?
  - NAT64 Implementation?
    - Some free software, e.g. Jool?
  - Is 10Gbps Ethernet enough?

Such parameters should be discussed on the v6ops mailing list!

Question to Decide

- Option A)
  - – Include scale-up test results
  - – Delay WGLC after IETF 112
- Option B)
  - – Leave out scale-up tests (add a pointer to a new draft)
  - – Initiate a WGLC right now (right after IETF 111)