Accelerated Site-to-Site VPN

Intermediate Talk

Maximilian Pudelko, M. Sc.

September 18, 2018

Chair of Network Architectures and Services
Department of Informatics
Technical University of Munich

Contents

Why VPNs are Important

- Todays business is multi-national and international
- Many distributed sites that need to be interconnected
- Provide a secure channel for communication over insecure medium
- Other usecases: VM interconnects, cell tower backbones, firm intra-nets

=> Need for high throughput solutions

Current state-of-the-art: $\ll$10 Gbit/s [1]

[1] 64 KB packets, no packet loss, hardware accelerated ciphers only

Chair of Network Architectures and Services
Department of Informatics
Technical University of Munich

Focus: Site-to-Site VPN

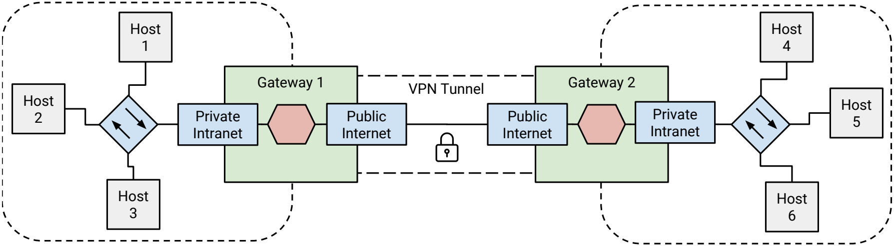

Figure 1: Overview of example Site-to-Site VPN setup

- Only two (or similar little) endpoints
- Very high bandwidth between them

Goals of this Thesis

- Create benchmark criteria for Site-to-Site setups
- Evaluate performance of common implementations
- Develop a general performance model for VPNs
- Explore different approaches for performance improvements

Overview of Existing Implementations

- OpenVPN
- IPsec
- WireGuard

Problems with them: Very slow under high load

Chair of Network Architectures and Services
Department of Informatics
Technical University of Munich

Insertgraphshere

Chair of Network Architectures and Services
Department of Informatics
Technical University of Munich

MoonWire

- DPDK network stack to bypass slow kernel
- Lua for fast prototyping and interfacing with libraries (crypto)

MoonWire Graphs

[bytes/cycle graph of different ciphers]

=> Cryptographic operations can become the bottleneck

Hard to improve, correctness is more important

Solution: work distribution to multiple cores

Symmetric Encryption in a Nutshell

enc(Shared key + Nonce) = Encrypted message

Correct nonce generation/handling is critical. Nonce reuse (under the same key) means K.O.

Nonce generation depends on size. IETF ChaCha20 nonce is 96 bit (10 byte). Too short to be randomly generated (birthday problem) => recommendation: counter++. Must be global over all threads/cores. Accessed for each packet => highly critical. Synchronization (mutex) and atomics are far too slow.

[Timing graphs for atomics/mutex vs. mpps]

Nonce Generation Tricks

- Partition nonce space per worker: 8 bit worker_id + 94 bit counter = 96 bit
  Worker_0: **0**123, **0**124, **0**125, **...**
  Worker_1: **1**123, **1**124, **1**125, **...**
  Beware of "overflows" into different worker partition
- Different cipher: XChaCha20 has 192 bit (24 byte) nonce
  Can be randomly generated safely. Each worker has own PRNG instance (seeded carefully) => Independent state, no sharing => fast
  Trade-off: messages get larger (by 10 bytes), incompatible with existing protocol

Nonce Generation Tricks

- Partition nonce space per worker: 8 bit worker_id + 94 bit counter = 96 bit
  Worker_0: **0**123, **0**124, **0**125, ...
  Worker_1: **1**123, **1**124, **1**125, ...
  Beware of "overflows" into different worker partition
- Different cipher: XChaCha20 has 192 bit (24 byte) nonce
  Can be randomly generated safely. Each worker has own PRNG instance (seeded carefully) => Independent state, no sharing => fast
  Trade-off: messages get larger (by 10 bytes), incompatible with existing protocol

Good news: Decryption is much easier. Message contains everything.

Multi-core Scaling Opportunities

Source IP & port identical over all packets => Simple RSS does not work

Remaining Work

- More benchmarking & measurements
- Try more other performance improvements
  - AVX512 cipher implementations & CPU downclocking
  - NUMA
- Thesis writing