Routing Information Base for the Global Data Plane

Project 17: Rahul Arya, Ja Wattanawong, Praveen Batra | CS 262a, advised by Prof. John Kubiatowicz and Nitesh Mor

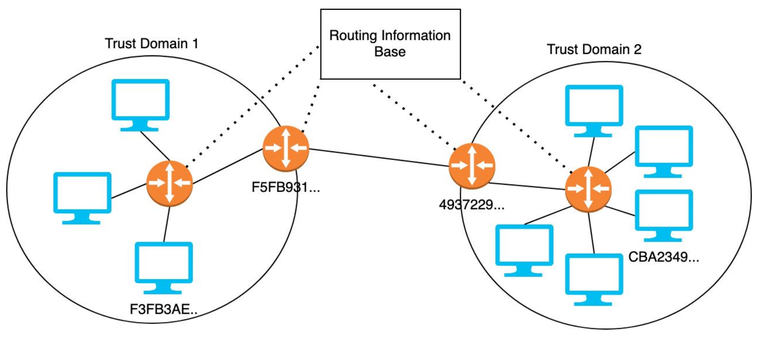
Problem

The Global Data Plane (GDP) is a system for secure distributed computing across a variety of heterogeneous and possibly untrustworthy resources. The GDP as a whole comprises several components, including cryptographically verified bundles of data (DataCapsules) and secure execution. Secure, scalable, and location-agnostic routing is needed to allow communication between GDP nodes based on their content/metadata and unique identifiers derived from it, rather than ephemeral or location-dependent identifiers such as IP. The system must scale to an order of 10^12 nodes to handle needs such as IoT and edge computing.

Background

There are existing GDP router implementations in Python and Click, but they either lack essential features such as support for hierarchical trust domains or are not performant. A faster, more feature-rich switching network and routing information base would allow the GDP to scale to match its vision.

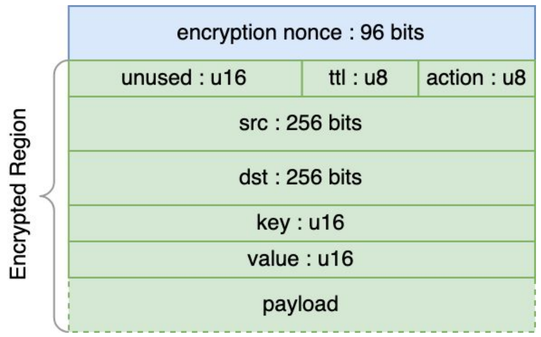
Our primary goal was to build an MVP of networking switches and a routing information base (RIB) in Rust using the Capsule framework, to achieve high performance with a variety of workloads including large and small packets and multiple-node routing paths with various topologies. Packets are routed based on their destination GDPname, a location-independent identifier mapped to an IP address by the RIB.



Solution

We have built a software switch and routing information base on top of the Capsule network function framework to tunnel GDP packets inside UDP datagrams.

GDP + Encryption Protocol Data Unit (PDU)



Currently the GDP header and payload are encrypted using AES-GCM with a pre-shared key. Each hop between switches is separately encrypted, so the route of a packet through the network cannot be traced by an external observer. We plan to build on top of this encryption system to implement full dTLS, but the current implementation is sufficient to demonstrate its performance impact on switching throughput.

Switching Logic

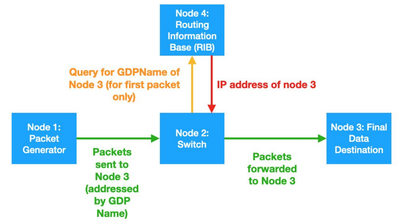
When a GDP packet is received, decrypt it and look up the next hop for its destination GDPname in the routing table. If there is a hit, forward the packet to this next hop. Otherwise, reply with a Nack to the source, and issue an RIB query for the destination. Once the RIB sends a reply, update the local routing table. We expect the source to retransmit the packet after a delay, at which point the next hop will be known to the switch and the packet can be forwarded.

Future Work

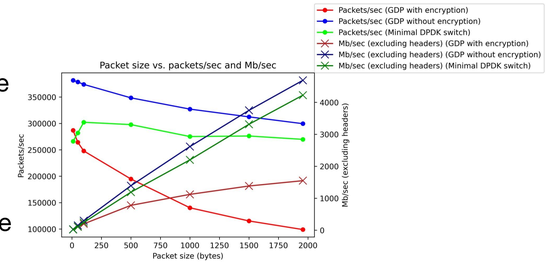
- Implement dTLS encryption
- Implement signature verification

Results

Our experimental setup consists of four GDP nodes on AWS EC2 in the topology shown to the right:.



Using a c5n.large instance with 1 core pinned to the switch, we tested performance with varying payload size and encryption.

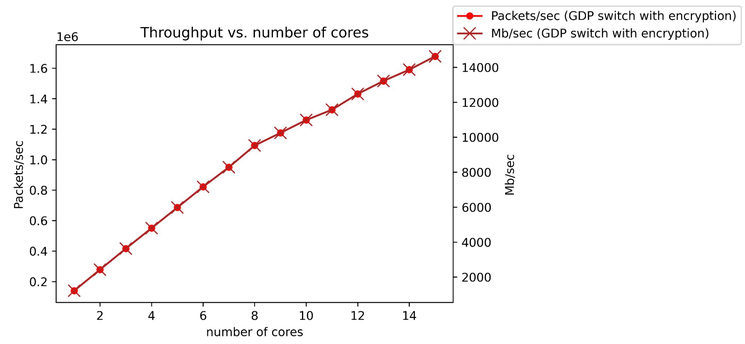


We achieved peak throughput of 1.55Gbps on the single-core switch with encryption, and 4.69Gbps without by using 1957 byte payloads.

Our unencrypted throughput slightly exceeds even that of a minimal DPDK-based software switch that forwards packets to a hardcoded destination, which could be attributed to both noise and our tuning.

To compare with existing implementations, the Click-based router achieved ~500Mbps using 1000 byte PDUs across four cores (2 physical cores with 2 hyperthreads each) [1]. At the same packet size, using only a single core, we achieved 1.12Gbps.

Furthermore, we tested our switch on the 16-core (8 physical) c5n.4xlarge to measure how throughput scales with more cores:



The above graph shows how adding additional cores to the switch improves throughput. There is a perfect linear increase when we use up to 8 physical cores. The other 8 cores are hyperthread siblings, and there is a reduced rate of increase when we add them. With 8 cores we get 8.74 Gbps with encryption, and with 15 cores we get 13.42 Gbps.

[1] https://people.eecs.berkeley.edu/~kubitron/cs262/handouts/papers/GDP_icdcs_19.pd (Fig. 6)