Synopsis

Telecom networks are the backbone of global communication, enabling voice, data, and multimedia services for billions of users. Traditionally reliant on proprietary hardware, the latest generation (5G) adopted Network Functions Virtualization (NFV) to enhance flexibility, scalability, and cost efficiency. The 5G packet core forms the core of the telecom network and comprises of multiple complex software based network functions (NFs). Each NF plays a specific role in facilitating data transfer between a mobile subscriber and the internet. Even though software based NFs have various advantages, they give rise to an interesting host of research problems, some of which this thesis tackles in the context of the 5G packet core.

With the advent of software based NFs, researchers have introduced various abstractions to ease the process of building an NF. These efforts have been focused on providing abstractions for either building a specific NF, e.g., router, middleware, etc., or a part of an NF, e.g., an L7 protocol parser that could be part of an intrusion detection system. The proposed abstractions are not enough to completely specify a complex NF that is part of a multi-tier system, e.g., the 5G packet core. Due to a lack of suitable options, the 5G packet core is specified in text based documents written in English. These documents are verbose, making them hard to maintain. Further, they are riddled with ambiguities, wherein the domain experts responsible for specifying the NFs miss out on key details while writing the specifications, leading to issues with interoperability between vendors responsible for building the NF from specifications.

This thesis presents Pyramis, a DSL, to ease the specification and development of large multi-tier systems. Domain experts write the system specification using the high-level abstractions and language constructs provided by Pyramis. These specifications can then be used to auto-generate optimized C++ reference implementations of the various components by the Pyramis translator. Specifications written using Pyramis are more concise and easier for non-programmers to work with as compared to a reference implementation in a general purpose language. Specifications in Pyramis are also precise and unambiguous by construction. We show that reference implementations auto-generated from Pyramis perform at least as well as hand-optimized implementations.

Further, running software based NFs on commercial hardware requires intelligent systems engineering to ensure a certain base level of performance, which is achievable with proprietary hardware. Previous work has focused increasingly on network stacks because of it being a significant bottleneck in an NF’s performance. Researchers have suggested improvements to the kernel network stack or explored novel network stack designs based on kernel bypass mechanisms like DPDK. This thesis discovers that novel network stack designs built over DPDK do not perform significantly better than the Linux network stack in the context of 5G packet core NFs, because of their compute intensive nature. Instead, this thesis observes that these NFs spend most of their time accessing and updating mobile subscriber state while processing incoming requests. Every NF is multithreaded, employing locks to access and update said state to maintain serial access. These locks act as a roadblock to the multicore scalability of these NFs.

This thesis develops AppSteer, which considers the problem of improving the multicore scalability of software network functions via application-aware packet steering. Prior work has dealt extensively with the problem of eliminating lock contention within the network stack by splitting kernel data structures into per-core slices and using RSS to maintain flow level affinity to a core. These network stacks were mainly developed on top of kernel bypass mechanisms like DPDK, and they lead to multicore scalability of the NF only if it maintains state at the granularity of a transport layer flow. AppSteer extends the scope of this problem by considering NFs running on top of the Linux kernel and NFs, which store state at the granularity of an application key that does not map to a transport layer flow. AppSteer exposes APIs that let an NF steer network traffic to application threads based on application identifiers in the packets and implements these APIs over the Linux kernel via eBPF programs and kernel changes. We modify the NFs of a production-grade 5G packet core implementation to maintain per-core state and operate in a lockfree manner running on top of AppSteer. We then compare the saturation throughput of our lockfree NFs with their optimized locking-based baselines running on vanilla kernel and find that the lockfree NFs have up to 15--18\% higher throughput on 16 cores.

Finally, network operators are responsible for sourcing NFs from vendors and deploying them. Operators are interested in monitoring metrics (queueing delay at switch ports), which could help them identify potential issues with the network (TCP incast or load imbalance). Researchers have made use of in-network compute to help the operator get access to such metrics. Operators in the 5G packet core are interested in application layer telemetry and monitoring application layer metrics (number of users serviced by an AMF instance) to improve packet core deployment. Operators tracking application layer metrics have traditionally not used in-network resources because of their limited compute capability and have relied on the NF to drive telemetry. Hence, operators in the 5G packet core, who receive black box NF binaries from vendors, are hamstrung in their ability to modify NF code and depend on the vendor to expose the necessary metrics. Operators could also procure NFs that log on every incoming application message, exposing a superset of metrics and trading NF performance for flexibility. Instead, this thesis explores telemetry techniques employed outside the application to allow the operator to perform application telemetry without being dependent on the vendor and without hurting NF performance. This thesis identifies that an on-path application telemetry solution is more viable than an off-path solution, which is plagued with packet copies and hinders NF performance.

To this end, this thesis develops DeepSight, which allows users to express queries to extract application-layer metrics and automatically compiles these to eBPF code via our novel intermediate representation. DeepSight allows network operators to gain insights about application performance by parsing application messages inside endhost kernels without any logging support from the application. Our evaluation of DeepSight shows that extracting application metrics in eBPF programs, where feasible, incurs lower performance overheads than logging all application messages while delivering the same level of flexibility. DeepSight paves the way for expressive and efficient in-network application telemetry for usecases like 5G analytics.