Synchronization and Concurrency in Software-Defined Networking

Goals: Design of a robust and efficient SDN policy composition.

Tools: Logic, algorithmic reasoning, programming.

Prerequisites: Some knowledge of algorithms and networking, basic concurrent and networking programming skills.

Computer networking currently goes through a transition phase: the paradigm of Software-Defined Networking (SDN) is discussed intensively, both in the industry and in the academia. In a nutshell, SDN out-sources the control over the network to a logically centralized software, called the *control plane*. The ability of the control plane to "program" the network (the *data plane*) opens new interesting opportunities to network operators and system designers.

While the perspective of a centralized controller simplifies network management, it comes with the usual drawbacks: single point of failure, scalability bottleneck, overhead due to indirection, etc. A fully centralized system may not adequately or cost-effectively provide the required levels of availability, responsiveness and scalability [3]. This raises the questions: What kind of a *distributed* control plane for SDN is needed?

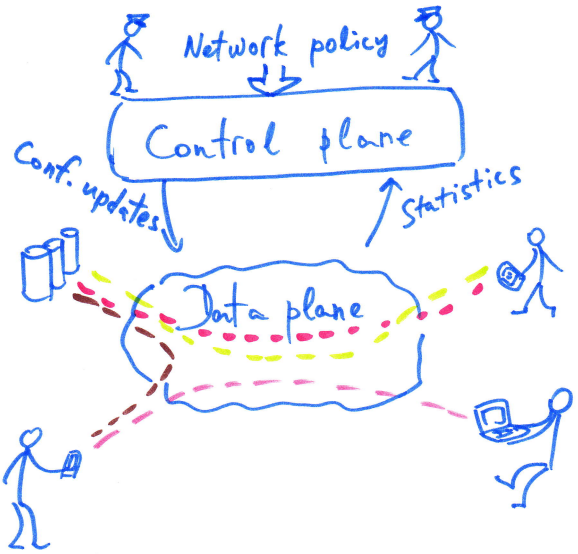

Figure 1: A view of an SDN system: the *control plane* accepts network policy updates from network operators and install them on the *data plane* so that the concurrent user traffic is not affected during the transition.

In this project we treat the control plane as a full-fledged distributed system, and balancing consistency, efficiency and availability in network management becomes a classical problem of distributed computing, extensively studied in multiple other settings. The aim of this project is to sort out the problem space. Our running example is the fundamental problem of the correct implementation of the *network policy*, i.e., the intended network behavior. In case when the network is manipulated by a distributed system of controllers, the question of consistent and resilient policy composition becomes central. Indeed, concurrent operation on the same data plane by distributed controllers may lead to conflicts and inconsistent network operation. So the classical problems of *synchronization* and *concurrency* raise in this novel networking context.

We introduce the abstraction of *Consistent Policy Composition (CPC)* [1, 2], which offers a *transactional* interface to address the issue of conflicting policy updates, in spired by the popular paradigm of software transactional memory (STM) [4]. We aim at devising a protocol that ensures both requirements, while providing optimal costs in terms of time and message complexity.

Contact

Prof. Petr Kuznetsov
`http://www.infres.enst.fr/~kuznetso/`
`petr.kuznetsov@telecom-paristech.fr`
INFRES, Télécom ParisTech
Office C213-2, 46 Rue Barrault

References

[1] M. Canini, P. Kuznetsov, D. Levin, and S. Schmid. The case for reliable software transactional networking. Technical report, arXiv TR 1305.7429, 2013.

[2] M. Canini, P. Kuznetsov, D. Levin, and S. Schmid. Software transactional networking: concurrent and consistent policy composition. In *HotSDN*, pages 1–6, 2013.

[3] D. Kreutz, F. M. V. Ramos, P. Verssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig. Software-defined networking: A comprehensive survey. Technical report, arXiv TR 1406.0440, 2014.

[4] N. Shavit and D. Touitou. Software transactional memory. *Distributed Computing*, 1997.