Towards Smart Intents with Robust and Flexible Routing

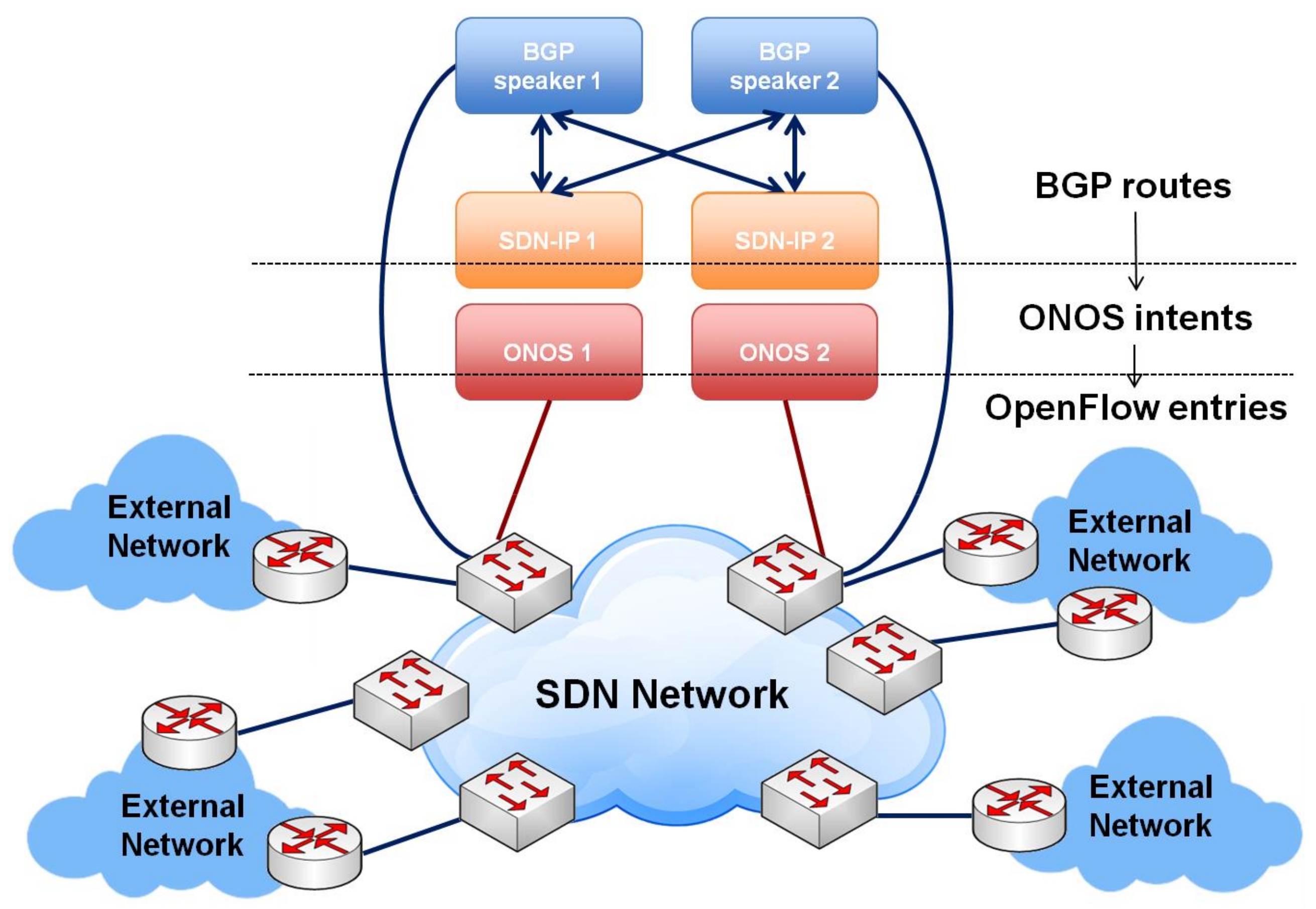
Davide Sanvito, Mattia Gulli, Ilario Filippini, Antonio Capone

Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano, Italy

Introduction

ONOS Intent Framework allows programmers to specify high-level policies which are then compiled to low-level configurations by the controller.
Intents gets re-compiled as a consequence of environment changes (e.g. link failures) to meet the objective. Without further constraints, intents are individually compiled to one of the shortest paths.
Can we consider jointly **multiple intents** in the **compilation**? Can we reactively take into account **flow-level statistics** events to optimize a global network objective, e.g. minimizing Maximum Link Utilization (MLU)?
In this demo we extended the ONOS **SDN-IP** [1] application and evaluate the benefits in terms of average MLU with real traffic traces.
Future plans include the definition of a **new Smart Intent** whose compiler monitors corresponding flows and periodically re-optimize the paths according to their statistics. This would allow other applications to transparently take benefit from the new re-compilation logic and the exposed parameters (such as minimum time between reconfiguration, level of robustness to variations, etc.).

SDN-IP application



- Connects a SDN network to legacy external networks via BGP
- **BGP routes** are received by internal BGP speakers, relayed to the ONOS app, translated to **MultiPointToSinglePointIntents** and then compiled to low-level OF messages

Extended SDN-IP application

TRAINING PERIOD

- Traffic is forwarded as in standard SDN-IP app
- AS-to-AS **Traffic Matrices** (TMs) are collected
  - TM endpoints inferred from BGP announcements
- Pairs of BGP routes are translated to **PointoPointIntents**

AT THE END OF THE TRAINING PERIOD

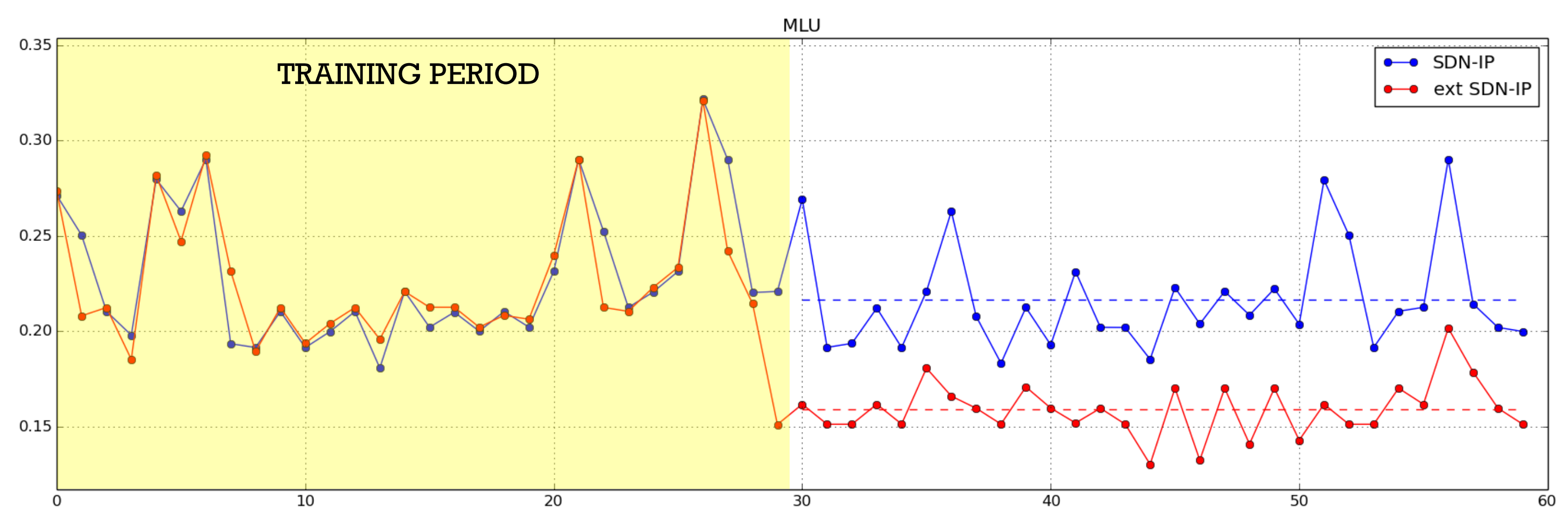
- Exploiting the quasi-periodicity of traffic, a new routing configuration is applied for the following period
  - Computed by solving [2] an **optimization model** taking into account flow statistics and minimizing the average **MLU**
  - Traffic deviations with respect to expected scenarios are coped with routing configurations **robust** over the TM space
- Optimization model and routing activation scheduling run by an **off-platform app**
- ONOS app exposes **REST APIs** to
  - Retrieve TM samples
  - Load a set of routing configurations
  - Apply a selected routing configuration

Demo description

- SDN-IP tutorial network fed with 2 days traffic from Abilene [3] replicated between pairs of Mininet hosts with iperf3
- MLU is monitored over the 2 days

  SDN-IP
  - traffic forwarding with standard intents for both days

  ext SDN-IP
  - traffic forwarding with standard intents during the 1st day
  - TMs collected during 1st day are used by the optimization model to generate robust routing configuration(s) for the 2nd day



A further step: Clustered Robust Routing (CRR)

- **Set** of **robust routing** configurations over the TM space
- Trade-off: number of reconfigurations vs robustness of routing
  - ONOS does not support consistent updates mechanism* during network update operations: a completely reactive approach can impair network performances!
- **two optimization models:**
  - Computation of a **set** of **robust routing** configurations
  - **TMs clustering** in time, space and routing domains to compute the proper routing activation times
  - By-design guarantees on number of re-configurations and the minimum duration for a network configuration.

*When a consistent updates mechanism will be available, our model already supports "broader transitions" between subsequent routings to relax the timelines requirements of the update mechanism.

Issues and future works

- **Splittable routing** in ONOS?
  - Faster model resolution (LP vs ILP) and better solution (lower OF)
  - OpenFlow's Group Tables? Advanced SDN data plane ([4],[5])?
- **Connection disruption** during network updates:
  - "Non-disruptive Intent Reallocation" from FBK CREATE-NET
- **REST API:**
  - gRPC more efficient with larger TMs and topologies?
- Transparent **failure recovery** by Intent Framework:
  - Paths enforced via LinkCollectionIntent not resilient
- Design data structures with **ONOS distributed primitives**
  - Current testbed runs a single ONOS instance
- Move the statistics-based recompilation logic from an off-platform app to the **ONOS Intent Framework**:
  - any application can transparently benefit
  - which parameters should we expose at Intent level?
  - heuristics as an alternative to the integration of optimization tools?

References

[1] https://wiki.onosproject.org/display/ONOS/SDN-IP
[2] http://www.gurobi.com
[3] A. Lakhina, K. Papagiannaki, M. Crovella, C. Diot, E. D. Kolaczyk, N. Taft, "Structural analysis of network traffic flows," ACM SIGMETRICS PER, June 2004
[4] G. Bianchi, M. Bonola, A. Capone, C. Cascone, "OpenState: programming platform-independent stateful OpenFlow applications inside the switch," SIGCOMM CCR, April 2014.
[5] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, D. Walker, "P4: programming protocol-independent packet processors". SIGCOMM CCR, July 2014