October 25, 2017

ONOS plug&play optimization and re-routing module

Antonio Capone, Davide Sanvito, Daniele Moro

POLITECNICO
MILANO 1863

Motivations

- ONOS Intent Framework allows to specify high-level policies
- Transparent re-compilation as a consequence of environment changes
- Can we reactively take into account flow-level statistics events to optimize a global network objective?
  - e.g. minimize Maximum Link Utilization (MLU)

Initial idea

- Definition of a new smart Intent whose compiler
  - monitors statistics of flows corresponding to a set of intents
  - periodically re-optimize their paths based on their flow statistics

- Any application can transparently benefit from the new re-compilation logic

PROBLEM: *the integration of optimization tools inside an ONOS instance kills the performance!*

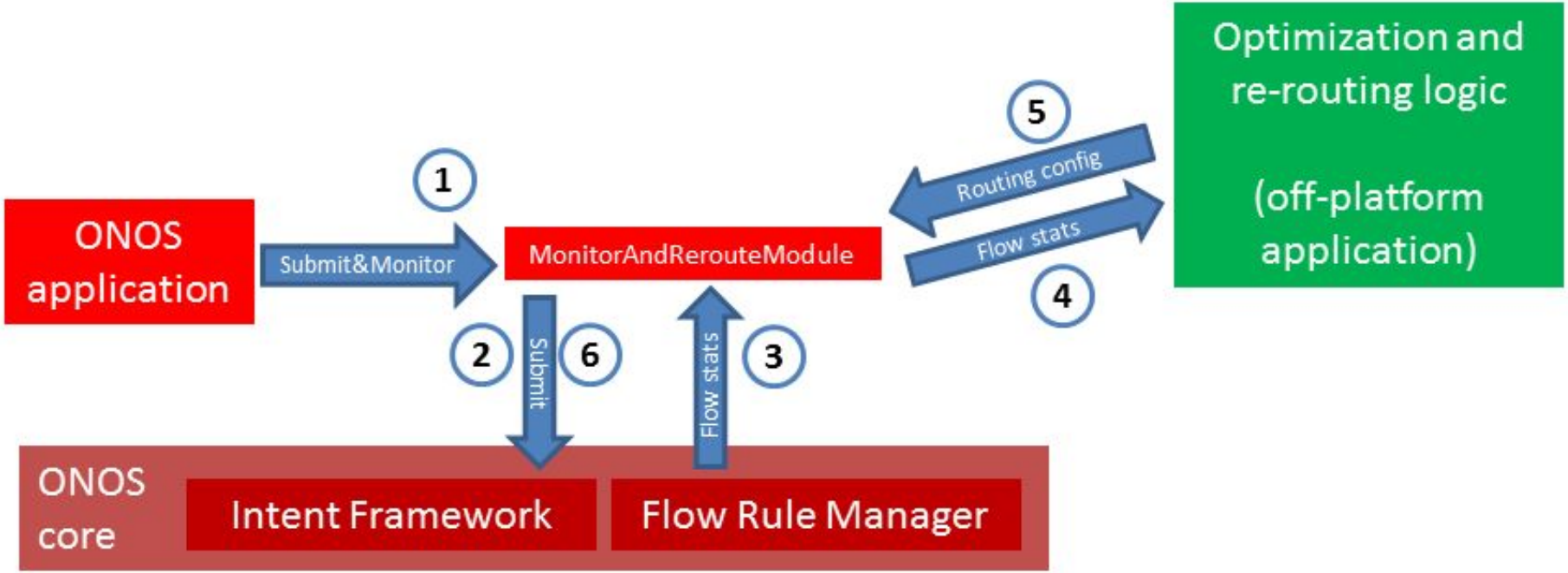
Proposed approach

- Split flow monitoring and path enforcing from routing logic
- (Re-)routing logic moved to an off-platform application
- Application developers/operators can define their own plug&play external routing logic
  - optimization tools/AI/ML based on traffic statistics
  - can re-use their existing TE tools and use ONOS to control the network

Big picture

- Application's developers submit a set of Intents to the new **MonitorAndReroute module** to **monitor** their stats
- The new module propagates the related **flow stats** to the off-platform application
- The new module applies the **new routing configuration,** (re-)computed and received from the off-platform application, via the Intent Framework

Application workflow example



New ONOS module: MonitorAndReroute

- Receives Intents to be submitted and monitored
  - offers a service to other ONOS applications
- Submits the Intent to the Intent Framework
- Maps Intent↔FlowRule to filter FlowRuleEvent to be propagated to the off-platform application
  - offers a REST/gRPC API
- Receives new routing configurations
  - offers a REST/gRPC API
- Enforce new routing configurations via Intent Framework

How to enforce a routing?

1. PathIntent/LinkCollectionIntent allow to specify an explicit path
   - *Path itself is part of the objective* => failures are not transparently recovered
2. The new module itself might handle failures similarly to PointToPointIntentCompiler*
   - *Against code reusability*: this functionality might be useful to other app developers!
3. Define a **new Intent** with “suggested” path(s)
   - The compiler checks if these path(s) are available and eventually fall backs to classic shortest paths
   - But this is very similar to PointToPointCompiler’s compile()! We might directly modify PointToPointIntent to include optional primary/backup paths!

*the compiler itself computes the backup path and configures both the flow rules and the fast-failover mechanism

Interactions ONOS module ↔ off-platform app (1)

Flow statistics propagation

- Retrieving statistics via REST implies a pull-based approach, so we would need to cache them while waiting for the off-platform app
  - high synchronization overhead between instances in case of big number of monitored intents
- gRPC allows to directly push them as soon as the FlowRuleEvent is triggered

Interactions ONOS module ↔ off-platform app (2)

New routing configuration

- The off-platform push them to the ONOS module
  - REST and gRPC are both viable approaches

⟶ Definition of a common interface with two implementations

(REST API + gRPC)

DEMO from ONOS Build 2017

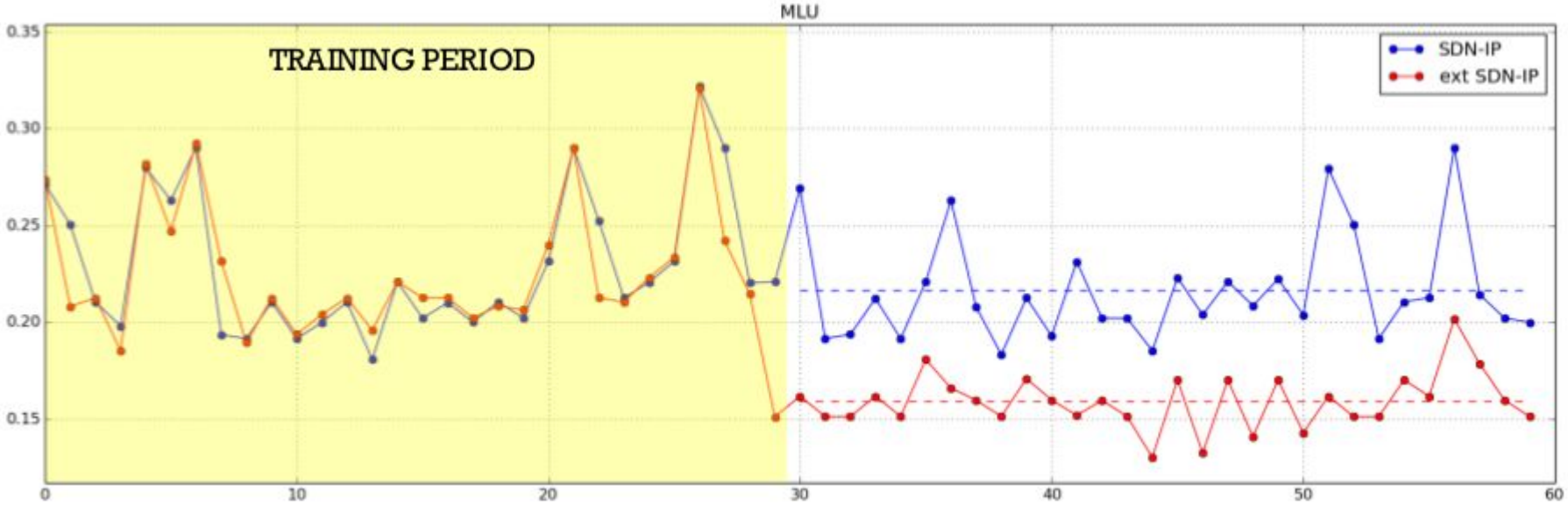
- PoC implementation built on top of SDN-IP application
- SDN-IP tutorial network fed with 2 days traffic from Abilene
- MLU is monitored over the 2 days
- SDN-IP
  - traffic forwarding with standard intents for both days
- extended SDN-IP
  - traffic forwarding with standard intents during the 1st day
  - TMs collected during 1st day are used by the optimization model to generate robust routing configuration(s) for the 2nd day

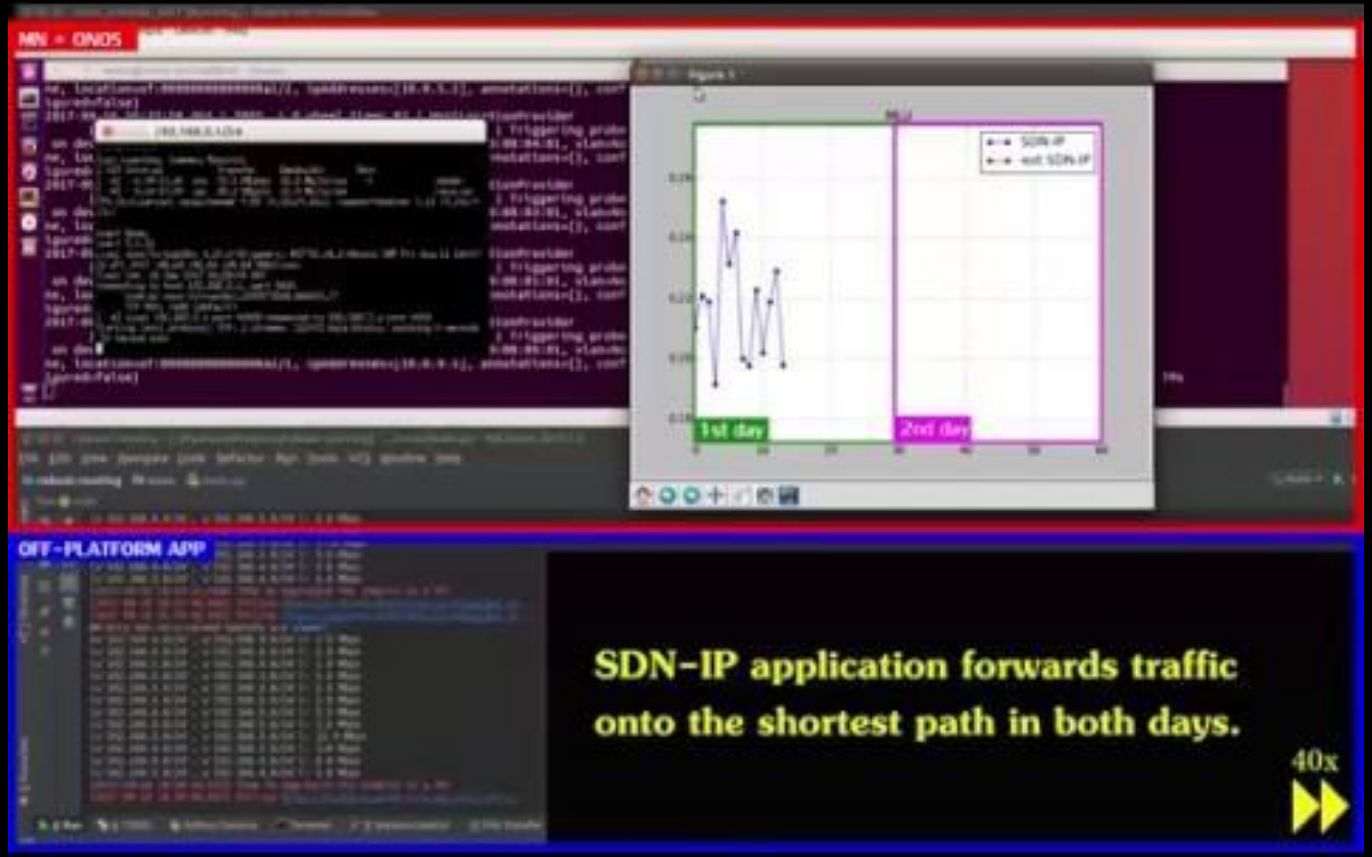
Routing optimization

- Traffic is monitored for a training period (e.g. a day) and a new routing configuration is computed and applied for the next period
  - exploiting traffic quasi-periodicity on a daily basis
  - defining an optimization model* to cluster TMs in time, space and routing domain based on flow stats to minimize avg MLU
  - routing configurations are robust over TM space to cope with traffic deviations w.r.t expected scenarios
  - Trade-off: number of reconfigurations vs robustness of routing

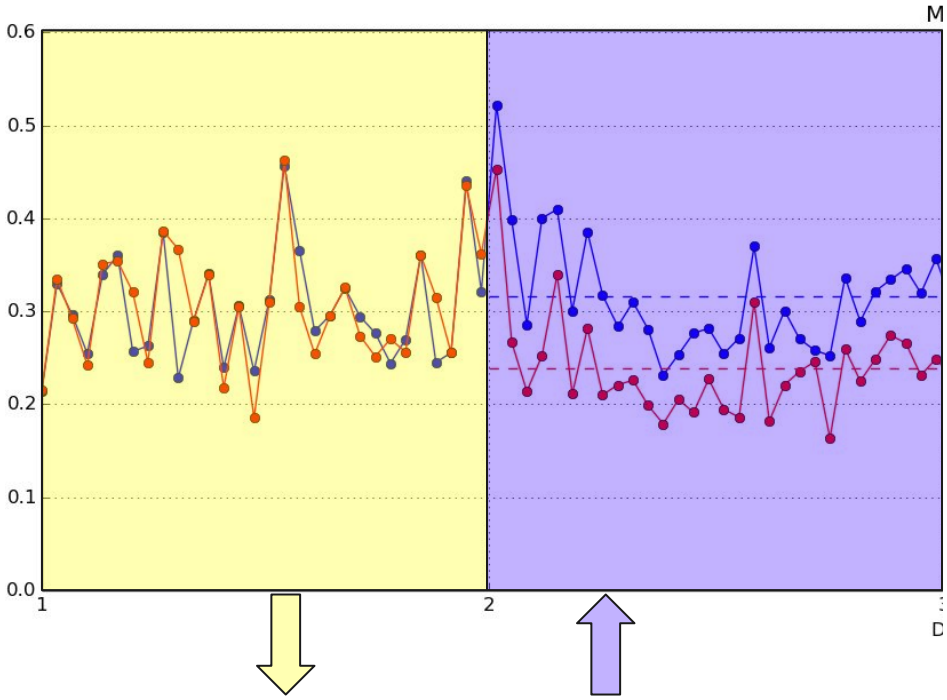
*joint work with France Research Center, Huawei Technologies Co. Ltd

SDN-IP vs extended SDN-IP comparison



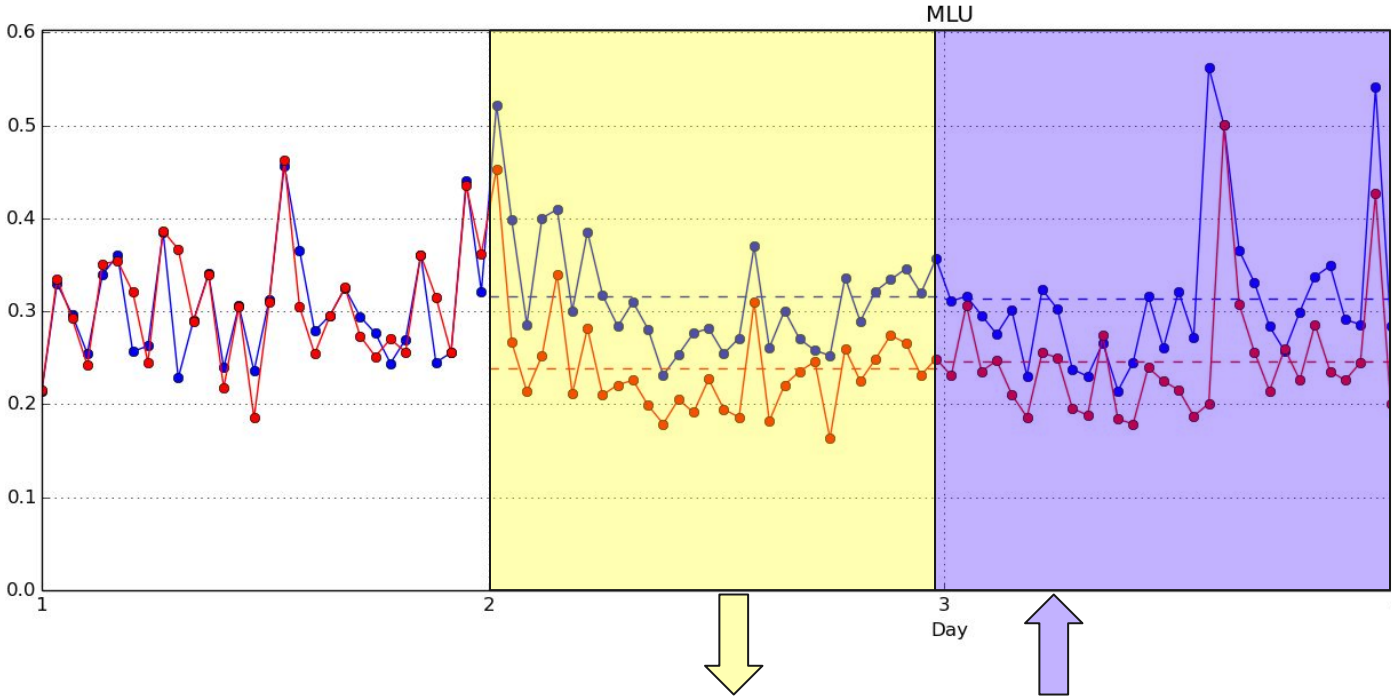

MN + ONOS
1st day
2nd day
OFF-PLATFORM APP
SDN-IP application forwards traffic
onto the shortest path in both days.
40x

The approach can be iterated!

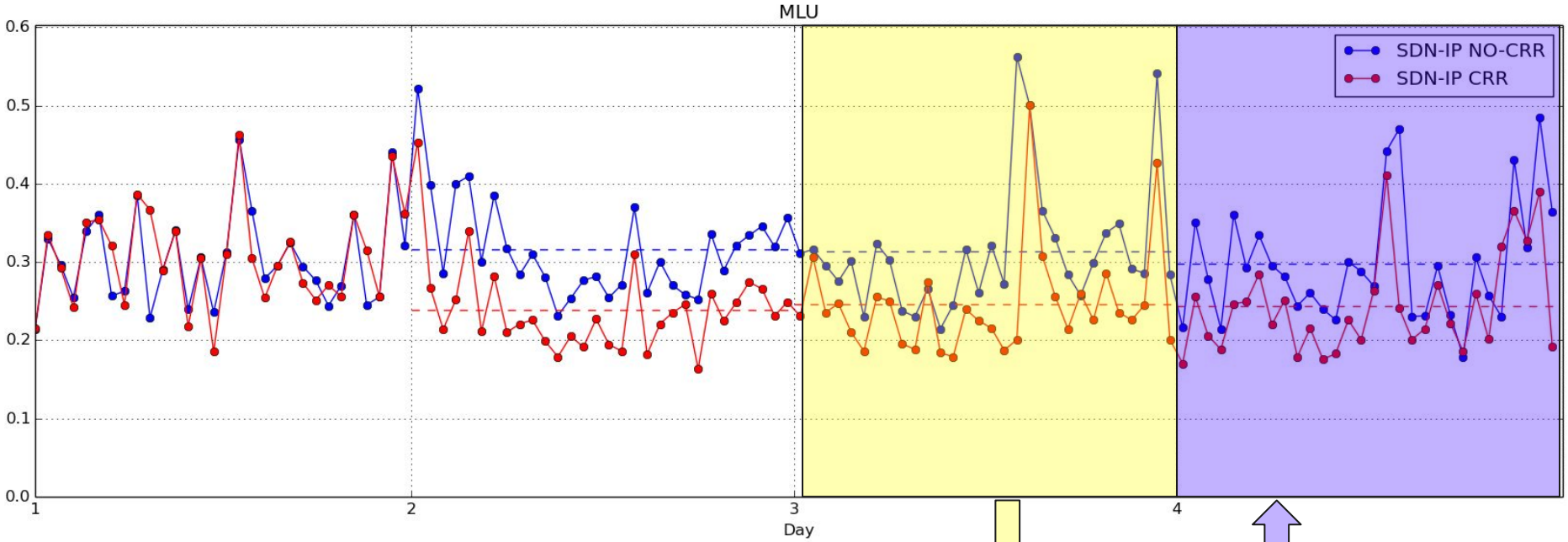


OPTIMIZATION MODEL

The approach can be iterated!



The approach can be iterated!



OPTIMIZATION MODEL

Feedbacks and collaboration

- Andrea Campanella and Carmelo Cascone helped us in the proposal definition and gave us interesting feedbacks
- Andrea will support us in the code review process

Release plan - FIRST RELEASE

Definition of the interface and implementation of the MonitorAndReroute module:

1. receives intents to be submitted and monitored
2. exposes a REST API to collect flow statistics of the monitored flows
3. exposes a REST API to configure the routings by specifying the explicit path (which will be submitted via existing intents, e.g. PathIntent)

Implementation of an off-platform application example to optimize flow routings by jointly considering the flow statistics of different flows

Release plan - SECOND RELEASE

Flow statistics are pushed to the off-platform application via gRPC.

The routings can be configured via gRPC.

New/modified intent to allow a seamless failure recovery.

gRPC is also used to propagate topology changes to make the external module aware of the latest state of the network.

Release plan - THIRD RELEASE

An application can request the monitoring of a treatment (e.g. HTTP traffic) rather than of a specific intent.

Our module will propagate to the off-platform application any flow statistics corresponding to flows matching one of the treatment to be monitored.

Open points

- The MonitorAndReroute module orchestrates application, Intent Framework, off-platform apps communication
  - we plan to implement it as an application offering a service
  - Is it a proper position in the ONOS architecture?
  - Should we implement it as a core service?
- How to enforce a routing configuration?
  - New Intent vs PointToPointIntent extension
- Intents are a “topology-independent network-centric abstraction”
  - is it formally correct to put a topology-dependent information (an explicit path) as a constraint?