The abstract art of composing SDN applications

Pedro A. Aranda – Telefonica

pedroa.aranda@telefonica.com

Programmable and virtualized networks...

- bring new enablers
- create new opportunities
- but also new challenges and requirements for their control and management.
  - more versatility
  - more automation,
  - new interactions/control models...

Therefore...

- Management of these networks need to evolve
- So... what are the foreseeable evolution paths?
  - Treat network functions as software libraries
  - Better abstractions
  - Integration of machine learning mechanisms

Compose-ability of network functions

- AKA Application composition
- The purpose is to integrate SW development techniques into the network creation process
- SDN applications yhat work should not be thrown away
- In comes the NetIDE architectire

Application composition - II

- But wait a second... this sort of looks like e.g. i2rs
- So it actually boils down to multi-headed environments
  - ie. multiple independent applications addressing the same network resource
- But wait... this spells out CONFLICT

Application composition - III

- Example of conflict
  - Two different applications in an i2rs environment try to set different next-hops for a prefix in a box
  - Hot topic: is assigning different priorities to the applications enough?
- Other implications
  - Nice: Application composition may simplify the design of network elements
  - However: we need a framework with well-defined semantics
  - Example: what if a block is silent?

Integration of machine learning mechanisms

- So... if we have SDN controllers gathering information from the network...
- Why don't we just use that information in a really intelligent way?
  - But what is *really* intelligent?
  - Learning from the past?
  - Since we humans have a hard time doing that, why not using machines
  - In comes machine learning techniques to solve complex issues

Integration of ML systems - II

- So... we have the SDN controller
  - It gathers information about the nework
    - Statistics
    - Input events

Integration of ML - III

- What if we feed all this information into a ML system?
- We could train it to detect complex events from different contexts
  - Note that this is something that would be very complex to implement as an App running in the SDN Controller
- This trained system could actually talk to different Apps running in the controller and coordinate them
- But how could we do that
  - This actions would be very high level
  - In comes *Intent* as a possible way of represent these high level actions

Better abstractions

- I will try to be provocative...
- But not too much

Intent

- Let me start with NON-intents
  - (Java, Python) Libraries are NOT intent
  - They are just (low level) abstractions
  - What purpose do they serve in an NBI of an SDN controller?

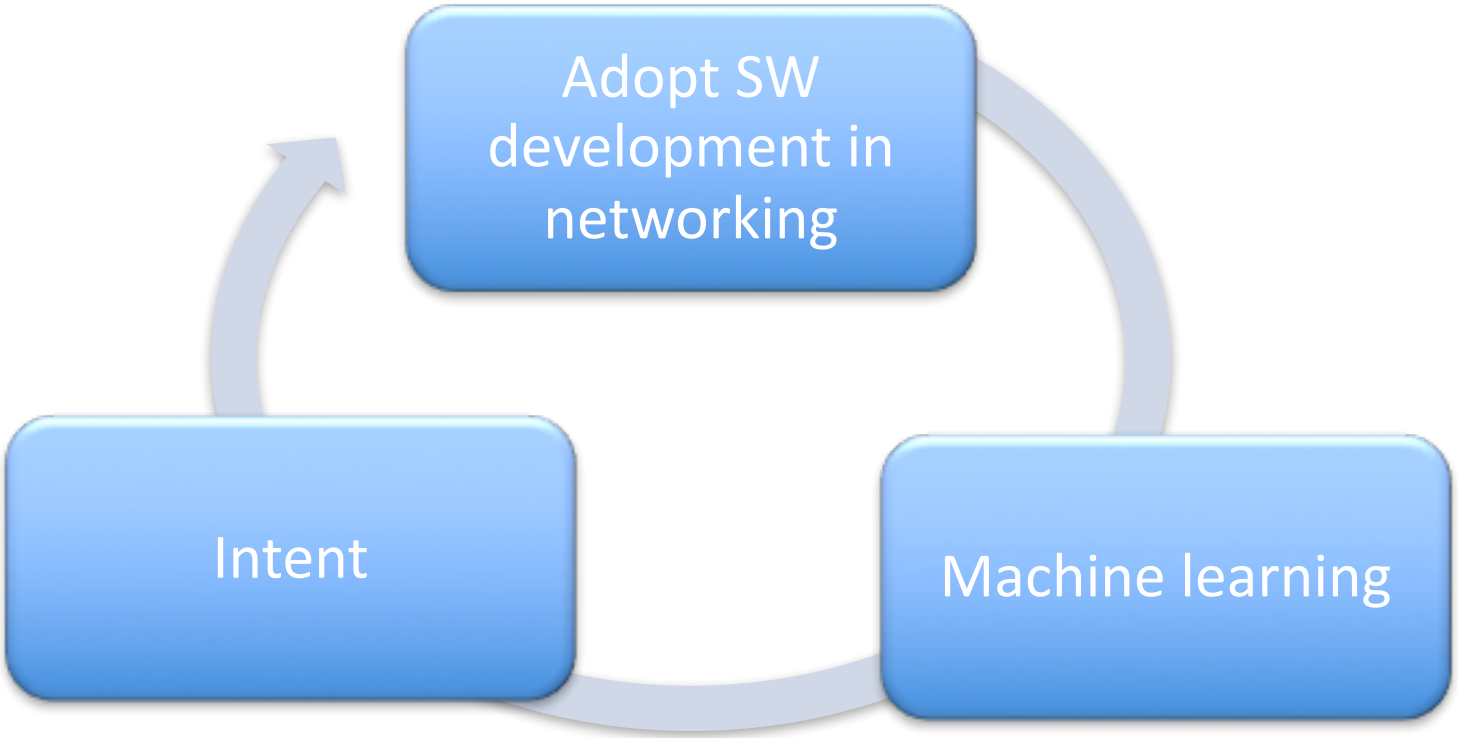
So what should Intent be? - I

- "A framework to express *network control desires* as policies..."
- A model that describes requests to alter network behaviour
- All good and nice and a very first step in the right direction
  - As long as the resulting construct is *not* just another library

What should Intent be? - II

- SDN controller independent NBIs with focus on semantics and data models
  - Declarative?
  - Imperative?
- Human readable and understadable (e.g. IBNEMO)
  - Nice for early adoption and debugging
  - However, should that be a must?
  - Nice approach that may be copied by new initiatives
    - Minimalistic approach: 20% of the language constructs that covers 80% of the use cases
- Build on models (e.g. ODL-NIC)
  - Support of YANG models seems to be ubiquitous
  - Minimalistic approach: something simple and filter through reality

Conclusion



Or was it the other way round?