Approaches to conflict resolution

Dr. Pedro A. Aranda Gutiérrez
pedroa.aranda@telefonica.com
Background

* Results from the NetIDE project
* Started as OF-centric, but were extended to cover other protocols
* Current status:
  * OF version-oblivious architecture
  * Extending to netconf
Short recap

* NetIDE architecture
  * Client/Server SDN controller paradigm of ONF
  * Network Application’s modules are given the runtime environment they expect in the client controller
  * Multi-controller support (OpenDaylight, Ryu, Floodlight, ONOS, ...)

* ONF SDN architectural concepts
* or should I say non-model?
* If you ever wanted a simple model, there you are
* Whether it is use-/meaningful is a different question
Our assumptions

* There is no conflict against network state

* Examples that may sound like conflict but are not:
  * Shut down an active interface
  * Change next hop for a given prefix

* If the result is unwanted
  * we face a bug in the application
  * but the network element should not try to correct this bug
So, when can we do anything?

* In comes the transaction concept:
* A network event and its resulting commands (or lack thereof) form a unit, we call \textit{transaction}

\[ \{Ev, Cmd^*\} \]
Conflicts become obvious with transactions

* When two or more transactions triggered by the same event are sent to the network
* Conflicts are easy to detect
* We can start thinking about automated remedies for them
Applying the results to application composition

- In SDN we should start thinking in other terms than stitching
- Example: Connecting a DMZ to the Internet
Composing a stealth firewall
Next steps

* As said before, integration of netconf
* An attempt at pro-active applications
  * Commands are not triggered by events
  * Normally when an application is started
Thank you!

Questions? Reactions...