How to Build a Rich SAL on Diverse Hardware

or

How I learned how to stop fearing and love OpenFlow 1.3+

Curt Beckmann (Brocade, ONF)

Colin Dixon (IBM, ODP)

OPEN DAYLIGHT SUMMIT

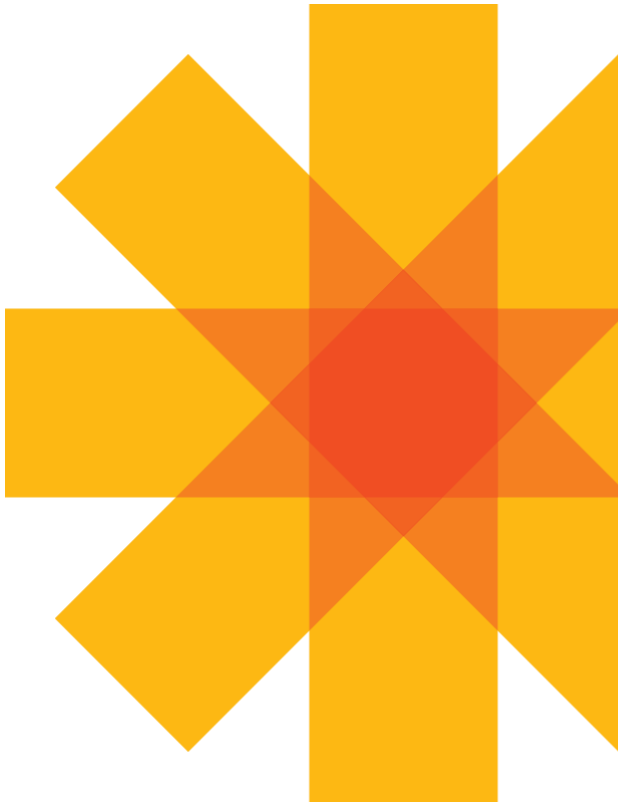
OF1.3 is Coming!

But it doesn’t solve all problems…

OPEN DAYLIGHT SUMMIT

OpenFlow 1.3+ is Coming

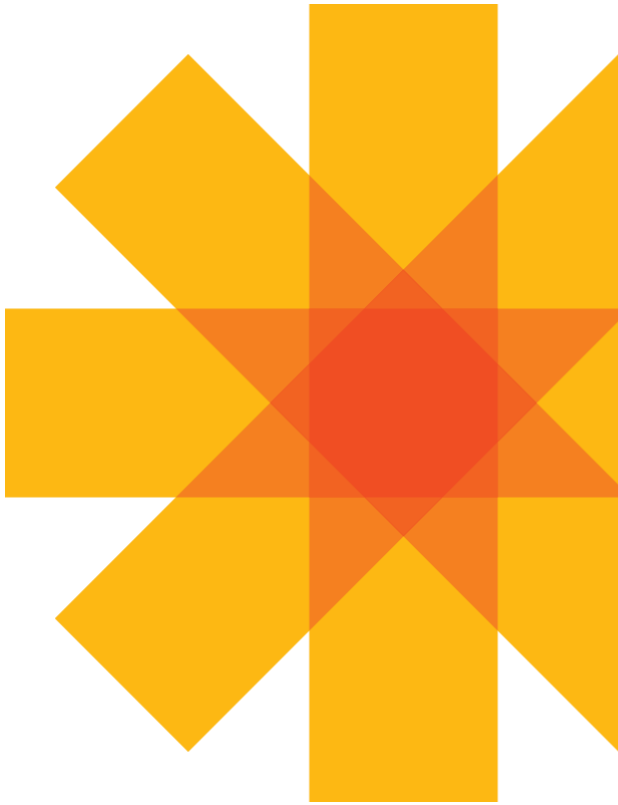
- Good news
  - Multiple tables => end of combinatorial explosion
    - i.e., separation of concerns into different tables
  - Hardware (ONF plugfest) and software support (Hydrogen, etc)
- Bad news
  - *Lots* of options and optional functionality
    - e.g., table numbers, supported matches, actions, config messages, etc.
  - How do we negotiate/discover these?
  - And map application functionality onto them?



OPEN DAYLIGHT SUMMIT

A Few (War) Stories

- IBM's OpenFlow 1.0 Implementation
  - Allows access to the L2 DMAC forwarding table
  - To get a rule in, you have to:
    1. Wildcard everything but DMAC, VLAN_ID, VLAN_PCP
    2. DMAC, VLAN_ID, and VLAN_PCP must be exact match
    3. Set the OpenFlow priority to 1000
  - This gets harder if you have more diverse h/w
- Testing OF1.3 support for OpenDaylight
  - Had to combine 3 soft switches and still couldn't get full coverage of OF1.3 features
  - Some messages rejected despite being valid



OPEN DAYLIGHT SUMMIT

Pipeline Mapping

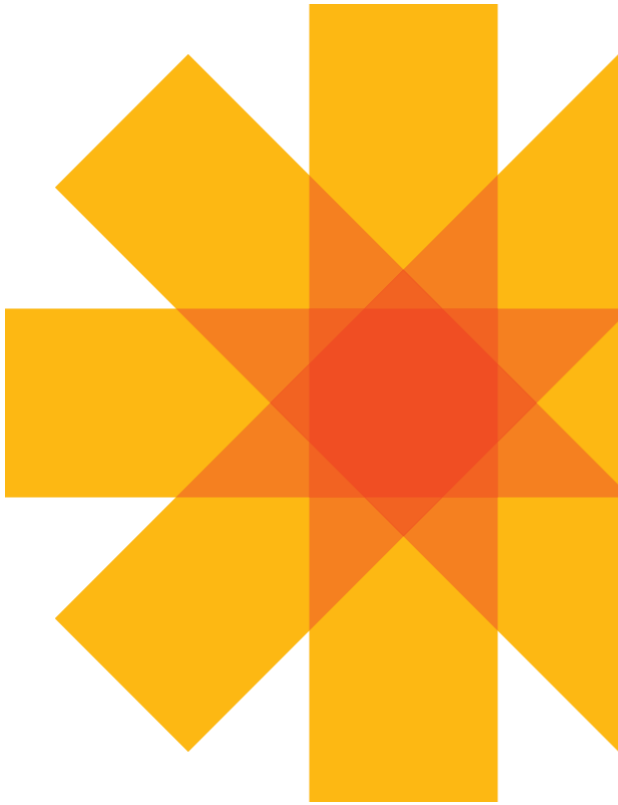
Dynamic mapping not recommended

Limits of OF1.3: Features, tables

- OF1.3 has >> new features, + multiple tables
- Many new features are optional

- How do applications code for variable hardware?
  - Answer: Hide the hardware behind abstraction layer!

- How does abstraction layer deal with it?
  - Glad you asked. We'll get back to you on that.



Multiple Tables == Pipeline

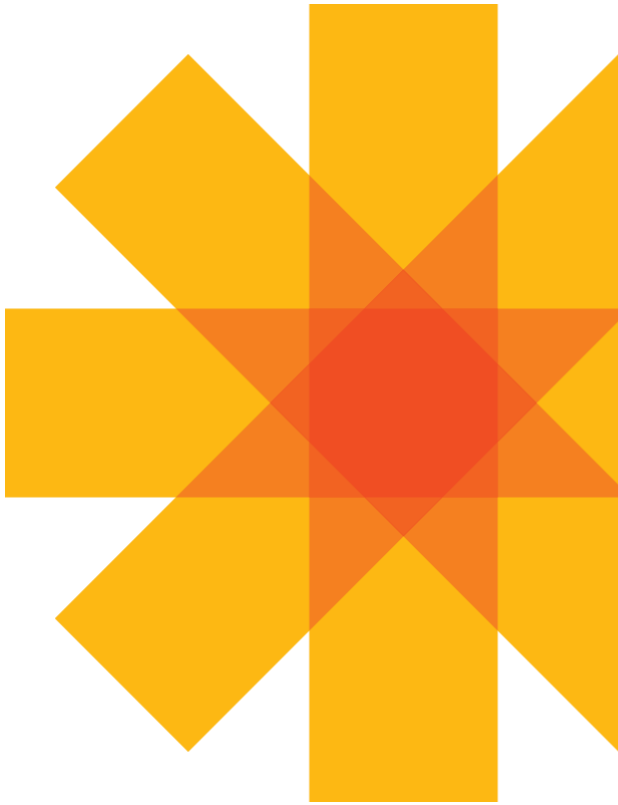
- OF1.3 is a control protocol, but…
  - The protocol language presumes a particular switch pipeline
  - OF1.0 also presumed a “pipeline” but a trivial one
    - The “trivial” pipeline was a subset of most ASIC pipelines
    - Switch vendors able to pre-resolve mapping of OF1.0 to ASIC
- But the OF1.3 pipeline exceeds all ASIC pipelines…
  - No way to pre-map the full OF1.3 pipeline on to ASICs
    - As it turns out, we shouldn’t have to…
    - Specific use cases need a subset of OF1.3 pipeline…a constrained pipeline
  - Sadly, OF1.3 did not offer a “pipeline constraint language”

OPEN DAYLIGHT SUMMIT

Image credit: http://commons.wikimedia.org/wiki/File:Empty_wave_at_Banzai_Pipeline.jpeg

We could retreat to one table

- With one table, OF1.3 mapping is (again) manageable
  - For some apps that's okay
  - But for many apps 1 table won't scale (combinatorial boom)
- Scalability → multiple flow tables…
  - We have multiple tables on ASICs
  - But it's often hard to map OF tables to ASIC tables



OPEN DAYLIGHT SUMMIT

Adapt on-the-fly! Mm, no.

- On-the-fly pipeline changes seem attractive…
  - Here, “on-the-fly” means “during flow-mod messaging time"
  - But on-the-fly is too dynamic, too “non-deterministic”
  - Operators want proven, tested, “baked”, known
- Dominant demand is actually for:
  - Control-on-the-fly
  - But with a “pre-baked” (e.g. pre-tested) pipeline
  - No single richly capable “pre-baked pipeline” is imminent
    - But… what if our framework allowed for > 1 “pre-baked pipeline”?

OPEN DAYLIGHT SUMMIT

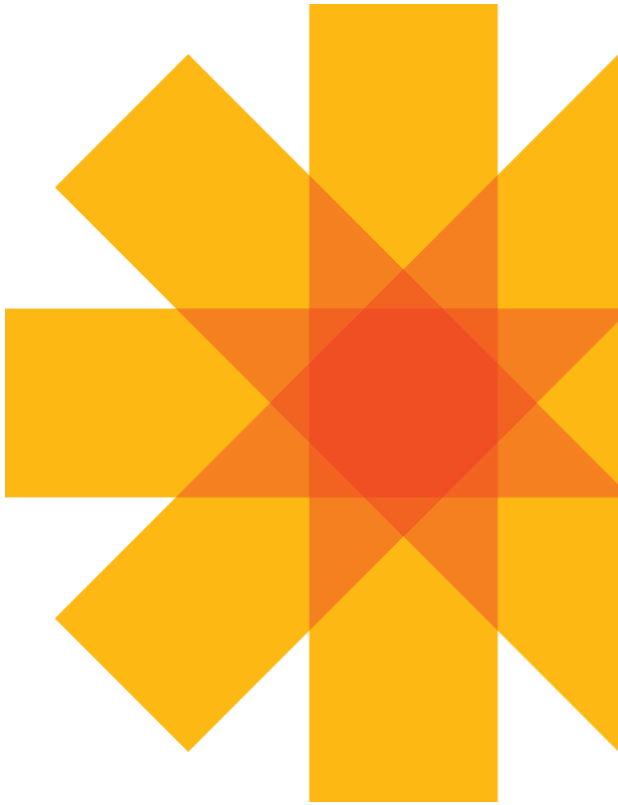
Image credit: http://www.visitrincon.com/surfing/etiquette.php

Quick Look at TTPs

OPEN DAYLIGHT SUMMIT

Pipeline mapping problems

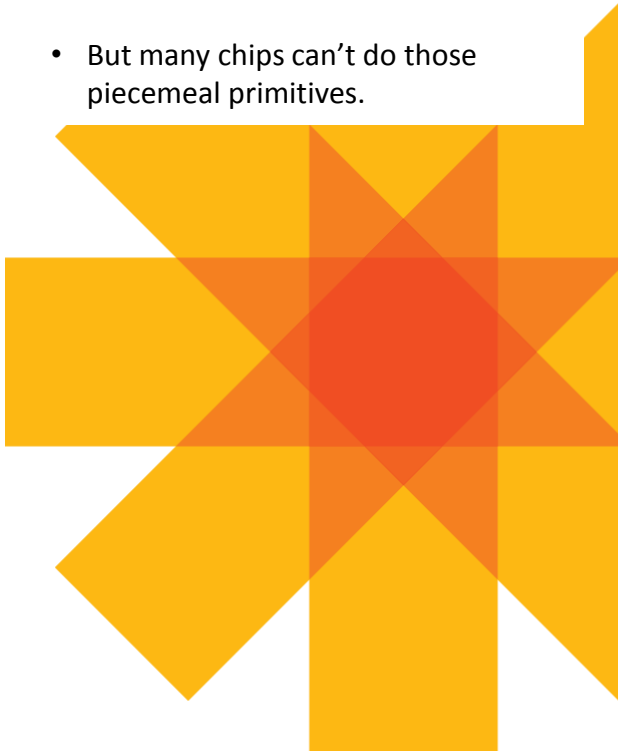
- Many many platform pipelines are deployed
- There is truly standard functionality most chips, but even for this there are cosmetic differences, e.g., table numbers
- Other features are similar, but not standard, e.g.,
  - How ACLs and PBR are exposed
  - Standard functionality can also be mixed in various ways
  - Other examples:
    - 1 router mac, or 1 per port?
    - 1 mac block, or independent?
    - LACP supported or not?
    - Both S and C tagging supported?
    - Internal (small/fast) or external (big/slow) TCAM?
    - Limited or flexible tunnel support?
    - Fully HW data plane?
    - Or some “exceptions” in SW?



Pipeline mapping solutions?

- Low-level (RISC-like) “instructions” for network devices
  - Seems like a good idea, but has issues
  - e.g., many ASICs have such primitives, but they can only be “mixed” in certain ways (that are difficult to specify precisely)
  - Like translating language word-by-word vs. sentence-by-sentence
- Richer capabilities exchange doesn’t help
- Future chips won’t fix it for a long time…
  - Consider the x in x86…
  - There’s a reason apps aren’t written in x86
- Moving problem out of run-time domain helps a lot…
  - Again, pre-baked pipelines

- CISC-like “Route” maps to RISC-like primitives:
- “SMAC rewrite” + “DMAC rewrite” + “TTL Decrement”
- But many chips can’t do those piecemeal primitives.

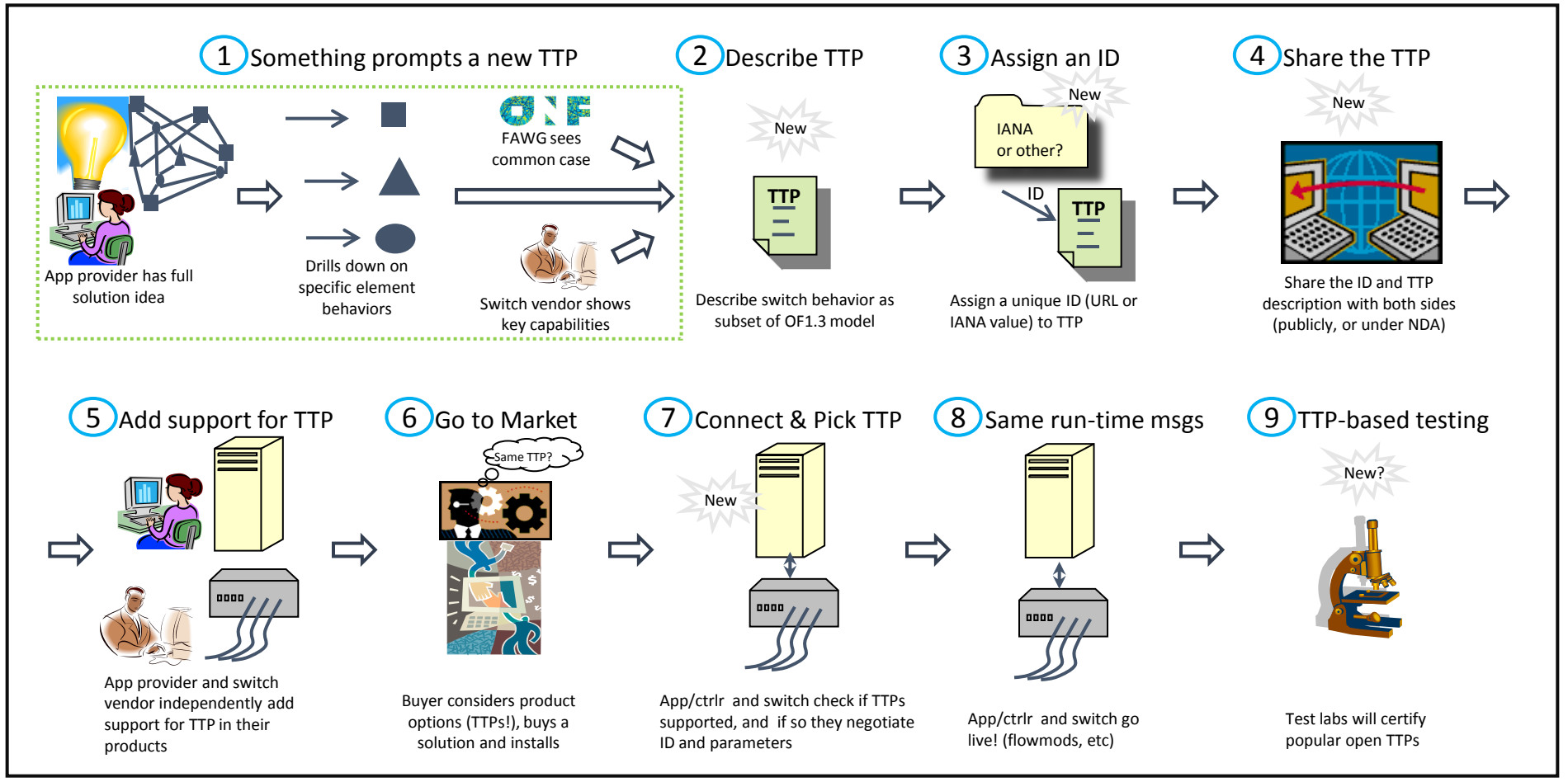


How TTPs Can Help

- TTPs are “Table Type Patterns” that market participants can define
  - TTPs are 1st gen of “Negotiable Datapath Models” (NDMs)
- TTPs = “pre-baked pipelines” specific switch funcs in OF1.x terms
  - With TTPs, pipelines can be mapped before run-time
  - Switches, controllers become deterministic (as they need to be)
    - Once TTP is agreed, Controller uses only TTP messages,
      Switch supports all TTP messages, All messages are valid OF1.x messages
- TTP Examples:
  - “VID Mapping L2 Switch”, “VXLAN Gateway”, “NVGRE Gateway”,
    “v4 Router w Ingress ACL”, “v6 Router w Egress ACL”, “MPLS Edge & Core Router”
- TTPs help sort out interoperability
  - Product sheets list supported TTPs, clarifies what works with what

OPEN DAYLIGHT SUMMIT

TTP “Lifecycle”



OPEN DAYLIGHT SUMMIT

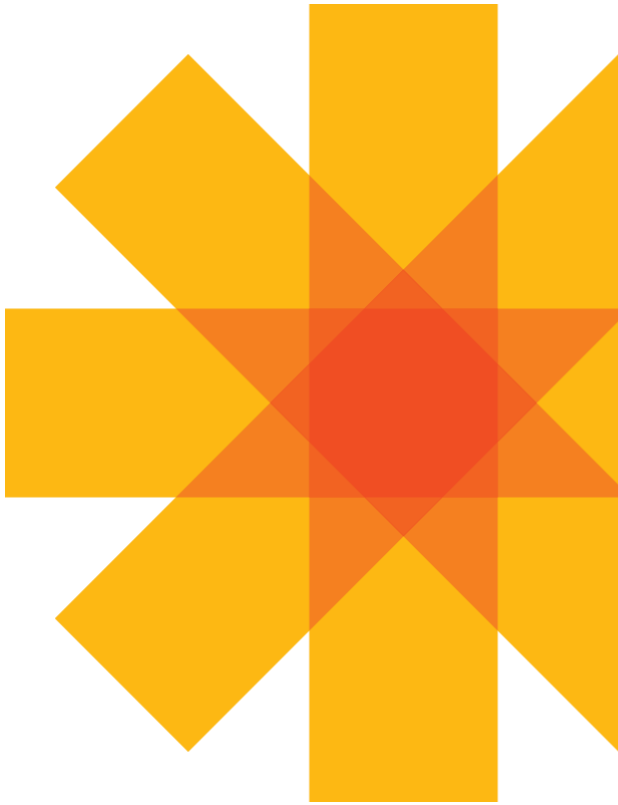
TTP's Impact on OpenDaylight

OPEN DAYLIGHT SUMMIT

The Implications for OpenDaylight

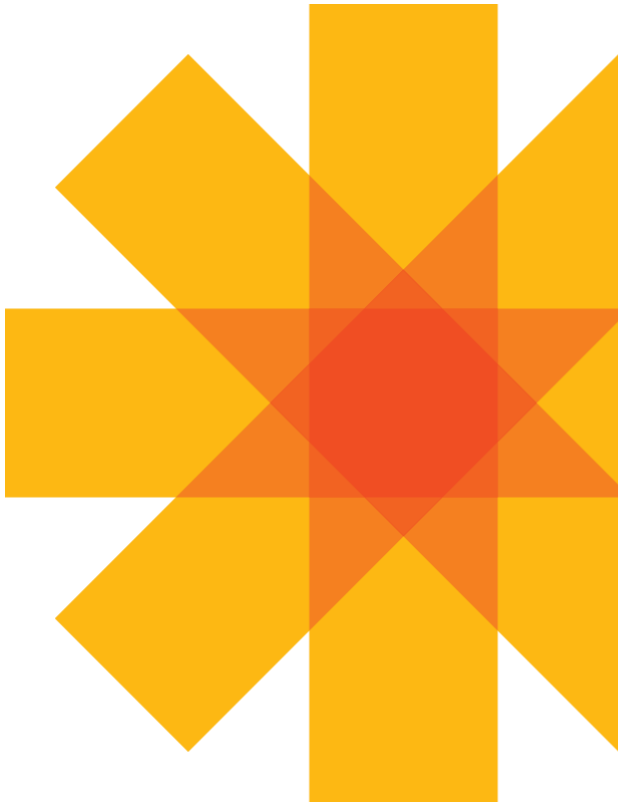
- TTPs assumed a controller would negotiate with a switch
  - Awareness that apps needed to be included in the picture
  - But no sense of how to handle that

- OpenDaylight is a controller, but it's also platform(-ish)
  - so it can – and should – negotiate on behalf of apps
  - How should we expose this?



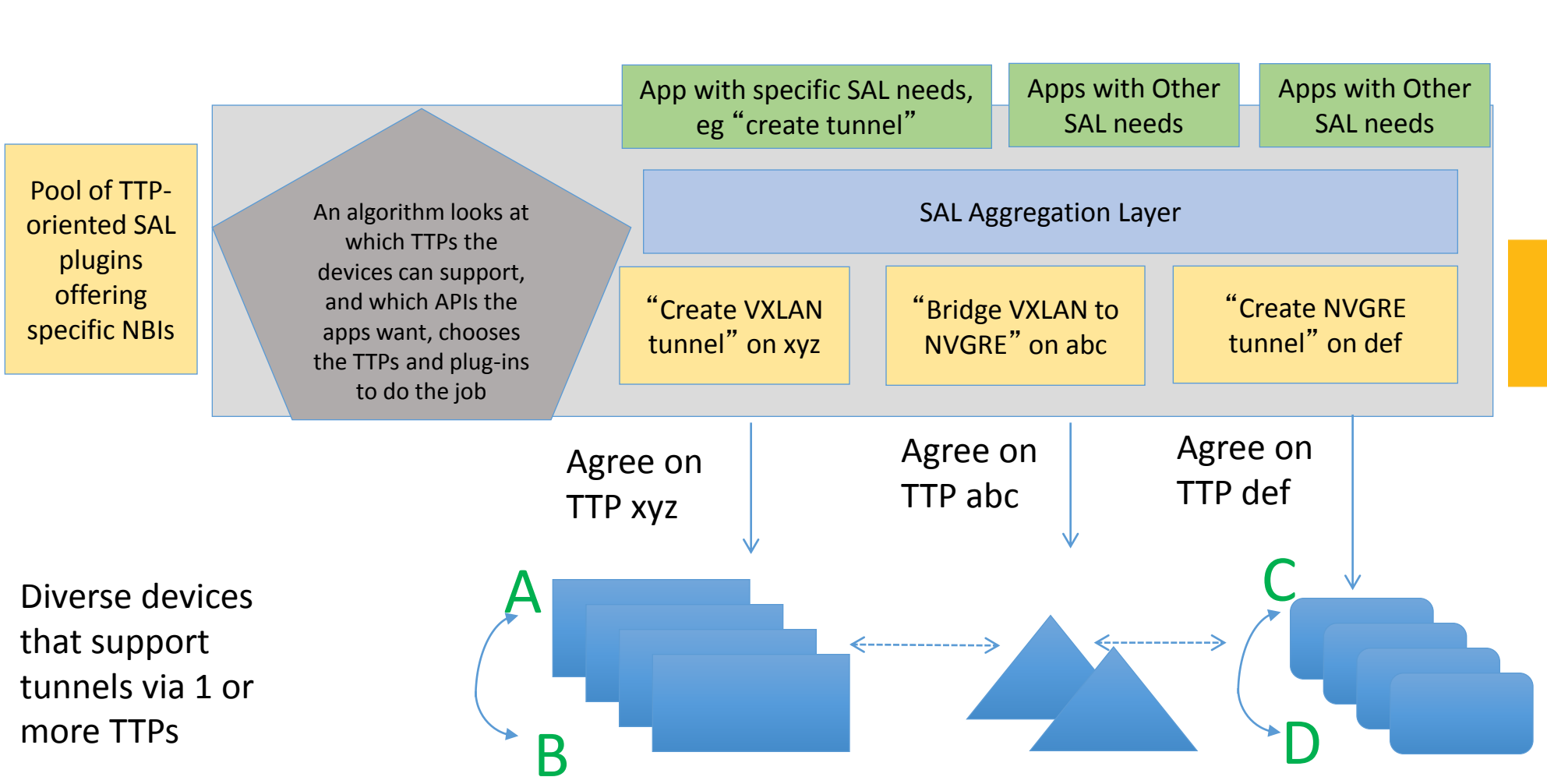
Looking at TTPs from NBI PoV

- Some apps will want to, say, “create tunnels”
  - At the implementation level, there many ways to do this.
  - Many (not all) TTPs will support “create tunnel”
  - The TTPs define a functional ifc to devices
  - Pool (library?) of TTP-oriented Service AL plug-ins offer the tunnel API northward
- Need ODL code that looks at the SAL and TTP options
  - Algorithm considers requested services and available TTPs
  - Then looks in library of SAL plug-ins to see what bridges the two
- The Point: Deliver requested services on the devices we have
- Thus we could provide a clean, abstract tunnel service on top of OF1.{3,4}



OPEN DAYLIGHT SUMMIT

TTPs and NBIs Pictorially

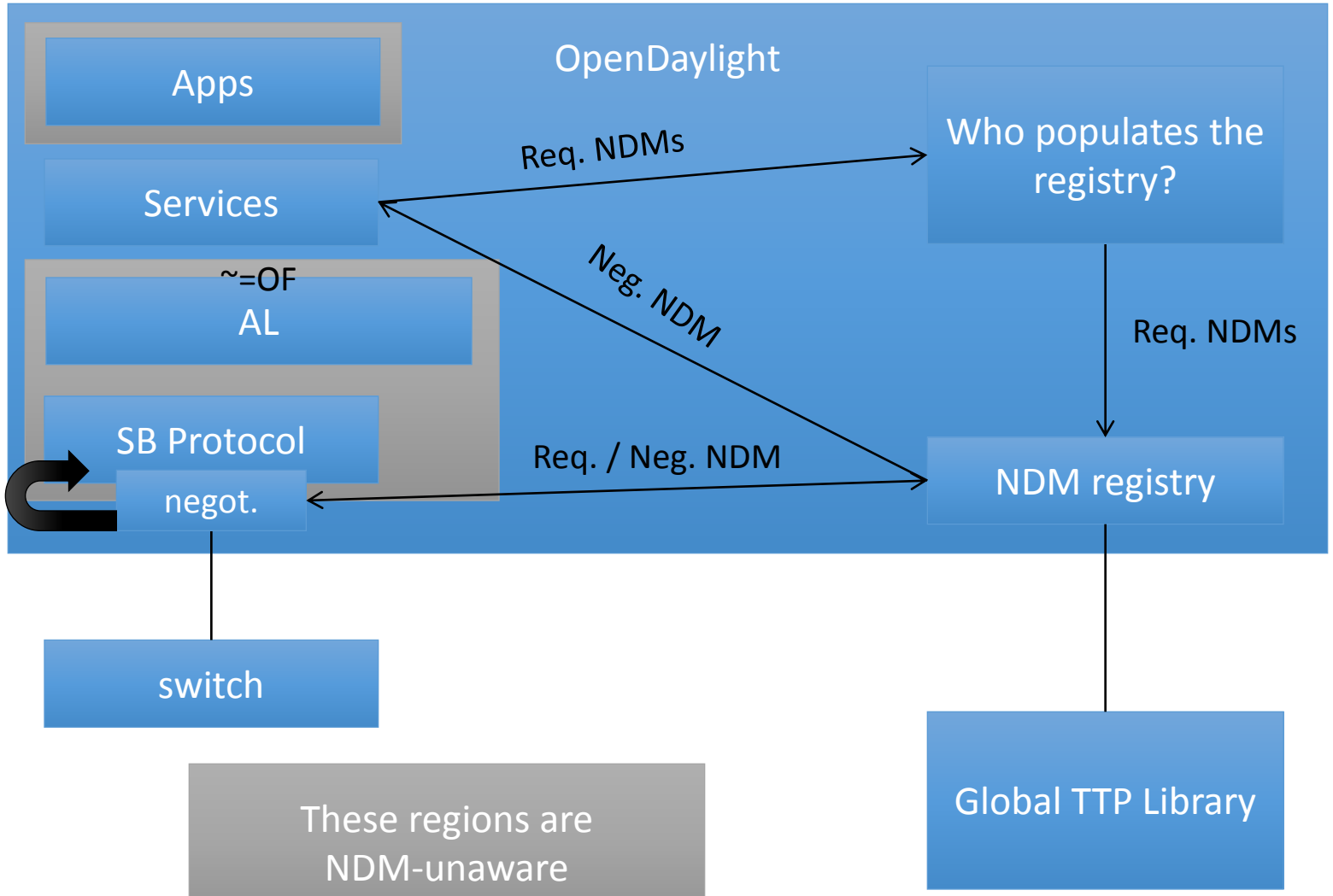


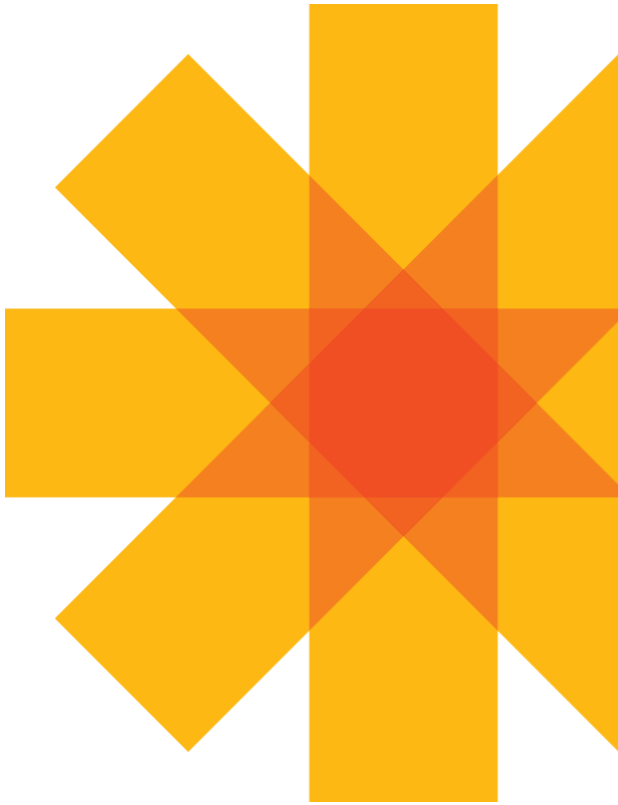
Diverse devices that support tunnels via 1 or more TTPs

Goal: Support pure VXLAN (A-B), or pure NVGRE (C-D), or even bridge between both (A-D)

OPEN DAYLIGHT SUMMIT

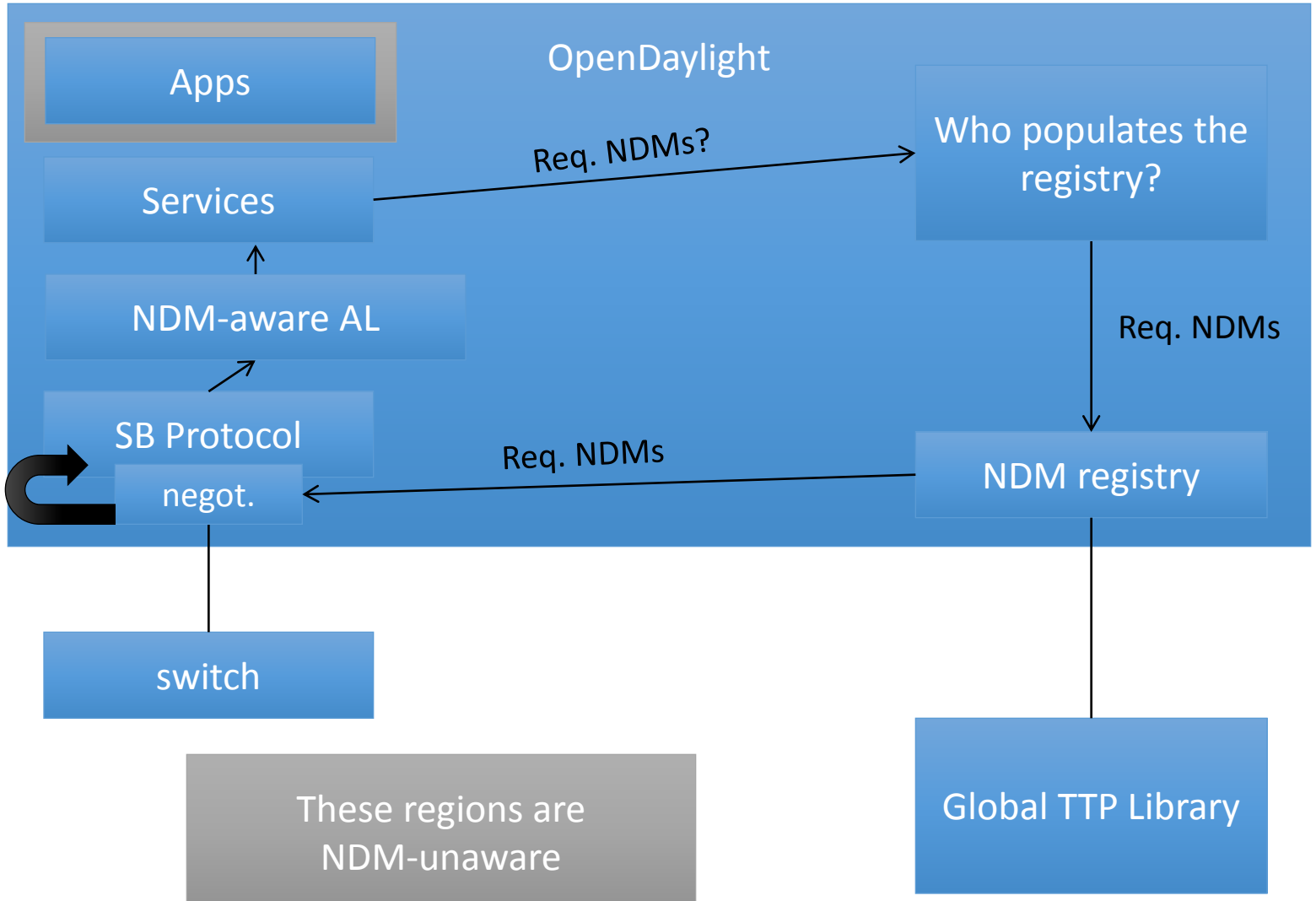
Option 1

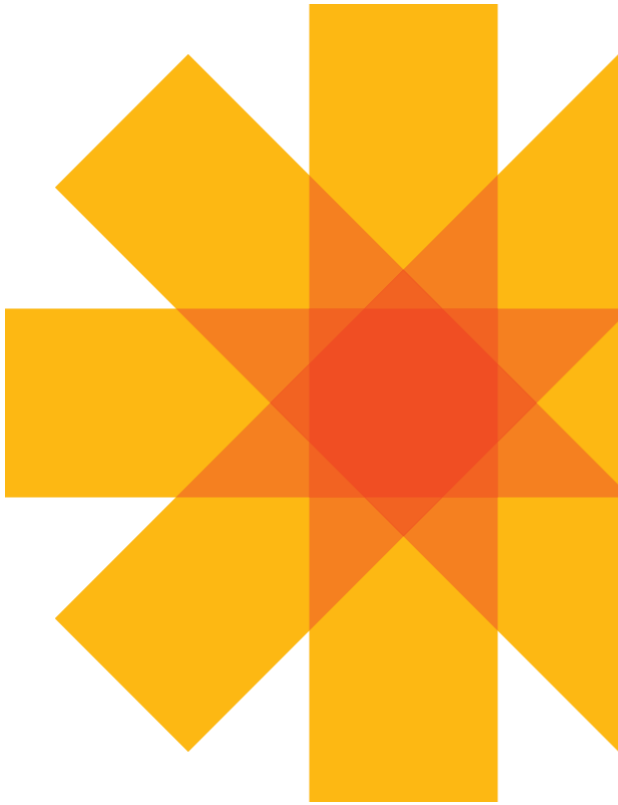




OPEN DAYLIGHT SUMMIT

Option 2





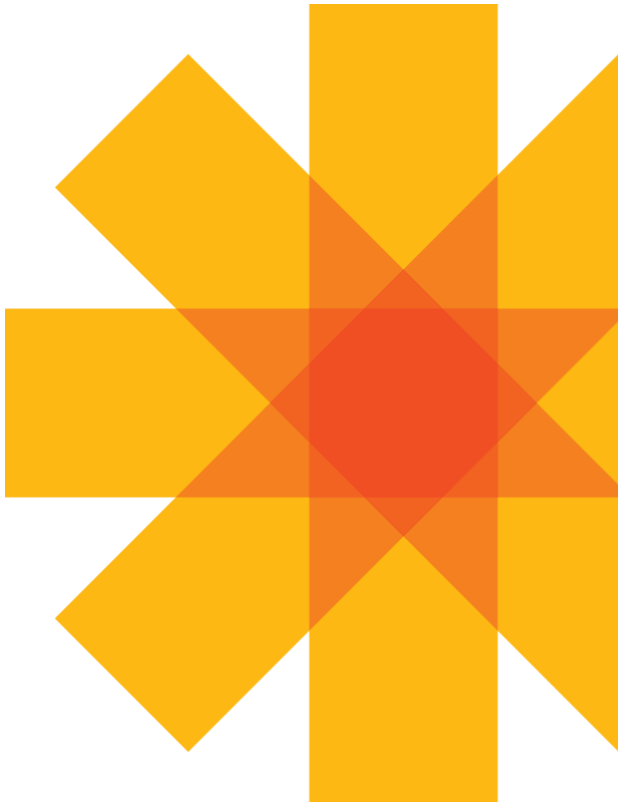
OPEN DAYLIGHT SUMMIT

Summary

OPEN DAYLIGHT SUMMIT

The main point

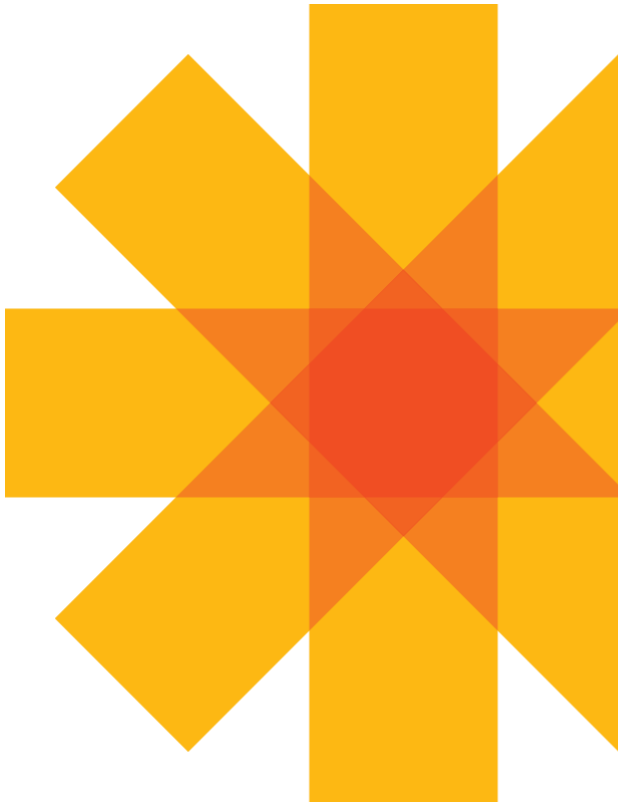
- Huge variety of devices, but “bucketable” by function (i.e. as TTPs)
- Apps want services
  - That means APIs that are more abstract than flow tables
  - Mapping service APIs onto unconstrained protocol [OF1.3] is a mess
- Mapping services onto devices depends on details of both
- Just as variety of apps will be supported by smaller set of APIs, variety of devices can be supported by smaller set of TTPs
- Controller should map bounded APIs to bounded TTPs/NDMs
  - And provide value by sorting out the abstract-to-device mapping
  - Other southbound interface could be NDMs too (I2RS, PCEP, etc)



OPEN DAYLIGHT SUMMIT

How to get Involved

- OpenDaylight Efforts
  - Either a separate project or part of openflowplugin
- Open Networking Foundation
  - Forwarding Abstractions Working Group
  - If you're an ONF member, you can:
    - get an early copy of the TTPs document
    - attend the work day
- ONF publication of TTPs should be here soon, meanwhile:
  - http://www.slideshare.net/US-Ignite/interoperable-open-flow-with-nd-ms-and-ttpsbeckmann

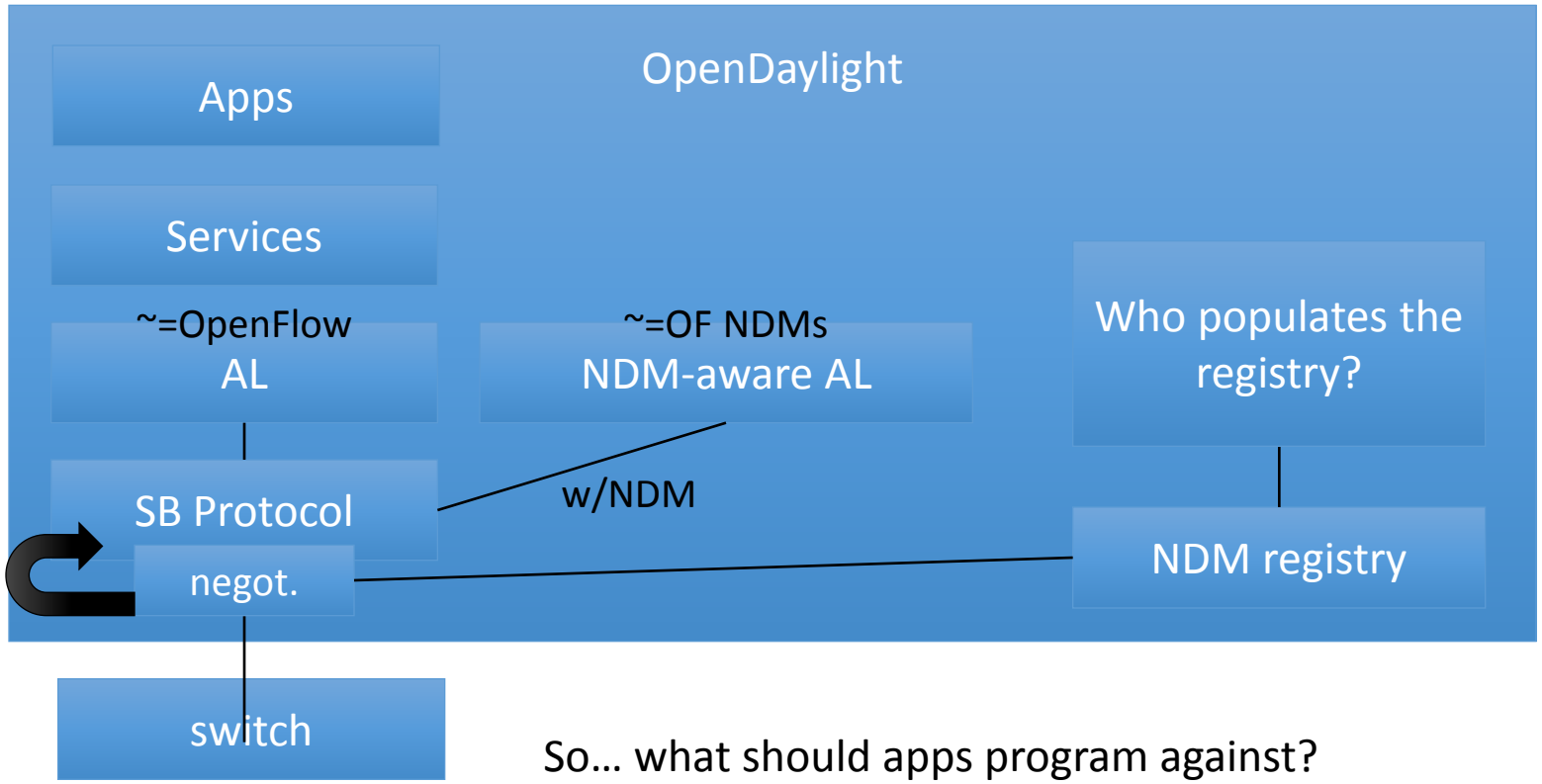


OPEN DAYLIGHT SUMMIT

Q & A

OPEN DAYLIGHT SUMMIT

My guess is that we do:
AL ***OR*** NDMaAL



For a given switch: assume one NDM active
How is it presented upward?
1.) the NDM
2.) any more specific NDM
3.) plain OF that throws errors on bad flows

Curt (and I think I) think that these are probably suboptimal to expose to apps. Maybe OK for services.

So… what should apps program against?
- Probably not NDMs b/c we want to be able to switch those out easily
- Think about the I2 NDDI use case with VLAN ID splicing? NDMs allow for us to avoid the Cartesian product.
- Maybe just paths?
- Do apps actually care about NDMs? Or services?
- Maybe make the services depend on NDMs and the apps depend on services which may or may not exist depending on NDMs.

OPEN DAYLIGHT SUMMIT