GENI OpenFlow Service Abstraction

Tim Upthegrove, GPO

June 22, 2014

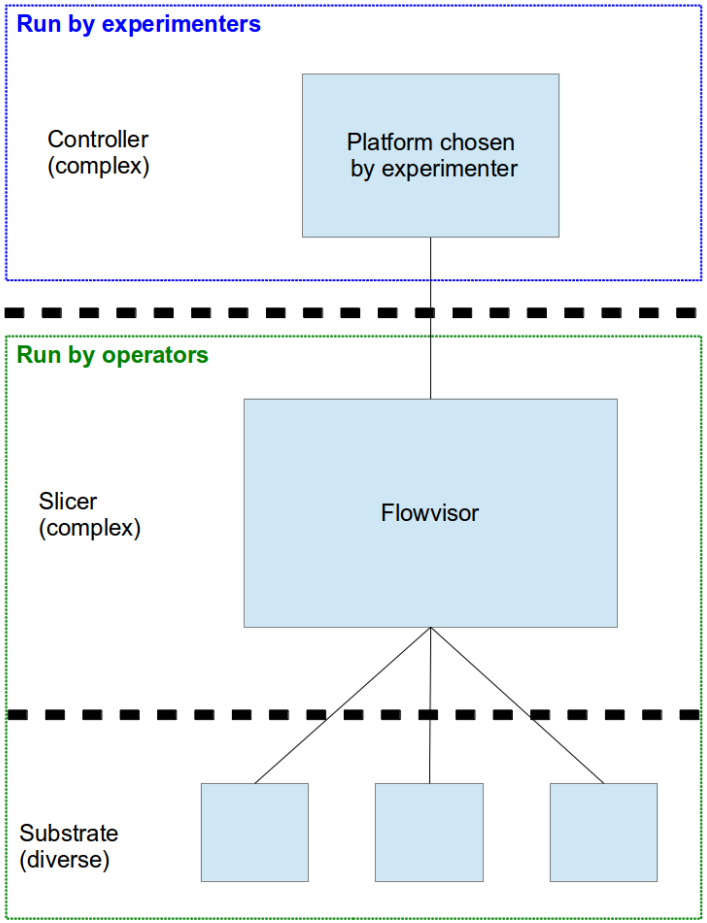
Sponsored by the National Science Foundation

Overview

- Existing solution
- Current problems
- Proposed solution
- Example deployment

Existing Solution

- Operators run and install slicer (Flowvisor) for OF aggregate
- Flowvisor can do a lot:
  - Can slice by anything
  - Flowmod augmentation
- Operators run a diverse substrate of OpenFlow gear
- Experimenters deal with complexity of substrate and limitations of slicer

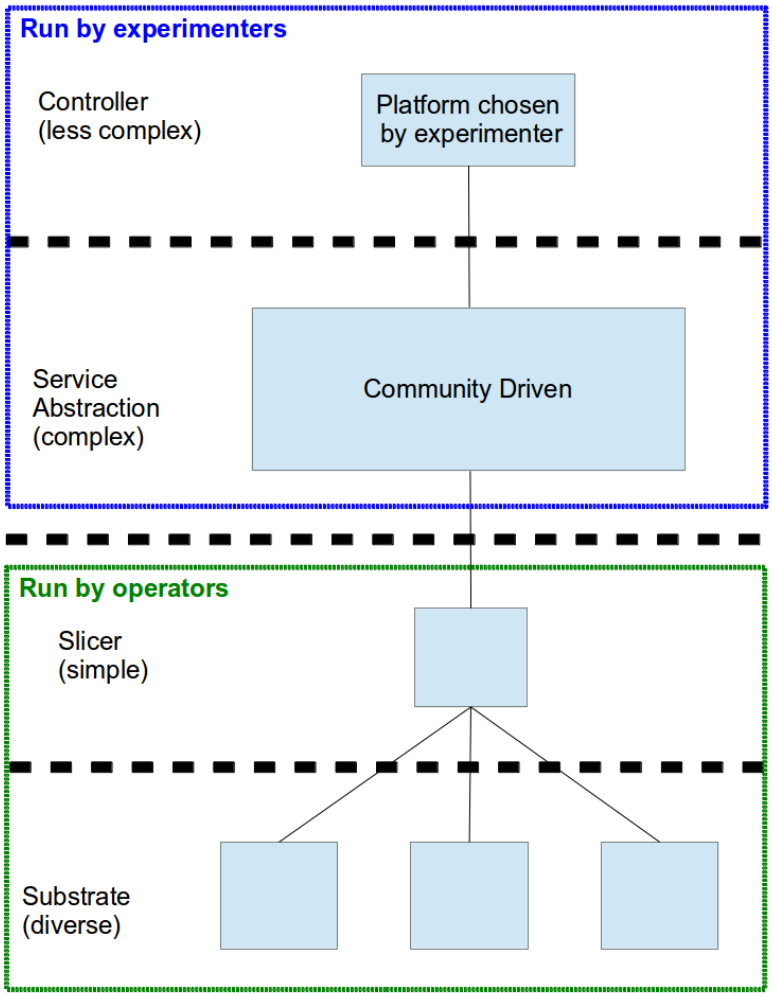


Current Problems

- The substrate is intentionally diverse and necessarily complex, but it shouldn't be so difficult to use
- Slicing layer is more complex than necessary, and it is difficult to update GENI-wide
- Researchers are working around the same GENI-specific problems that have nothing to do with their research...
- And aggregate developers with OpenFlow substrates are solving some of those same problems!

Proposed Solution

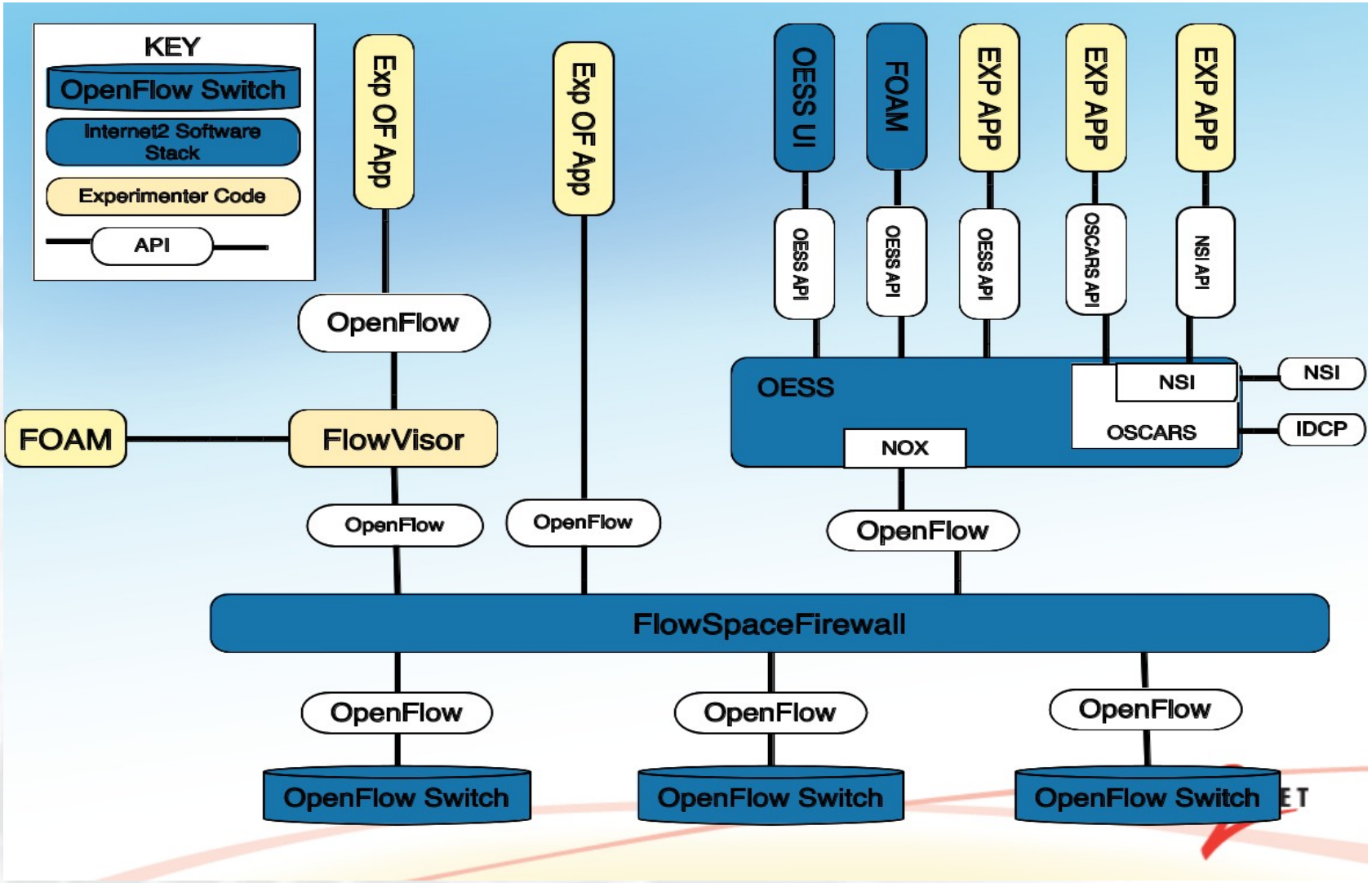
- Operators run and install simpler slicer with fewer requirements and features
- Add community driven service abstraction layer for solving GENI-specific issues
- Operators run a diverse substrate of OpenFlow gear
- Experimenter can **focus on research problem**



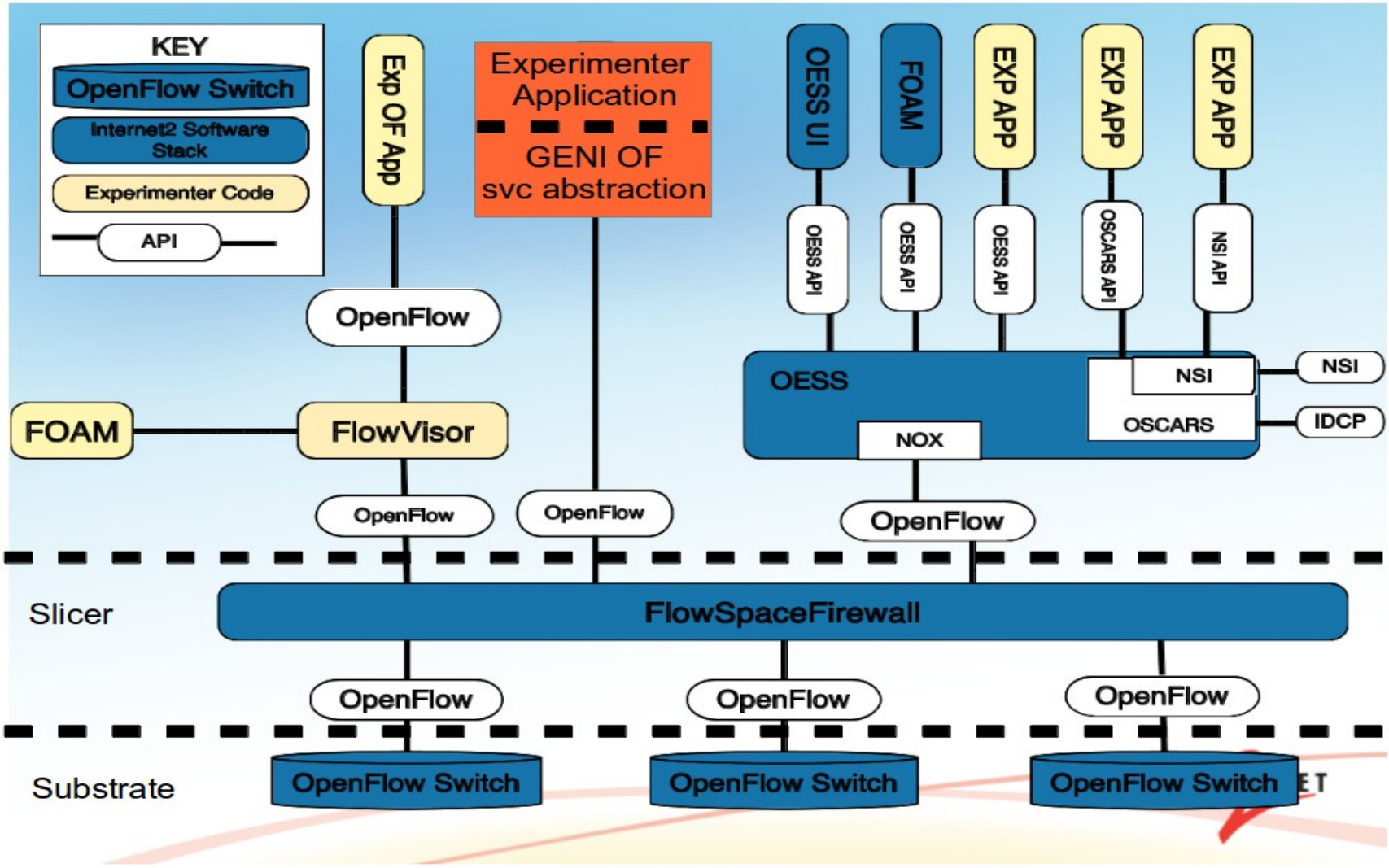
Proposed Solution (cont'd)

- **Overarching goals**: Hide complexity where it is not required and increase stability
- Potential features of service abstraction layer
  - Normalize experimenter view of network
  - Layer of indirection for compilers or translators
- Benefits of splitting service abstraction layer from slicer
  - Slicer will require fewer GENI-wide updates
  - Service abstraction layer can develop at its own pace
  - Service abstraction layer can be used to develop GENI services regardless of choice of slicer
- The service abstraction layer is **not required** for experimenters, developers, or operators who want to deal directly with substrate

Example Deployment: Internet2



Example Deployment: Internet2



Questions?

Opinions on slicer or service abstraction layer?

Experimenter Problems

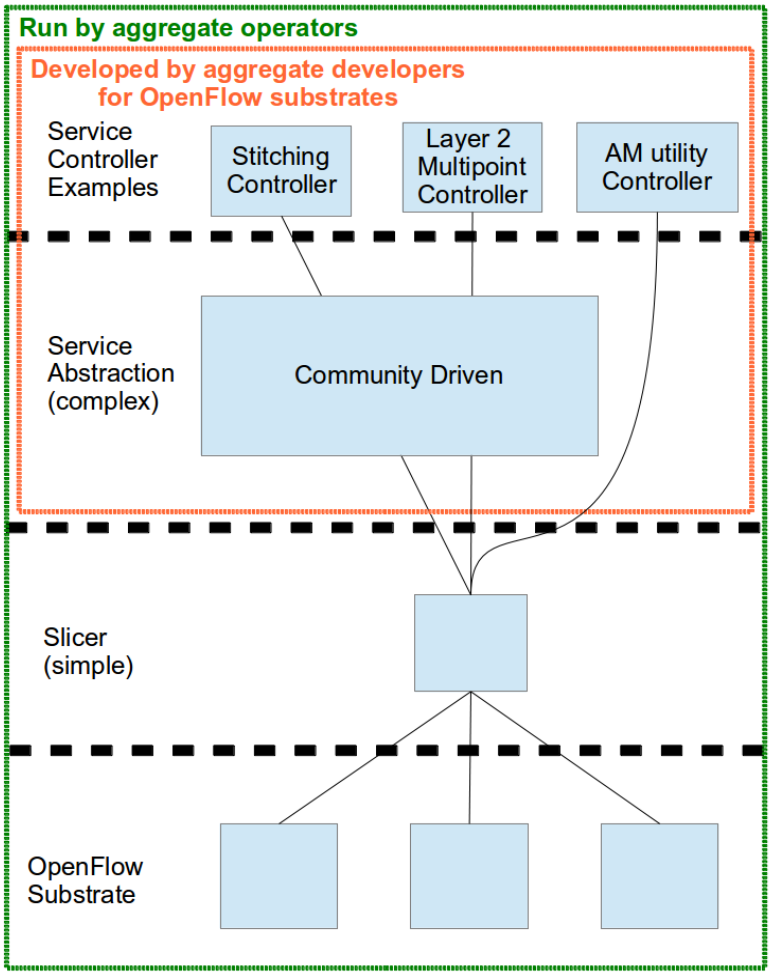
- Diverse set of switches
  - Different match support, action support, and capabilities
  - VLAN hybrid vs. port hybrid vs. non-hybrid
- All data plane traffic must be VLAN tagged, but control plane traffic sometimes has VLAN tags and sometimes doesn't
- VLAN IDs change across data plane of a slice (and sometimes sliver!)
- Parts of spec not supported due to slicer limitations or bugs
- Many controller platform features don't work out of the box

Observations about Operations

- Slicing is the lowest fundamental layer of control plane in GENI
  - MUST be stable
  - Can decrease complexity of code at slicing layer if we consistently slice infrastructure in the same way (for GENI, VLAN ID)
  - Experimenter abstractions should be solved on top of slicing function
- Updating OpenFlow slicing layer across GENI is difficult
  - Testing and validation have to happen for each bug fix or feature release
  - Many different operators must take action
- Most OpenFlow substrates can support many different services
  - Experimenter OpenFlow control
  - GENI stitching point-to-point transport

Developer Use Case

- If substrate is complex, can simplify service controllers
- Can still directly access slicer for important operational functions
- Allows for diversification of substrate over time with less new glue code
- In the end, the entire stack is run by operators



Complexity Sausage

RFC 6670 Section 3.4:

"... simplification in one element of the system introduces an increase (possibly a non-linear one) in complexity elsewhere. This creates the 'squashed sausage' effect, where reduction in complexity at one place leads to significant increase in complexity at a remote location."