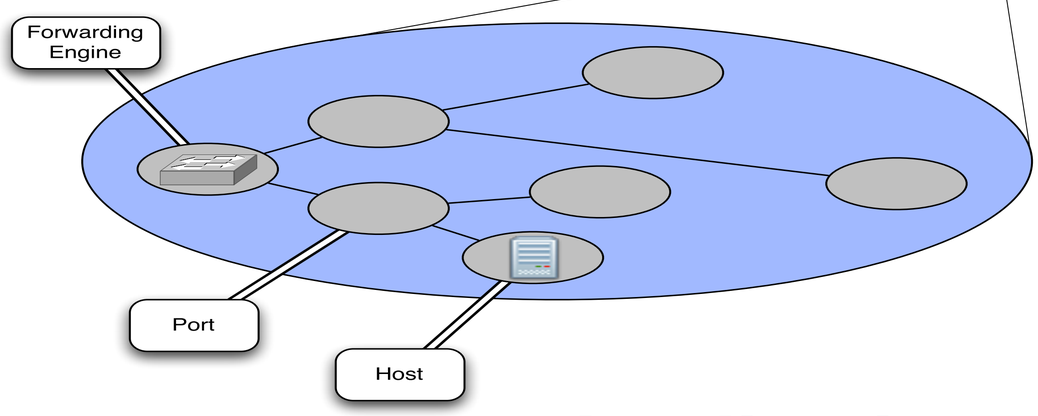
Detecting Network Invariant Violations with CLINT

Colin Scott, Andreas Wundsam, Justine Sherry, and Scott Shenker

Background

Software-Defined Networking (SDN) control applications are functions over an abstract network view

f(view) => configuration



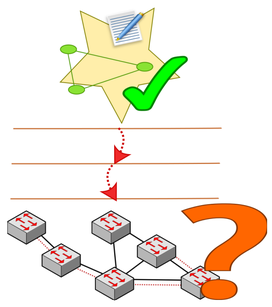
- View encapsulated in the Network Object Model (NOM):
  - Predefined entities (e.g. Switch, Port)
  - Handlers for abstract event types (e.g. PortDown)
- NOM may represent physical or virtual entities
- Applications configure the network by examining and manipulating entities in the NOM

Implications for Troubleshooting

The Good

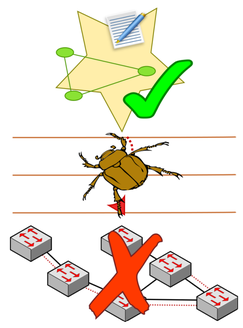
Abstractions facilitate concise specifications of behavior

Easy to reason about state



The Bad

Additional distance to 'the metal' makes it difficult to understand low-level behavior



The Ugly

Platform itself is complex

Bugs in platform affect behavior of control applications

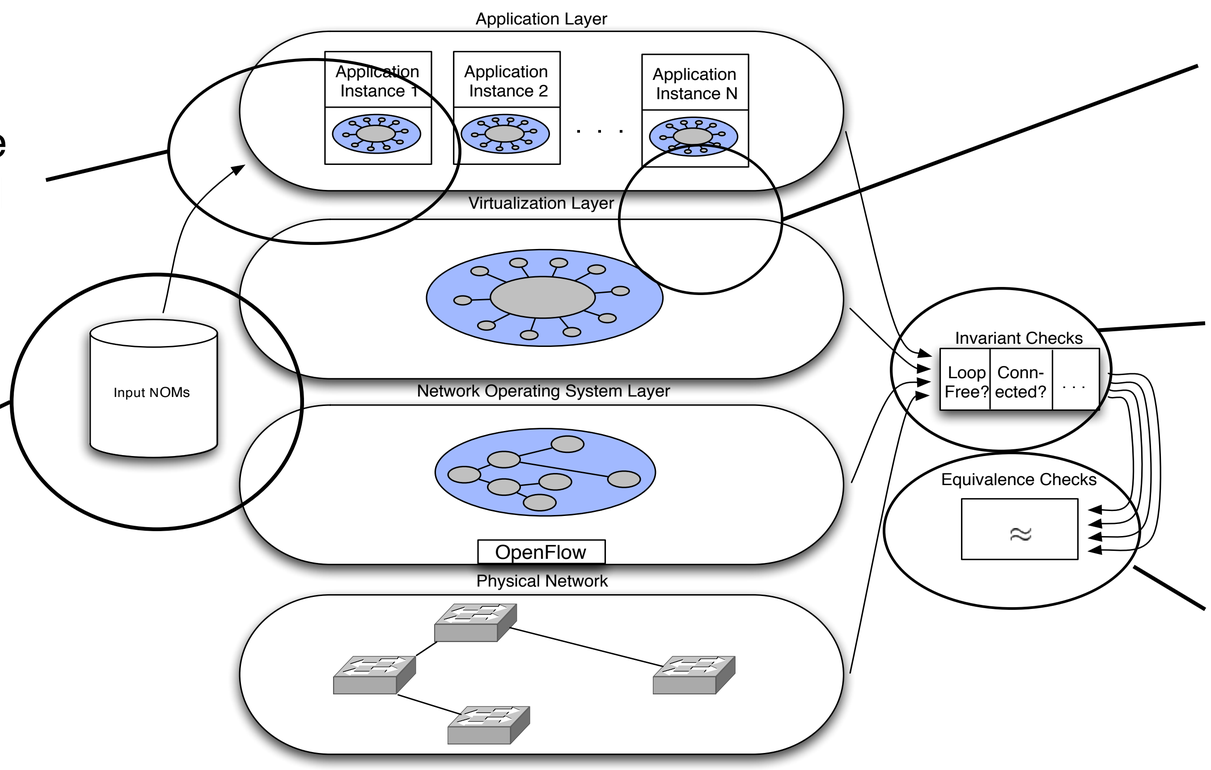
Goal: mechanism to verify all layers of SDN stack for any given control application

Approach

Key insights:

Since applications are functions, we can treat them as black boxes: feed in NOMs, and record output. Later, check that output conforms to expected invariants.

Virtualized NOMs are intentionally simple, so the state space of relevant inputs is tractable



Well-defined interfaces between layers allows us to check each layer separately.

Well-defined data structure (NOM) makes invariants easy to test

Application output represents expected behavior of network; verify correspondence of each layer to intended configuration

Classes of Invariants

Pre-defined

Some invariants apply to all network control applications. For example:

- Loop-freeness
- Graph connectivity
- Per-packet routing consistency

User-defined

Application developers may define their own invariants, such as:

- ACL placement requirements
- Link utilization limits
- Minimum redundancy levels

Application-derived

Expected behavior of the network is codified in the application's output.

Translate between each layer's representation of configuration state, and verify correspondence.