Software Defined Networking

SDN Controller
Building and Programming

Yotam Harchol

December 2013

Outline

- Floodlight SDN controller
- Indigo OpenFlow Switch
- Problems in controller development
- Real-life SDN applications

Please ask questions during my talk

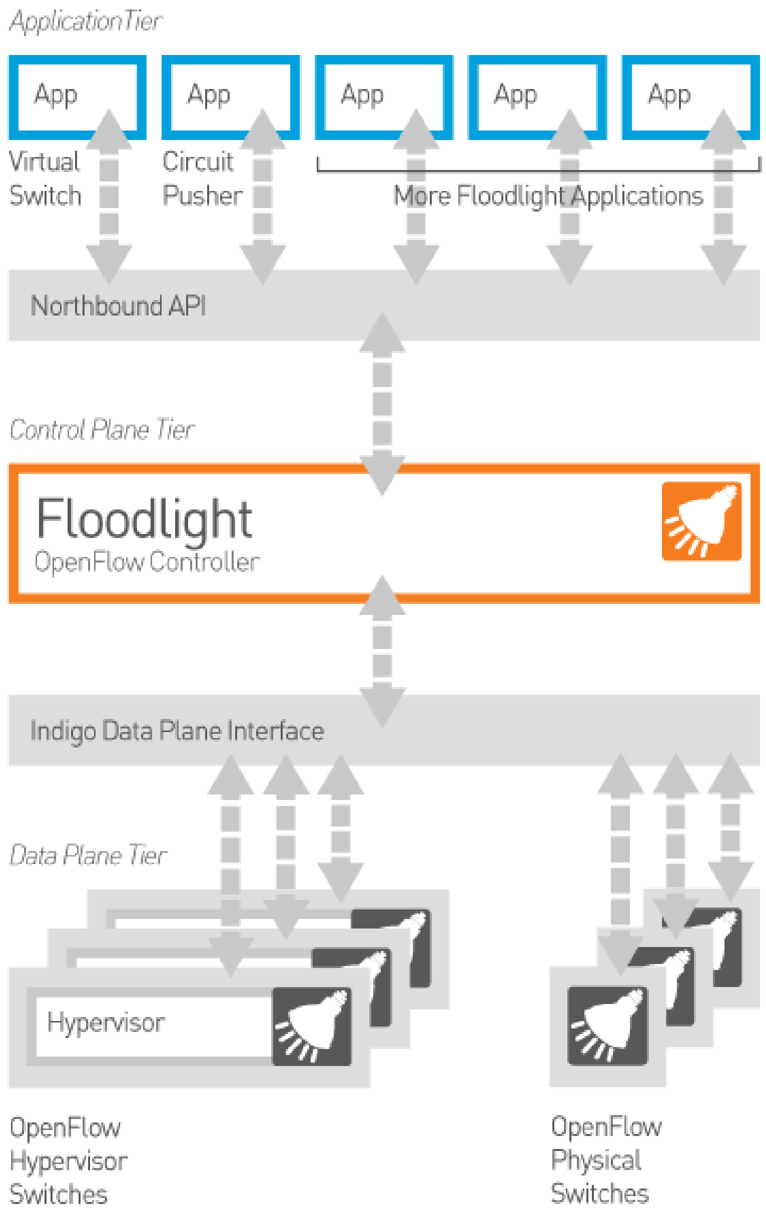
About Me

- Ph.D. student at the Hebrew University
- Advisers:
  - Prof. Anat Bremler-Barr (IDC)
  - Dr. David Hay (HUJI)
- Research areas: networking, middlebox performance, SDN, network security
- Spent last summer at Big Switch Networks

Floodlight

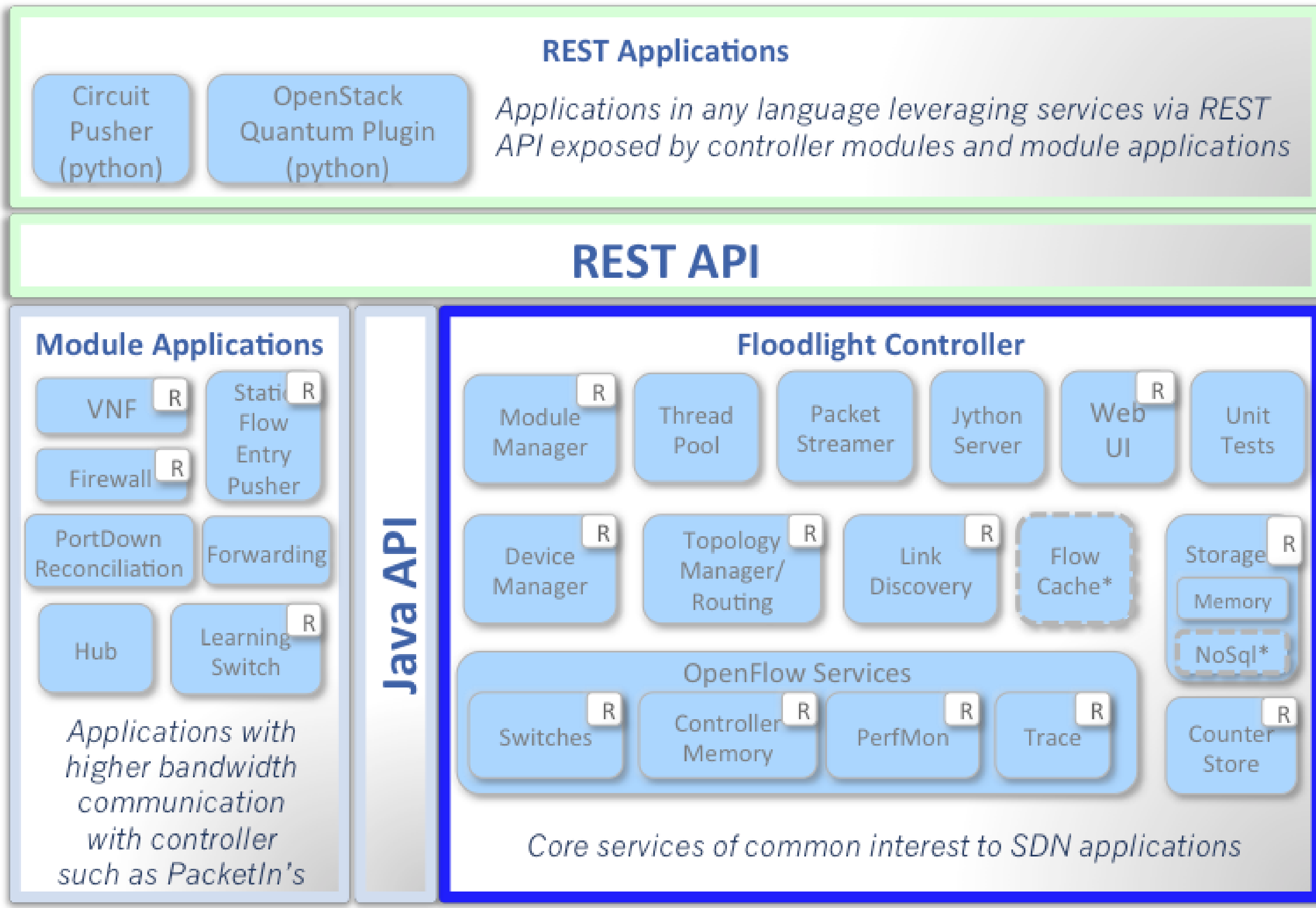
General Architecture

System Architecture



Source: projectfloodlight.org

Controller Architecture



* Interfaces defined only & not implemented: FlowCache, NoSql

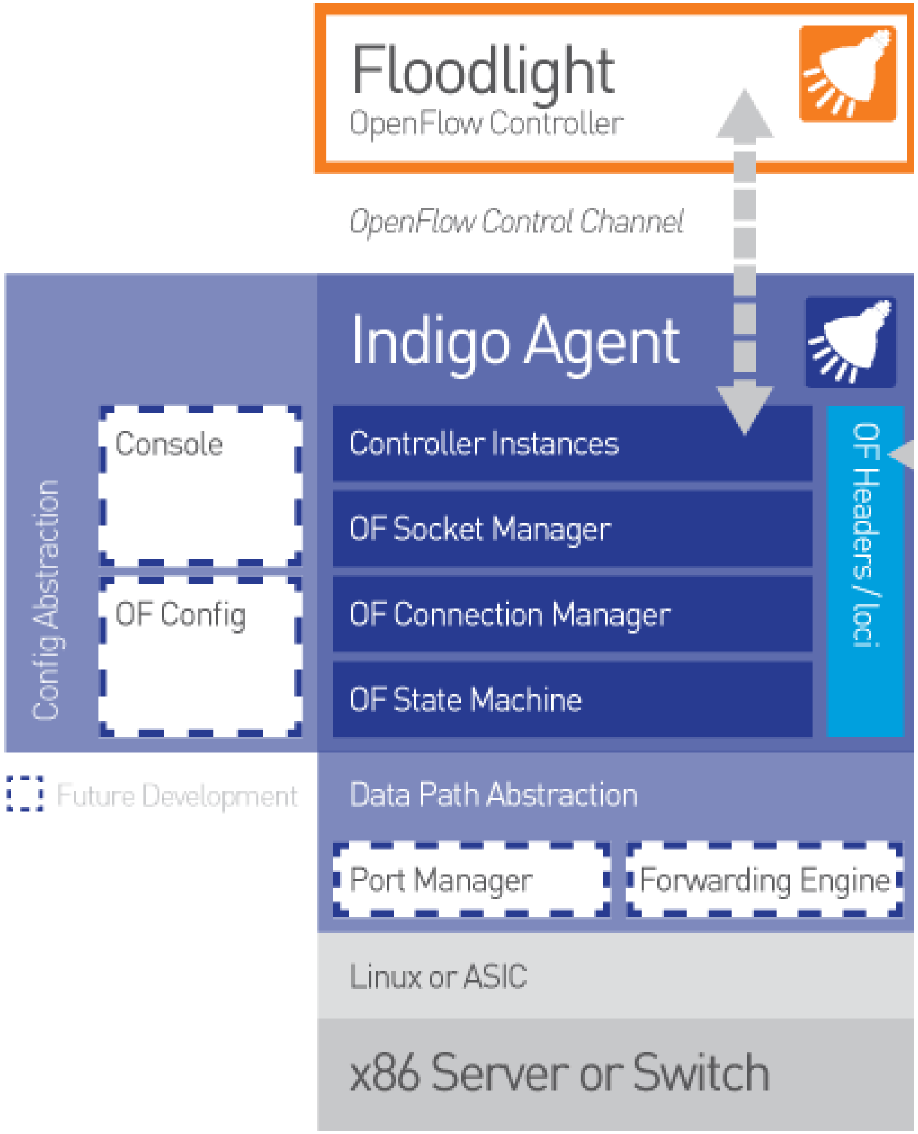
Source: projectfloodlight.org

Indigo

Open source OpenFlow switch

For software and hardware implementations

Indigo Architecture



IVS
Indigo Virtual Switch

SwitchLight
Open source physical switch

Source: projectfloodlight.org

Problems

Problems in controller (and switch) development

Testing and Verification

- **Unit tests** – every class has its own unit test. All tests are executed before code is merged into main branch
- **External tests** – these tests are more comprehensive and use mininet and physical switches to test that functionality is maintained (runs after merge and rebuild)
- QA

Vendor Extensions

- OpenFlow is not enough
- Extensions should be supported by the data plane
- Data plane is manufactured separately
- Possible solution: extend both controller and switch software

Protocol Evolvement

- The OpenFlow protocol evolves quickly and has dramatic changes between some of the versions (e.g. 1.0 and 1.2, 1.3)
- This requires adaptations in controller, applications, and the switches (virtual or physical)
- Backward compatibility is a major concern as well (e.g. new controller, old switches…)

LoxiGen

- LoxiGen is a tool that generates OpenFlow protocol libraries for a number of languages
- Frontend parses wire protocol descriptions (Currently, for versions 1.0, 1.1, 1.2, 1.3.1)
- Backend for each supported language (currently C, Python, and Java, with an auto-generated wireshark dissector in Lua on the way)
- Results with code for floodlight controller libraries, indigo switch libraries
- Written in python, open-source

LoxiGen

```
package ${msg.package};

//:: include("_imports.java", msg=msg)

class ${impl_class} implements ${msg.interface.inherited_declaration()} {
//:: if genopts.instrument:
    private static final Logger logger = LoggerFactory.getLogger(${impl_class}.class);
//:: #endif
    // version: ${version}
    final static byte WIRE_VERSION = ${version.int_version};
//:: if msg.is_fixed_length:
    final static int LENGTH = ${msg.length};
//:: else:
    final static int MINIMUM_LENGTH = ${msg.min_length};
//:: #endif

//:: for prop in msg.data_members:
    //:: if prop.java_type.public_type != msg.interface.member_by_name(prop.name).java_type.public_type:
    //::    raise Exception("Interface and Class types do not match up: C: {} <-> I: {}".format(prop.java_type.public_type, msg.int
    //:: #endif
    //:: if prop.default_value:
        private final static ${prop.java_type.public_type} ${prop.default_name} = ${prop.default_value};
    //:: #endif
//:: #end

    // OF message fields
//:: for prop in msg.data_members:
    private final ${prop.java_type.public_type} ${prop.name};
//:: #endfor
//
//:: if all(prop.default_value for prop in msg.data_members):
    // Immutable default instance
    final static ${impl_class} DEFAULT = new ${impl_class}(
        ${", ".join(prop.default_name for prop in msg.data_members)}
    );
//:: #endif

    //:: if msg.data_members:
    // package private constructor - used by readers, builders, and factory
    ${impl_class}(${
        ", ".join("%s %s" %(prop.java_type.public_type, prop.name) for prop in msg.data_members) }) {
//:: for prop in msg.data_members:
        this.${prop.name} = ${prop.name};
//:: #endfor
    }
```

Source: github.com/floodlight/loxigen

Applications ("Northbound") API

- Currently –
  Thin API, mainly exposes OpenFlow protocol directly and event handler registration for OpenFlow events
- Future –
  Rich API with:
  - Sophisticated flow table management and caching
  - Virtualization and encapsulation of underlying network
  - More… (on next slides)

Multiple Applications

- Simple example:
  - 2 applications
  - First application sets:
    (IP_DST = 192.168.1.* -> forward to port 3)
  - Second application sets:
    (TCP_DST = 80 -> forward to port 4)
- What will happen with a TCP packet to IP 192.168.1.1 port 80?
- Is expansion of all possible combinations a valid solution?
  - Add higher priority rule:
    (IP_DST=192.168.1.*, TCP_DST=80 -> forward to ports 3,4)
  - Exponential growth in number of rules
- What if rules contradict?
  - Third application: (TCP_DST=80 -> drop)

Fault Tolerance

- Application fault:
  - Wrong logic
  - Malicious logic
  - Misconfiguration (e.g. creating loops)
- Controller fault
- Switch fault
  - If switch went down or rebooted and "forgot" its flow table – who is responsible?

- No good answers as of today…

Caching

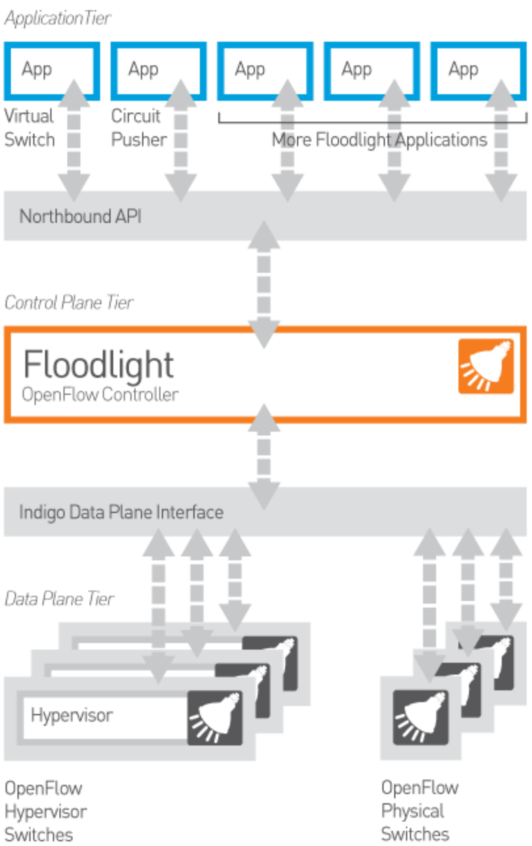
(can be viewed as part of "fault tolerance")

- Prevent redundant flow_mod messages from applications to the switches
- Allow recovery for applications and switches
- Cache results of queries to the switches

- Relates also to high availability issues, replication, etc.

Transactional Models

- Allow rollback of previous operations of the same transaction in case of failure
  - Controller-Switch channel
  - Application-Controller-Switch path



Source: projectfloodlight.org

Interesting SDN Architectures and Applications

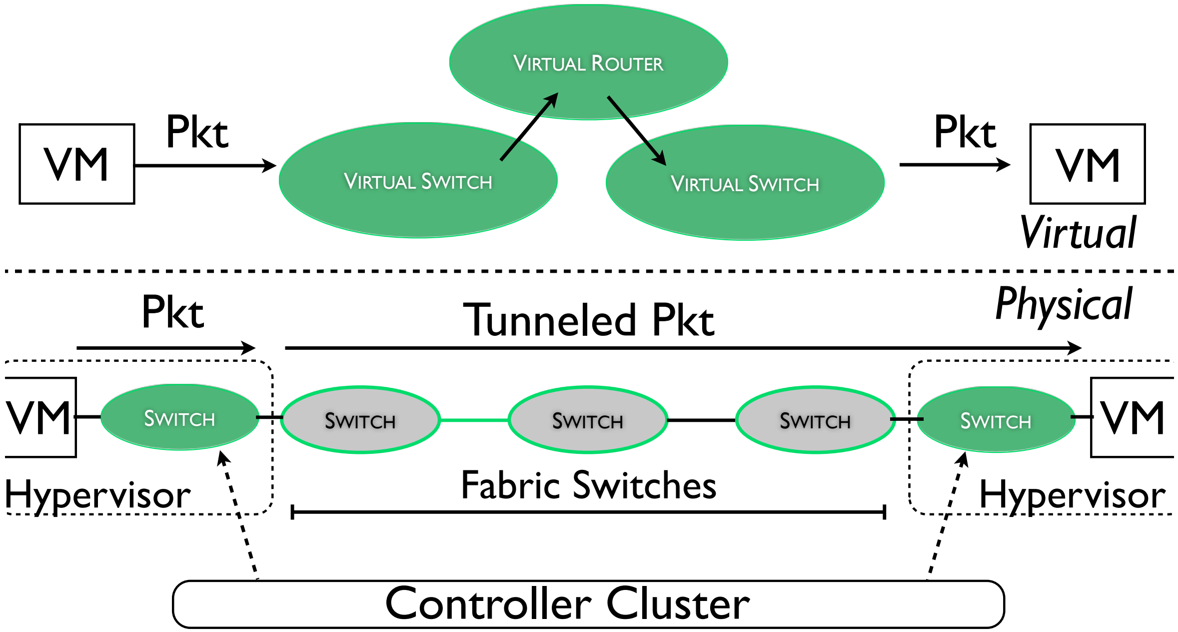
What's going on out there?

Overlay Networks

- Aim: inside a data center, have the flexibility of SDN for hosted VMs
  - Easily create tunnels
  - Control endpoint routing
  - Services: NAT, filtering, ACL, etc.
- Problem: hypervisor machines are connected on a non-SDN network
  - Would not like to replace the network equipment of the whole data center
  - Might not fully trust the new SDN technology
- Solution: virtualize the network as well!

Overlay Networks

- Overlay SDN:
  - Put a virtual (software) switch as the gateway of each hypervisor
  - Central control manages all virtual switches
  - Virtual switches are connected through the legacy fabric

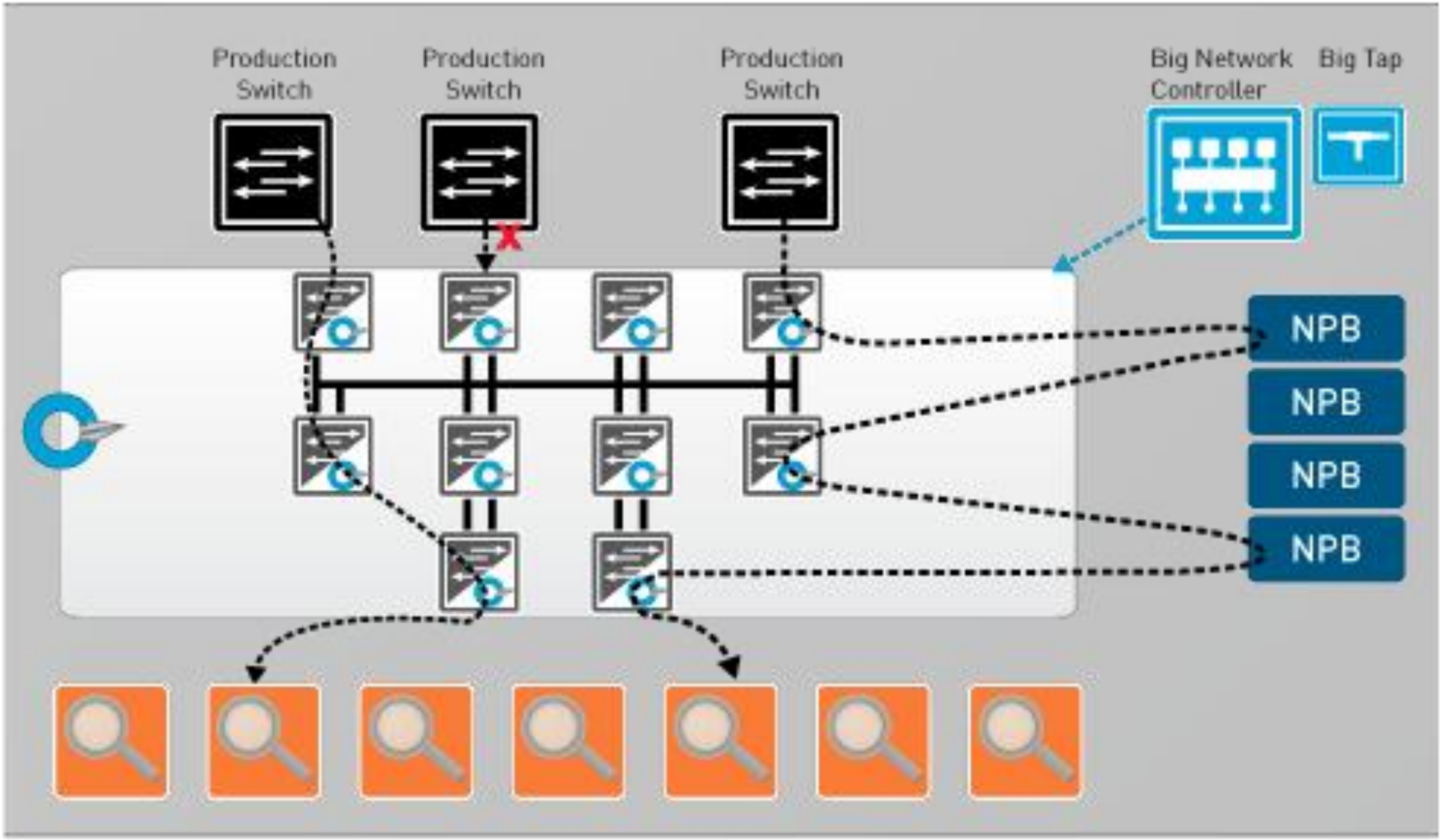


From Teemu Koponen (Nicira/VMWare)

Monitoring Networks

- Monitoring is a big deal for network operators
- So far: tapped selected points in network and sent data to adjacent monitoring devices
  - Requires lots of monitoring devices
  - Each tapping and monitoring point is managed separately
  - Multiple moderators must cooperate in order to use the same equipment together

Monitoring Networks



Source: bigswitch.com

Big Switch Networks – Big Tap