OpenFlow 1.3: Protocol, Use Cases, And Building a Fault Tolerant Application

Geddings Barrineau
Ryan Izard
Clemson University
Niky Riga
GENI Project Office
Game Plan

• Pre-work for hands on demo
• OpenFlow 1.3
  – Motivation (vs. 1.0)
  – Popular features
  – Community support
  – Other details
• GENI Cinema OpenFlow 1.3 use case
• Hands-on with OpenFlow 1.3 groups
Up Next

• Pre-work for hands on demo
• OpenFlow 1.3
  – Motivation (vs. 1.0)
  – Popular features
  – Community support
  – Other details
• GENI Cinema OpenFlow 1.3 use case
• Hands-on with OpenFlow 1.3 groups
Before We Get Started...

1. Login to GENI @ http://portal.geni.net and select the GRWBU2016 project (which you should be a member of already)
2. Create a new slice using the GENI RSPEC file located @ http://tinyurl.com/fast-failover-rspec
3. Bind your resources to an InstaGENI rack
4. Reserve your resources
5. Go grab some lunch
6. Login to your resources
Up Next

• Pre-work for hands on demo
• **OpenFlow 1.3**
  – Motivation (vs. 1.0)
  – Popular features
  – Community support
  – Other details
• GENI Cinema OpenFlow 1.3 use case
• Hands-on with OpenFlow 1.3 groups
Evolution of the OpenFlow Protocol

• OpenFlow 1.0
  + What you know and love!

• OpenFlow 1.1
  + Multiple tables and group tables
  + Some more matches and actions

• OpenFlow 1.2
  + The OpenFlow Extensible Match (OXM)

• OpenFlow 1.3
  + Meters
  + Table features
Evolution of the OpenFlow Protocol

- OpenFlow 1.4
  + Bundles
  + Flow table synchronization
  + Flow monitoring
- OpenFlow 1.5
  + More fine-grained matches and actions
  + Egress tables
  + Packet type aware pipeline & pipeline registers
  + Group/meter table improvements
- ...But we struggle to keep up...
Why OpenFlow 1.3?

• OF 1.0 primary complaint = too rigid
• OF 1.3 gains
  ✓ Greater match and action support
  ✓ Instructions add flexibility and capability
  ✓ Groups facilitate advanced actions
  ✓ Meters provide advanced counters
  ✓ Per-table features
  ✓ Custom table-miss behavior
  ✓ ...and more!
Note...

OpenFlow 1.1 and 1.2 introduced some of the features we will discuss.

However, due to the relative lack in adoption of OpenFlow 1.1 and 1.2, we will consider such features as OpenFlow 1.3 features.
Up Next

- Pre-work for hands on demo
- **OpenFlow 1.3**
  - Motivation (vs. 1.0)
  - **Popular features**
  - Community support
  - Other details
- GENI Cinema OpenFlow 1.3 use case
- Hands-on with OpenFlow 1.3 groups
OpenFlow 1.3 Matches

- OXM = OpenFlow Extensible Match
  - *Variable-length* list of matches, in *any order*
  - Contrast to rigid match structure of OF1.1-
Reason for the OXM

OpenFlow 1.0  OpenFlow 1.1  OpenFlow 1.2+

http://flowgrammable.org/sdn/openflow/message-layer/
OpenFlow 1.3 Matches

- Increased match support w/OXM
  - Ingress port
  - Ethernet
  - VLAN
  - IPv4
  - TCP
  - UDP
  - ARP
  - MPLS
  - PBB
  - ICMPv4
  - ICMPv6
  - IPv6
  - Tunnel
  - SCTP
  - Metadata
  - Custom/Experimenter
OpenFlow 1.3 Actions

- Set field
  - Any OXM
- Push/Pop
  - VLAN
  - MPLS
  - PBB
- Set queue
- Goto group
- Output
- TTL
  - Set
  - Decrement
- Custom/Experimentator
OpenFlow 1.3 Instructions

• Apply actions
  – List of actions to perform immediately

• Write actions
  List of actions to perform later

• Clear actions
  – Clear list of accumulated “write actions”

• Meter
  – Send to an installed meter

• Goto table
  – Send to another table in the switch

• Write metadata
  – Store some “data” associated with the packet as it traverses table(s)
OpenFlow 1.3 Meters

- Monitor and rate-limit packets
- Multiple meter “bands” define different rate thresholds

```c
if (rate > t1) do_this;
else if (rate > t2) do_that;
else if (rate > t3) drop_it;
else do_nothing;
```
OpenFlow 1.3 Groups

- Allow more complex actions
- Bucket = (list of actions) + (optional params)
- Actions can be unique per bucket
ALL Group

- Copy packet to *all buckets*
- No special parameters
- Use case: GENI Cinema uses ALL groups

```
Group
<table>
<thead>
<tr>
<th>ID</th>
<th>Type=ALL</th>
<th>Counters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bucket</td>
<td>Actions</td>
<td></td>
</tr>
<tr>
<td>Bucket</td>
<td>Actions</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bucket</td>
<td>Actions</td>
<td></td>
</tr>
</tbody>
</table>
```
SELECT Group

- Distribute packet to a single bucket
- Weight assigned to each bucket
- Distribution algorithm is switch-dependent
  - Weighted round-robin?
INDIRECT Group

- All packets go to a single bucket
- No special parameters
- Consolidate common actions amongst a set of flows
  - Reduce complexity
  - Reduce memory footprint
FAST-FAILOVER Group

- Use a single bucket for all packets
- Watch port/group as parameters
  - Determines “liveness” of bucket
  - If port/group is up, bucket can be used
  - If port/group is down, bucket cannot be used
Up Next

- Pre-work for hands on demo
- **OpenFlow 1.3**
  - Motivation (vs. 1.0)
  - Popular features
  - **Community support**
  - Other details
- GENI Cinema OpenFlow 1.3 use case
- Hands-on with OpenFlow 1.3 groups
Community Support

• Great software switch support
  – OVS supports everything* except meters
    • Present protocol support for meters
    • Table features supported in 2.4.0
    • Groups fully supported in 2.3.1
  – ofsoftswitch supports meters but does not support all other OpenFlow 1.3 features
• Hit-and-miss support with HW vendors
  – Some vendors technically do, but can be buggy
• Wide controller support

*to my knowledge
Floodlight Controller

- Java-based OpenFlow controller
  - OpenFlow 1.0 and 1.3 support
  - Experimental OpenFlow 1.1, 1.2, 1.4
  - Lots of included modules

- Home
  - [http://projectfloodlight.net](http://projectfloodlight.net)

- Documentation and tutorials
  - [http://floodlight.atlassian.net/wiki](http://floodlight.atlassian.net/wiki)

- Large, active community
  - floodlight-dev@openflowhub.org
  - Google Group
Up Next

- Pre-work for hands on demo
- **OpenFlow 1.3**
  - Motivation (vs. 1.0)
  - Popular features
  - Community support
  - Other details
- GENI Cinema OpenFlow 1.3 use case
- Hands-on with OpenFlow 1.3 groups
Multi-Version OpenFlow Handshake

- **Handshake**
  - Message-exchanging process to establish an OpenFlow channel between a controller and a switch
  - Need to negotiate common OpenFlow version

- **Algorithm**
  - Switch says “Hello version\_X” with OF version X
  - Controller says “Hello version\_Y” with OF version Y
  - Switch and controller each pick lower version of X and Y
    - (theirs < mine) ? theirs : mine; e.g. (X < Y) ? X : Y;

- **Caveat…**
  Algorithm requires support for each OF version up to and including the “Hello” version advertised
  Not the case in implementation/practice

- **Fix** for (controller >= OF1.3) && (switch >= OF1.3)
  Hello advertises highest version + version bitmap for negotiation
OpenFlow Auxiliary Connections

- Multiple control connections per switch
  - Parallelize some operations
  - Negotiated on a per-switch basis
  - Aux ID 0 = main; Aux ID > 0 = other

- Controller chooses which connection to use
  - Main
  - Aux 1
  - Aux 2
  - ...etc.
  - ID=0 (main)
  - ID=1
  - ID=2
OpenFlow 1.3 Controller Roles

• OpenFlow 1.3 integrates roles in protocol
  – Role = controller read/write permissions for each switch
  – MASTER + SLAVE
    • Exactly one master controller per switch
    • Zero or more slaves per switch
    • Only the master controller can write
    • All (other) slave controllers can read
  – EQUAL
    • All controllers can read and write
    • Likely requires synchronization between controllers (e.g. HA)

• But, doesn’t Nicira has role extension for OF 1.0?
  – Same idea for MASTER and SLAVE
  – Nicira’s OTHER role = OpenFlow 1.3’s EQUAL role
OpenFlow Multipart Messages

- Steady-state controller-to-switch “queries”
- Efficiently process large requests
- Flow stats, port stats, group stats, meter stats, table features...
- Request and reply pairs with same XID
- OFPMPF_REQ_MORE flag for more messages
Table Miss Behavior

What to do if a packet matches no flows?

• Previously, a property of the flow table
  – Typically, send to the controller

• In OpenFlow 1.3, defined by a flow
  – Zero-priority and fully-wildcarded match
  – User-defined actions and instructions
  – Can send to controller/drop (most common)
  – Or, can do what YOU want
Table Features

• Many OpenFlow features are optional, not required

• **Problem:** How to determine switch features?

• **Solution:** Table Features specify capabilities of each table
  – Matches, actions, instructions, etc.

• Do table features indicate match co-dependencies or hardware vs. software support?
  NO.
Up Next

• Pre-work for hands on demo
• OpenFlow 1.3
  – Motivation (vs. 1.0)
  – Popular features
  – Community support
  – Other details
• GENI Cinema OpenFlow 1.3 use case
• Hands-on with OpenFlow 1.3 groups
What is GENI Cinema?

- Live video streaming service
  - Operates entirely over GENI
  - OpenFlow 1.0 and 1.3 based solution
- Offers channel changing abilities
- Scalable and efficient
What’s special about GENI Cinema?

- Software defined networking based
  - Heavy use of OpenFlow 1.3 ALL groups
- “Fast” channel changing
  - GENI Cinema = ~2s average
  - YouTube live = ~5s average
  - Digital cable TV = ~3s average
  - Satellite TV = ...
- Event *perspective* in hands of user
  - Multiple cameras available to select
GENI Cinema ALL Groups

Open vSwitch Client-to-Channel Mapper

Default Flow Table
Channel A: goto Group 1
Channel B: goto Group 2
Channel C: goto Group 3
Channel D: goto Group 4
Channel E: goto Group 5
... Channel n: goto Group n

Group 1: Client a, Client b, Client c, Client d, Client e, Client f
Group 2: Client g, Client h, Client i
Group 3: [No Clients in Bucket List]
Group 4: Client j, Client k, Client l, Client m, Client n, Client o, Client p, Client q
Group n: Client r, Client s, Client t, Client u, Client v, Client w, Client x, Client y, Client z, Client aa, Client bb, ...

UDP Packets In

UDP Packets Out
Quick Demonstration

https://cinema.smartfuture2015.net
Up Next

- Pre-work for hands on demo
- OpenFlow 1.3
  - Motivation (vs. 1.0)
  - Popular features
  - Community support
  - Other details
- GENI Cinema OpenFlow 1.3 use case
- Hands-on with OpenFlow 1.3 groups
Your Turn!

• Problem
  – A single path from point A to point B leaves a single point of failure in any topology. Upon link failure, end-to-end communication is impossible.

• Solution
  – Use redundant links to allow end-to-end connections to persist in the event of a link failure.
Tutorial Topology
Hands-on Tutorial

https://floodlight.atlassian.net/wiki/display/floodlightcontroller/How+to+Work+with+Fast-Failover+OpenFlow+Groups

Or...

http://tinyurl.com/fast-failover-demo