LOOM
An Open Flow Controller
Marc Sugiyama, Erlang Solutions
Problem

Build an extensible, robust OpenFlow Controller for thousands of switches
Why?

• Explosion in the number of IP devices

• Virtual Hosts, Mobile

• Internet of things: Sensors, Body Monitors, Cars, Lightbulbs

• “The network is the computer” - John Gage
OpenFlow

Separates the Controller...

... from the Switching hardware

Together they do what traditional ethernet switches can do.
What is LOOM?

- Large scale OpenFlow controller
- Global view, deliberate, network aware applications
- Not a traditional IP switch controller
- Open Source
  
  https://github.com/FlowForwarding/loom
Where’s the code?

- http://github.com/FlowForwarding
- Libraries: of_protocol, of_msg_lib, of_driver, ofs_handler, enetconf, of_config, ofs_config
- Examples: icontrol, tapestry, stats_poller
## Getting Started

### Custom Controller

<table>
<thead>
<tr>
<th>Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>ofs_handler</td>
</tr>
<tr>
<td>of_driver</td>
</tr>
<tr>
<td>of_msg_lib</td>
</tr>
<tr>
<td>of_protocol</td>
</tr>
</tbody>
</table>

### Custom Config Point

<table>
<thead>
<tr>
<th>Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>ofs_config</td>
</tr>
<tr>
<td>enetconf</td>
</tr>
<tr>
<td>of_config</td>
</tr>
</tbody>
</table>
Asynchronous Send

Protocol Flow

www.websequencediagrams.com
Synchronizing Send

Protocol Flow

Application ➔ of_driver ➔ Switch

flow_mod(XID1) ➔ flow_mod(XID1) ➔ flow_mod(XID2) ➔ flow_mod(XID2) ➔ flow_mod(XID3) ➔ flow_mod(XID3) ➔ barrier request ➔ error(XID2) ➔ barrier reply ➔ error(XID2)

www.websequencediagrams.com
Asynchronous Messages

Protocol Flow
Callbacks
Sending Request

- Create message record with of_msg_lib
- ofs_handler:sync_send
- Process replies, if any
Subscriptions

- Handling async messages
- `ofs_handler:subscribe(DPID, Module, What)`
- `atom, {atom, fun/1}`
Example: adding flow

Matches = [{in_port, <<1:32>>}],
Instructions = [{apply_actions,
    [{output, 2, no_buffer}]}],
Opts = [{table_id, 0},{priority,100},
    {idle_timeout,0},
    {cookie,<<0,0,0,0,0,0,0,0,0,10>>}],
    {cookie_mask,<<0,0,0,0,0,0,0,0>>}],
Request = of_msg_lib:flow_add(4,
    Matches, Instructions, Opts),
...
Example: adding flow

\[
\text{Reply} = \text{ofs}\_\text{handler:sync}\_\text{send}(\text{DPID}, \text{Request}).
\]

Success:
\[
\{\text{ok, noreply}\}
\]

Example error:
\[
\{\text{ok, \{ofp_message, 4, error, 0, ofp_error_msg, bad_action, bad_out_port, <<>>>\}}\}
\]
packet_in

- ofs_handler:subscribe(DPID, Module, packet_in)
- Module:handle_message({packet_in, Xid, Body}, State)
Danger!

- Calls from callbacks can cause deadlocks
Current Status

- PlugFest Indianapolis May 2014
- Successfully tested with switches from five vendors, Ixia, and Spirent
- Managed 2000 simulated switches on Ixia - flows between 1000 switches
- Demonstrated stats_poller in large topology test
Future

• Fault tolerance
• Switch level data store
• Graph store
• Network Executive
• Flow compiler
Learn More

• http://www.opennetworking.org
• http://www.flowforwarding.org
• http://github.com/FlowForwarding
Contribute!

- Learning switch
- Topology discovery (LLDP)
- Expand icontrol, more stats poller integration
- OpenFlow Config
- Documentation
Questions?

• http://github.com/FlowForwarding

• Marc Sugiyama
  marc.sugiyama@erlang-solutions.com