



Troubleshooting ODL using Log analyser

Prem Sankar, Principal Engineer, Ericsson

Deepthi VV, Software Engineer, Ericsson

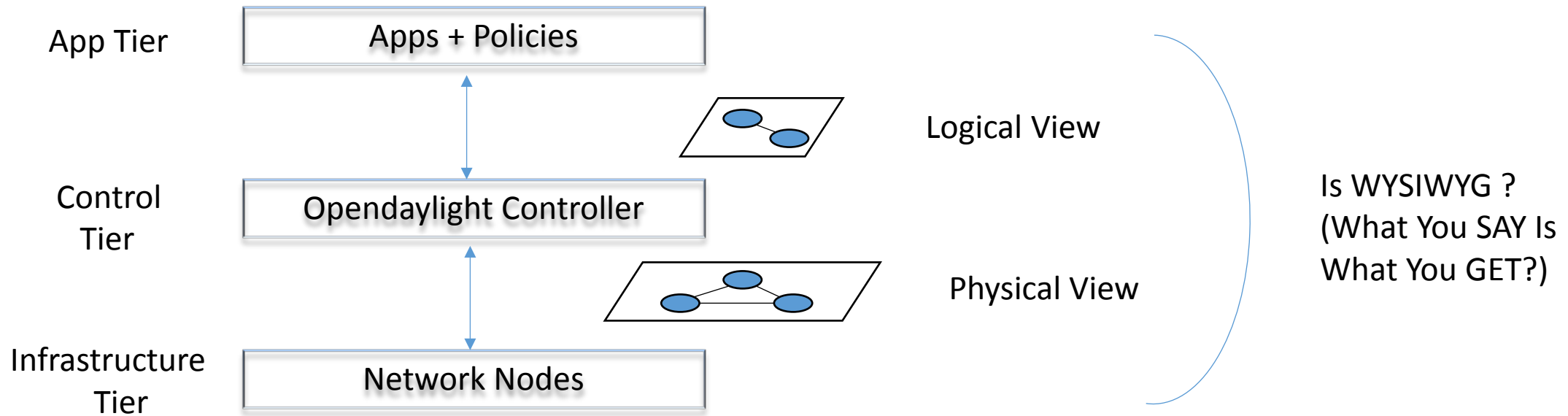
Agenda

- Troubleshooting challenges
- Troubleshooting using log analyzer
- ELK overview
- Installation and configuration
- Demo



#ODSummit

SDN Deployment



ODL Troubleshooting Challenges

- Challenges
 - Competence areas
 - Varied nature of projects and competence
 - Correlation of events at various stages to debug a problem
 - (NSF -> MDSAL -> PLUGIN)
 - Module state detection/Race condition
 - Bundle dependency and related issues
 - Log messages issues
 - Sheer volume of log messages



SDN Troubleshooting Challenges

- Complexity at each stage
 - App level
 - Scope limited to analysis of NBI Calls
 - Control tier level
 - Scope limited to analysis of inter module messages and southbound plugin messages
 - Infrastructure level
 - Device specific logs
- Correlation of logs
 - Complex due to different level of abstraction

Log analyzer (Warning : Trivial facts)

- Benefits- Log contains extensive volume of useful information as listed below that can be mined to arrive at conclusion
 - Module startup
 - State transition information
 - Exceptions and errors ...
- Conventional method – Tail, Awk, sed
 - Time consuming
 - Correlation is difficult
- Automatic log analyzer – Splunk, ELK
 - Powerful and has variety of plugins for transformation and visualization
 - Proactive fault detection

#ODSummit



Troubleshooting using log analysis

- Multiphased approach
 - Tiered log analysis
 - Use ELK to do the initial level of log analysis
 - Correlation
 - Define rules for correlation
 - Machine learning
 - Using WEKA for predictive issue reporting

Steps

Transform
(Logstash)

Persist
(Elasticsearch)

Analysis
(Kibana)



#ODSummit

ELK

- Elastic Search
 - Store the data that LogStash processed and provide full-text index
- Logstash
 - Collecting and parsing log files. Transform unstructured log into meaningful and searchable
- Kibana
 - Web console for user to interact with Elasticsearch

ELK components

- Logstash
 - Input
 - File, syslog, **log4j**, redis, websocket,
 - Filter
 - Clone, drop, grok, mutate, geoip
 - Output
 - **Elasticsearch**, graphine, statsd, file
- Elasticsearch
 - Mapping
 - SearchAPI
 - Aggregations
 - Bucketizes based on various category
- Kibana
 - Discover
 - Visualize

#ODSummit



Prerequisite for ODL

Edit etc/org.ops4j.pax.logging.cfg

Change rootlogger line to "log4j.rootLogger=INFO, out, ELKTransform, osgi:*"

log4j.appender.ELKTranform=org.apache.log4j.net.SocketAppender

log4j.appender.ELKTransform.port=9500

log4j.appender.ELKTransform.remoteHost=127.0.0.1



Logstash

Installation

- `sudo wget http://download.elastic.co/logstash/logstash/packages/debian/logstash_1.5.3-1_all.deb`
- `sudo dpkg -i logstash_1.5.3-1_all.deb`
- `sudo update-rc.d logstash defaults 95 10`



Logstash

- Edit `/etc/logstash/conf.d/karaf.conf`

```
input {
  log4j {
    mode => "server"
    port => 9500
  }
}
output {
  elasticsearch {
    host=> "localhost"
  }
}
```

- `logstash -f /etc/logstash/conf.d/karaf.conf`

#ODSummit



Elasticsearch

- `sudo wget https://download.elastic.co/elasticsearch/elasticsearch/elasticsearch-1.7.0.deb`
- `sudo dpkg -i elasticsearch-1.7.0.deb`
- `sudo update-rc.d elasticsearch defaults 95 10`
- Add `network.host:localhost` to `/etc/elasticsearch/elasticsearch.yml`
- `sudo /etc/init.d/elasticsearch restart`

Kibana

- `sudo wget https://download.elasticsearch.org/kibana/kibana/kibana-4.1.1-linux-x64.tar.gz`
- `sudo tar xvfz kibana-4.1.1-linux-x64.tar.gz`
- `sudo ln -s kibana-4.1.1-linux-x64 kibana`
- Edit `kibana.yml`
 - Port:5601
 - Host:"localhost"



DEMO



#ODSummit

ELK + ODL

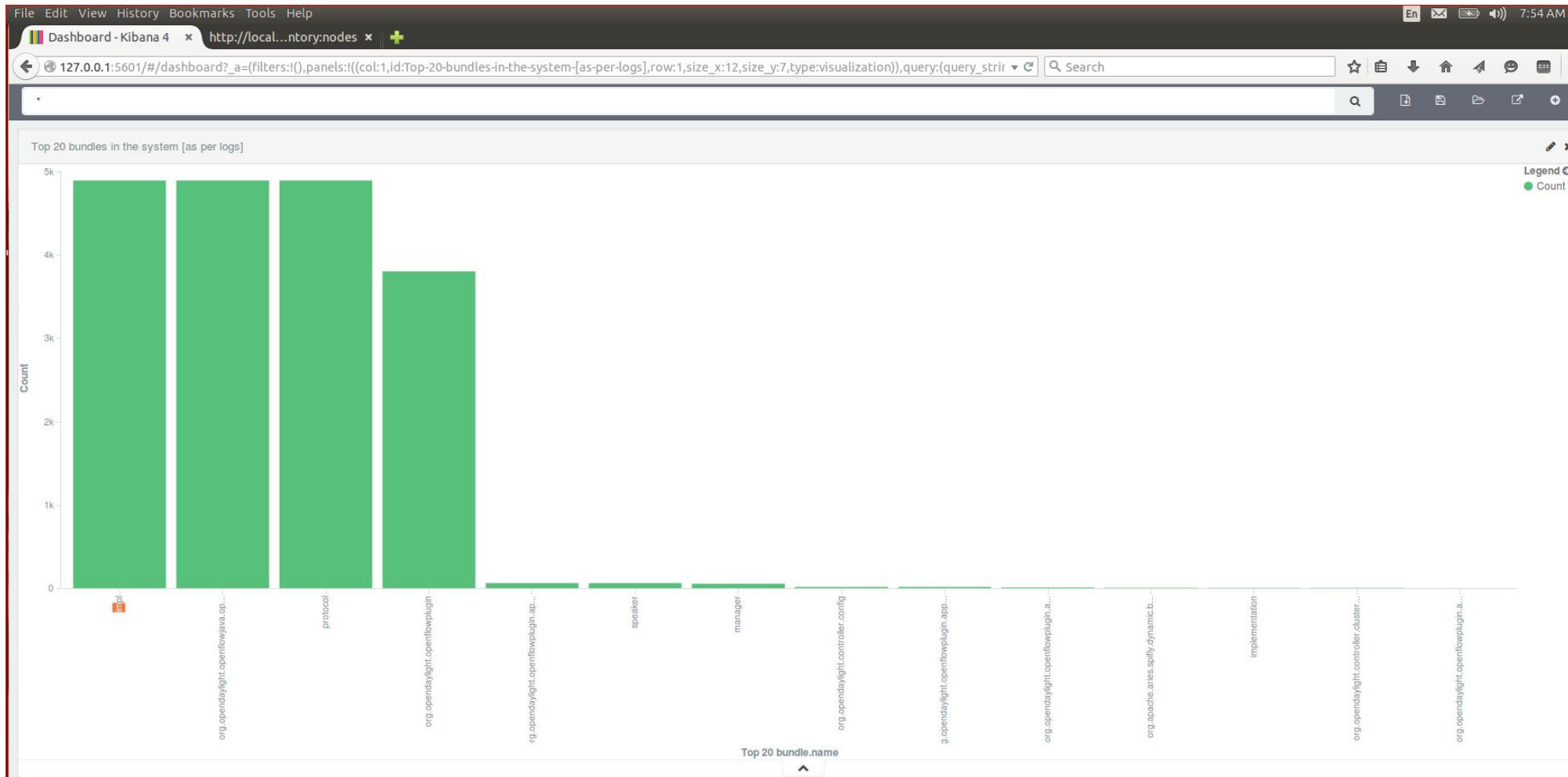
The screenshot shows the Kibana 4 interface for a logstash-* dashboard. The top navigation bar includes 'Discover', 'Visualize', 'Dashboard', and 'Settings'. The main area displays a bar chart for the time range 'July 26th 2015, 07:56:55.000 - July 26th 2015, 07:57:00.000' with 2,872 hits. The chart shows a distribution of counts over time. Below the chart is a table of log entries with columns for Time, host, bundle.name, priority, logger_name, thread, and message.

Time	host	bundle.name	priority	logger_name	thread	message
July 26th 2015, 07:56:58.328	127.0.0.1:36263	org.opendaylight.openflowplugin	TRACE	org.opendaylight.openflowplugin.openflow.m.d.queue.TicketProcessorFactoryImpl	OFmsgProcessor-1	translator: org.opendaylight.openflowplugin.openflow.m.d.core.translator.MultipartReplyTableFeaturesToTableUpdatedTranslator@1b34843 elapsed time 2951 ns
July 26th 2015, 07:56:58.328	127.0.0.1:36263	org.opendaylight.openflowplugin	DEBUG	org.opendaylight.openflowplugin.openflow.m.d.queue.TicketProcessorFactoryImpl	OFmsgProcessor-1	message processing done (type: MultipartReplyMessage, ticket: 2101626228)
July 26th 2015, 07:56:58.327	127.0.0.1:36263	org.opendaylight.openflowplugin	TRACE	org.opendaylight.openflowplugin.openflow.m.d.queue.TicketProcessorFactoryImpl	OFmsgProcessor-1	translator: org.opendaylight.openflowplugin.openflow.m.d.core.translator.MultipartReplyTranslator@6bd8fd92 elapsed time 168450 ns
July 26th 2015, 07:56:58.327	127.0.0.1:36263	org.opendaylight.openflowplugin	DEBUG	org.opendaylight.openflowplugin.openflow.m.d.core.translator.MultipartReplyTranslator	OFmsgProcessor-1	Converted aggregate flow statistics : AggregateFlowStatisticsUpdate [_byteCount=Counter64 [_value=0], _flowCount=Counter32 [_value=0], _id=Uri [_value=openflow:2], _packetCount=Counter64 [_value=0], _transactionId=TransactionId [_value=72], _moreReplies=false, augmentation=[]]
July 26th 2015, 07:56:58.326	127.0.0.1:36263	org.opendaylight.openflowplugin	DEBUG	org.opendaylight.openflowplugin.openflow.m.d.core.translator.MultipartReplyTranslator	OFmsgProcessor-1	Received aggregate flow statistics response from openflow 1.3+ switch



#ODSummit

ELK + ODL



#ODSummit



Thank You



#ODSummit