ONOS基础教程——QuickStark with a VM

本文档由http://sdnhub.cn译自：

> https://wiki.onosproject.org/display/ONOS/Basic+ONOS+Tutorial

在本教程中，你将完成专门设计的用于解释ONOS的主要概念的一系列练习，理解如何使用ONOS的基本特性。

1 介绍

1.1 前提条件

1. 计算机：2G内存和5G自由空间，快速的处理器可以加速虚拟机的运行性能。
2. 能运行VirtualBox的操作系统。
3. 安装VirtualBox，需要管理员权限。
4. SDN，OpenFlow和Mininet的基本概念。请事先完成OpenFlow tutorial和Mininet workthrough
5. 并非必须，但是强烈建议完成FlowVisor tutorial。熟悉Apache Karaf将会很有帮助。

1.2 卡住了?返现了bug? 有问题?

Email:onos-discuss@googlegroups.com
高效的提交bug报告：https://wiki.onosproject.org/display/ONOS/ONOS+Mailing+Lists

2 环境设置

2.1 安装需要的软件

下载并安装VirtualBox最新版，下载Tutorial VM：http://downloads.onosproject.org/vm/onos-tutorial-1.0.0r161-ovf.zip

2.2 创建虚拟机

解压缩下载的Tutorial VM，双击OVF文件将自动启动VitualBox和导入虚拟机对话框。导入虚拟机，即可。

虚拟机内的Guest OS帐号和密码如下：
USERNAME:tutorial1
PASSWORD:tutorial1

2.3 重要的命令提示符

ONOS的命令提示符：

```
onos>
```

Mininet的命令提示符：

```
mininet>
```

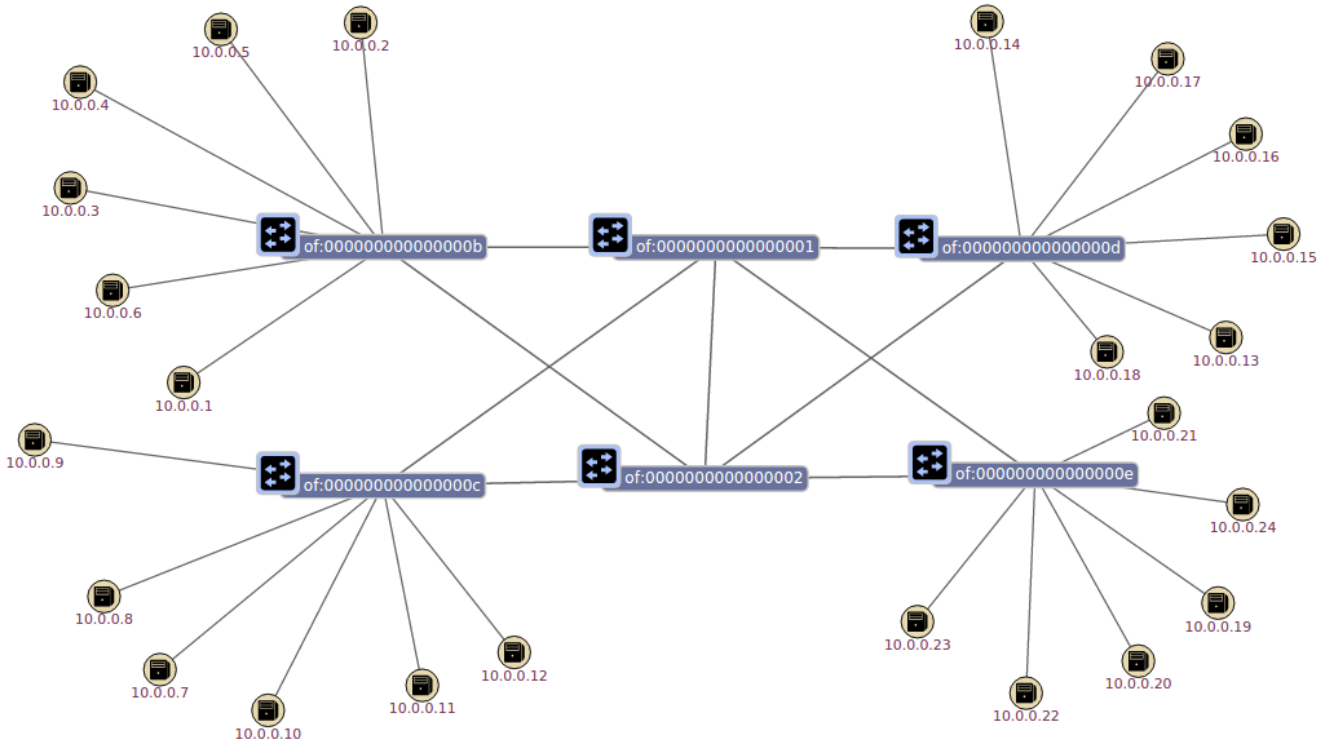
2.4 重置

桌面上的Reset快捷方式用户快速重置Tutorial所做的所有改动为初始状态。

2.5 启动Mininet

本教程所有的联系都是使用相同的网络拓扑(如下图所示)。网络中有6台交换机，4台外边的交换机上连接了6台主机，这4台交换机又连接到中间互连的2台交换机上。

启动Mininet（双击桌面的Mininet图标）即可启动此网络拓扑。



3 激活数据包转发（Reactive Forwarding）

双击桌面上的ONOS图标，打开一个ONOS命令行提示符。

3.1 No ping? Why?

上面的网络拓扑中主机之间是不能ping通的。在mininet提示符中输入下面的命令：

```
mininet>h11 ping -c3 h41
```

得到如下的结果：

```
mininet> h11 ping -c3 h41
PING 10.0.0.19 (10.0.0.19) 56(84) bytes of data.
From 10.0.0.1 icmp_seq=1 Destination Host Unreachable
From 10.0.0.1 icmp_seq=2 Destination Host Unreachable
From 10.0.0.1 icmp_seq=3 Destination Host Unreachable
--- 10.0.0.19 ping statistics ---
3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 2009ms
```

"Reactive Forwarding"是一个非常简单的SDN应用程序，它为每一个被送到控制器的数据包安装流表。但是此应用默认没有被加载。使用如下命令查看已经被加载的应用程序列表：

```
onos> list
110 | Active |  80 | 1.0.0.SNAPSHOT   | onos-rest
116 | Active |  80 | 1.0.0.SNAPSHOT   | onos-of-api
117 | Active |  80 | 1.0.0.SNAPSHOT   | onos-of-ctl
118 | Active |  80 | 1.0.0.SNAPSHOT   | onos-lldp-provider
119 | Active |  80 | 1.0.0.SNAPSHOT   | onos-host-provider
120 | Active |  80 | 1.0.0.SNAPSHOT   | onos-of-provider-device
121 | Active |  80 | 1.0.0.SNAPSHOT   | onos-of-provider-packet
122 | Active |  80 | 1.0.0.SNAPSHOT   | onos-of-provider-flow
136 | Active |  80 | 1.0.0.SNAPSHOT   | onos-cli
137 | Active |  80 | 8.1.15.v20140411 | Jetty :: Websocket
138 | Active |  80 | 1.0.0.SNAPSHOT   | onos-gui
151 | Active |  80 | 1.0.0.SNAPSHOT   | onos-core-net
```

```
152 | Active |  80 | 1.0.0.SNAPSHOT   | onos-core-trivial
```

3.2 加载Reactive Forwarding应用

在ONOS提示符中输入如下命令（可以使用tab键提示输入内容）：

```
ONOS>feature:install onos-app-fwd
```

在Mininet提示符中再次输入ping命令，即可ping通：

```
mininet> h11 ping h41
PING 10.0.0.19 (10.0.0.19) 56(84) bytes of data.
64 bytes from 10.0.0.19: icmp_req=1 ttl=64 time=9.12 ms
```

3.3 启动和停止

在ONOS命令行提示符中输入如下的命令停止和启动应用：

```
onos> stop onos-app-fwd
```

停止应用程序后，不能ping通。

```
onos> start onos-app-fwd
```

启动应用程序后，能ping通。

4 ONOS CLI命令

ONOS有很多命令，本节学习最常用命令。获得命令行帮助的方法如下：

```
onos>help onos
```

4.1 devices命令

如果没有设备去控制的话，SDN控制器就没什么用。列出控制其中现有的设备列表：

```
onos> devices
id=of:0000000000000001, available=true, role=MASTER, type=SWITCH, mfr=Nicira, Inc., hw=Open vSwitch, sw=2.1.3, serial=None, protoc
id=of:0000000000000002, available=true, role=MASTER, type=SWITCH, mfr=Nicira, Inc., hw=Open vSwitch, sw=2.1.3, serial=None, protoc
id=of:000000000000000b, available=true, role=MASTER, type=SWITCH, mfr=Nicira, Inc., hw=Open vSwitch, sw=2.1.3, serial=None, protoc
id=of:000000000000000c, available=true, role=MASTER, type=SWITCH, mfr=Nicira, Inc., hw=Open vSwitch, sw=2.1.3, serial=None, protoc
id=of:000000000000000d, available=true, role=MASTER, type=SWITCH, mfr=Nicira, Inc., hw=Open vSwitch, sw=2.1.3, serial=None, protoc
```

4.2 links命令

显示系统中设备间的连接信息：

```
onos> links
src=of:000000000000000e/1, dst=of:0000000000000001/5, type=DIRECT, state=ACTIVE
src=of:000000000000000d/1, dst=of:0000000000000001/4, type=DIRECT, state=ACTIVE
。。。。。。。。
src=of:0000000000000001/1, dst=of:0000000000000002/1, type=DIRECT, state=ACTIVE
```

4.3 hosts命令

列出网络中的主机列表：

```
onos> hosts
id=00:00:00:00:00:01/-1, mac=00:00:00:00:00:01, location=of:000000000000000b/3, vlan=-1, ip(s)=[10.0.0.1]
id=00:00:00:00:00:13/-1, mac=00:00:00:00:00:13, location=of:000000000000000e/
```

4.4 flows命令

流命令允许你查看系统中当前注册的流表项，流表项有如下几个状态：

- **PENDING_ADD** - 流被提交和转发给交换机。
- **ADDED** - 流已被添加到交换机。
- **PENDING_REMOVE** - 请求删除已被提交和转发给交换机的流。
- **REMOVED** - 流表规则已被删除

首先，在网络中开始发送一些流量：

```
mininet> h11 ping h41
```

接着在ONOS的提示符中输入流命令，以查看流表信息：

```
onos> flows
deviceId=of:0000000000000001, flowRuleCount=1
   id=30000b889cb32, state=ADDED, bytes=8722, packets=89, duration=89, priority=10, appId=org.onlab.onos.fwd
      selector=[ETH_TYPE{ethType=800}, ETH_SRC{mac=00:00:00:00:00:01}, ETH_DST{mac=00:00:00:00:00:13}, IN_PORT{port=2}]
      treatment=[OUTPUT{port=5}]
deviceId=of:0000000000000002, flowRuleCount=1
   id=30000b889cf4d, state=ADDED, bytes=8624, packets=88, duration=88, priority=10, appId=org.onlab.onos.fwd
      selector=[ETH_TYPE{ethType=800}, ETH_SRC{mac=00:00:00:00:00:13}, ETH_DST{mac=00:00:00:00:00:01}, IN_PORT{port=5}]
      treatment=[OUTPUT{port=2}]
deviceId=of:000000000000000b, flowRuleCount=2
   id=30000b88a8321, state=ADDED, bytes=8722, packets=89, duration=89, priority=10, appId=org.onlab.onos.fwd
      selector=[ETH_TYPE{ethType=800}, ETH_SRC{mac=00:00:00:00:00:13}, ETH_DST{mac=00:00:00:00:00:01}, IN_PORT{port=2}]
      treatment=[OUTPUT{port=3}]
   id=30000b88a833e, state=ADDED, bytes=8722, packets=89, duration=89, priority=10, appId=org.onlab.onos.fwd
      selector=[ETH_TYPE{ethType=800}, ETH_SRC{mac=00:00:00:00:00:01}, ETH_DST{mac=00:00:00:00:00:13}, IN_PORT{port=3}]
      treatment=[OUTPUT{port=1}]
deviceId=of:000000000000000c, flowRuleCount=0
deviceId=of:000000000000000d, flowRuleCount=0
deviceId=of:000000000000000e, flowRuleCount=2
   id=30000b88a8e45, state=ADDED, bytes=8722, packets=89, duration=89, priority=10, appId=org.onlab.onos.fwd
      selector=[ETH_TYPE{ethType=800}, ETH_SRC{mac=00:00:00:00:00:01}, ETH_DST{mac=00:00:00:00:00:13}, IN_PORT{port=1}]
      treatment=[OUTPUT{port=3}]
   id=30000b88a8e82, state=ADDED, bytes=8722, packets=89, duration=89, priority=10, appId=org.onlab.onos.fwd
      selector=[ETH_TYPE{ethType=800}, ETH_SRC{mac=00:00:00:00:00:13}, ETH_DST{mac=00:00:00:00:00:01}, IN_PORT{port=3}]
      treatment=[OUTPUT{port=2}]
```

从上面的输出中可以看出，ONOS提供了很多交换机上流表的详细信息。例如：每一条流表项定义一个“selector”和一个“treatment”，他们被流表项进行流量匹配，和匹配的流量如何被处理。每一个流表项被appId标记，指出此流表项被那个应用程序安装，这是非常有用的特性，因为它能帮助管理员区分那个应用程序行为不当或耗费较多的资源。

4.5 apps命令

列出当前在ONOS上运行的应用程序列表：

```
onos> apps
id=1, name=org.onlab.onos.net.intent
id=2, name=org.onlab.onos.fwd
```

如上所示，目前为止，本tutorial加载了2个应用程序。

4.6 paths命令

对于给定的网络拓扑，ONOS计算所有任意两个节点见的最短路径。
这一点对于那些获取路径信息用于流表安装的应用程序特别有用。当然也可用于其他用途。paths命令有两个参数（分别是两个设备）。可以使用ONOS的tab键的命令补全功能。

```
onos> paths <TAB>
of:0000000000000001   of:0000000000000002   of:000000000000000b
```

```
of:000000000000000c   of:000000000000000d   of:000000000000000e
```

ONOS会列出可用的设备列表，方便找到你需要的设备。下面是paths命令的一个输出结果示例：

```
onos> paths of:000000000000000b of:000000000000000e
of:000000000000000b/1-of:0000000000000001/2==>of:0000000000000001/5-of:000000000000000e/1; cost=2.0
of:000000000000000b/2-of:0000000000000002/2==>of:0000000000000002/5-of:000000000000000e
```

4.7 Intent命令

ONOS中的intent命令允许我们查看系统中的intents。“intents”存在如下几个状态：

- **SUBMITTED** - “intent”已经被提交，很快将被处理。
- **COMPILING** - “intent”正在被编译，这是一个瞬态（transient state）。
- **INSTALLING** - 安装的“intent”正在被处理。
- **INSTALLED** - 安装的“intent”安装成功。.
- **RECOMPILING** - "intent"遭遇失败后，正在被重新编译。
- **WITHDRAWING** - “intent”正在被撤销。
- **WITHDRAWN** - “intent”已经被删除。
- **FAILED** - 由于“intent”不能被满足，而处于失败状态。

更多关于Intents的详细，参考：https://wiki.onosproject.org/display/ONOS/The+Intent+Framework

```
onos> intents
id=0x0, state=INSTALLED, type=HostToHostIntent, appId=org.onlab.onos.gui
    constraints=[LinkTypeConstraint{inclusive=false, types=[OPTICAL]}]
id=0x1, state=WITHDRAWN, type=HostToHostIntent, appId=org.onlab.onos.cli
    constraints=[LinkTypeConstraint{inclusive=false, types=[OPTICAL]}]
```

使用上面的命令在添加“intent”之前，是看不到任何“intent”。 在本教程的下一节中，你将加载“intent reactive forwarding”应用程序，它可以在需要的时候自动添加intents。

下面的命令（intents -i）能输出intent被编译后的“sub-intents”。

```
onos> intents -i
id=0x2, state=INSTALLED, type=HostToHostIntent, appId=org.onlab.onos.ifwd
    constraints=[LinkTypeConstraint{inclusive=false, types=[OPTICAL]}]
    installable=[
PathIntent{id=0x4, appId=DefaultApplicationId{id=2, name=org.onlab.onos.ifwd},
    selector=DefaultTrafficSelector{criteria=[ETH_SRC{mac=00:00:00:00:00:0D}, ETH_DST{mac=00:00:00:00:00:07}]},
    treatment=DefaultTrafficTreatment{instructions=[]}, constraints=[LinkTypeConstraint{inclusive=false, types=[OPTICAL]}],
    path=DefaultPath{src=ConnectPoint{elementId=00:00:00:00:00:0D/-1, portNumber=0},
                    dst=ConnectPoint{elementId=00:00:00:00:00:07/-1, portNumber=0}, type=INDIRECT, state=ACTIVE, durable=false}},
PathIntent{id=0x5, appId=DefaultApplicationId{id=2, name=org.onlab.onos.ifwd},
    selector=DefaultTrafficSelector{criteria=[ETH_SRC{mac=00:00:00:00:00:07}, ETH_DST{mac=00:00:00:00:00:0D}]},
    treatment=DefaultTrafficTreatment{instructions=[]}, constraints=[LinkTypeConstraint{inclusive=false, types=[OPTICAL]}],
    path=DefaultPath{src=ConnectPoint{elementId=00:00:00:00:00:07/-1, portNumber=0},
                    dst=ConnectPoint{elementId=00:00:00:00:00:0D/-1, portNumber=0}, type=INDIRECT, state=ACTIVE, durable=false}}]
```

例如，对于“host to host”的“intent”已经被编译成2条“path intents”，这2条“path intents”具有适当的流量选择和动作（with the appropriate traffic selections and actions computed on your behalf）

5 Intent Reactive Forwarding

ONOS中的另一个范例应用是“intent reactive forwarding”。与前面看到的，基于每个数据包推送流表项不同，“intent reactive forwarding”应用提供“intent”。特殊的，它提供一个“host to host”的“intent”（一个简单的connectivity intent），用于两个主机间的连通性。

5.1 除旧迎新

首先，移除原先加载的“reactive forwarding”应用，并且加载“intent reactive forwarding”应用。

```
onos> feature:uninstall onos-app-fwd
```

```
onos> feature:install onos-app-ifwd
```

注意两个应用的名字差了一个“i”。：）

OK，查看一下应用是否加载正确：

```
onos> apps
id=0, name=org.onlab.onos.net.intent
id=1, name=org.onlab.onos.fwd
id=2, name=org.onlab.onos.ifwd
```

如上所示，“intent reactive forwarding”被正确加载。The appId“Reactive Forwarding”应用的appId还在，所以如果你重新加载此应用，它将得到与先前相同的appId。

5.2 Intentionally React

接下来，让我们转发一些流量：
mininet> h21 ping h31
PING 10.0.0.13 (10.0.0.13) 56(84) bytes of data.
64 bytes from 10.0.0.13: icmp_seq=1 ttl=64 time=25.7 ms
64 bytes from 10.0.0.13: icmp_seq=2 ttl=64 time=1.73 ms
64 bytes from 10.0.0.13: icmp_seq=3 ttl=64 time=0.191 ms
64 bytes from 10.0.0.13: icmp_seq=4 ttl=64 time=0.079 ms
^C
--- 10.0.0.13 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 0.079/6.926/25.700/10.859 ms

使用flows命令可以列出“intent”所安装的流表项。“intent reactive forwarding”与其他应用，虽然最终结果是一样的，但是他们从根本上是不同的。“intent reactive forwarding”应用在h21和h31之间安装了一个“intent”，可以使用“intent -i”命令看到：

```
onos> intents -i
id=0x0, state=INSTALLED, type=HostToHostIntent, appId=org.onlab.onos.ifwd
    constraints=[LinkTypeConstraint{inclusive=false, types=[OPTICAL]}]
    installable=[
PathIntent{id=0x1, appId=DefaultApplicationId{id=2, name=org.onlab.onos.ifwd},
    selector=DefaultTrafficSelector{criteria=[ETH_SRC{mac=00:00:00:00:00:0D}, ETH_DST{mac=00:00:00:00:00:07}]},
    treatment=DefaultTrafficTreatment{instructions=[]}, constraints=[LinkTypeConstraint{inclusive=false, types=[OPTICAL]}],
    path=DefaultPath{src=ConnectPoint{elementId=00:00:00:00:00:0D/-1, portNumber=0},
                     dst=ConnectPoint{elementId=00:00:00:00:00:07/-1, portNumber=0}, type=INDIRECT, state=ACTIVE, durable=false}},
PathIntent{id=0x2, appId=DefaultApplicationId{id=2, name=org.onlab.onos.ifwd},
    selector=DefaultTrafficSelector{criteria=[ETH_SRC{mac=00:00:00:00:00:07}, ETH_DST{mac=00:00:00:00:00:0D}]},
    treatment=DefaultTrafficTreatment{instructions=[]}, constraints=[LinkTypeConstraint{inclusive=false, types=[OPTICAL]}],
    path=DefaultPath{src=ConnectPoint{elementId=00:00:00:00:00:07/-1, portNumber=0},
                     dst=ConnectPoint{elementId=00:00:00:00:00:0D/-1, portNumber=0}, type=INDIRECT, state=ACTIVE, durable=false}}]
```

从上面的输出结果中的appId可以看出Intent是由“intent reactive forwarding”应用推送的。intent是一个“host to host”的intent，它指出了路径（被安装的流）的细节。关于intents或intents框架更详细的，请参看：https://wiki.onosproject.org/display/ONOS/The+Intent+Framework

继续往下进行前，让我们删除刚刚安装的intent，以免影响下面的实验。

```
onos> remove-intent <TAB>
```

表示tab键，它自动不全关的id，类似的bash中tab键功能。删除后再使用intents命令查看一下：

```
onos> intents
```

确保没有intents存在，一面后面的实验出现问题。

6 State your intentions

使用intents比简单的使用流表项去编程网络的好处是：intents可以跟踪网络的状态，并且为了满足你的intention对网络进行重新配置。例如：如果某处的link断开了，intent框架可以变更（reroute）你的intent（也就是你的流表）到另外可用的路径上。但是，如果没有另外可用的路径该怎么办呢？在这种情况下，intent将进入失败状态，并且保持这种状态直到出现一个可用的路径。很酷吧，确实很牛拜！Let's go。

让我们从查看ONOS中已知的主机开始，如果你是按照上面的教程一路走来，你将看到4个主机：

```
onos> hosts
id=00:00:00:00:00:01/-1, mac=00:00:00:00:00:01, location=of:000000000000000b/3, vlan=-1, ip(s)=[10.0.0.1]
id=00:00:00:00:00:07/-1, mac=00:00:00:00:00:07, location=of:000000000000000c/3, vlan=-1, ip(s)=[10.0.0.7]
id=00:00:00:00:00:0D/-1, mac=00:00:00:00:00:0D, location=of:000000000000000d/3, vlan=-1, ip(s)=[10.0.0.13]
id=00:00:00:00:00:13/-1, mac=00:00:00:00:00:13, location=of:000000000000000e/3, vlan=-1, ip(s)=[10.0.0.19]
```

选择任意的两个主机，为他们安装“host to host”的intent：

```
onos> add-host-intent 00:00:00:00:00:01/-1 00:00:00:00:00:13/-1
```

上面的这条命令在主机10.0.0.1（h11）和主机10.0.0.19（h41）之间建立一条路径，用如下命令查看安装的intent：

```
onos> intents
id=0x9, state=INSTALLED, type=HostToHostIntent, appId=org.onlab.onos.cli
    constraints=[LinkTypeConstraint{inclusive=false, types=[OPTICAL]}]
```

现在，intent已经被安装，让我们来看一下他使用的是什么路径（注意：你的数据结果可能与下面的不同，这是因为所有可选的路径具有相同的花费equal cost，所以ONOS随机的选择一个可用的路径）：

```
onos> flows
deviceId=of:0000000000000001, flowRuleCount=2
   id=10000c364dd58, state=ADDED, bytes=0, packets=0, duration=1781, priority=123, appId=org.onlab.onos.net.intent
      selector=[IN_PORT{port=2}, ETH_SRC{mac=00:00:00:00:00:01}, ETH_DST{mac=00:00:00:00:00:13}]
      treatment=[OUTPUT{port=5}]
   id=10000c364ddb2, state=ADDED, bytes=0, packets=0, duration=1781, priority=123, appId=org.onlab.onos.net.intent
      selector=[IN_PORT{port=5}, ETH_SRC{mac=00:00:00:00:00:13}, ETH_DST{mac=00:00:00:00:00:01}]
      treatment=[OUTPUT{port=2}]
deviceId=of:0000000000000002, flowRuleCount=0
deviceId=of:000000000000000b, flowRuleCount=2
   id=10000c3659528, state=ADDED, bytes=0, packets=0, duration=1781, priority=123, appId=org.onlab.onos.net.intent
      selector=[IN_PORT{port=1}, ETH_SRC{mac=00:00:00:00:00:13}, ETH_DST{mac=00:00:00:00:00:01}]
      treatment=[OUTPUT{port=3}]
   id=10000c3659564, state=ADDED, bytes=0, packets=0, duration=1781, priority=123, appId=org.onlab.onos.net.intent
      selector=[IN_PORT{port=3}, ETH_SRC{mac=00:00:00:00:00:01}, ETH_DST{mac=00:00:00:00:00:13}]
      treatment=[OUTPUT{port=1}]
deviceId=of:000000000000000c, flowRuleCount=0
deviceId=of:000000000000000d, flowRuleCount=0
deviceId=of:000000000000000e, flowRuleCount=2
   id=10000c365a06b, state=ADDED, bytes=0, packets=0, duration=1781, priority=123, appId=org.onlab.onos.net.intent
      selector=[IN_PORT{port=1}, ETH_SRC{mac=00:00:00:00:00:01}, ETH_DST{mac=00:00:00:00:00:13}]
      treatment=[OUTPUT{port=3}]
   id=10000c365a0a7, state=ADDED, bytes=0, packets=0, duration=1781, priority=123, appId=org.onlab.onos.net.intent
      selector=[IN_PORT{port=3}, ETH_SRC{mac=00:00:00:00:00:13}, ETH_DST{mac=00:00:00:00:00:01}]
      treatment=[OUTPUT{port=1}]
```

从上面的输出结果可以看出，在 dpid 00 00 00 00:01 (s1) 和 00 00 00 00:0b (s11) 之间建立了流量流（从流表项可以看出来）。在GUI图形界面很容易查看。

接下来，让我们拆除（teardown）s1和s11之间的连接。你必须拆除s2和s11之间的连接，注意观察flows命令的输出信息。使用如下的命令：

```
mininet> link s1 s11 down
```

让我们再次查看一下流表：

```
onos> flows
deviceId=of:0000000000000001, flowRuleCount=0
deviceId=of:0000000000000002, flowRuleCount=2
   id=10000c364e119, state=ADDED, bytes=0, packets=0, duration=1, priority=123, appId=org.onlab.onos.net.intent
```

```
      selector=[IN_PORT{port=2}, ETH_SRC{mac=00:00:00:00:00:01}, ETH_DST{mac=00:00:00:00:00:13}]
      treatment=[OUTPUT{port=5}]
   id=10000c364e173, state=ADDED, bytes=0, packets=0, duration=1, priority=123, appId=org.onlab.onos.net.intent
      selector=[IN_PORT{port=5}, ETH_SRC{mac=00:00:00:00:00:13}, ETH_DST{mac=00:00:00:00:00:01}]
      treatment=[OUTPUT{port=2}]
deviceId=of:000000000000000b, flowRuleCount=2
   id=10000c3659547, state=ADDED, bytes=0, packets=0, duration=1, priority=123, appId=org.onlab.onos.net.intent
      selector=[IN_PORT{port=2}, ETH_SRC{mac=00:00:00:00:00:13}, ETH_DST{mac=00:00:00:00:00:01}]
     treatment=[OUTPUT{port=3}]
   id=10000c3659565, state=ADDED, bytes=0, packets=0, duration=1, priority=123, appId=org.onlab.onos.net.intent
      selector=[IN_PORT{port=3}, ETH_SRC{mac=00:00:00:00:00:01}, ETH_DST{mac=00:00:00:00:00:13}]
      treatment=[OUTPUT{port=2}]
deviceId=of:000000000000000c, flowRuleCount=0
deviceId=of:000000000000000d, flowRuleCount=0
deviceId=of:000000000000000e, flowRuleCount=2
   id=10000c365a08a, state=ADDED, bytes=0, packets=0, duration=1, priority=123, appId=org.onlab.onos.net.intent
      selector=[IN_PORT{port=2}, ETH_SRC{mac=00:00:00:00:00:01}, ETH_DST{mac=00:00:00:00:00:13}]
      treatment=[OUTPUT{port=3}]
   id=10000c365a0a8, state=ADDED, bytes=0, packets=0, duration=1, priority=123, appId=org.onlab.onos.net.intent
      selector=[IN_PORT{port=3}, ETH_SRC{mac=00:00:00:00:00:13}, ETH_DST{mac=00:00:00:00:00:01}]
      treatment=[OUTPUT{port=2}]
```

从上面的输出结果可发现，流变成从00 00 00 00:01 to 00 00 00 00:02 (s2)，其余的流没有发生变化。这一切是怎么发生的呢？当拆除s1和s11之间的连接后，ONOS检测到这个变化（时间），并且通知所有于对此事件有兴趣的people“连接断开了”。所以，intent服务收到这个信息后可以知道到受此变化影响的intent，所以，intent服务根据这个变化重新编译intent，最终导致这个intent被安装到一条不同的路径上。

这个简单的示例展示了intent比单纯的安装流表更加强大的地方。Intents维护你的意图（intention，这也是intent名称的来历）同时保留安装他们的能力，这是可能的和最高效的。

6.1 Up down up down

If you wish you can take down more links and see what happens. Obviously, if you partition the network then no flows will be installed, sadly ONOS doesn't grow links between switches yet. You can bring up links in mininet by:

```
mininet> link s1 s11 up
```

Have fun!

7 ONOS 图形用户接口

首先，重新加载reactive forwarding应用。

```
onos> feature:uninstall onos-app-ifwd
onos> feature:install onos-app-fwd
```

ONOS自带GUI，允许你用简单的方法维护网络。加载ONOS内置的GUI：

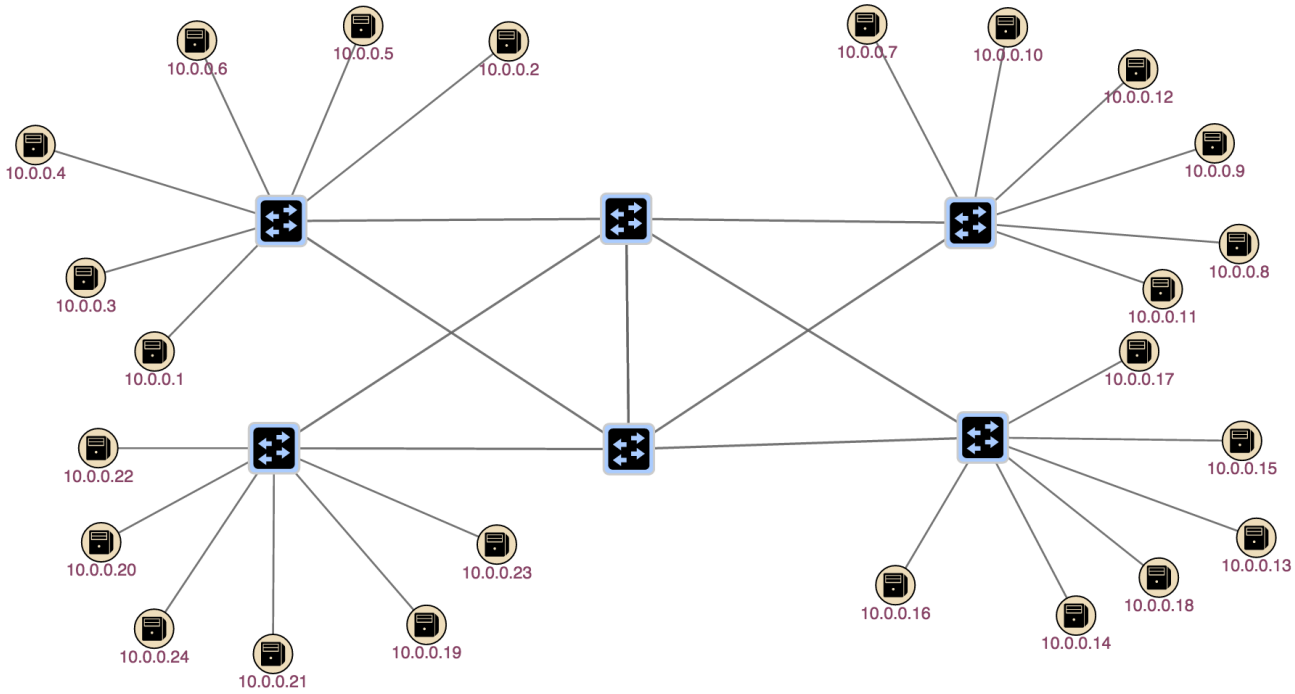
```
onos> feature:install onos-gui
```

之后，双击桌面上的'ONOS GUI' 图标，启动web浏览器。初始化完成后，看到浏览器上有一个基于USA地图背景的网络拓扑（使用‘b’键隐藏或显示地图背景），在浏览器上输入“/”键，显示快速参考。

接下来，我们让主机出现在这个UI中。在Mininet提示符中使用pingall命令让主机和网络进行回话：

```
mininet> pingall
*** Ping: testing ping reachability
h11 -> h12 h13 h14 h15 h16 h21 h22 h23 h24 h25 h26 h31 h32 h33 h34 h35 h36 h41 h42 h43 h44 h45 h46
h12 -> h11 h13 h14 h15 h16 h21 h22 h23 h24 h25 h26 h31 h32 h33 h34 h35 h36 h41 h42 h43 h44 h45 h46
。。。。
```

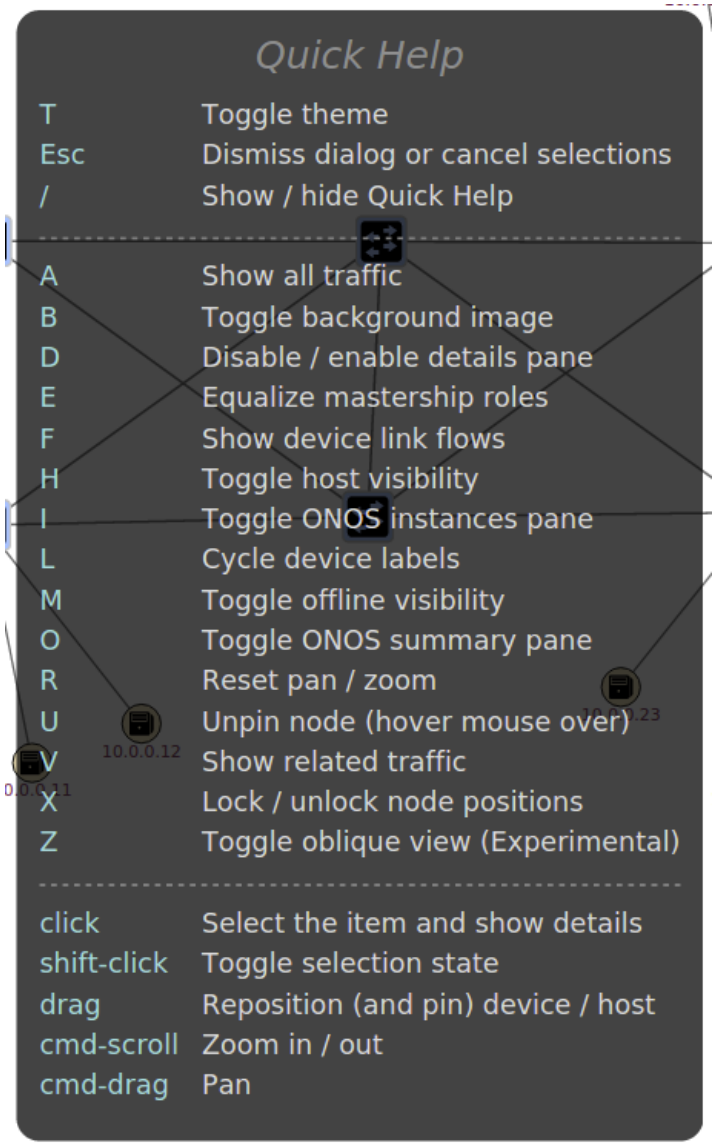
注意初始时主机不会再UI中显示，使用‘h’键来显示主机。如下图所示：



7.1 GUI特性

7.1.1 GUI速查表

任何时候，在浏览器中使用‘/’键，都可以显示出GUI快捷键的速查表。



7.1.2 摘要面板

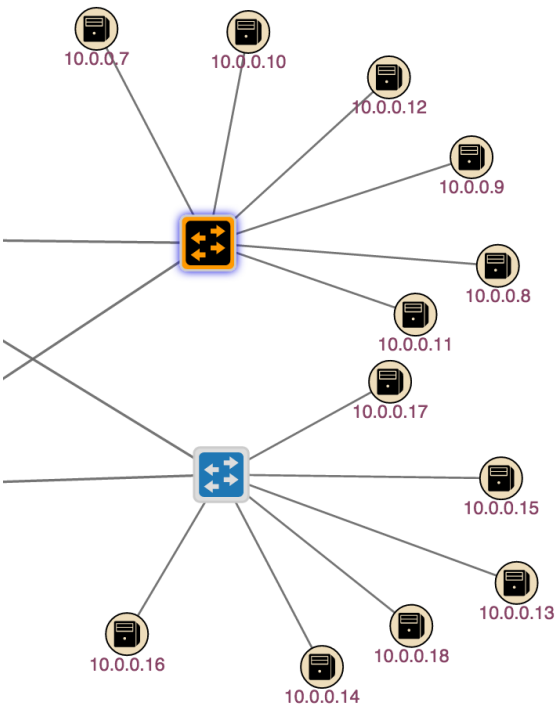
在GUI界面中有一个非常有用的摘要面板，显示了在你的ONOS集群上的摘要信息。

ONOS Summary

Devices :	6
Links :	20
Hosts :	24
Topology SCCs :	1
Paths :	44
Intents :	0
Flows :	0
Version :	1.0.0*

7.1.3 交换机详细信息

当在GUI中点击一台交换机时，在右边会出现一个交换机详细信息面板。如下图所示：



of:000000000000000c

URI :	of:000000000000000c
Vendor :	Nicira Networks, Inc.
H/W Version :	Open vSwitch
S/W Version :	1.4.6
Serial Number :	None
Master :	local
Latitude :	
Longitude :	
Ports :	9
Flows :	0

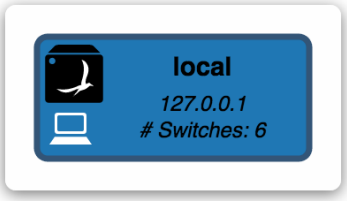
Show Related Traffic

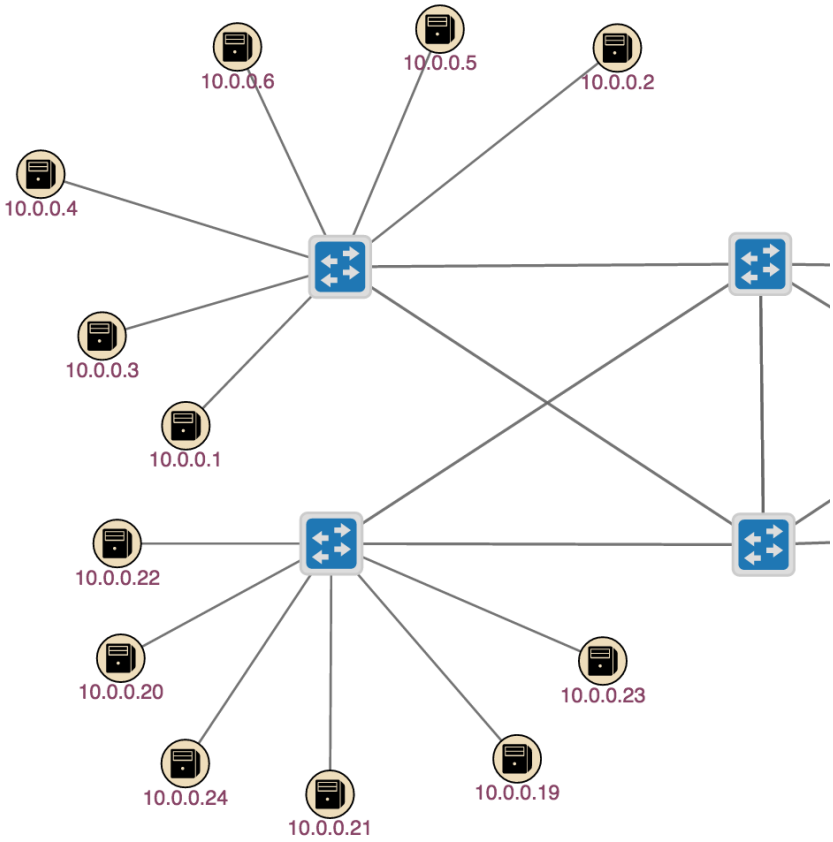
Show Device Flows

从上图中可以发现，交换机详细信息面板中显示有9个端口，但是在GUI中只显示了8个。这是引文，OpenFlow交换机具有虚拟端口（virtual ports），这些端口很难显示在UI中。在交换机上右击鼠标，将隐藏交换机详细信息面板。

7.1.4 实例（Instances）

ONOS的GUI界面可以显示哪一个ONOS实例是活动的（默认此功能是打开的，在左上角）。看使用'i'键打开或关闭当前活动示例的显示功能。

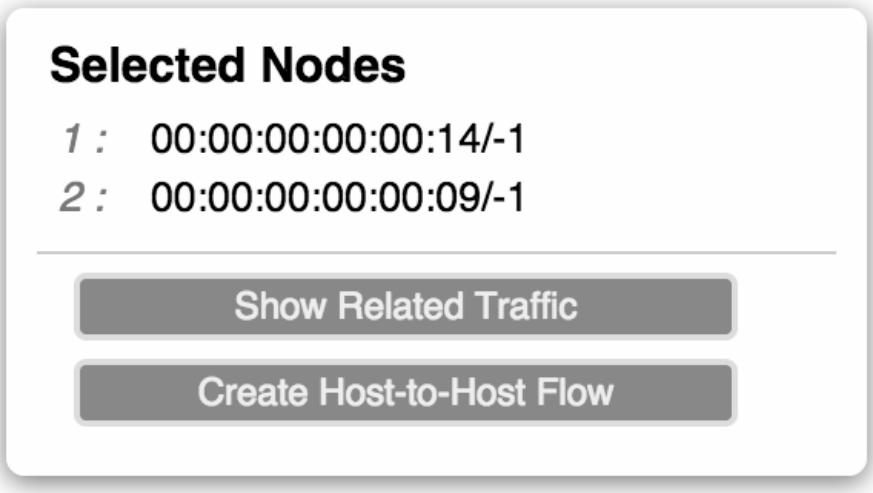




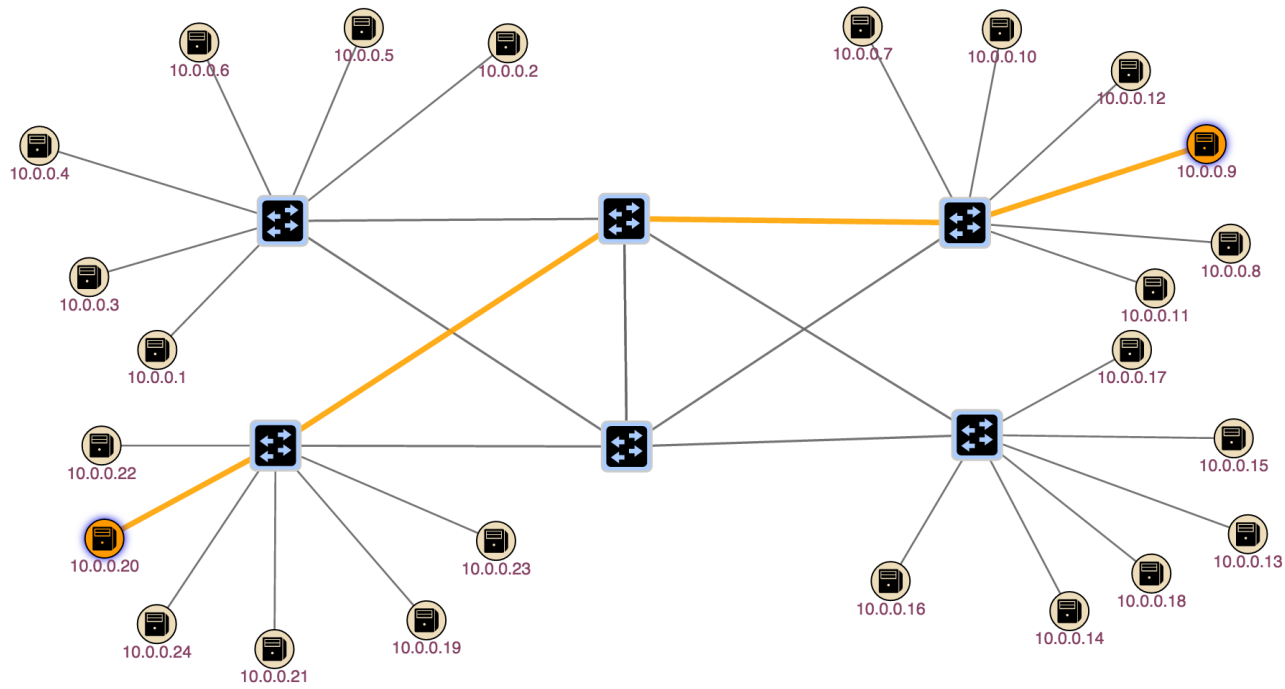
注意：交换机颜色和示例的颜色一致，表示交换机被哪个实例控制。这有利于总体上浏览实例和交换机的控制关系。

7.1.5 安装Intent

使用GUI可以安装intent。首先，选择两台主机（先选中一个主机，然后按下shift键选择另一个主机。）。我们选择10.0.0.20和10.0.0.9两台主机。现在，GUI的右边会显示一个面板，如下图所示：



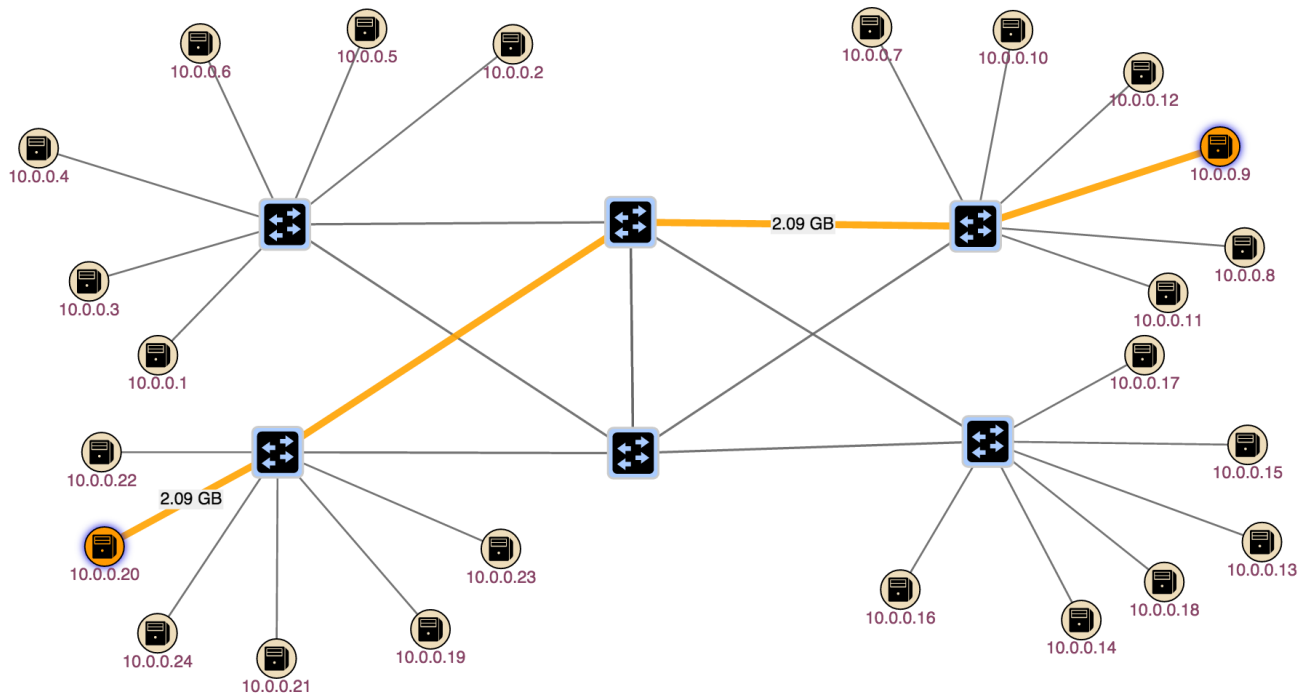
点击'Create Host-to-host Flow'，这将实际的规定一个host to host的intent，并且将此intent路径高亮显示。如下图所示：



可以通过命令行检查使用ONOS GUI安装的intent：

```
onos> intents
    id=0x223838ca, state=INSTALLED, type=HostToHostIntent, appId=org.onlab.onos.gui
    [LinkTypeConstraint{inclusive=false, types=[OPTICAL]}]
```

现在，可以在此intent上发送一些流量。在GUI上可以以动画的方式动态的显示在这个路径上的流量和流量的计数值。如下图所示：(在实验室，此部分的实时流量显示有问题，达不到预想的效果！)



7.1.6 显示所有的流量 traffic

使用‘a’键在ONOS GUI上显示“All Traffic”。显示运行网络中的任何流量。

8.进一步探索

本教程仅仅介绍了一些ONOS的皮毛。强烈建议大家进一步使用ONOS和开发你自己的应用程序,Application tutorial。