BPFCollector中是对遥测的实现

SRRRouting是对路由算法的实现

data/influx中是系统运行收集到的数据

mininet根据topo中的文件生成拓扑

mininet.rar中是对数据层的具体实现

onos1.rar-onos4.rar中是对控制层的实现，由于太大，被拆分成了4块

系统部署好后，系统的运行流程如下：

## 6.2基于可信身份的多维路由与分级管控

### 6.2.1基于可信身份多维路由机制

|  |  |
| --- | --- |
| 测试目的 | 基于IPv6实现多维路由转发。 |
| 测试设计 | 除了基于目的地址前缀的默认路由，可以在数据平面为特定可信身份的源IPv6地址、目的IPv6地址和目的端口的数据流生成指定的转发路径。 |
| 测试组网 | C:\Users\zhangcc\Desktop\1.png |
| 预置条件 | 网络搭建成功，主机均已经被发现，主机1、2、3的IP验证为真实可信。 |
| 测试步骤 | 1、控制端、数据端：  分别打开一个命令行窗口，输入ifconfig确保两个机器在192.168.\*\*\*.\*\*\*子网内，互相ping测试连通性。  2、控制端：  export ONOS\_ROOT=/home/sdn/onos修改环境变量  cd onos进入控制器主目录  bazel run onos-local -- clean debug编译并启动控制器  待出现“127.0.0.1 state to READY”后控制器启动  打开浏览器进入控制器页面  3、控制端：  打开另一命令行窗口，输入onos，进入控制器命令行窗口，输入：  app activate zdy.labproject.v4v6ECMP  app activate zdy.labproject.pipelines.srv6\_INT  启动管道和路由应用  4、控制端：  打开另一命令行窗口，输入  cd ~/onos/tools/package/runtime/bin  ./onos-netcfg 127.0.0.1 /home/sdn/netcfg.json  配置网络信息  5、数据端：  sudo -E python /home/sdn/topo/topo.py --onos-ip=192.168.\*\*\*.\*\*\* --pipeconf-id=zdy.labproject.pipelines.srv6\_INT，启动Mininet网络拓扑  （出现问题后ctrl+c, 输入exit后重新启动，成功后可在控制器页面中看到拓扑结构）  6、数据端：  Mininet中h11 ping h51  一段时间后ctrl+c停止  输入h51 ping h11，一段时间后ping通，h11和h51被发现  之后停止，h12 ping h51, h13 ping h51,发现所有主机  （在控制器页面中按H键可以查看主机）  7、数据端：  Mininet中输入xterm h11，xterm h12，xterm h13打开三个主机的命令行窗口  8、数据端：  在h11、h12、h13窗口中分别输入  h11 -> iperf -c 2001:1:5::1 -u -V -P1 -b0.1M -t10000 -i10 -p 5002  h12 -> iperf -c 2001:1:5::1 -u -V -P1 -b0.1M -t10000 -i10 -p 5002  h13 -> iperf -c 2001:1:5::1 -u -V -P1 -b0.1M -t10000 -i10 -p 5003  在控制器页面中按A键位可以看到数据流，按L键显示路由器编号，进入s02节点可以看到多维流表有数据包经过。  停止h11发送流，输入  h11 -> iperf -c 2001:1:5::1 -u -V -P1 -b0.1M -t10000 -i10 -p 5003  数据流仍然正常通过 |
| 预期结果 | 1. 在h11的xterm窗口中，向h51的5002端口发送IPv6流量可以正常转发。 2. 改变源主机，在h12的xterm窗口中，向h51的5002端口发送IPv6流量可以正常转发。 3. 既改变源主机又改变目的端口，在h13的xterm窗口中，向h51的5003端口发送IPv6流量可以正常转发。 4. 改变目的端口，在h11的xterm窗口中，向h51的5003端口发送IPv6流量可以正常转发。 |
| 测试记录 | 1. 可观察数据平面设置了主机1、主机2和主机3的可信身份的IPv6源地址的多维路由转发规则      1. 使用iPerf测试，界面展示主机1、主机2、主机3可信身份的IPv6源地址通过不同路径到该目的地址可以连通。      1. 网络控制器显示三个数据流被认定为三个不同的单位经过不同的路径转发。 |
| 测试结果 | ☑Passed  □Failed  □N/A |

## 6.7应用5：服务定制化可靠路由

### 6.7.1 可靠定制请求的分析与定制备份路径段列表的计算

|  |  |
| --- | --- |
| 测试目的 | 使用带内遥测延迟数据，结合IP源地址真实性定制备份路径。 |
| 测试设计 | 在一定时间周期内（10S），获取当前网络遥测延迟数据，结合节点和链路可靠信息，绘制当前网络状态。处理主机服务定制可靠路由请求，判断IP源地址真实性等，使用定制算法为特定主机选择SRv6中TI-LFA 的PQ空间节点，计算出定制段列表。 |
| 测试组网 |  |
| 预置条件 | 网络搭建成功，分段路由信息配置成功，主机均已经被发现。 |
| 测试步骤 | 1、控制端、数据端：  分别打开一个命令行窗口，输入ifconfig确保两个机器在192.168.\*\*\*.\*\*\*子网内，互相ping测试连通性。  2、控制端：  export ONOS\_ROOT=/home/sdn/onos修改环境变量  cd onos进入控制器主目录  bazel run onos-local -- clean debug编译并启动控制器  待出现“127.0.0.1 state to READY”后控制器启动  打开火狐浏览器进入控制器页面  3、控制端：  打开另一命令行窗口，输入onos，进入控制器命令行窗口，控制器命令行窗口中：  app activate zdy.labproject.v4v6ECMP  app activate zdy.labproject.pipelines.srv6\_INT  app activate org.onosproject.inbandtelemetry  启动管道、遥测和路由应用  4、控制端：  打开另一命令行窗口，输入  cd ~/onos/tools/package/runtime/bin  ./onos-netcfg 127.0.0.1 /home/sdn/netcfg.json  配置网络信息  5、数据端：  sudo -E python /home/sdn/topo/topo.py --onos-ip=192.168.\*\*\*.\*\*\* --pipeconf-id=zdy.labproject.pipelines.srv6\_INT，启动Mininet网络拓扑  6、数据端：  Mininet中h11 ping h51，一段时间后ctrl+c停止  输入h51 ping h11，一段时间后ping通，h11和h51被发现  之后停止，h12 ping h51, h13 ping h51,发现所有主机  （在控制器页面中按H键可以查看主机）  7、Python项目expect.sh文件中：  将IP置换为控制端IP  8、数据端：  新命令行窗口：influx，进入InfluxDB命令行  InfluxDB命令行窗口：drop database INTdatabase  create database INTdatabase  重置INTdatabase数据库  9、数据端：  新命令行窗口：  sudo sysctl net/core/bpf\_jit\_enable=1  sudo ip link add veth\_1 type veth peer name veth\_2  sudo ip link set dev veth\_1 up  sudo ip link set dev veth\_2 up  配置虚拟网卡用于遥测  打开P4Runtime：simple\_switch\_CLI --thrift-port `cat /tmp/bmv2-s09-thrift-port`  输入mirroring\_add 500 5  10、控制端：  控制器页面，进入遥测窗口，配置遥测信息：  127.0.0.1 54321  10.1.1.1 10.1.5.1 空 空 UDP  SwitchID, HopLatency, EgressTimeStamp  11、数据端：  新命令行窗口：  sudo python BPFCollector/InDBClient.py veth\_2  12、数据端：  Mininet窗口：link s02 s09 down，使链路断开  Mininet窗口：xterm h11，进入h11窗口  H11窗口：iperf -c 10.1.5.1 -u -P5 -b2M -t10000 -i60 > /home/sdn/Desktop/iperf.txt，使h11向h51发送ipv4流量  13、数据端：  InfluxDB窗口：using INTdatabase  show measurements确认是否有遥测数据  （若没有，则重新启动网络控制器和数据平面，但不用重置数据库）  14、数据端：  Python项目运行FlowConfigWriter文件写入流信息  15、数据端：  Mininet窗口：  link s02 s09 up，使链路恢复  xterm h11进入h11窗口，h11窗口：iperf -c 2001:1:5::1 -u -V -P1 -b0.025M -t10000 -i10  xterm h12进入h12窗口，h12窗口：iperf -c 2001:1:5::1 -u -V -P1 -b0.025M -t10000 -i10  16、Python项目运行GUI，输入两个定制请求  2001:1:1::1 0 1  2001:2:1::1 0 1  17、Python项目运行SRComputor，获取段列表 |
| 预期结果 | 1. 数据端获得网络的带内遥测信息存放在InfluxDB数据库中。 2. 使用服务定制客户端可以输入用户的服务定制可靠路由请求。 3. 判定为非可信源地址IP的服务定制请求将被忽视不进行处理。 4. 存在可信服务定制请求的时候，根据当前带内遥测信息计算出定制保护路径。 |
| 测试记录 | 1. 获得时间周期内带内遥测的延迟数据；     说明：前面的表记录的是一个交换机的延时数据，后面5个表记录的是路径的延时数据    说明：第一列为时间戳，第二列为路径延时，第三列为流的路径   1. 可以使用可靠路由定制客户端录入定制请求，且认定为不可信的源IP的定制请求将被忽略；       说明：上图第三列0表示普通保护路由，1表示定制保护路由    说明：用户的定制需求被载入内核   1. 计算出满足定制请求的备份路径SRv6段列表；     说明:TDPath为定制路由的段列表，ODPath为普通路由的段列表 |
| 测试结果 | ☑Passed  □Failed  □N/A |

### 6.7.2 使用ONOS P4Runtime下发二维SRv6备份路径流表

|  |  |
| --- | --- |
| 测试目的 | 获取段列表后，使用ONOS将其转化为二维SRv6备份路径流表。 |
| 测试设计 | 在获取到备份路径的段列表后，将其使用ONOS的P4 Runtime转化为保护节点的二维SRv6的段插入流表，且在不发生链路故障时仍使用之前的流表。 |
| 测试组网 | C:\Users\zhangcc\Desktop\1.png |
| 预置条件 | 定制保护段列表已经计算完成，并已经通过scp发送到控制端。 |
| 测试步骤 | 1. 控制端：   控制器页面，观察s02保护节点流表   1. 控制端：   控制器页面，观察流量实际发送情况 |
| 预期结果 | 1. 在ONOS页面的s02交换机上看到TD\_srv6\_transit和OD\_srv6\_transit两个保护流表，且不被启用。 2. 不发生s02->s09链路故障时，IPv6流量通过s02->s09最短路径正常转发。 |
| 测试记录 | 1. 将段列表转化为二维SRv6保护流表，且提前安装在s02节点上；      1. 将段列表转化为普通TI-LFA保护流表，且提前安装在s02节点上；      1. s02->s09链路完好时，IPv6流量经过该链路正常转发，不启用任何保护流表。   C:\Users\zhangcc\Desktop\2.png  说明：h11发往h51数据流、h12发往h51数据流的带宽均为2pps，s02-s09链路的带宽为5pps，所以两条流走相同的路径 |
| 测试结果 | ☑Passed  □Failed  □N/A |

### 6.7.3 链路故障和定制流量恢复

|  |  |
| --- | --- |
| 测试目的 | 验证链路发生故障，定制流量经过二维SRv6保护流表保护，非定制流量通过普通TI-LFA保护流表保护 |
| 测试设计 | 使转发IPv6的最短路径链路发生故障，删除保护节点的正常转发流表，启用二维SRv6保护流表，在Wireshark发现定制流量的IPv6数据包内已经插入正确的段列表。流量经过预先定制的路径转发。 |
| 测试组网 |  |
| 预置条件 | 两种保护流表已经在s02安装，IPv6流量通过s02->s09正常转发。 |
| 测试步骤 | 1. 数据端：   Mininet窗口：输入link s02 s09 down，使IPv6转发链路发生故障。   1. 控制端：   网络控制器页面：查看定制流量和非定制流量的实际保护路径，查看s02交换机二维SRv6保护流表和普通TI-LFA保护流表的启用状态。 |
| 预期结果 | 1. 当数据端输入s02->s09使链路发生故障后，ONOS页面的s02交换机上普通SRv6流表被删除，TD\_srv6\_transit和OD\_srv6\_transit两个保护流表启用。 2. ONOS页面上看到IPv6流量经过特定的保护路径被保护，流量继续传输。 |
| 测试记录 | 1、故障链路后，s02节点原IPv6分段路由流表被删除；  2、二维SRv6保护流表被启用，用来保护定制流量；    说明：“54”表明该定制流表项有流匹配   1. 普通TI-LFA保护流表被启用，用来保护普通流量；   普通TI-LFA保护流表被启用，用来保护普通流量；    说明：“13”表明该普通流表项有流匹配   1. 流量根据自身定制与否按照不同的路径被保护；      1. 在Wireshark发现定制流量的IPv6数据包内已经插入保护段列表； |
| 测试结果 | ☑Passed  □Failed  □N/A |