**数据中心两种常用流量模型运用mininet的实现**

[**https://www.sdnlab.com/11079.html**](https://www.sdnlab.com/11079.html)

!!! 注意Python脚本的空格问题

在网络性能评估中一个巨大的挑战就是如何生成真实的网络流量，还好可以通过程序来创造人工的网络流量，通过建立测试环境来模拟真实的状况。本文就以数据中心网络为目标场景，来在mininet仿真环境中尽可能地还原数据中心内部的真实流量情况。目前有两种常用的流量模型:

* **随机模型**：主机向在网络中的另一任意主机以等概率发送数据包
* **概率模型**：在网络中，编号为m的主机分别以概率Pt 、Pa 、Pc 、向主机编号为(m+i)、(m+j)、(m+k)的主机发送数据包

我们使用mininet中的iperf工具在网络中生成UDP流量，iperf客户端传送数据流到iperf的服务端，由服务端接收并记录相关信息。mininet自带控制台可供使用的命令虽然已经比较丰富，但却并未给出较为明确的API接口来支持用户自定义命令。在诸如数据中心这样复杂、网络节点较多的仿真环境中做一些批处理的工作就需要非常大的，比如通过iperf在所有主机之间发生流量。所以我们需要将自定义命令添加到mininet中，在mininet中完成新命令的拓展。

**一、 mininet功能拓展**

在mininet中进行功能拓展主要分为3步:  
1. **修改mininet/net.py**: net模块实现Mininet类，是仿真平台mininet的主体类，该类完成节点添加配置、网络基本功能和一些选项功能的实现。我们需要将我们自定义的函数定义在该类中。

|  |  |
| --- | --- |
| 1  2  3 | class Mininet(object):      def function(self, \*\*kwargs):          #function code |

2. **修改mininet/cli.py**: cli模块定义了CLI类，为mininet提供命令行接口，用于解析用户输入的命令，之前定义的自定义命令需要在CLI类中通过注册函数注册这条自定义命令。

|  |  |
| --- | --- |
| 1  2  3 | class CLI(Cmd):      def do\_function(self, line):          #do\_function code |

完成命令注册与绑定。

3. **修改bin/mn**: 在完成命令注册与绑定后，需要在mininet执行解释器中注册命令与对应执行函数的映射关系。

|  |  |
| --- | --- |
| 1  2  3  4 | ALTSPELLING = { 'pingall':'pingAll',              'pingpair':'pingPair',              'iperfudp':'iperfUdp',              'function':'function} |

4．net.py和cli.py均在mininet/mininet目录，mn文件在在mininet/bin目录中。在代码修改完成后需要重新编译安装一遍mininet，即运行：

|  |  |
| --- | --- |
| 1 | $~/mininet/util/install.sh –n |

**二、 两种流量模型在mininet中的实现**

**2.1 随机模型**

任意一台主机以等概率随机地向另外一台主机发起一条UDP数据流。

**修改mininet/net.py**

首先，先在两个主机之间进行iperf测试，并且在server端记录，实现iperf\_single函数：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | def iperf\_single( self,hosts=None, udpBw='10M', period=60, port=5001):          """Run iperf between two hosts using UDP.             hosts: list of hosts; if None, uses opposite hosts             returns: results two-element array of server and client speeds"""          if not hosts:              return          else:              assert len( hosts ) == 2          client, server = hosts          filename = client.name[1:] + '.out'          output( '\*\*\* Iperf: testing bandwidth between ' )          output( "%s and %s\n" % ( client.name, server.name ) )          iperfArgs = 'iperf -u '          bwArgs = '-b ' + udpBw + ' '          print "\*\*\*start server\*\*\*"          server.cmd( iperfArgs + '-s -i 1' + ' > /home/zg/log/' + filename + '&')          print "\*\*\*start client\*\*\*"          client.cmd(              iperfArgs + '-t '+ str(period) + ' -c ' + server.IP() + ' ' + bwArgs              +' > /home/zg/log/' + 'client' + filename +'&') |

接着为mininet添加自定义命令iperfmulti，依次为每一台主机随机选择另一台主机作为iperf的服务器端，通过调用iperf\_single,自身以客户端身份按照指定参数发送UDP流，服务器生成的报告以重定向的方式输出到文件中，使用iperfmulti命令，主机随机地向另一台主机发起一条恒定带宽的UDP数据流。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | def iperfMulti(self, bw, period=60):      base\_port = 5001      server\_list = []      client\_list = [h for h in self.hosts]      host\_list = []      host\_list = [h for h in self.hosts]        cli\_outs = []      ser\_outs = []        \_len = len(host\_list)  for i in xrange(0, \_len):   ##   server = client   ##    while( server == client ):  ##            server = random.choice(host\_list) ##随机选取主机作为服务器端通信          server\_list.append(server)            self.iperf\_single(hosts = [client, server], udpBw=bw, period= period, port=base\_port)          sleep(.05)          base\_port += 1        sleep(period)      print "test has done" |
|  |  |

**修改mininet/cli.py**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | def do\_iperfmulti( self, line ):      """Multi iperf UDP test between nodes"""      args = line.split()      if len(args) == 1:          udpBw = args[ 0 ]          self.mn.iperfMulti(udpBw)      elif len(args) == 2:          udpBw = args[ 0 ]          period = args[ 1 ]          err = False          self.mn.iperfMulti(udpBw, float(period))      else:          error('invalid number of args: iperfmulti udpBw period\n' +                 'udpBw examples: 1M 120\n') |

**修改bin/mn**

在mininet/bin目录下修改mn文件，将iperfmulti加入到对应的列表中。

|  |  |
| --- | --- |
| 1  2  3  4  5 | ALTSPELLING = { 'pingall': 'pingAll',              'pingpair': 'pingPair',              'iperfudp': 'iperfUdp',              'iperfUDP': 'iperfUdp',              'iperfmulti':'iperfMulti' } |

最后，进入mininet/util目录，重新编译安装mininet

|  |  |
| --- | --- |
| 1 | $~/mininet/util/install.sh -n |

重启mininet，输入iperf，可用table补全iperfmulti，从而可使用iperfmulti进行流量随机模型的测试。

**2.2 概率模型**

为mininet添加自定义命令iperfpb，依次为每一台主机（编号为m）分别以概率Pt 、Pa 、Pc 向主机编号为(m+i)、(m+j)、(m+k)的主机发送数据包，通过调用iperf\_single,自身以客户端身份按照指定参数发送UDP流，服务器生成的报告以重定向的方式输出到文件中，使用iperfpb命令，主机按概率向其他被选择的主机发起一条恒定带宽的UDP数据流。

**概率选择函数**

为完成以一定概率选择主机，我们需要实现一个概率选择函数randompick，这个函数可用于以不同的概率从一个列表中随机地选择一些元素。下面为randompick的实现过程：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | def random\_pick( self, \_list, probabilities):      x = random.uniform(0,1)      p = None      cumulative\_probability = 0.0      for item, item\_probability in zip(\_list, probabilities):          cumulative\_probability += item\_probability          p = item          if x < cumulative\_probability:break      return p |

**修改mininet/net.py**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | def iperfPb (self, bw, period = 60, i = 1,j = 4,k = 64,pt = 0.5,pa = 0.3):      base\_port = 5001      server\_list = []      client\_list = []      client\_list = [h for h in self.hosts]      cli\_outs = []      ser\_outs = []      host\_list = []      host\_list = [h for h in self.hosts]      pc = 1 - pt - pa      p\_list = [pt,pa,pc]      \_len = len(self.hosts)      for key in xrange(\_len):          client = host\_list[key]          access\_host = [host\_list[(key+i)%\_len],host\_list[(key+j)%\_len],host\_list[(key+k)%\_len]]          server = self.random\_pick(access\_host,p\_list)          server\_list.append(server)          self.iperf\_single(hosts = [client, server], udpBw=bw, port=base\_port)          sleep(.05)      sleep(period)      print "test has done" |

**修改mininet/cli.py**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | def do\_iperfpb(self, line):      """Multi iperf UDP test with probablity"""      args = line.split()      if len(args) == 1:          udpBw = args[ 0 ]          self.mn.iperfMulti(udpBw)      elif len(args) == 2:          udpBw = args[ 0 ]          period = args[ 1 ]          err = False          self.mn.iperfPb(udpBw, float(period))      else:          error('invalid number of args: iperfmulti udpBw period\n' +                 'udpBw examples: 1M 120\n') |

**修改bin/mn**

在mininet/bin目录下修改mn文件，将iperfpb加入到对应的列表中。

|  |  |
| --- | --- |
| 1  2  3  4  5 | ALTSPELLING = { 'pingall': 'pingAll',              'pingpair': 'pingPair',              'iperfudp': 'iperfUdp',              'iperfUDP': 'iperfUdp',              'iperfpb':'iperfPb' } |

最后，进入mininet/util目录，重新编译安装mininet：

|  |  |
| --- | --- |
| 1 | $~/mininet/util/install.sh -n |

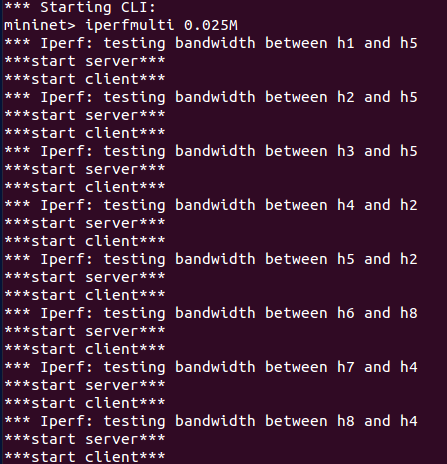
重启mininet，输入iperf，可用table补全iperfpb，从而可使用iperfpb进行流量的概率模型的测试。

**测试结果**

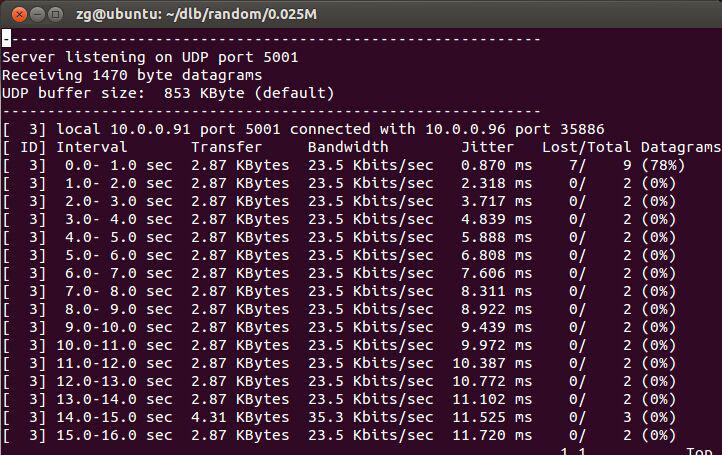
我们使用mininet仿真数据中心网络的fattree拓扑场景，包含128台主机与80台交换机的8叉fattree网络。流量负载被定义为边缘交换机与主机之间链路上所占用的带宽。下面我们来看看之前我们完成的两种流量模型的结果。

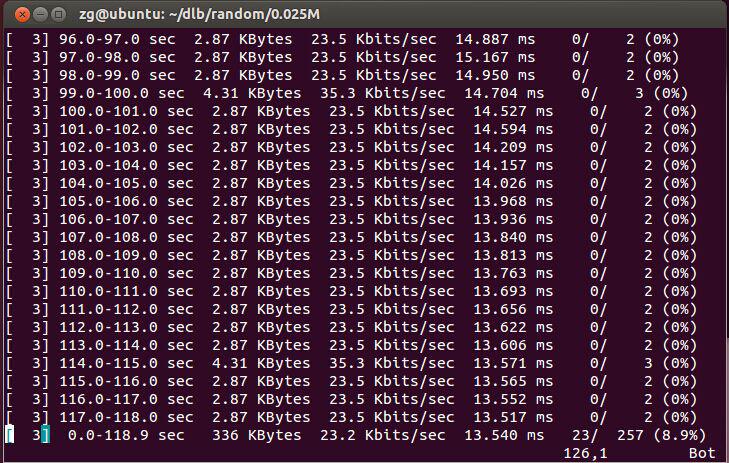
**随机模型**

在mininet中执行iperfmulti命令，设置带宽参数为0.025M，如下图所示，我们就能看到8台主机随机地向另外一台主机发送数据包：



下面是iperf服务器端生成的报告：

[](http://7xnzbp.com2.z0.glb.qiniucdn.com/wp-content/uploads/2015/04/iperf%E6%9C%8D%E5%8A%A1%E5%99%A8%E7%AB%AF%E7%94%9F%E6%88%90%E7%9A%84%E6%8A%A5%E5%91%8A1.jpg)

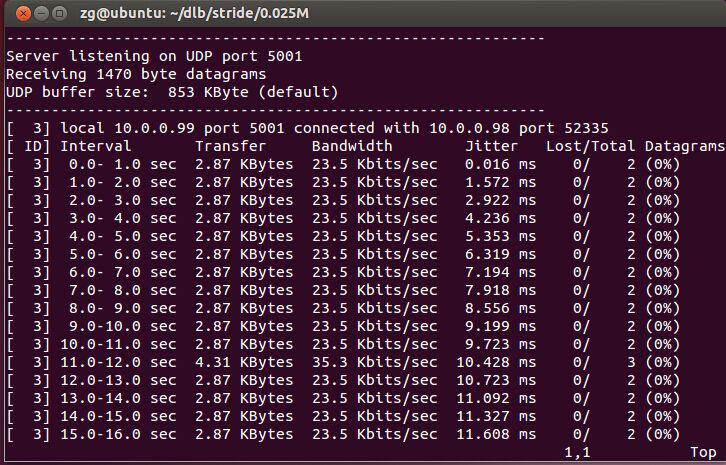
[](http://7xnzbp.com2.z0.glb.qiniucdn.com/wp-content/uploads/2015/04/iperf%E6%9C%8D%E5%8A%A1%E5%99%A8%E7%AB%AF%E7%94%9F%E6%88%90%E7%9A%84%E6%8A%A5%E5%91%8A2.jpg)

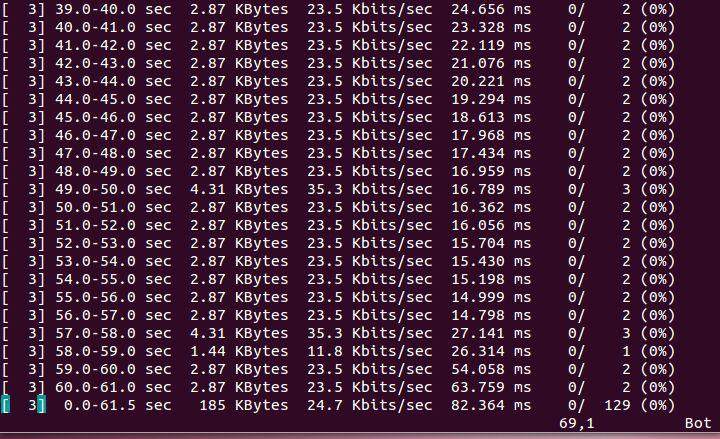
可以看出，这是ip为10.0.0.91的主机向10.0.0.96的主机发送数据包，每隔一秒打印信息，包括带宽，延时抖动，丢包情况等数据。最后，打印测试时间内服务器端总共接收的数据量、平均带宽数据、平均延时抖动和平均丢包情况。

**概率模型**

在mininet中执行iperfPb命令，设置带宽参数为0.025M，依照概率模型，为每一台主机选取对应主机，发送数据包，如下图所示：

下面是iperf服务器端生成的报告：

[](http://7xnzbp.com2.z0.glb.qiniucdn.com/wp-content/uploads/2015/04/%E6%A6%82%E7%8E%87%E6%A8%A1%E5%9E%8Biperf%E6%9C%8D%E5%8A%A1%E5%99%A8%E7%AB%AF%E7%94%9F%E6%88%90%E7%9A%84%E6%8A%A5%E5%91%8A1.jpg)

[](http://7xnzbp.com2.z0.glb.qiniucdn.com/wp-content/uploads/2015/04/%E6%A6%82%E7%8E%87%E6%A8%A1%E5%9E%8Biperf%E6%9C%8D%E5%8A%A1%E5%99%A8%E7%AB%AF%E7%94%9F%E6%88%90%E7%9A%84%E6%8A%A5%E5%91%8A2.jpg)

可以看出，这是ip为10.0.0.99的主机向10.0.0.98的主机发送数据包，每隔一秒打印信息，包括带宽，延时抖动，丢包情况等数据。最后，打印测试时间内服务器端总共接收的数据量、平均带宽数据、平均延时抖动和平均丢包情况。