Wireshark 拥塞控制实验教程

一、设置 Wireshark 捕获环境

1. **选择网络接口**

- 打开 Wireshark，选择正在使用的网络接口（如 Wi-Fi 或以太网卡）。
- 若需过滤特定设备流量，使用过滤器（如 tcp and host 192.168.1.100）

2. **启动捕获**

点击“开始捕获”按钮记录流量。

二、触发网络拥塞

- **大文件传输**：通过云盘上传/下载大文件。
- **视频流媒体**：同时播放多个高清视频。
- **网络压力工具**：使用 iperf3 生成高带宽流量

```
iperf3 -s

iperf3 -c <服务器 IP> -t 60 -P 4   # 启动 4 条并行 TCP 连接
```

三、识别拥塞的关键指标

1. 重传包（Retransmissions）

同一序列号（Seq）的数据包被多次发送。

- **过滤语法**：`tcp.analysis.retransmission`
- **分析路径**：
  `Statistics > TCP Stream Graphs > Time-Sequence Graph`

2. 重复 ACK（Duplicate ACK）

接收方多次发送相同的 ACK 号（期望下一个未到达的数据包）

- 过滤语法：`tcp.analysis.duplicate_ack`
- 分析路径：
  `Analyze > Expert Information`

3. 往返时间（RTT）增加

现象：数据包从发送到确认的时间明显变长。

- 分析路径：
  `Statistics > TCP Stream Graphs > Round Trip Time`

4. 拥塞窗口缩小（cwnd reduction）

- 观察字段：`tcp.window_size`
- 分析路径：
  `Statistics > TCP Stream Graphs > Window Scaling`

四、实验示例：

1.慢启动与拥塞避免

启动一个大型文件下载，观察初始阶段窗口指数增长（慢启动），随后转为线性增长（拥塞避免）。

2.快速恢复

人为制造丢包（如使用 tc 命令），观察 3 次重复 ACK 后触发的快速重传，而非超时重传。

Mininet 与 ns-3 仿真环境搭建教程

一、Mininet 安装与使用

1. 安装步骤（Ubuntu）

推荐使用预打包的 Mininet 虚拟机，避免依赖冲突，下载地址：

https://github.com/mininet/mininet/wiki/Mininet-VM-Images

使用 VMware 或 VirtualBox 导入镜像，默认用户/密码为 mininet。

或者原生安装（不建议）：

```
git clone https://github.com/mininet/mininet
cd mininet
./util/install.sh -a   # 安装所有依赖和组件
```

详细安装教程可参考：

http://mininet.org/walkthrough/

https://blog.csdn.net/weixin_42896572/article/details/109447236

2. 验证安装

```
sudo mn --test pingall   # 测试默认拓扑连通性
```

3. 连接 Mininet 到互联网（NAT 配置）

命令行方式：

```
sudo mn --nat   # 自动添加 NAT 网关
```

python 脚本（示例）：

```
from mininet.cli import CLI
from mininet.topolib import TreeNet
from mininet.log import setLogLevel

if __name__ == '__main__':
    setLogLevel('info')
    net = TreeNet(depth=1, fanout=2)
    net.addNAT().configDefault()  # 添加 NAT 功能
    net.start()
    CLI(net)
    net.stop()
```

二、ns-3 安装与配置

1. 安装依赖

安装必要工具链与库：

```
sudo apt-get install gcc g++ python python-dev mercurial bzr cmake libsq
lite3-dev libxml2-dev libgtk2.0-dev
```

完整依赖列表参考：

https://www.nsnam.org/wiki/Installation

2. 下载与编译

使用 Tarball(推荐):

```
wget https://www.nsnam.org/release/ns-allinone-3.xx.tar.bz2

tar xjf ns-allinone-3.xx.tar.bz2

cd ns-allinone-3.xx

./build.py  # 自动编译
```

使用 Bake 工具(开发版):

```
hg clone http://code.nsnam.org/bake

cd bake

./bake.py configure -e ns-3.xx

./bake.py download

./bake.py build
```

详细安装教程可参考：

https://www.nsnam.org/

https://icode.best/i/11753311742211

3. 验证安装

```
./waf --run hello-simulator  # 输出"Hello Simulator"即成功
```

三、实验示例：TCP 拥塞控制仿真

1. mininet 脚本（python 代码示例）

```
from mininet.net import Mininet
from mininet.topo import Topo
from mininet.link import TCLink
```

```
from mininet.log import setLogLevel
import time

class CongestionTopo(Topo):
    def build(self):
        s1 = self.addSwitch('s1')
        h1 = self.addHost('h1')
        h2 = self.addHost('h2')
        self.addLink(h1, s1, cls=TCLink, bw=10, delay='20ms', max_queue_size=100)
        self.addLink(h2, s1, cls=TCLink, bw=10, delay='20ms', max_queue_size=100)

def run_experiment():
    setLogLevel('info')
    net = Mininet(topo=CongestionTopo())
    net.start()

    h1, h2 = net.get('h1', 'h2')

    # 设置 TCP 算法，h1 使用 cubic，h2 使用 reno
    h1.cmd('sysctl -w net.ipv4.tcp_congestion_control=cubic')
    h2.cmd('sysctl -w net.ipv4.tcp_congestion_control=reno')

    # 启动抓包
    h1.cmd('tcpdump -i h1-eth0 -w h1_traffic.pcap &')

    # 启动 iperf 服务器和客户端
    h2.cmd('iperf -s &')
    time.sleep(1)
    h1.cmd(f'iperf -c {h2.IP()} -t 30 -i 1 > iperf_results.txt &')

    # 等待实验完成
    time.sleep(35)
    net.stop()

if __name__ == '__main__':
    run_experiment()
```

2. ns-3 脚本（C++代码）

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/applications-module.h"
#include "ns3/point-to-point-module.h"
```

```
using namespace ns3;

int main() {
    // 创建节点与链路
    NodeContainer nodes;
    nodes.Create(2);
    PointToPointHelper p2p;
    p2p.SetDeviceAttribute("DataRate", StringValue("10Mbps"));
    p2p.SetChannelAttribute("Delay", StringValue("10ms"));
    NetDeviceContainer devices = p2p.Install(nodes);

    // 配置 TCP 协议栈
    InternetStackHelper stack;
    stack.Install(nodes);
    Ipv4AddressHelper address;
    address.SetBase("10.1.1.0", "255.255.255.0");
    Ipv4InterfaceContainer interfaces = address.Assign(devices);

    // 启动 FTP 应用
    BulkSendHelper source("ns3::TcpSocketFactory", InetSocketAddress(interfaces.GetAddr
ess(1), 9));
    ApplicationContainer apps = source.Install(nodes.Get(0));
    apps.Start(Seconds(1.0));
    apps.Stop(Seconds(10.0));

    // 启用 PCAP 抓包
    p2p.EnablePcapAll("tcp-congestion");
    Simulator::Run();
    Simulator::Destroy();
    return 0;
}
```

3. 编译与运行

```
./waf --run tcp-congestion
```