说明

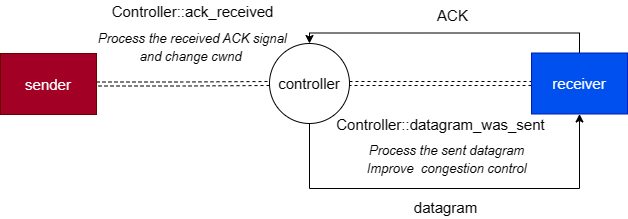
**Description**

建立拥塞控制算法，算法要求其在仿真蜂窝网络上运行时能最大限度地提升传输速度。网络模拟从简单的一个UDP发送者和接收者开始，也就是只有一个接收端和发送端，读写数据报文的大部分工作都已经由现有代码完成。

你主要要做的就是修改拥塞控制器的行为（修改controller.cc代码中的内容），控制发送方何时发送数据报。而一般情况下，发送的数据报越多，吞吐量越大，但产生拥塞的可能性越高。

**拥塞控制算法的运行示意图如下：**

// 框架加入mahimahi



**Procedure**

1. **学习拥塞控制算法**
2. **安装相应软件**

虚拟机或物理机Linux版本要求：Ubuntu GNU/Linux 14.04以上

//在其终端中运行如下代码：

//安装mahimahi

$sudo apt-get install build-essential git debhelper autotools-dev dh-autoreconf iptables protobuf-compiler libprotobuf-dev pkg-config libssl-dev dnsmasq-base ssl-cert libxcb-present-dev libcairo2-dev libpango1.0-dev iproute2 apache2-dev apache2-bin iptables dnsmasq-base gnuplot iproute2 apache2-api-20120211 libwww-perl （这些依赖项在[mahimahi / debian / control](https://github.com/keithw/mahimahi/blob/master/debian/control" \l "L5)中列出，另外还有一些用于比赛的依赖项）  
$git clone <https://github.com/ravinet/mahimahi> （Mahimahi工具提供了我们模拟的蜂窝网络和测量工具）  
$cd mahimahi  
$./autogen.sh && ./configure && make （用通常的方式编译mahimahi）  
$sudo make install （若要检查是否安装成功，可运行$mm\_delay 20，若出现delay 20ms字样，说明安装成功）  
$cd ..

//安装sourdough  
$git clone https://git.ruc.edu.cn/akth/cc-training   
$cd cc-training  
$./autogen.sh && ./configure && make （按常规方式编译代码）

1. **运行示例代码**

通过模拟VerizonLTE连接大约两分钟，运行完成时会生成日志。

$sudo sysctl -w net.ipv4.ip\_forward=1（必须启用Linux的IP转发才能使mahimahi工作）  
$cd datagrump

$./run-contest *[scheme\_name]*（scheme\_name就是你给自己文件命名的名称）

$mm-throughput-graph 500 ./contest\_uplink\_log（分析日志，输出吞吐等信息）

run-contest文件负责将测试文件通过模拟Verizon downlink生成日志，并使用mahimahi仿真，提供传输过程中的性能信息（如下），并显示queueing delay和throughput随时间变化的折线图。

Average capacity:平均容量

Average throughput:平均吞吐量

95th percentile per-packet queueing delay:95%排队延迟

95th percentile signal delay:95%信号延迟

//如果需要模拟不同的网络场景，可以执行以下命令:

//配置BBR

$ sysctl net.ipv4.tcp available congestion controlet.ipv4.tcp available congestion control = reno cubic  
$ sysctl net.core.default gdisc=fqsysctl: permission denied on key "net.core.default qdisc", ignoring

$ sudo sysctl net.core.default gdisc=fq  
$ sudo sysctl net.ipv4.tcp congestion control=bbret.ipv4.tcp congestion control = bbr

$ sysctl net.ipv4.tcp available congestion controlet.ipv4.tcp available congestion control = reno cubic bbr

$ sudo sysctl net.ipv4.tcp congestion controlet.ipv4.tcp congestion control = bbr

//切换回cubic

$ sudo sysctl net.ipv4.tcp congestioncontrolet.ipv4.tcp congestion control = bbr  
$ sudo sysctl net.core.default gdiscet.core.default qdisc = fq

$ sudo sysctl net.ipv4.tcp congestion control=cubicet.ipv4.tcp congestion control = cubic

$ sudo sysctl net.core.default gdisc=pfifo fastet.core.default gdisc = pfifo fast

**算法提交方法：通过TransHub平台**