竞赛指南

Transhub安装指南

本次竞赛最终结果需要提交在本网站“算法提交”处并查看排名，但为了方便参赛者修改和调试自己的拥塞控制算法代码，需要各位在本地配置Transhub运行环境，以下是安装指南(在第二节对Transhub代码框架进行了简单介绍)：

硬件环境：一台Linux主机（可以是实体机或虚拟机）

虚拟机或物理机Linux版本要求：Ubuntu GNU/Linux 14.04以上，建议是18.04版本，其他版本可能有不兼容问题。

1. **安装mahimahi**

//在其终端中运行如下代码：

$ `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 中列出，另外还有一些用于比赛的依赖项； 下载过程中如果太慢可以考虑替换 apt 源，可参考 https://zhuanlan.zhihu.com/p/61228593）

$ `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make工具编译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（测试完毕需要输入 exit 来退出 mm-delay 生成的 shell）。注意 mm-delay 运行前需要执行命令 sudo sysctl -w net.ipv4.ip_forward=1）

2. **安装Transhub本地代码**

$ `cd ..`

//安装

$ `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）

3. **运行测试**

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` （前提已经在cc-training目录下）

$ `./run-contest [scheme_name]` （scheme_name 就是你给自己文件命名的名称,测试阶段`scheme name`可以起名为 test）

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

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

95(th) percentile signal delay:95%信号延迟

$ `mm-throughput-graph 500 ./contest_uplink_log`（分析日志，输出吞吐等信息）

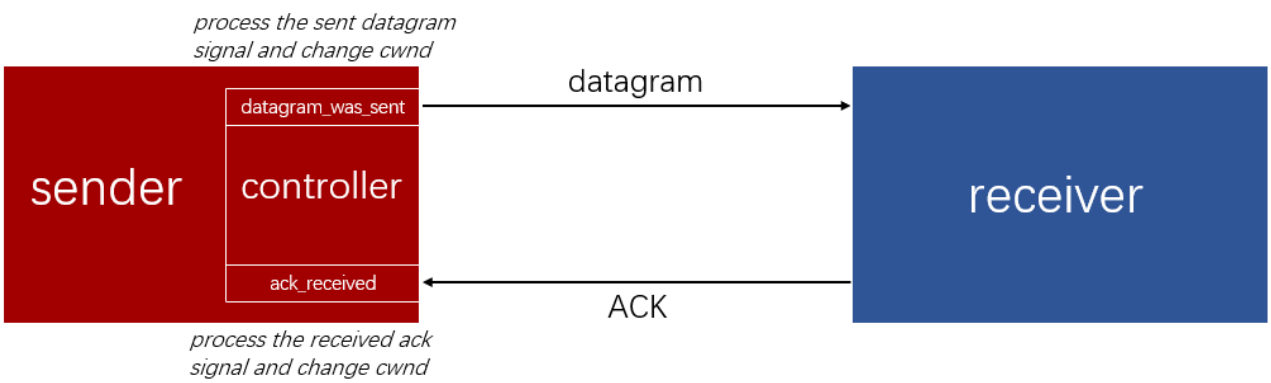
Transhub框架介绍

将Transhub代码克隆到本地后，大家需要关心/cc-training/datagrump目录下的内容，该目录下的内容与比赛息息相关，以下是对datagrump目录下各文件的简单介绍：

- contest_message文件：规定数据报文的格式。
  在通信之前，我们需要事先在通信双方约定一个通信语法，也就是对应着一个报文格式标准，用来指定传输过程中数据的组织形式。报文格式标准没有固定的要求，唯一的要求就是发送端和接收端都接受相同的标准。
  数据报文（Packet）总体来说分为两个部分：报文头部和包内数据。报文头部主要包含描述报文特性的一些元数据，例如报文序列号、报文的发送时间等等。比赛测试场景是固定发送端和接收端，因此报文头元数据部分更加简单，具体有以下一些字段：
  1. `sequence_number`：数据报文的序列号
  2. `send_timestamp`：数据报文的发送时间
  3. `ack_sequence_number`：确认收到的数据报文序列号
  4. `ack_send_timestamp`：确认报文(ACK)的发送时间
  5. `ack_recv_timestamp`：确认报文(ACK)的接收时间
  6. `ack_payload_length`：数据报文的有效载荷大小

- sender.cc文件：模拟发送端的行为。
  发送端的行为遵循两个准则，一是如果发送端还有发送窗口空余，发送端就发送更多的数据报文来填满发送窗口；二是如果发送端收到了ACK，处理该确认报文并通知controller，进行拥塞控制。此外，如果超过了一定的时间(超时时间)发送端还没收到ACK，发送端会认为这个报文丢失了，会进行重传。具体细节请参看sender.cc文件。

- receiver.cc文件：模拟接收端的行为。
  接收端会为收到的每个数据报文发送ACK，`ack_sequence_number`就是数据报文的序列号`sequence_number`，具体细节请参看receiver.cc文件和contest_message.cc文件中的`transform_into_ack`函数。
- controller.cc文件：模拟进行拥塞控制的文件，也是参赛者们需要修改的文件。
  有以下四个接口函数：
  1. `window_size()`：返回给发送端当前的发送窗口大小。
  2. `datagram_was_sent()`：发送端发送数据报文时调用controller中的该函数，参数中有`const bool after_timeout`标志位，标记该报文是否是因为超时发送的。
  3. `ack_received()`：发送端收到ACK时会调用controller中的该函数，根据ACK报文中的信息，如确认序列号、数据报文的发送时间戳、ACK的接收时间戳等信息，计算往返时延等变量，进行拥塞控制。
  4. `timeout_ms()`：设置超时时间，返回给发送端。
  拥塞控制示意图如图一所示。



图一 拥塞控制示意图

- run-contest文件：测试脚本。

Transhub评分标准

算法评分主要衡量两个指标——网络的平均吞吐量和排队时延。

平均吞吐量(average_throughput)：单位时间内成功传输的数据量。

95分位单向时延(95th percentile one-way delay)：统计每个数据包在网络中传输的单向时延，取95分位的值作为指标。

得分公式定义如下：

$$score = \ln\left(\frac{\text{average\_throughput}}{\text{95th percentile one-way delay}}\right)$$

一共有两套测试环境，分别代表较好的网络环境和较差的网络环境，最后的总得分将取两套测试环境得分的平均值。