数据库/网络安全方向课程设计大作业答辩

邢元勋　　李欣宜

大连理工大学软件学院

https://github.com/Lixinyi-DUT/sctp_simulation

2015 年 7 月 22 日

概览

主要工作

- 了解 SCTP 协议工作机制
- 了解基于 SCTP 多宿主特性的 CMT 传输机制
- 寻找 SCTP 在无线网络中的应用
- 提出可能的改进方案
- 针对几种典型的 SCTP 有线网络结构进行模拟得到
  - 丢包率
  - 吞吐量
  - 端到端时延
  - 拥塞控制窗口大小

  等参数，并进行分析

SCTP协议 [1]

流传输控制协议：Stream Control Transmission Protocol

- 与应用广泛的 TCP 、 UDP 协议相似，SCTP 作为一种传输层协议在计算机网络中使用。
- 基于基于不可靠传输业务的协议之上的可靠的数据报传输协议
- 面向消息(message-oriented): 以消息（比特组）的形式传输数据，将数据和控制信息划分为组块 (chunk)
- 多流 (multi-streaming)：并行传送多个独立的数据流
- 在一个数据流中，给每个消息指定序列号，以保证不同流的消息是有序的，也引入了拥塞控制

SCTP协议特性

- 多宿主 (multi-homing) 支持：一个连接的端点可以拥有不止一个 IP 地址，缓解一条路径故障带来的威胁
- 消除了头阻塞 (head-of-line blocking)
- 路径选择，主传输路径选择，测试传输路径的连通性
- 引入有效性和确认机制对抗洪泛攻击（四次握手），提供数据块重复和丢失通知
- 使用 SACK 进行拥塞控制

多宿主支持

增加冗余 IP 地址，提高可达性

节点使用心跳 (heartbeat) 检查对方的主地址和冗余地址的可达性，当收到 heartbeat 时需要确认回复 ACK

发送消息时，主 IP 由主机（而不是 SCTP）决定

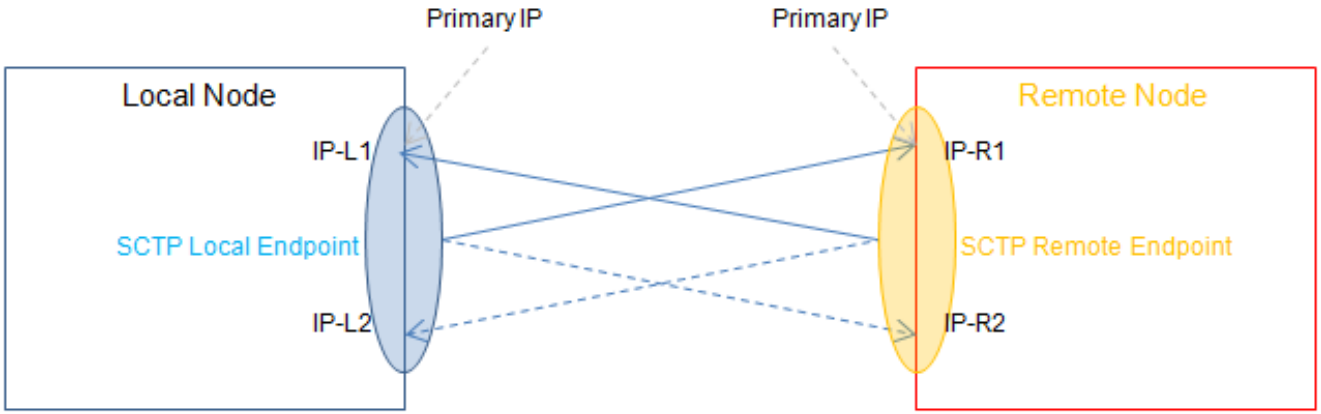


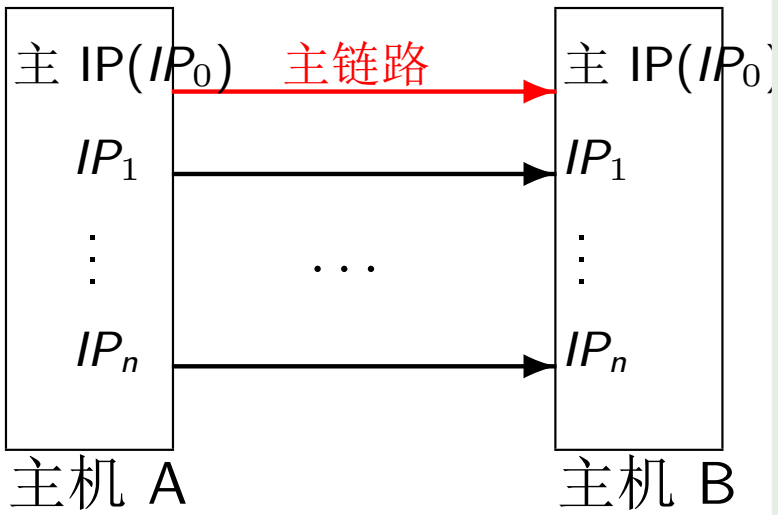
Figure: 对称多宿主

CMT:并行多路传输

Concurrent Multipath Transfer

能被同时分配多个 IP 的主机是多宿主的

利用 SCTP 的多宿主特性，使用多个端到端路径同时传送数据 [2]



- 用于提高吞吐量
- 瓶颈队列互相独立
- 使用 SFR-CACC 算法消除不必要的快速重传
- CUC 算法解决拥塞窗口更新太少的问题
- DAC 算法解决 ack 流量增加问题

无线 SCTP

mSCTP: Mobile SCTP

- ADDIP 扩展：满足动态建立连接要求，移动不再受到限制
- 移动节点和通信节点都支持动态地址配置扩展

cSCTP: Celluar SCTP

在封包内引入变量 handoff_mode，切换效果更佳。

Layer3 Host-Agent: 与接入路由 (AR) 通信

Layer4 cSCTP:SCTP 增加动态地址设置扩展和切换流程

Layer5 应用层 SIP User Agent

无线多跳网络 [3]

- IEEE 802.11(DCF) CSMA/CA

优化方案

无线多跳网络环境下 SCTP 协议的吞吐量表现

- 采用 OPNET 模拟，通过改变接收方拥塞窗口的大小，源和目的地之间的跳数来分析实验结果，吞吐量随着跳数的增加而降低。
- 在跳数较少的情况下，需在博监听对吞吐量的增加帮助不大，丢包会导致 SCTP 超时，降低工作效率。
- 如果丢包的原因不是因为网络拥塞和 SCTP 窗口过小，就会很不公平。这种情况被称为“SMALL WINDOW SYNDROME”。

解决该问题的办法是在传输的空闲阶段 (idle period) 传输丢失的数据报。

优化方案

多路径并行传输中的吞吐量改进算法

- 多路径并行传输是一种有线情况下的传输过程，分为理想和非理想状态
- 理想状态下不考虑 $throughput_r = ck_r / RTT_r$
  - 即吞吐量等于 r 轮传输成功的数据包总和除以 r 轮中往返的时间的总和
- 非理想状态下要考虑超时重传。
- 一般来说，一个路径发生故障时，会导致数据报传输失败，当重传次数超过检测阈值（PMR）时，路径会变为 Failed 状态。

优化方案

多路径并行传输中的吞吐量改进算法

CMT-PF 算法解决了上述问题：

- 当超时传输在传输路径上个产生时，则认为其进入 PF（潜在失败）状态，并终止数据包的传输
- 进入 PF 状态的路径会发送 HeartBeat 包，若 HB 超过 PMR 值，则彻底进入 Failed 状态
- 否则路径仍为 Active 状态

SCTP vegas 算法

- 以 RTT 的变化来判断网络拥塞的发生
- 由 TCP vegas 算法扩展而来

发送端每收到一个数据报的 ACK 时，就会获得一个新的 RTT 样本，利用平均 RTT 作为判别标准，对 RTT 进行更新。

优化方案

慢启动方法的改进

- 当发送速率接近网络带宽时，逐渐减小拥塞窗口的增长速率，使其平滑的逼近网络带宽
- $cwnd < ssthreshold/2$ 时，与慢启动的原理一致
- $cwnd > ssthreshold/2$ 且 $ssthreshold - cwnd > 1$ 时，每个 RTT 内拥塞窗口增大
- $ssthreshold - cwnd < 1$ 时，进入拥塞避免

优化方案

SCTP拥塞控制机制的研究与改进

$$cwnd(t+T) = \begin{cases} 2 * cwnd(t) & \text{当} cwnd < ssthresh/2 \\ cwnd + Rtt(Rtt + MaxRtt) * (ssthresh - cwnd(t)) & \\ \text{当} cwnd \geq ssthresh/2 \text{且} ssthresh - cwnd > 1 & \\ ssthresh & \text{其他} \end{cases}$$

实验环境/工具

实验环境

- Windows 8.1
- Cygwin (Easy 2007.3.21)
- ns-2.34

实验工具

tcl/otcl	脚本编写模拟测试
gawk	读写/管理测试结果数据
R/ggplot2	计算/分析/作图

结果参数

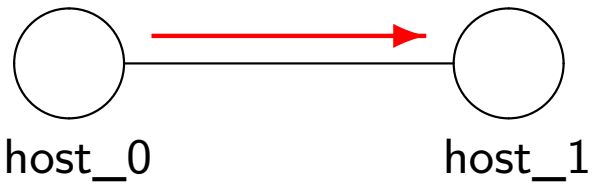
丢包率 丢失数据包占所发数据组的比率

吞吐量 单位时间内，某个节点发送和接受的数据量

端到端时延 数据包从离开源点时算起一直到抵达终点时一共经历了的时间，包括打包和解包的时延，和网络传输时延

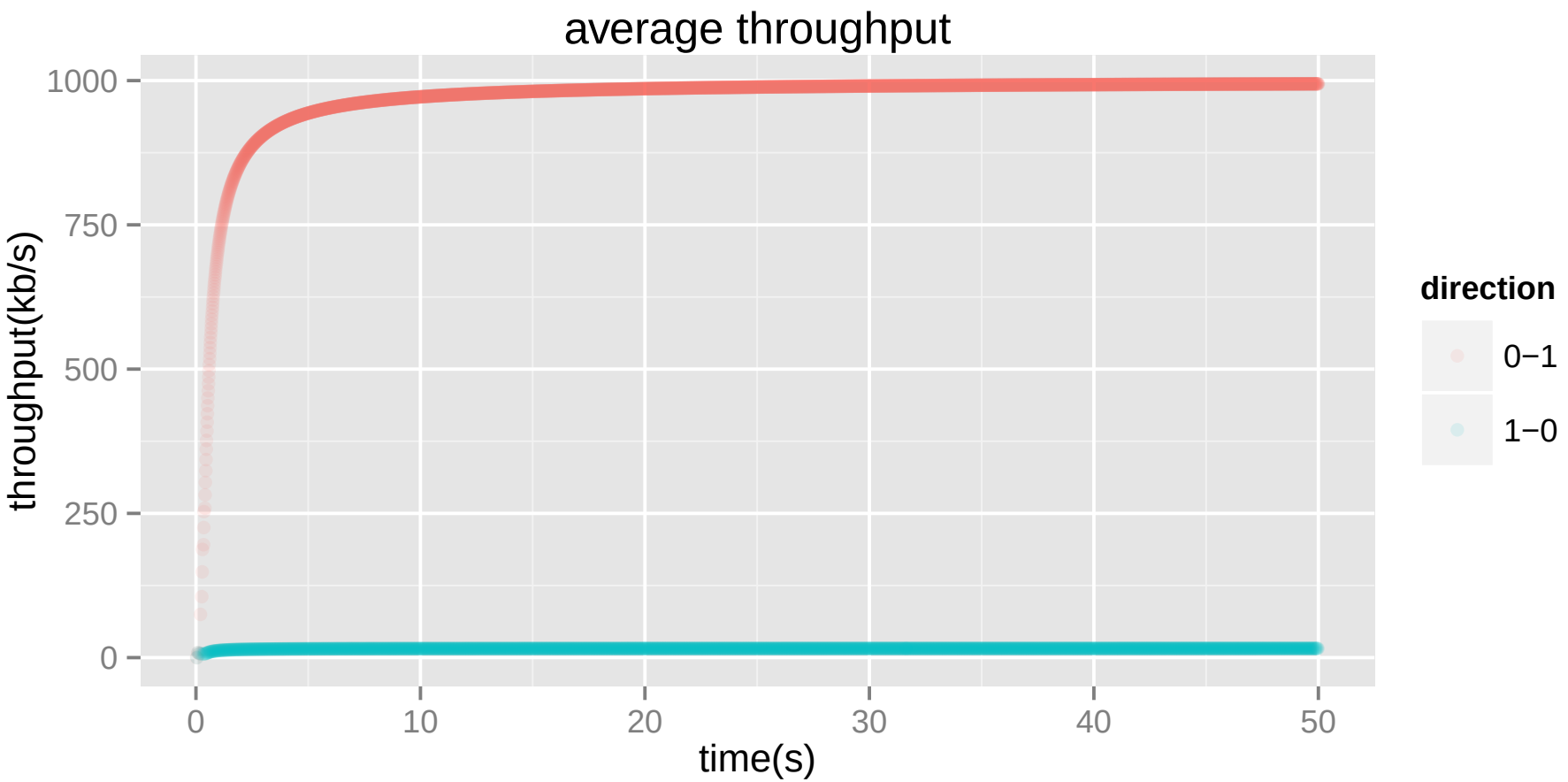
cwnd 拥塞窗口，大小取决于网络的拥塞程度，且动态变化

实验 1

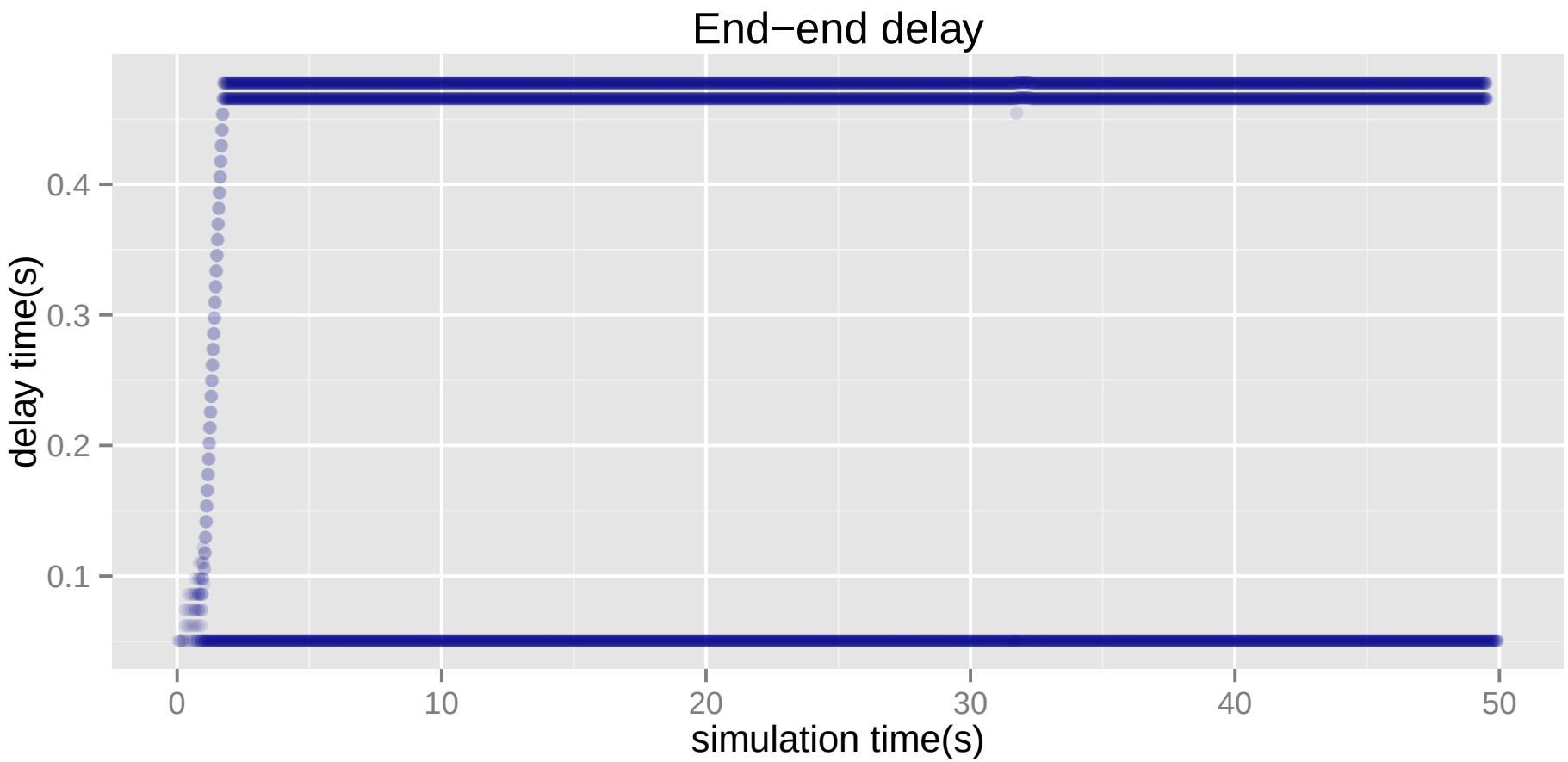


参数	值
传输层	SCTP
应用层	FTP
带宽	1Mb
时延	50ms
丢包率	0.01
MTU	1500
数据组块大小	1468

实验 1：吞吐量

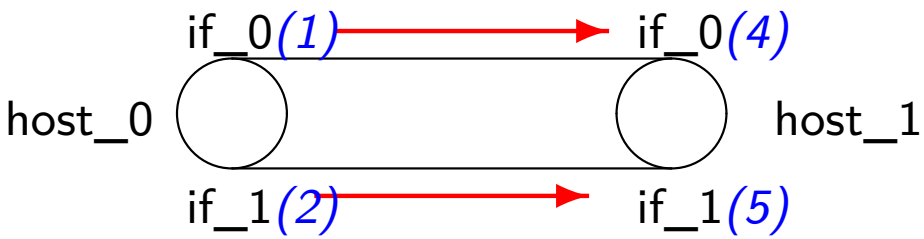

average throughput
throughput(kb/s)
1000
750
500
250
0
0
10
20
30
40
50
time(s)
direction
0−1
1−0

实验 1：时延


End−end delay
delay time(s)
0.4
0.3
0.2
0.1
0
10
20
30
40
50
simulation time(s)

实验 2

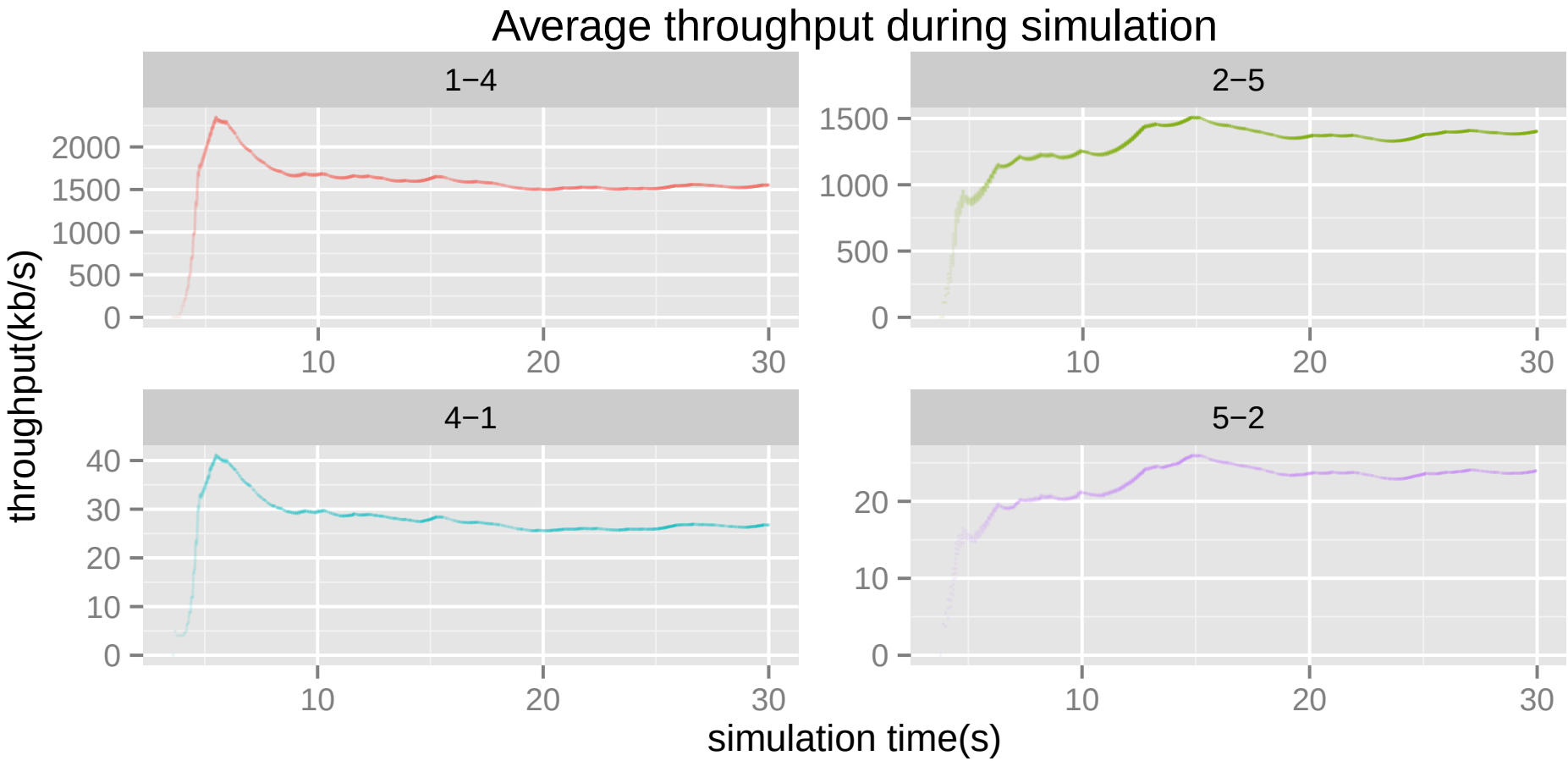
CMT模拟，使用 ReOrdering、CUC、AUC算法，引入 PF机制



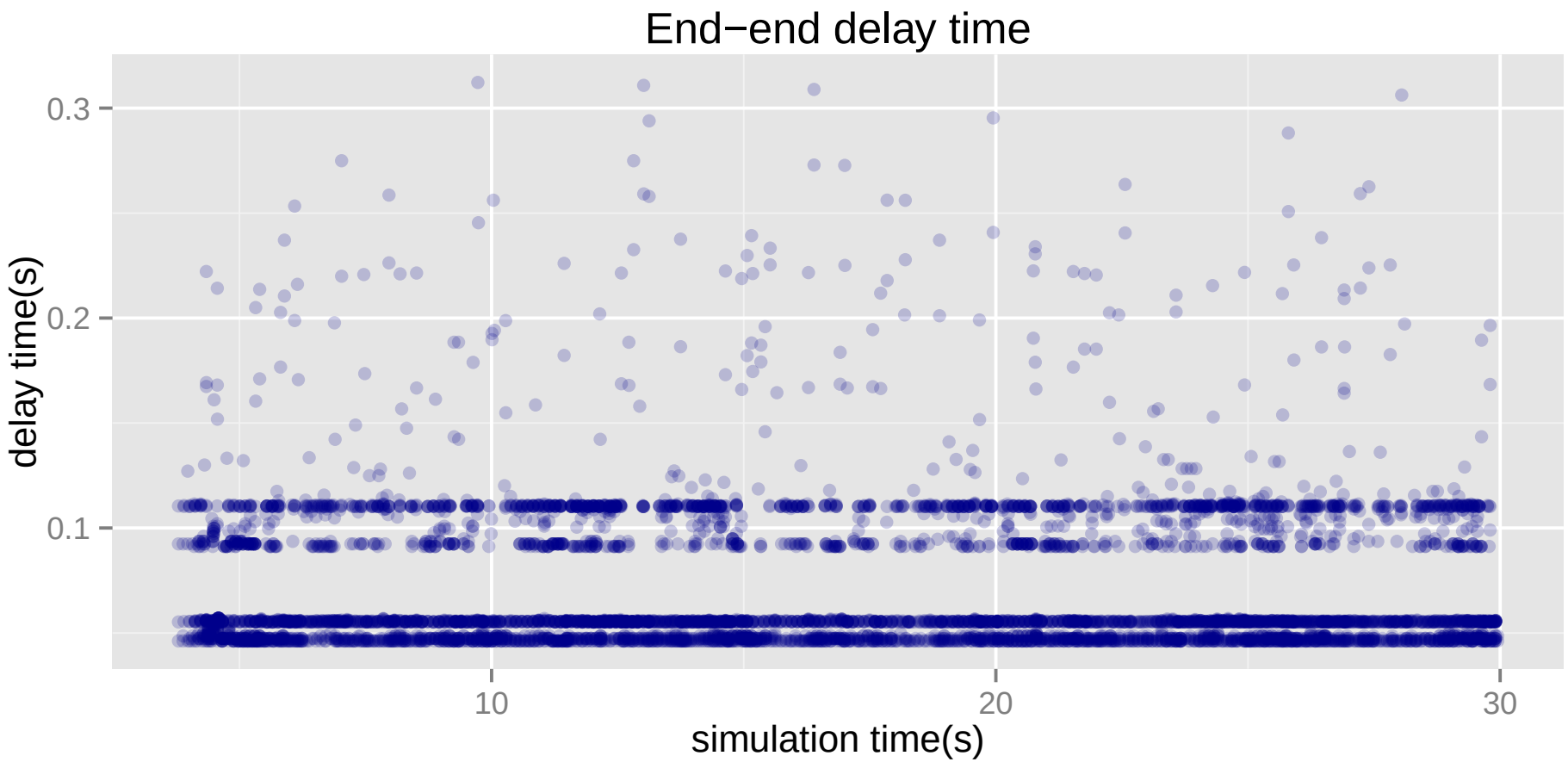
* 收到 *HB-ACK* 后拥塞窗口大小更新为原来的 *2* 倍

参数	值
应用层	FTP
丢包率	0.01
主地址	if_0(4)
带宽 (1-4)	10Mb
时延 (1-4)	45ms
带宽 (2-5)	50Mb
时延 (2-5)	55ms
MTU	1500
数据组块大小	1468
**cmtPFCwnd*	2

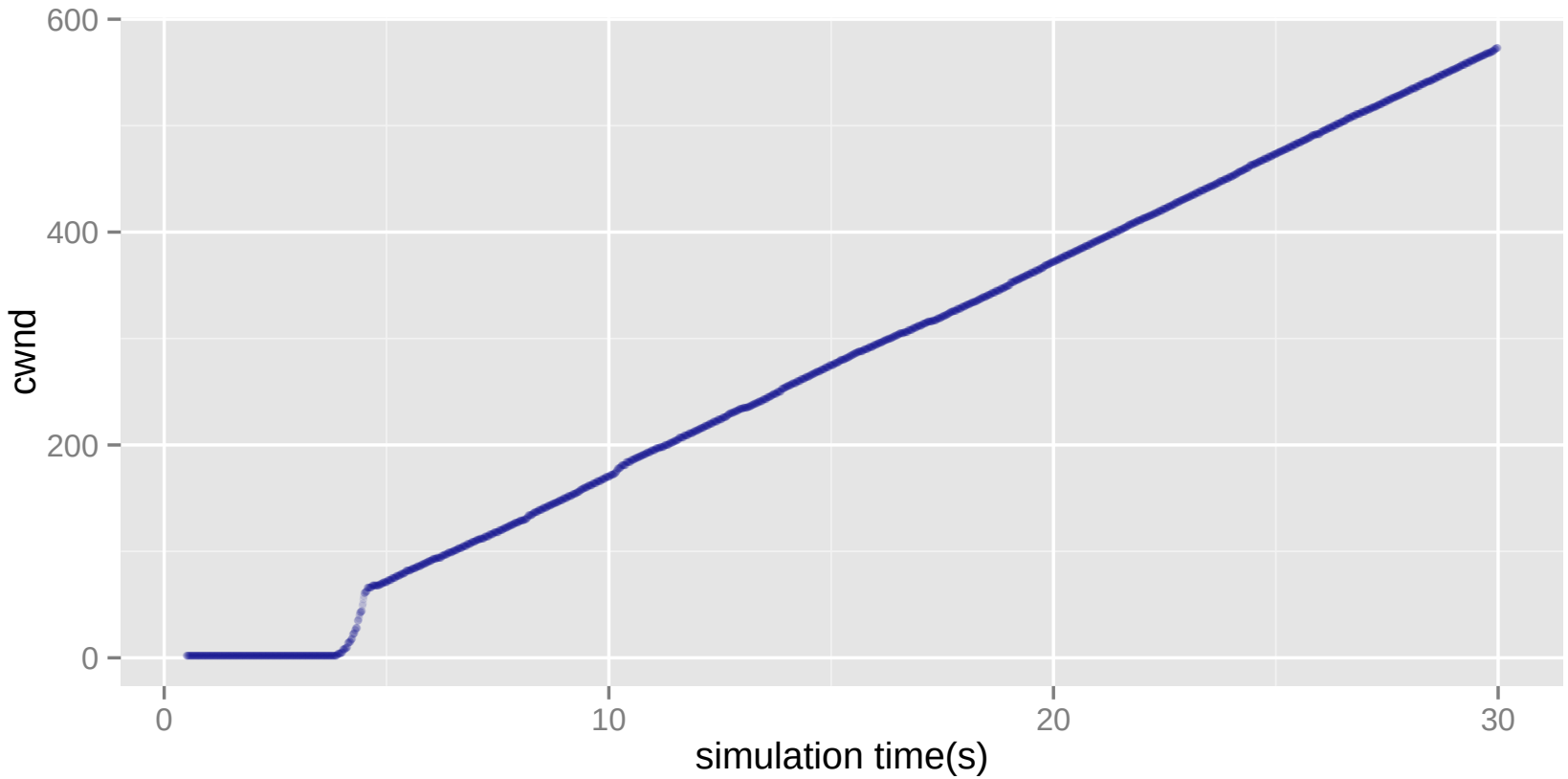
实验 2: 吞吐量


Average throughput during simulation
1−4
2−5
4−1
5−2
throughput(kb/s)
simulation time(s)
0
500
1000
1500
2000
10
20
30
40

实验 2: 时延

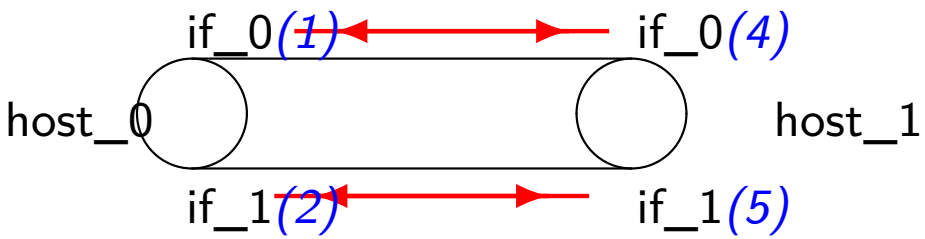

End−end delay time
delay time(s)
0.3
0.2
0.1
10
20
30
simulation time(s)

实验 2: cwnd



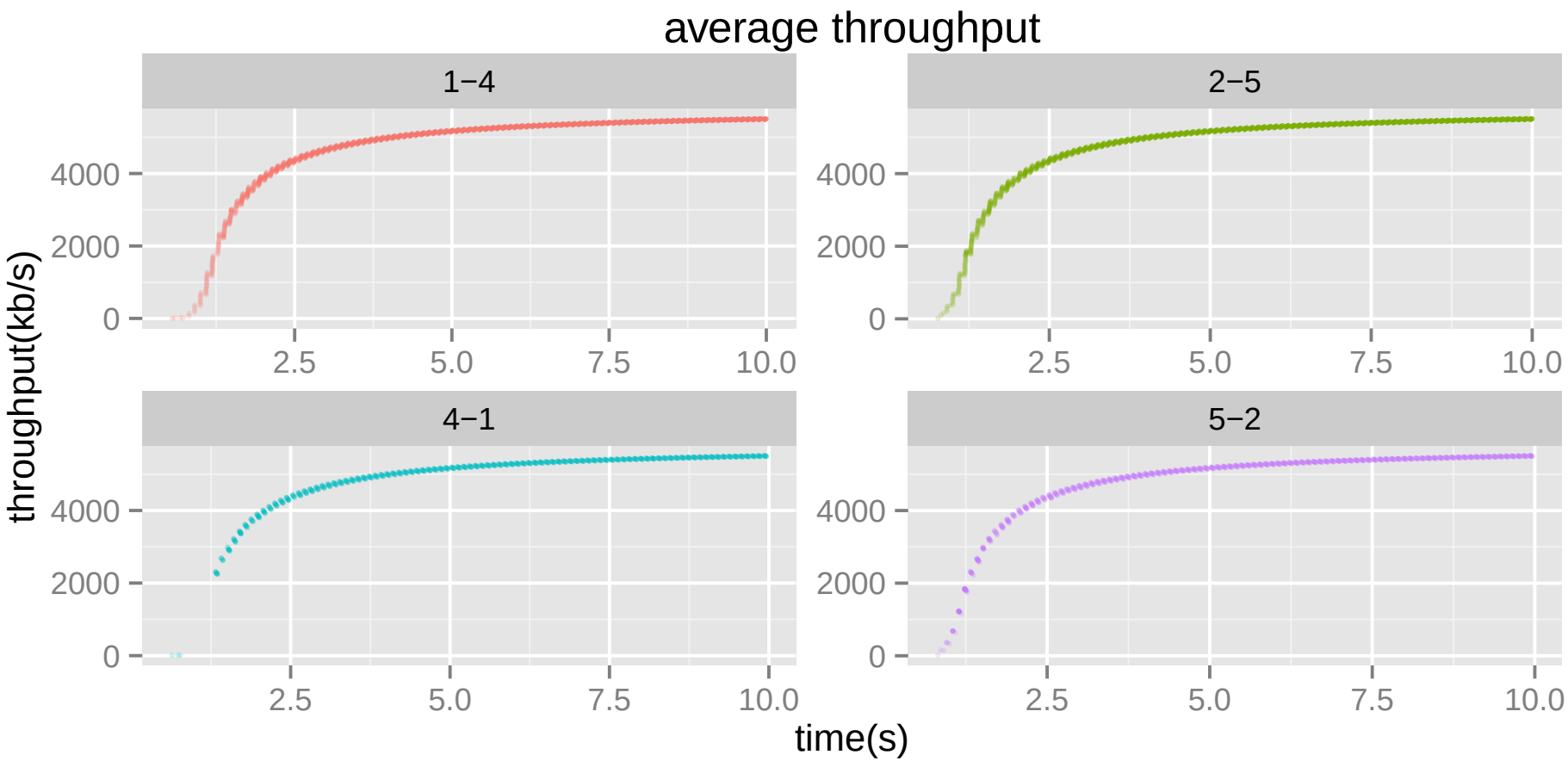
实验 3

CMT双向平行链路同带宽

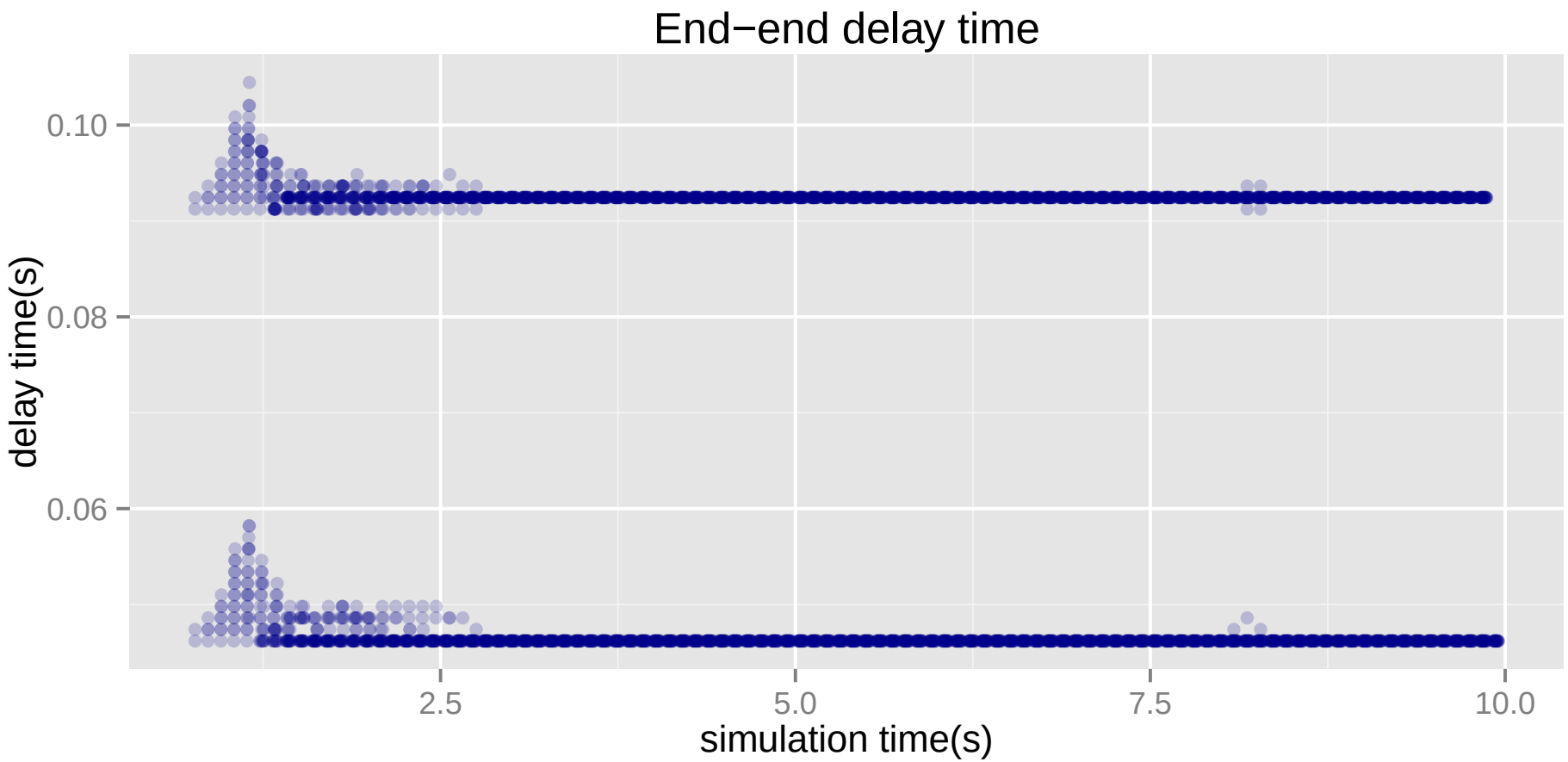


参数	值
应用层	FTP
主地址	if_0(4)
带宽 (1-4)	10Mb
时延 (1-4)	45ms
带宽 (2-5)	10Mb
时延 (2-5)	45ms
MTU	1500
数据组块大小	1468

实验 3：吞吐量

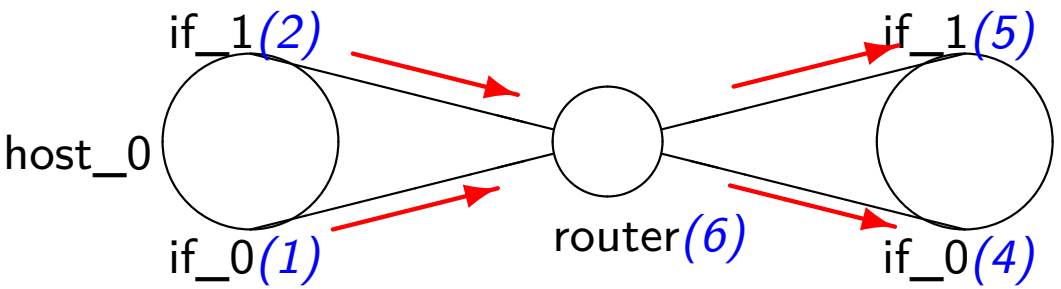

average throughput
1−4
2−5
4−1
5−2
throughput(kb/s)
time(s)
0
2000
4000
2.5
5.0
7.5
10.0

实验 3：时延


End−end delay time
delay time(s)
0.10
0.08
0.06
2.5
5.0
7.5
10.0
simulation time(s)

实验 4

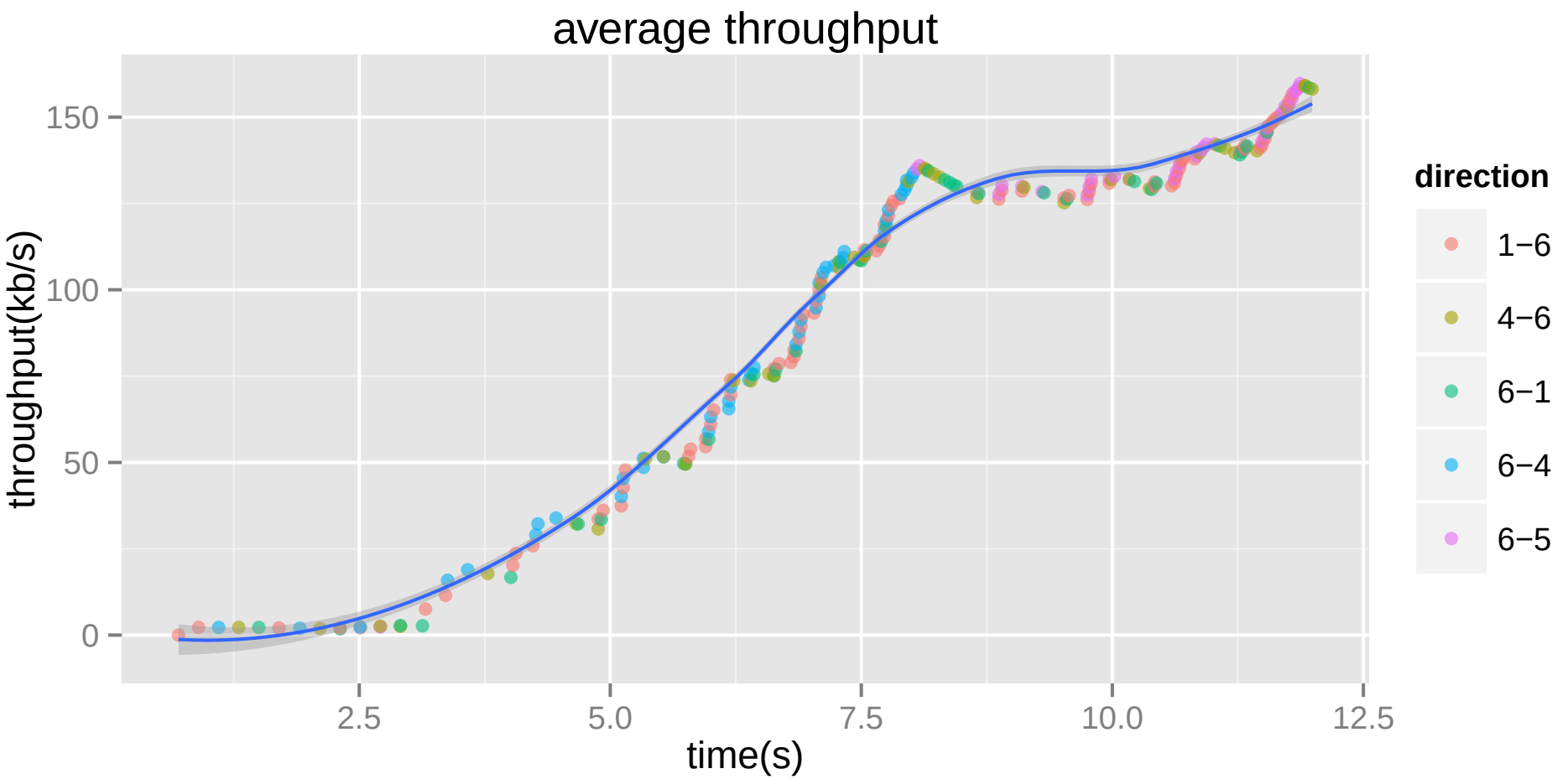
通过路由连接，使用心跳，中途更换主地址



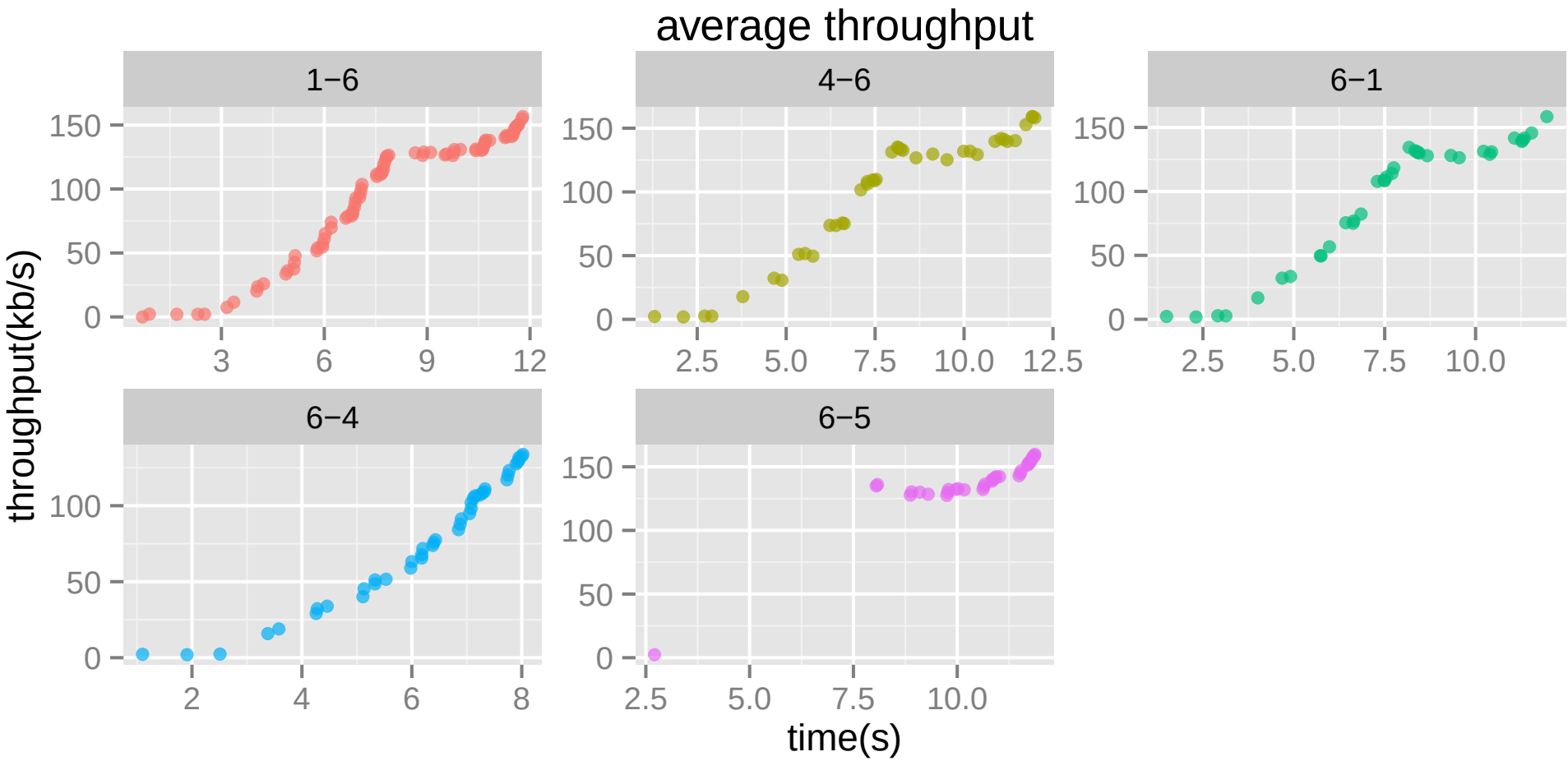
* 主地址在第 7.5s 时更换为 if_1(5)

参数	值
应用层	FTP
* 主地址	if_0(4)
带宽	0.5Mb
时延	200ms
MTU	1500
数据组块大小	1468

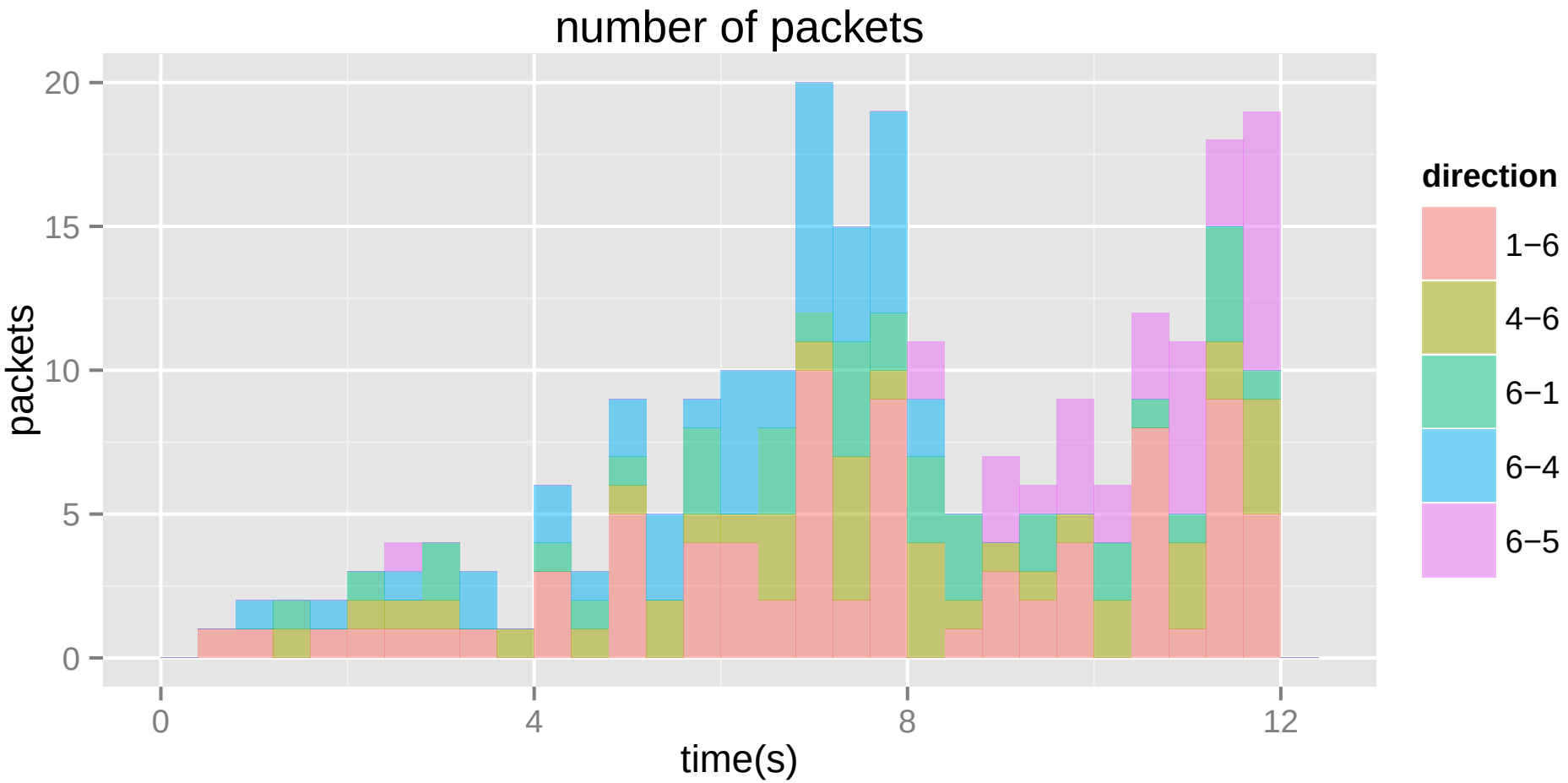
实验 4：吞吐量


average throughput
throughput(kb/s)
150
100
50
0
2.5
5.0
7.5
10.0
12.5
time(s)
direction
1−6
4−6
6−1
6−4
6−5

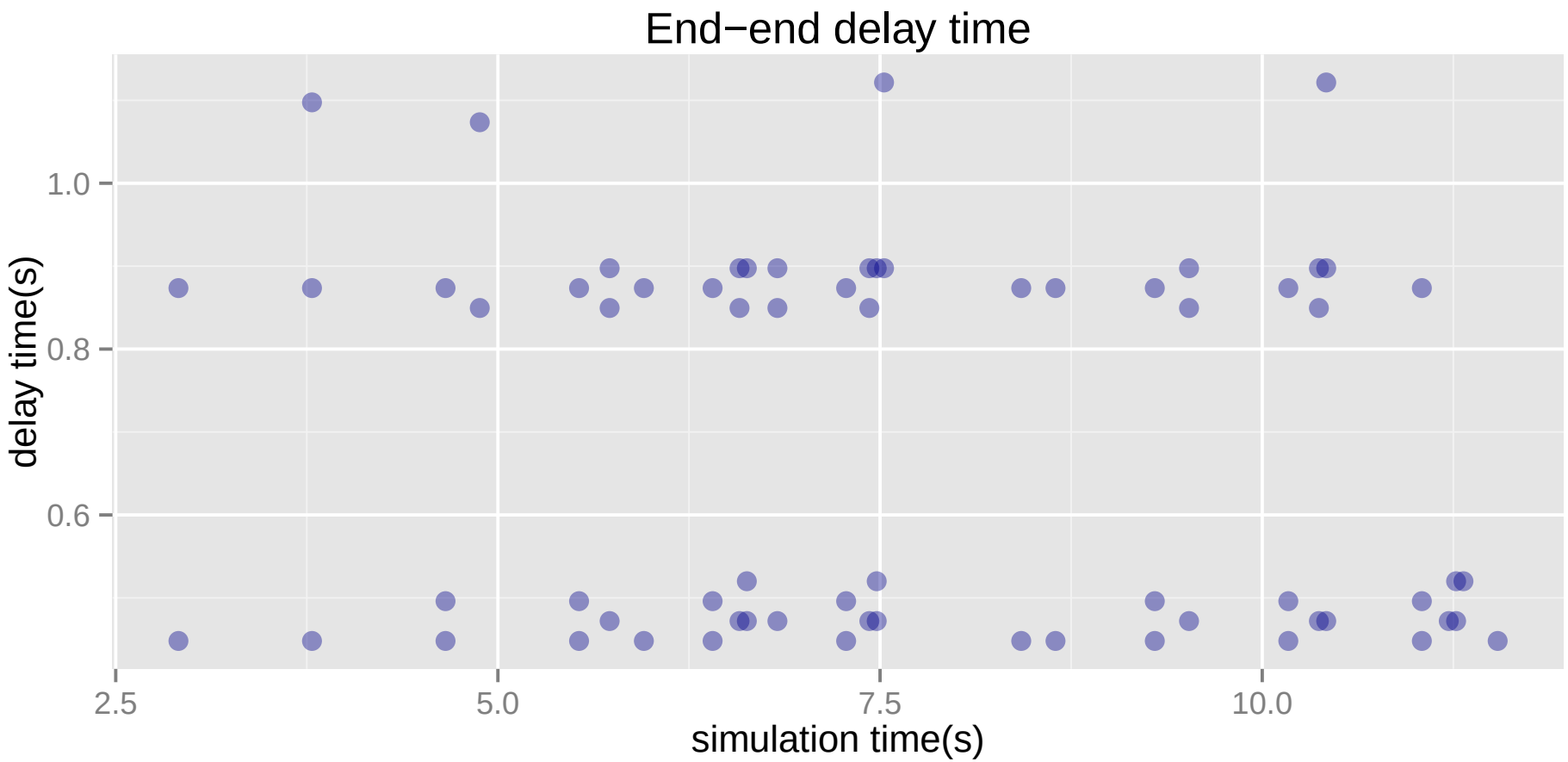
实验 4：吞吐量


average throughput
1−6
4−6
6−1
6−4
6−5
throughput(kb/s)
time(s)
150
100
50
0
3
6
9
12
2.5
5.0
7.5
10.0
12.5
2
4
8

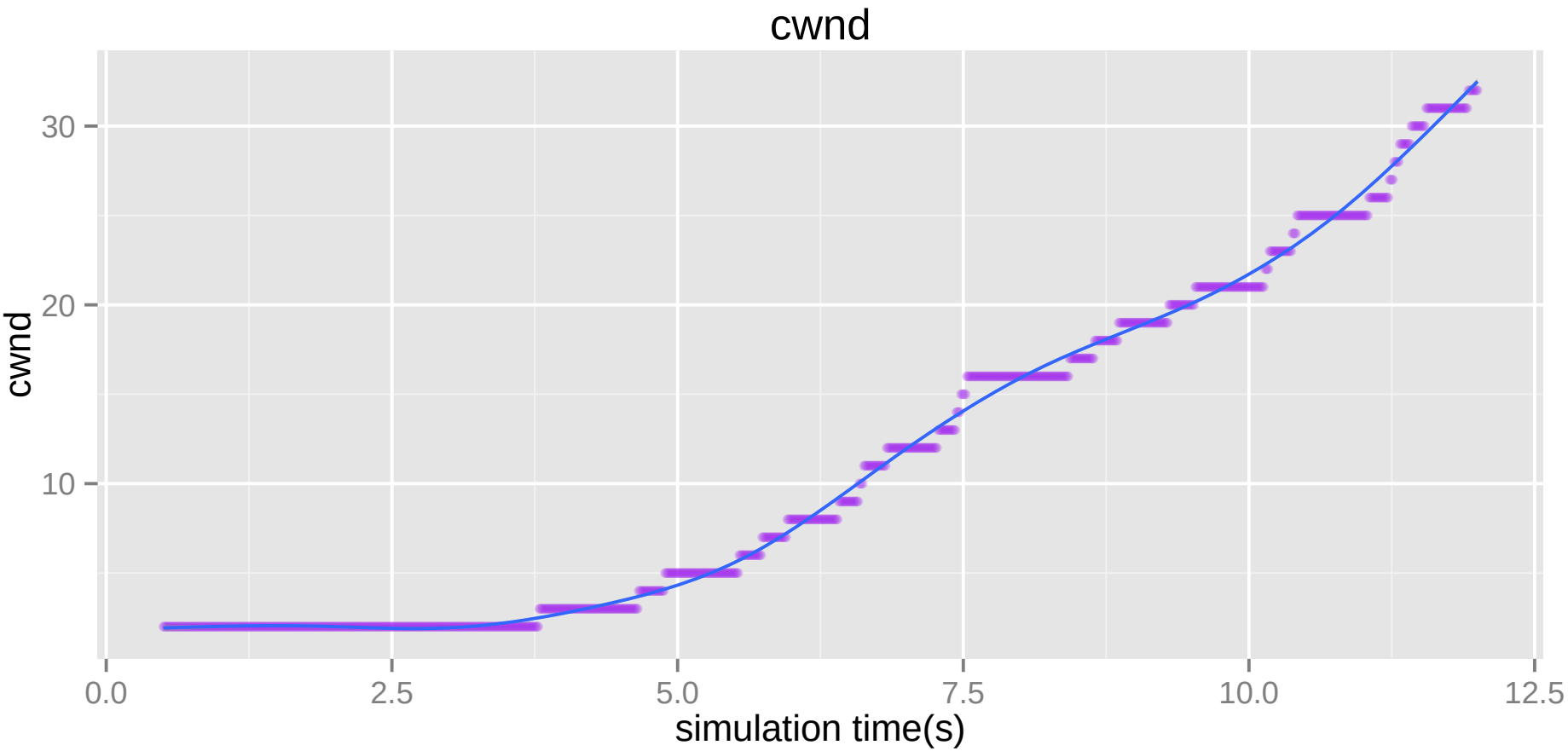
实验 4：封包


number of packets
packets
20
15
10
5
0
0
4
8
12
time(s)
direction
1−6
4−6
6−1
6−4
6−5

实验 4：时延


End−end delay time
delay time(s)
1.0
0.8
0.6
2.5
5.0
7.5
10.0
simulation time(s)

实验 4：cwnd



参考文献

- R. Stewart *et al.*, "Stream control transmission protocol. proposed standard, rfc2960, internet engineering task force (ietf), october 2000."
- J. anardhan Iyengar, "Concurrent multipath transfer using sctp multihoming," *Multihomed Communication with SCTP (Stream Control Transmission Protocol)*, p. 99, 2012.
- I. Aydin and C.-C. Shen, "Performance evaluation of concurrent multipath transfer using sctp multihoming in multihop wireless networks," in *Network Computing and Applications, 2009. NCA 2009. Eighth IEEE International Symposium on*, pp. 234–241, IEEE, 2009.

The End