UDT: A High Performance Data Transport Protocol
Высокоэффективный протокол передачи данных

Yunhong GU
gu@lac.uic.edu

Национальный центр поиска данных
Университет Иллинойса и Чикаго
10 октября, 2005 г.
Обновлено 8 августа 2009 г.

ВСТУПЛЕНИЕ

ПРОЕКТ ПРОТОКОЛА И ЕГО РЕАЛИЗАЦИЯ

УПРАВЛЕНИЕ ПЕРЕГРУЗКОЙ

ОЦЕНКА ЭФФЕКТИВНОСТИ

СОСТАВНОЙ UDT

ЗАКЛЮЧЕНИЕ

Обоснования

- Широкое распространение высокоскоростных сетей (1Gb/s, 10Gb/s, и т. д.) позволило создавать приложения с распределенными данными
  - Недорогие волокна и продвинутые оптические сетевые технологии (например DWDM - Dense Wavelength Division Multiplexing, технология плотного спектрального уплотнения)
  - 10Gb/s сейчас является обычной скоростью на испытательных площадках, уже появляются 40Gb/s

- Наборы данных большого объема
  - Метеоданные со спутников
  - Астрономические наблюдения
  - Сетевой мониторинг

- Необходим новый протокол передачи!

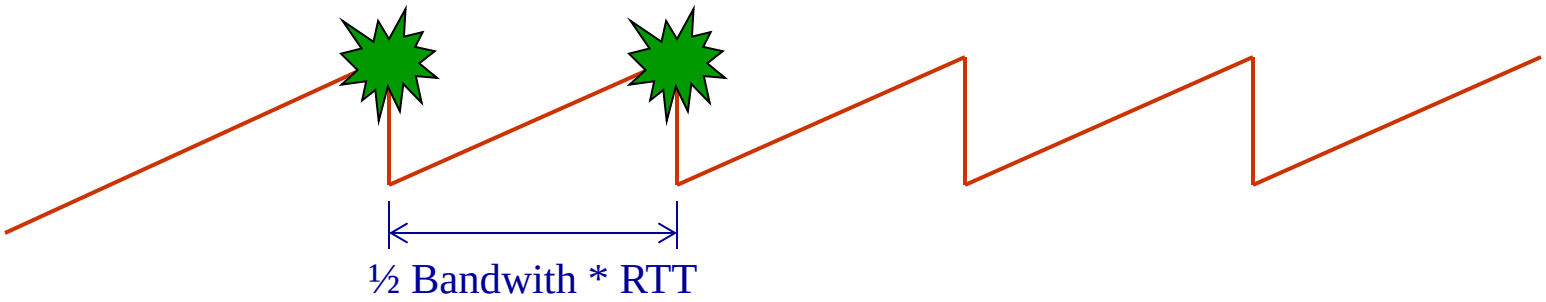
Протокол Передачи Данных

- Функциональные возможности
  - Потоки, обмен сообщениями
  - Надежность
  - Своевременность
  - Unicast vs. multicast
- Управление перегрузкой
  - Эффективность
  - Честность
  - Сходимость
  - Распределенность

TCP

- Надежный, поток данных, unicast

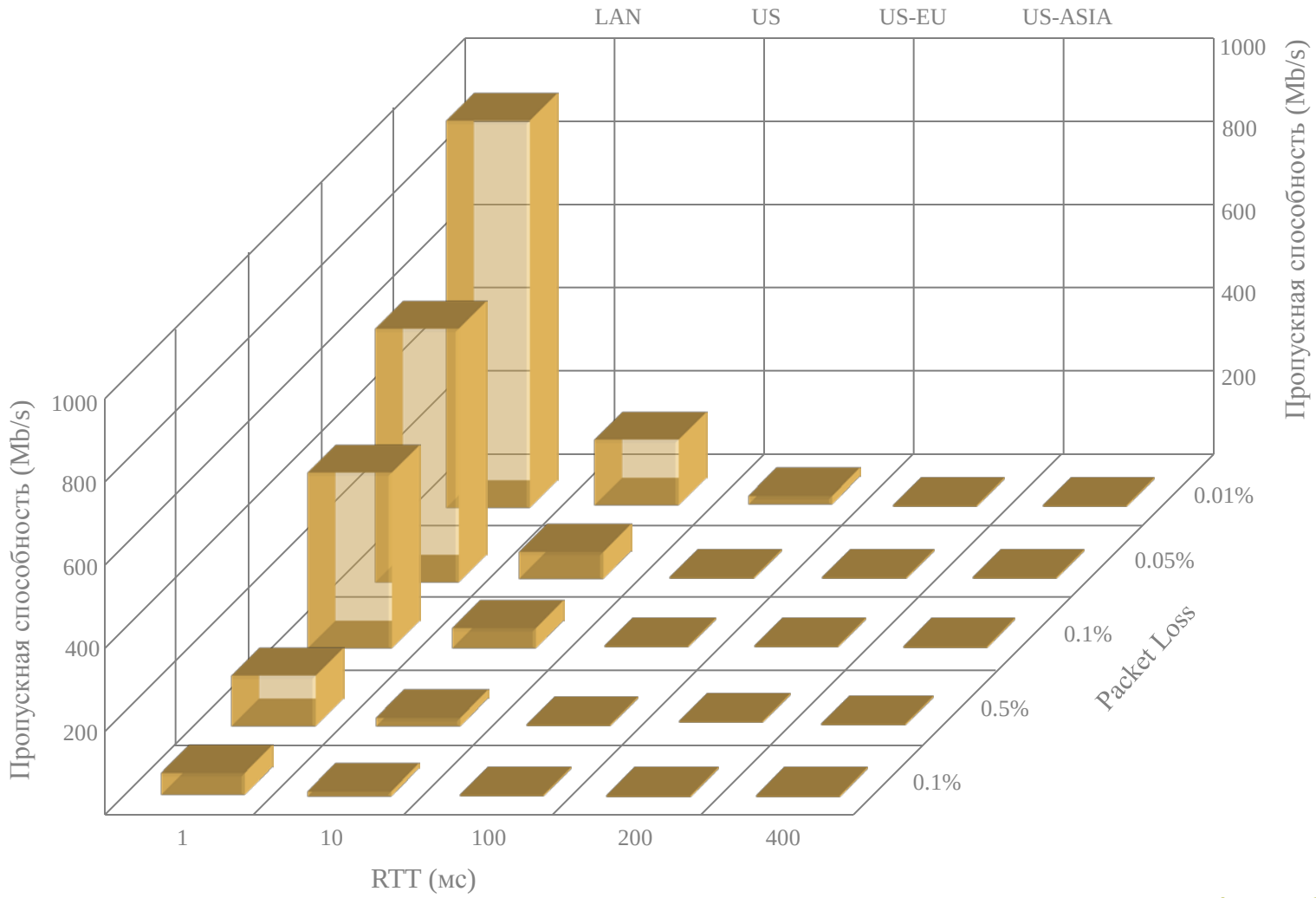
- Управление перегрузкой AIMD (*Additive Increase Multiplicative Decrease - аддитивное увеличение, мультипликативное уменьшение*)
  - Увеличение размера окна перегрузки (*cwnd*) на один полноразмерный пакет за RTT
  - Уменьшение вдвое *cwnd* за каждое loss event



- Слабо эффективен в сетях с высоким BDP (*bandwidth-delay product – произведение полосы пропускания на задержку*)

- Плохо влияет на потоки с большим RTT

TCP


LAN
US
US-EU
US-ASIA
Пропускная способность (Mb/s)
1000
800
600
400
200
Пропускная способность (Mb/s)
1000
800
600
400
200
0.01%
0.05%
0.1%
0.5%
0.1%
Packet Loss
1
10
100
200
400
RTT (мс)

Родственные решения

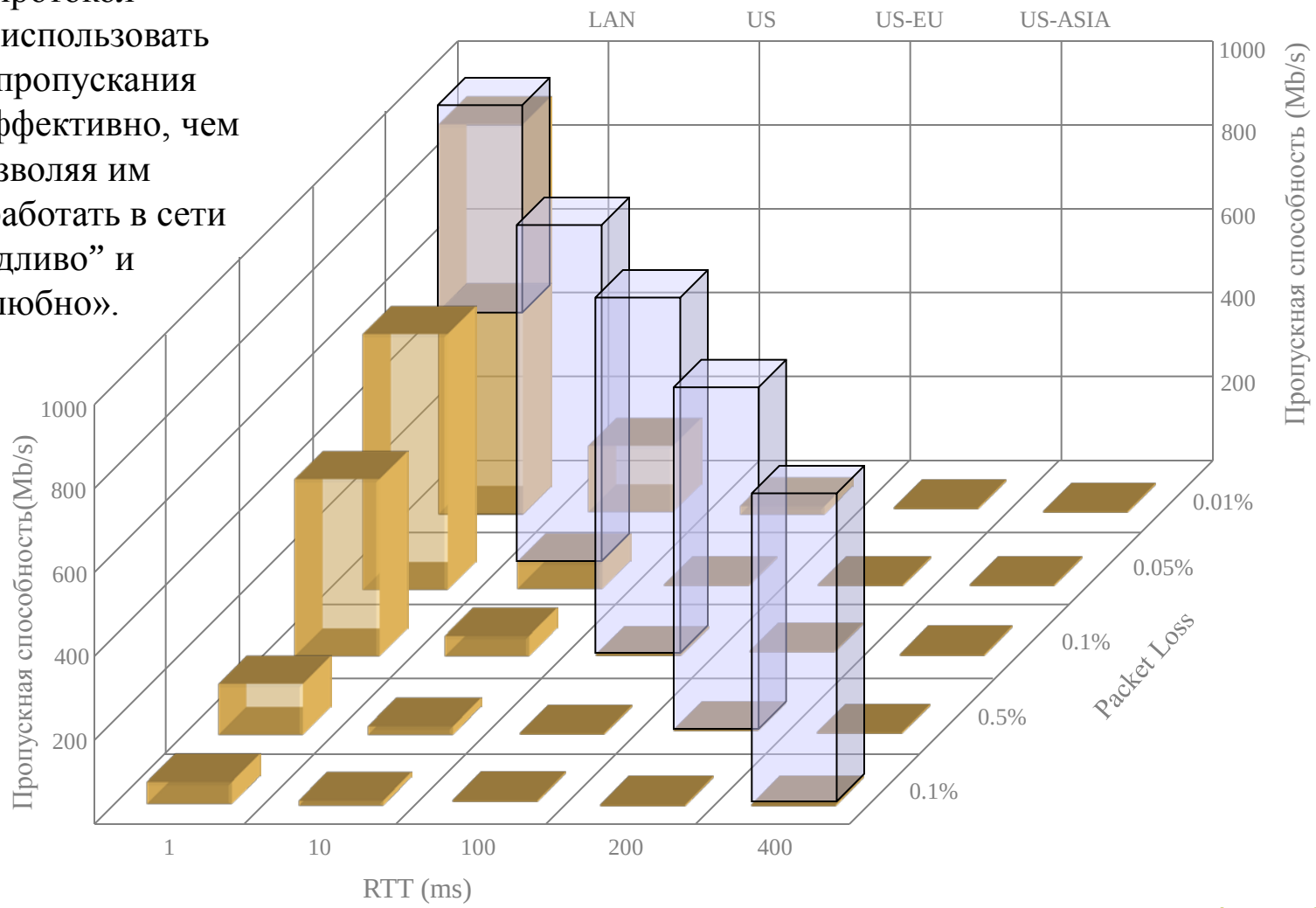
- TCP варианты (*модификации исходного алгоритма AIMD, большие параметры увеличения и меньшие факторы уменьшения*)
  - HighSpeed, Scalable, BiC, Cubic, FAST, H-TCP, L-TCP
  - *сложно сделать в закрытых ОС Windows, пересборка ядра в Linux*
- Параллельный TCP (*модификации в пользовательском пространстве*)
  - PSockets, GridFTP
- Основанный на скорости надежный UDP
  - RBUDP, Tsunami, FOBS, FRTP (основан на SABUL), Hurricane (основан на UDT)
- XCP (*использует TCP-флаги, устанавливаемые роутерами, сложен во внедрении*)
- SABUL (*прототип UDT*)

Проблемы в работе

- Довольно тяжел в применении
  - TCP варианты и XCP
  - Необходимы изменения в ядре ОС и/или в роутерах
- Не могут использоваться в коллективных (shared) сетях
  - Большинство надежных протоколов основанных на UDP
- Низкая “справедливость / честность” (fairness)
  - Intra-протокола “справедливость”
  - RTT “справедливость”
- Ручная настройка параметров

Новый протокол

Новый протокол должен использовать полосу пропускания более эффективно, чем TCP, позволяя им обоим работать в сети “справедливо” и «дружелюбно».



UDT (UDP-based Data Transfer Protocol, Основанный на UDP Протокол Передачи Данных)

- Прикладной уровень, основан на UDP
  - может быть легко установлен
- Аналогичная TCP функциональность
  - Ориентирован на соединение, надежный, дуплексная передача, unicast, поток данных
- Новый проект протокола и его реализация
- Новый алгоритм управления перегрузкой
- Конфигурируемая структура управления перегрузкой

Прямая задача и то, что не входит в планы

- Прямая задача
  - Для distributed data intensive applications в высокоскоростных сетях
  - Небольшое количество потоков делят полосу пропускания
  - Эффективный, честный, «дружелюбный» (Efficient, fair, and friendly)
  - Конфигурируемый
  - Легок в установке и применении

- То, что не входит в планы
  - Заменить TCP в Internet

Проект UDT

- С открытым исходным кодом (udt.sourceforge.net)
- Разработка и исполнение протокола UDT
- Разработка алгоритма управления перегрузкой в UDT
- Экспериментальным путем определить производительность UDT
- Разработка и реализация конфигурируемой протокольной структуры основанной на UDT (составной UDT)

ВСТУПЛЕНИЕ

>> ПРОЕКТ ПРОТОКОЛА И ЕГО РЕАЛИЗАЦИЯ

УПРАВЛЕНИЕ ПЕРЕГРУЗКОЙ

ОЦЕНКА ЭФФЕКТИВНОСТИ

СОСТАВНОЙ UDT

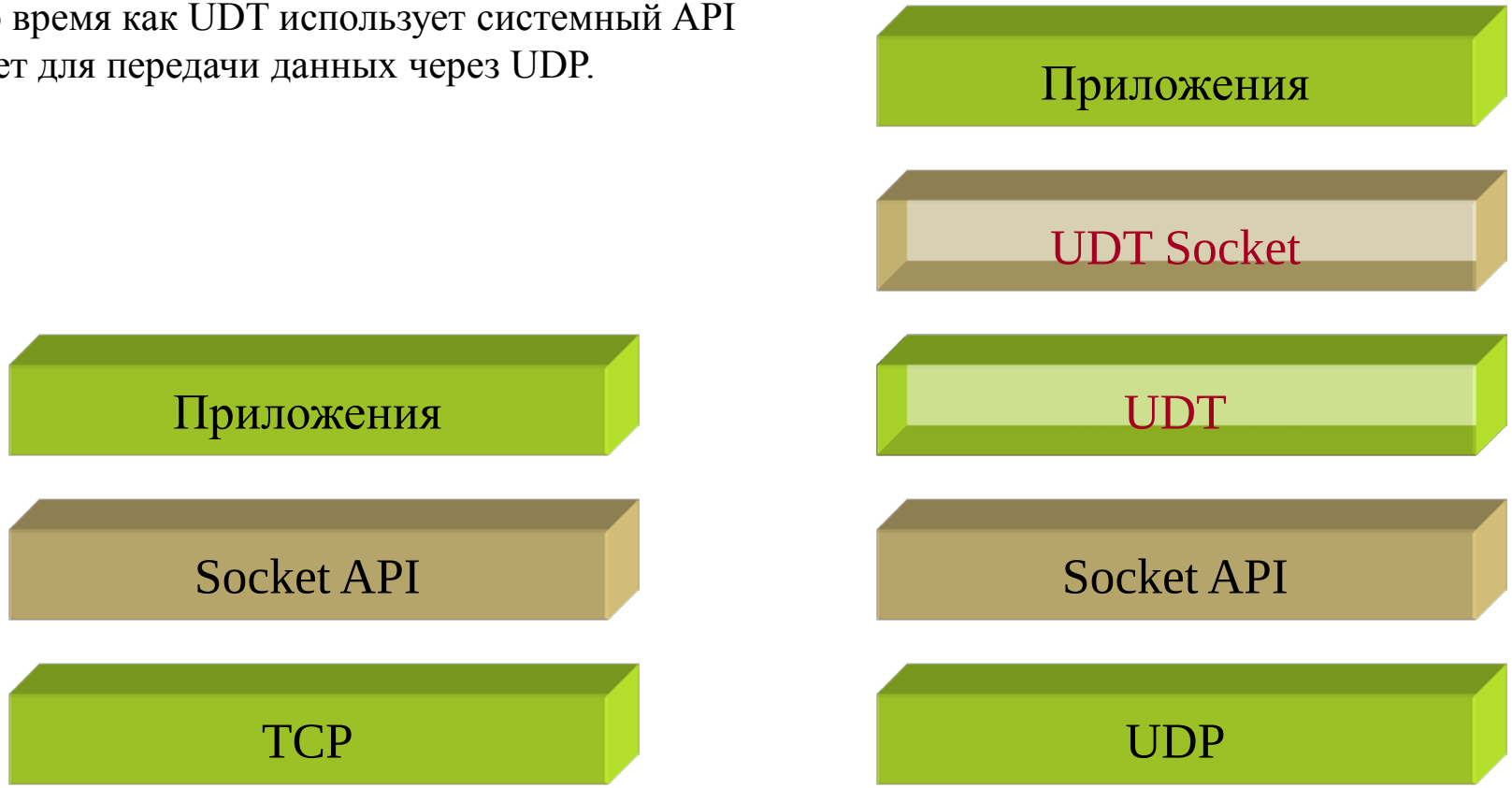
ЗАКЛЮЧЕНИЕ

Краткий обзор UDT

- Два пересекающихся элемента
  - Протокол UDT
  - Алгоритм управления перегрузкой UDT
- Проект и реализация протокола
  - Функциональность
  - Эффективность
- Алгоритм управления перегрузкой
  - Эффективность, честность, стабильность

Краткий обзор UDT

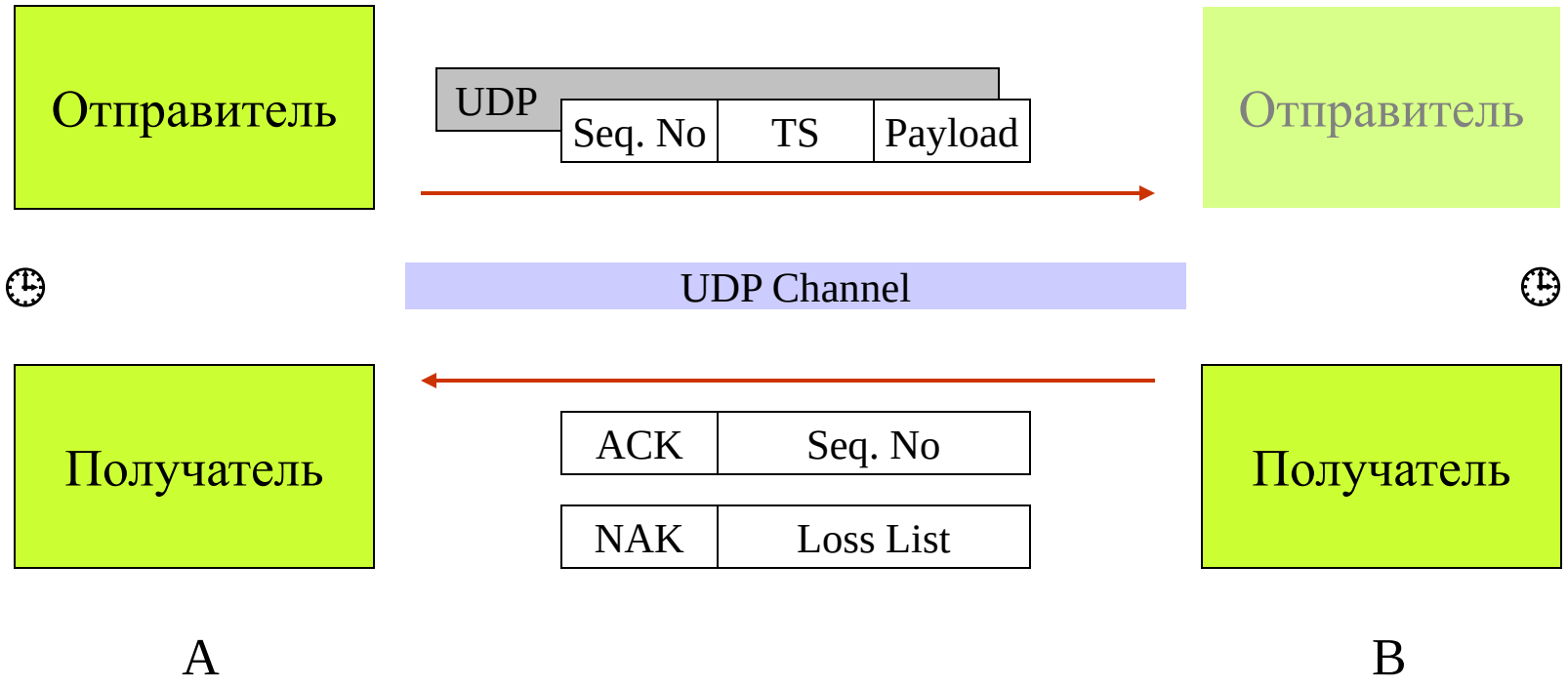
Приложения используют TCP через API сокеты.
Приложения используют UDT с API сокетом UDT, в то время как UDT использует системный API сокет для передачи данных через UDP.



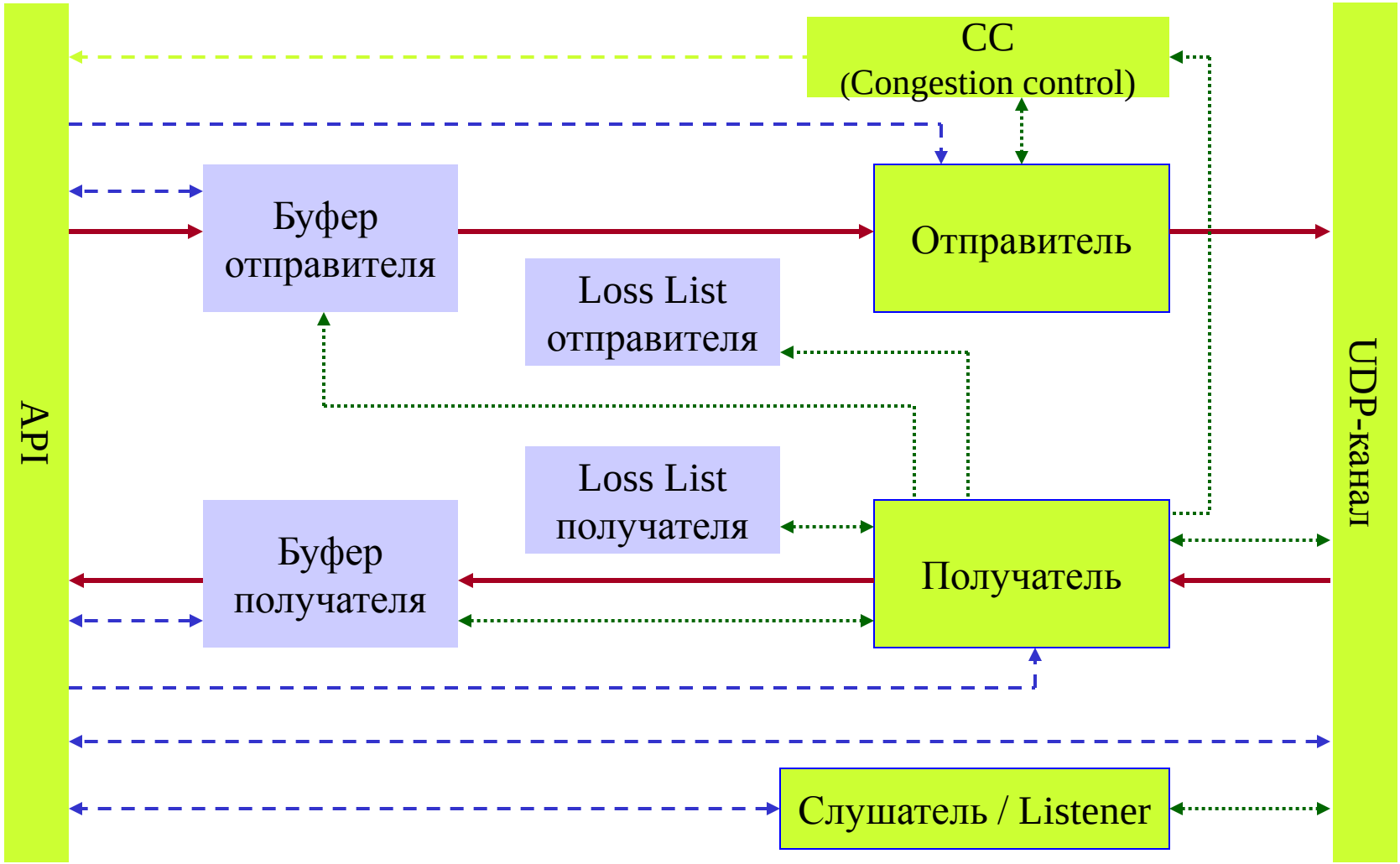
Функциональность

- Надежность
  - Пакетно-ориентированное упорядочивание
  - Подтверждение и сообщение о потере от получателя
  - ACK sub-sequencing
  - Ретрансмиссия (основана на сообщениях о потерях и таймерах)
- Потоки и обмен сообщениями
  - Управление буфером/памятью
- Обеспечение соединения
  - Рукопожатие, сообщение активности (keep-alive), сообщение о прекращении работы (teardown)
- Дуплексная передача
  - Каждый участник UDT является и отправителем и получателем

Структура протокола



Структура ПО



Оценка эффективности

- Меньшее количество пакетов
  - Подтверждение основанное на таймерах
- Меньшее время работы CPU
  - Уменьшенное время обработки пакета
  - Уменьшается копирование памяти
  - Уменьшается время обработки loss list
  - Light ACK vs. regular ACK
- Параллельная обработка
  - Нитевая структура
- Меньшее количество разрывов при обработке
  - Равномерное распределение времени обработки

Application Programming Interface (API)

- Socket API

- Новый API
  - sendfile/recvfile: эффективная передача файлов
  - sendmsg/recvmsg: обмен сообщениями с частичной надежностью
  - selectEx: более эффективная версия “select”

- Рандеву-соединение
  - Обход фаерволла

ВСТУПЛЕНИЕ

ПРОЕКТ ПРОТОКОЛА И ЕГО РЕАЛИЗАЦИЯ

>> УПРАВЛЕНИЕ ПЕРЕГРУЗКОЙ

ОЦЕНКА ЭФФЕКТИВНОСТИ

СОСТАВНОЙ UDT

ЗАКЛЮЧЕНИЕ

Обзор

- Управление перегрузкой vs. управление потоком
  - Управление перегрузкой: эффективное использование полосы пропускания
  - Управление потоком: предохраняет получателя от переполнения входящими пакетами

- Управление окном vs. управление скоростью
  - Управление окном: настраивается максимальное число пакетов передаваемых без подтверждения (TCP)
  - Управление скоростью: настраивается время между передачами пакетов (UDT)

- AIMD: additive increases multiplicative decreases - исключение перегрузки, управление размером cwnd

- Обратная связь
  - Packet loss (большее количество TCP вариантов, UDT)
  - Delay (Vegas, FAST)

AIMD с уменьшающимися увеличениями

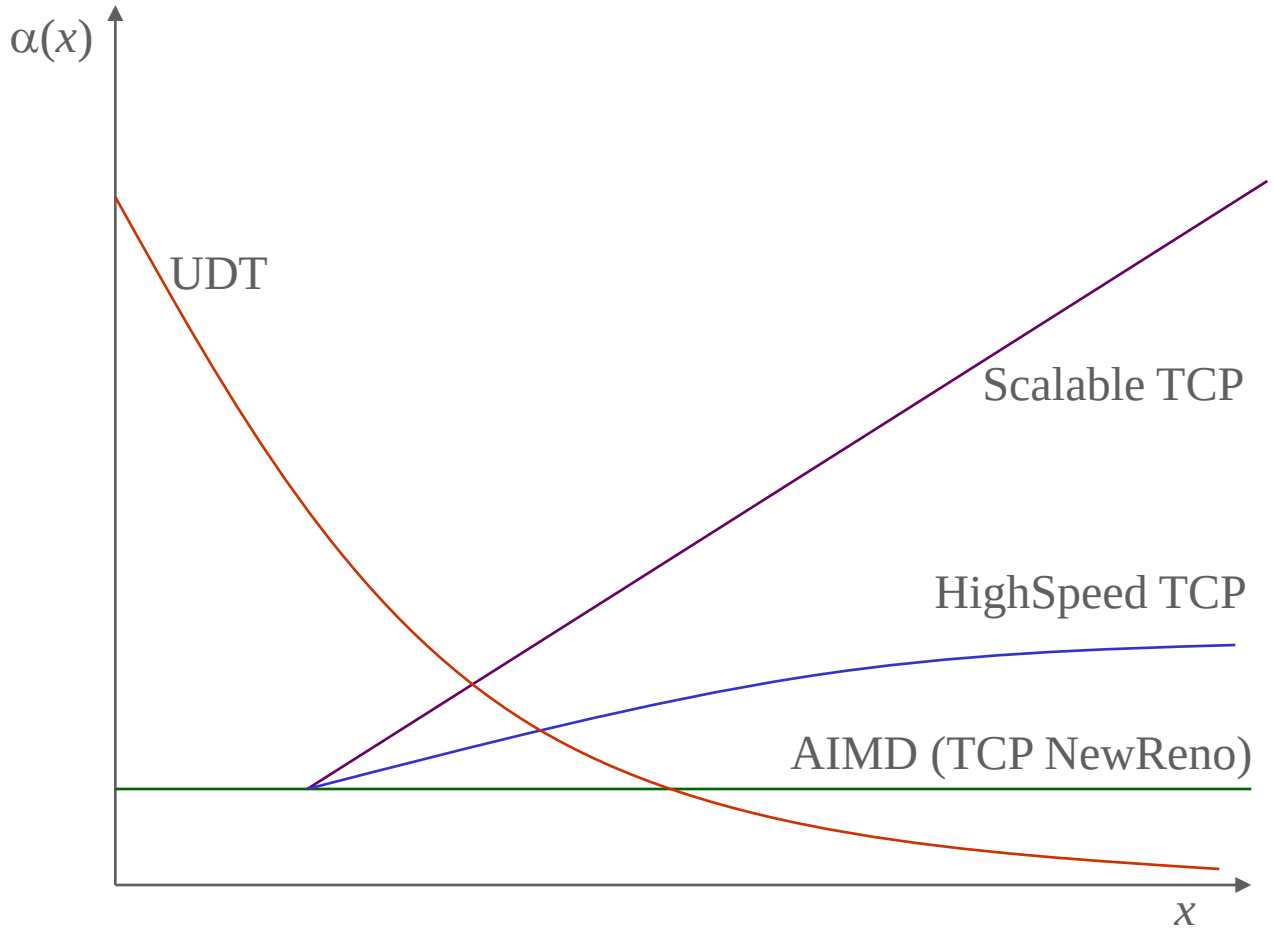
- AIMD
  - $x = x + \alpha(x)$, для всех постоянных интервалов (например, RTT)
  - $x = (1 - \beta)\, x$, когда произошло событие packet loss

  где x это скорость отправления пакета.

- TCP
  - $\alpha(x) \equiv 1$, увеличивающимся интервалом является RTT.
  - $\beta = 0.5$

- AIMD с уменьшающимися увеличениями
  - $\alpha(x)$ не увеличивается, и $lim_{x->+\infty}\, \alpha(x) = 0.$

AIMD с уменьшающимися увеличениями



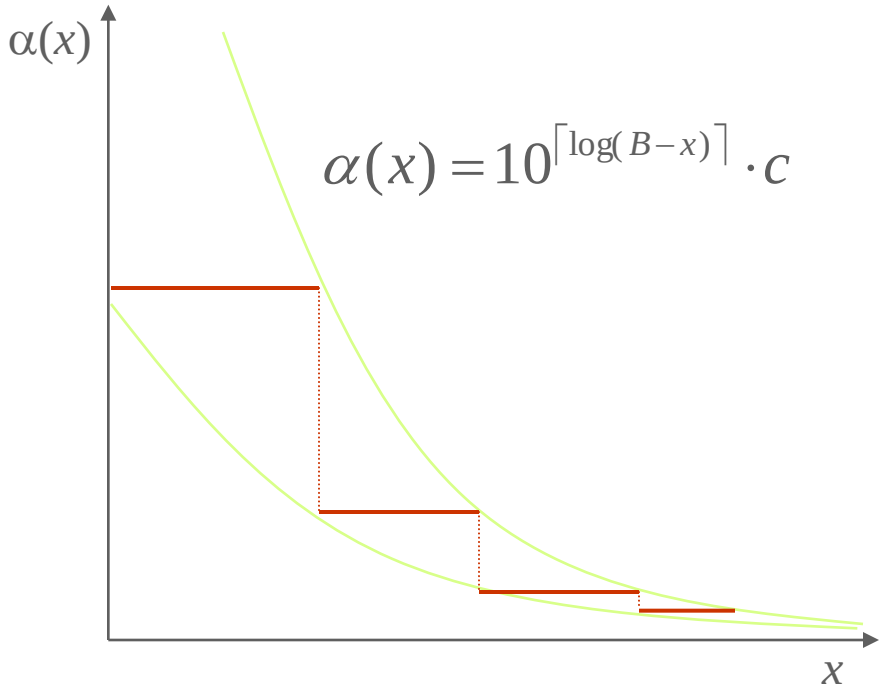
Алгоритм управления UDT

- Увеличение
  - $\alpha(x) = f(B - x) * c$

    где B – это пропускная способность соединения, c – постоянный параметр

- Постоянный интервал управления скоростью (*SYN*), не равный RTT
  - SYN = 0.01 сек.

- Уменьшение
  - Случайный коэффициент уменьшения
  - $\beta = 1 - (8/9)^n$



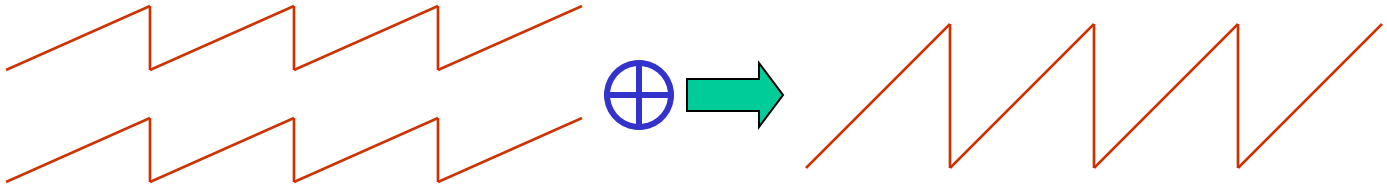
Формула увеличения: пример

Полоса пропускания (*B*) = 10 Gbps, Размер пакета = 1500 bytes

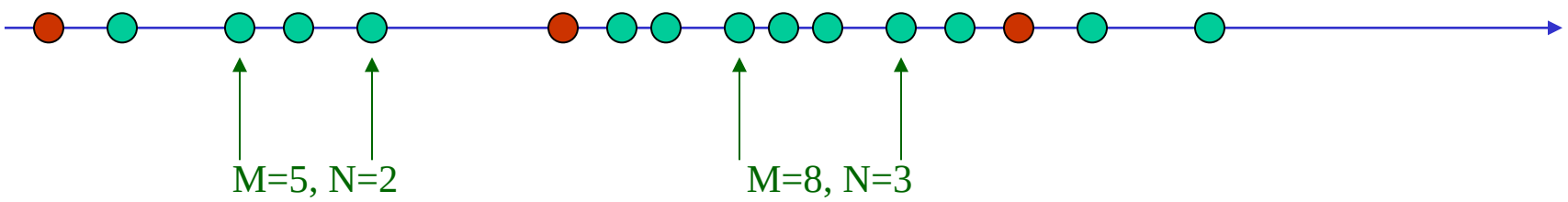
x (Mbps)	*B* - *x* (Mbps)	Прибавка (pkts/SYN)
[0, 9000)	(1000, 10000]	10
[9000, 9900)	(100, 1000]	1
[9900, 9990)	(10, 100]	0.1
[9990, 9999)	(1, 10]	0.01
[9999, 9999.9)	(0.1, 1]	0.001
9999.9+	<0.1	0.00067

Ситуация с потерей пакета(Packet Loss)

- Синхронизация при потере
  - Метод рандомизации



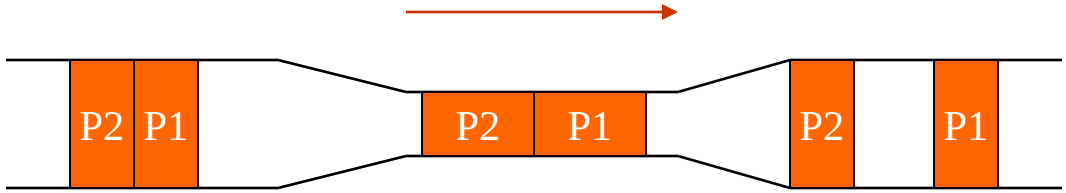
- Потеря без перегрузки
  - Скорость отправки не уменьшается при первой потере пакета



- Перезапрос пакета

Определение полосы пропускания

- Пара пакетов



Packet Size / Space ≈ Bottleneck Bandwidth

- Фильтры
  - Пересечение трафика
  - Смешанные препятствия
  - Устойчивость к ошибкам вычислений

- Случайный интервал между отправлениями пар пакетов

ВСТУПЛЕНИЕ

ПРОЕКТ ПРОТОКОЛА И ЕГО РЕАЛИЗАЦИЯ

УПРАВЛЕНИЕ ПЕРЕГРУЗКОЙ

>> ОЦЕНКА ЭФЕКТИВНОСТИ

СОСТАВНОЙ UDT

ЗАКЛЮЧЕНИЕ

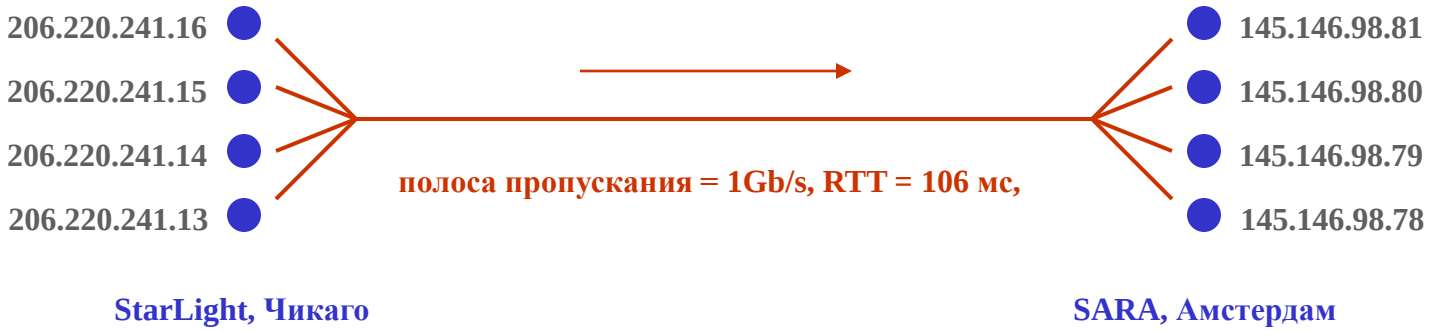
Характеристики производительности

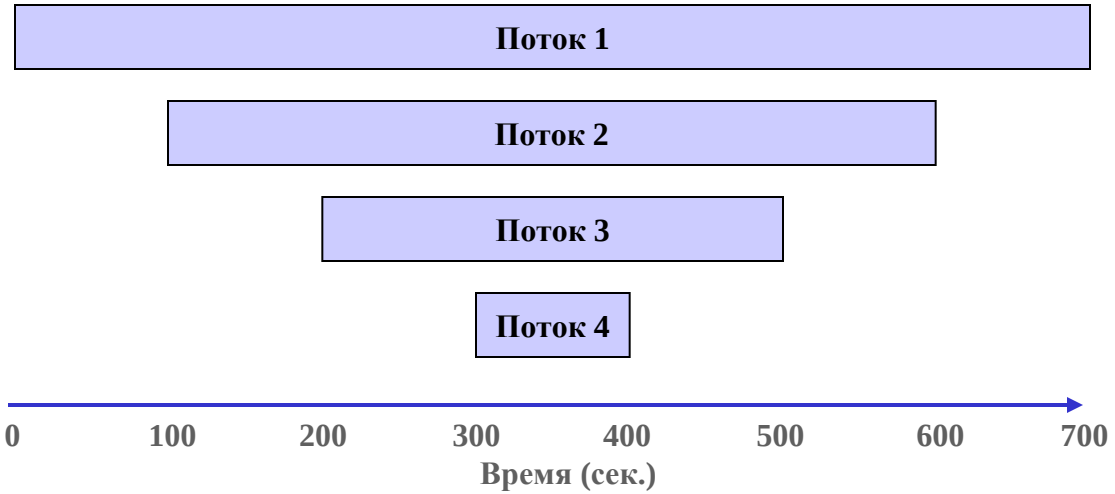
- Эффективность
  - Большее использование полосы пропускания, меньшее использование CPU
- Межпротокольная честность
  - Max-min честность
  - Индекс честности Jain‘a
- TCP «дружелюбие»
  - Bulk TCP поток vs. Bulk UDT поток
  - Short-lived TCP поток vs Bulk UDT поток
- Устойчивость (колебания)
  - Индекс устойчивости (стандартное отклонение)

Стратегии оценки

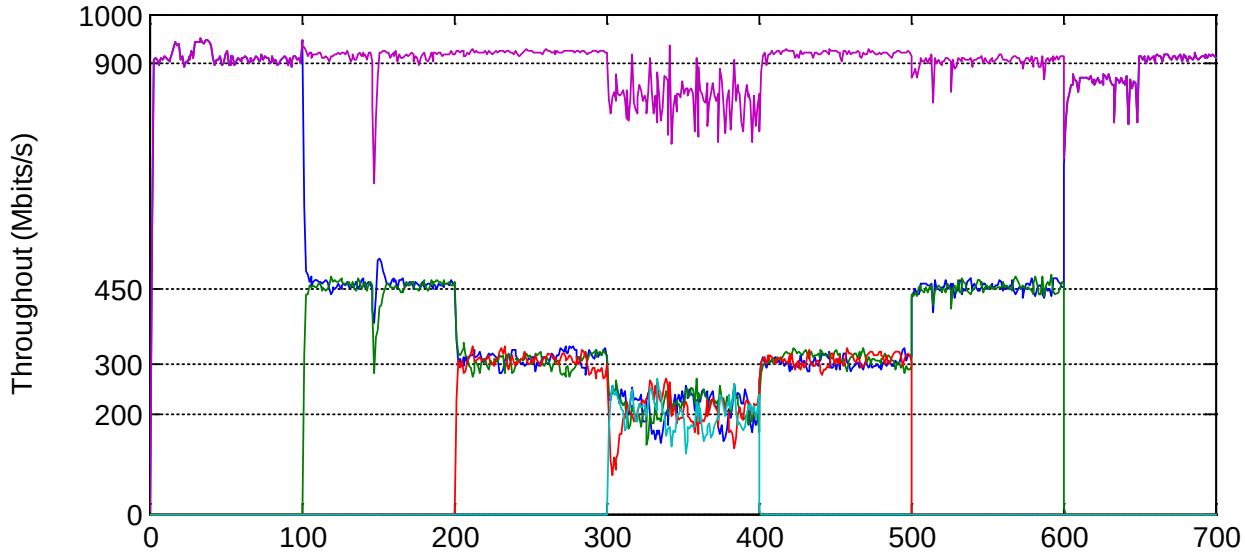
- Моделирование vs. эксперимент
  - Симулятор сети NS2, тестовая площадка NCDM Teraflow
- Система
  - Топология сети, полоса пропускания, дистанция, организация очередей, и т.д.
  - Согласованность (количество параллельных потоков)
- Сравнение (с TCP)
- Возможные применения
  - Передач данных SDSS, высокопроизводительное получение потоковых данных, и т.д.
- Независимая оценка
  - SLAC, JGN2, UvA, Unipmn (Италия), и т.д.

Эффективность, честность и стабильность



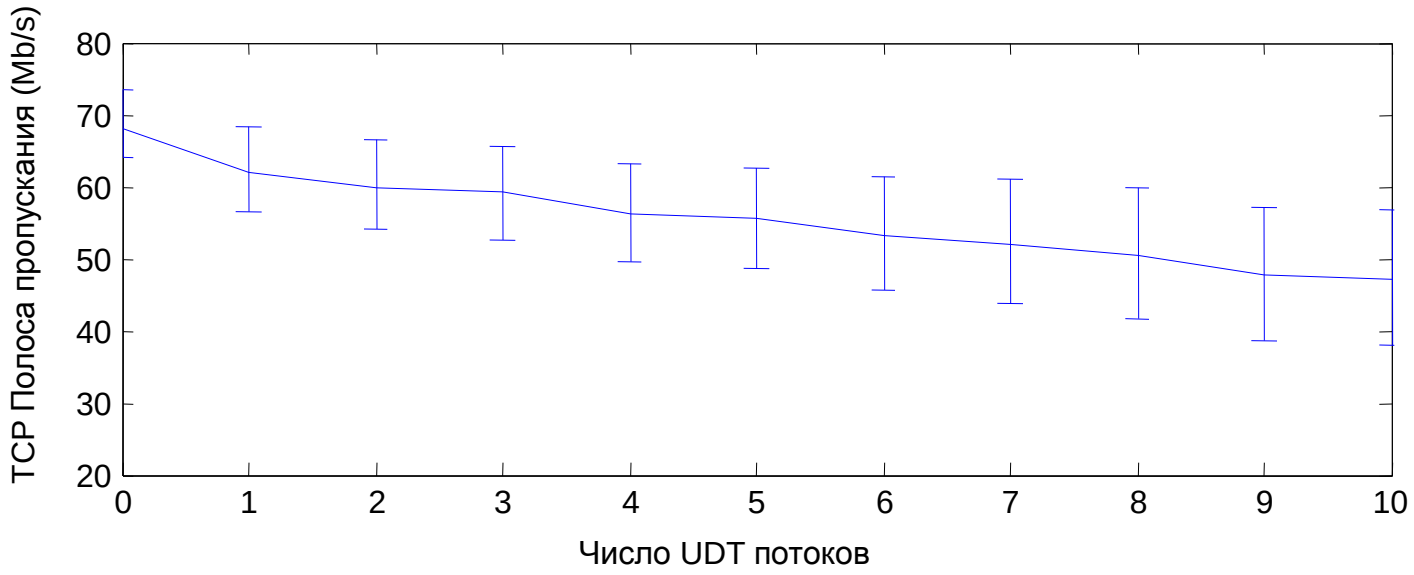


Эффективность, честность и стабильность



Поток 1	902	466	313	215	301	452	885
Поток 2		446	308	216	310	452	
Поток 3			302	202	307		
Поток 4				197			
Эффективность	902	912	923	830	918	904	885
Честность	1	0.999	0.999	0.998	0.999	1	1
Стабильность	0.11	0.11	0.08	0.16	0.04	0.02	0.04

TCP «дружелюбие»



- 500 1MB TCP потоков vs. 0 – 10 bulk UDT потоков
- 1Gb/s между Чикаго и Амстердамом

ВСТУПЛЕНИЕ

ПРОЕКТ ПРОТОКОЛА И ЕГО РЕАЛИЗАЦИЯ

УПРАВЛЕННИЕ ПЕРЕГРУЗКОЙ

ОЦЕНКА ЭФФЕКТИВНОСТИ

>> СОСТАВНОЙ UDT

ЗАКЛЮЧЕНИЕ

Составной UDT - Задачи

- Легкая реализация и внедрение новых алгоритмов управления
- Легкая оценка новых алгоритмов управления
- Поддержка осведомленности приложений и динамическая конфигурация

Составной UDT - Методология

- Управление отправкой пакетов
  - Основанное на окне, основанное на скорости, гибридное
- Управление событиями
  - onACK, onLoss, onTimeout, onPktSent, onPktRecved, и т.д.
- Доступ к параметрам протокола
  - RTT, RTO, и т.д.
- Пакетное расширение
  - Управление пакетами осуществляется пользователем

Составной UDT - Оценка

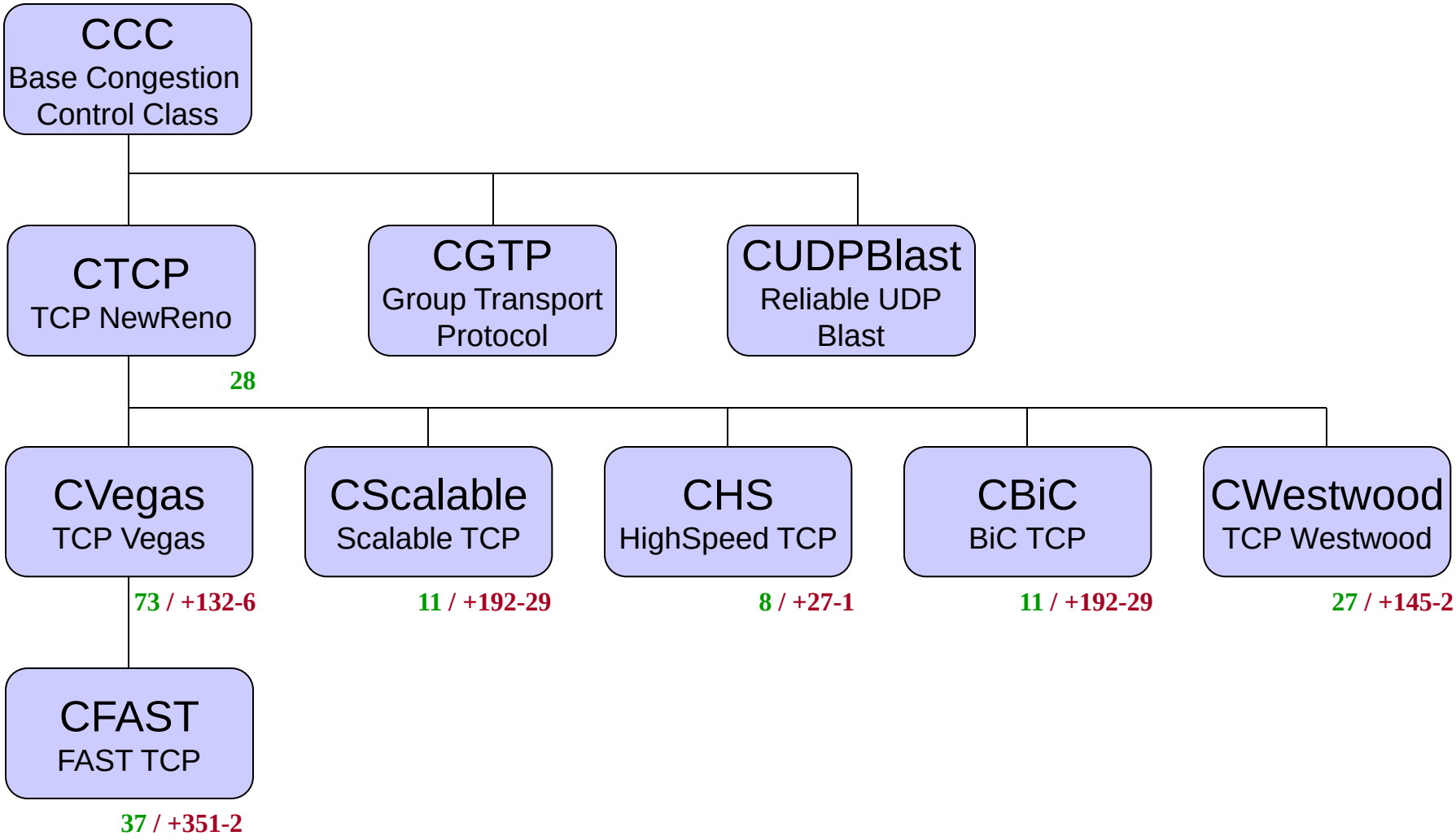
- Простота
  - Возможна ли простота в использовании?
- Многозначительность
  - Может ли быть использован для реализации большинства управляющих протоколов?
- Сходство
  - Могут ли реализации, основанные на составном UDT, быть столь же эффективными как их «родные» реализации?
- Накладные расходы
  - Будут ли накладные расходы слишком большими?

Простота и многозначительность

- Восемь обработчиков событий, четыре функции управления протоколом, и одна функция мониторинга производительности.

- Поддержка большого ряда протоколов
  - Надежный UDT blast
  - TCP и его варианты (основанные как на потерях, так и на задержке)
  - Протоколы передачи групп

Простота и многозначительность



Сходства и накладные расходы

- Сходства
  - Как реализации, основанные на составном UDT, имитируют свои «родные» реализации?

- CTCP vs. Linux TCP

Поток #	Проп. способность		Честность		Стабильность	
	TCP	CTCP	TCP	CTCP	TCP	CTCP
1	112	122	1	1	0.517	0.415
2	191	208	0.997	0.999	0.476	0.426
4	322	323	0.949	0.999	0.484	0.492
8	378	422	0.971	0.999	0.633	0.550
16	672	642	0.958	0.985	0.502	0.482
32	877	799	0.988	0.997	0.491	0.470
64	921	716	0.994	0.996	0.569	0.529

- Использование CPU
  - Отправитель: CTCP использует около 100% больше времени CPU чем Linux TCP
  - Получатель: CTCP использует около 20% больше CPU чем Linux TCP

ВСТУПЛЕНИЕ

ПРОЕКТ ПРОТОКОЛА И ЕГО РЕАЛИЗАЦИЯ

УПРАВЛЕНИЕ ПЕРЕГРУЗКОЙ

ОЦЕНКА ЭФФЕКТИВНОСТИ

СОСТАВНОЙ UDT

>> ЗАКЛЮЧЕНИЕ

Сотрудничество

- Высокопроизводительный протокол передачи данных и сопутствующие реализации
  - Протокол UDT
  - Open source UDT библиотека (*udt.sourceforge.net*)
  - Пользователи: НИИ и промышленность
- Эффективный и «честный» алгоритм управления
  - DAIMD и алгоритм управления UDT
  - Методы манипулирования потерей пакетов
  - Использование метода расчета полосы пропускание при управлении перегрузкой
- Конфигурируемая структура протокола передачи данных
  - Составной UDT

Использованные публикации

- Papers on the UDT Protocol
  - ***UDT: UDP-based Data Transfer for High-Speed Wide Area Networks,*** Yunhong Gu and Robert L. Grossman, Computer Networks (Elsevier). Volume 51, Issue 7. May 2007.
  - ***Supporting Configurable Congestion Control in Data Transport Services***, Yunhong Gu and Robert L. Grossman, SC 2005, Nov 12 - 18, Seattle, WA.
  - ***Experiences in Design and Implementation of a High Performance Transport Protocol***, Yunhong Gu, Xinwei Hong, and Robert L. Grossman, SC 2004, Nov 6 - 12, Pittsburgh, PA.
  - ***An Analysis of AIMD Algorithms with Decreasing Increases***, Yunhong Gu, Xinwei Hong and Robert L. Grossman, First Workshop on Networks for Grid Applications (Gridnets 2004), Oct. 29, San Jose, CA.
- Internet Draft
  - ***UDT: A Transport Protocol for Data Intensive Applications***, Yunhong Gu and Robert L. Grossman, draft-gg-udt-02.txt.

Коммерциализация

- Baidu Hi Messenger
- Maidsafe
- Movie2Me by broadcasting center europe
- NiFTy TV
- Sterling File Accelerator (SFA) by Sterling Commerce
- Tideworks
- PowerFolder
- GridFTP
- etc.

Поддержка и сервис

- Онлайн форум
  - https://sourceforge.net/forum/?group_id=115059
  - Поддержка Английского и Китайского языков

- Услуги консультации
  - Выделенный сервис для проектов заказчика
  - Обеспечивается разработчиками UDT

Достижения

- SC 2002 Bandwidth Challenge “Best Use of Emerging Network Infrastructure” Award
- SC 2003 Bandwidth Challenge “Application Foundation” Award
- SC 2004 Bandwidth Challenge “Best Replacement for FedEx / UDP Fairness” Award
- SC 2006 Bandwidth Challenge Winner
- SC 2008 Bandwidth Challenge Winner

Будущее UDT

- Близкое
  - Практическое решение для приложений с распределенными данными в средах с высоким BDP.

- В дальнейшем
  - Развитие при помощи новых технологий (open source & open standard)
  - Большая функциональность
  - Платформа для сетевых исследований(в т.ч., быстрое изготовление прототипов и оценка новых алгоритмов управления)

Спасибо!

Yunhong Gu, 10 Октября, 2005

Обновлено 8 Августа, 2009