

Лекция № 10. Протокол ТСР

Содержание

1. Воспоминания о
возможностях протокола,
формате пакета,
установлении соединения.
2. Повышение производительности ТСР:
 - управление таймаутами
 - борьба с перегрузками в сети
 - борьба с мелким окном
 - иные методы повышения эффективности

Возможности ТСП

- базовый обмен данными (Basic Data Transfer);
- надежность (Reliability);
- управление потоком данных (Flow Control);
- мультиплексирование (Multiplexing);
- соединения (Connections);
- предпочтения и безопасность (Precedence and Security).

Формат заголовка

| | | | | | | | | |
|---------------------|--------|-------------|-------------|---------------------|-------------|-------------|-------------|------|
| Порт назначения | | | | Порт получателя | | | | |
| Порядковый номер | | | | | | | | |
| Номер подтверждения | | | | | | | | |
| Смещение данных | Резерв | u r g | a c k | p s h | r s t | s y n | f i n | Окно |
| Контрольная сумма | | | | Указатель срочности | | | | |
| Опции | | | | | | Заполнение | | |
| Данные | | | | | | | | |

Возможности ТСР: надежность

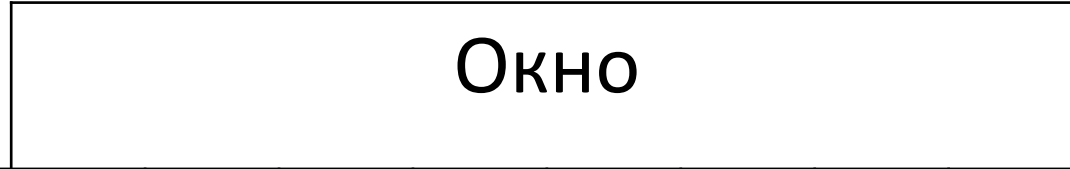
| | | | |
|-------------|--------|--------------|----------------------|
| Повреждение | Потеря | Дублирование | Нарушение порядка |
|-------------|--------|--------------|----------------------|

Присвоение порядковых номеров и квитирование

Вычисление контрольной суммы

Возможности ТСР: управление потоком

Поток
октетов →



| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|-------------------------------|---|---|-----------------------------------|---|---|---|---|--|----|----|---|----|----|
| Отправленные и подтвержденные | | | Отправленные, но неподтвержденные | | | | | Неотправленные, но могут быть отправлены | | | Пока не могут быть отправлены. Отправить можно после смещения окна. | | |

Скользящее окно – механизм управления потоком данных, позволяющий контролировать объем данных, передаваемых отправителем.

Возможности ТСР: соединения

Соединение – комбинация сведений, включающая сокет, порядковые номера и размеры окна.

Соединение однозначно идентифицируется парой сокетов.

Информация о соединении хранится в структуре данных TCB (Transmission Control Block).

Соединение открывается вызовом OPEN.

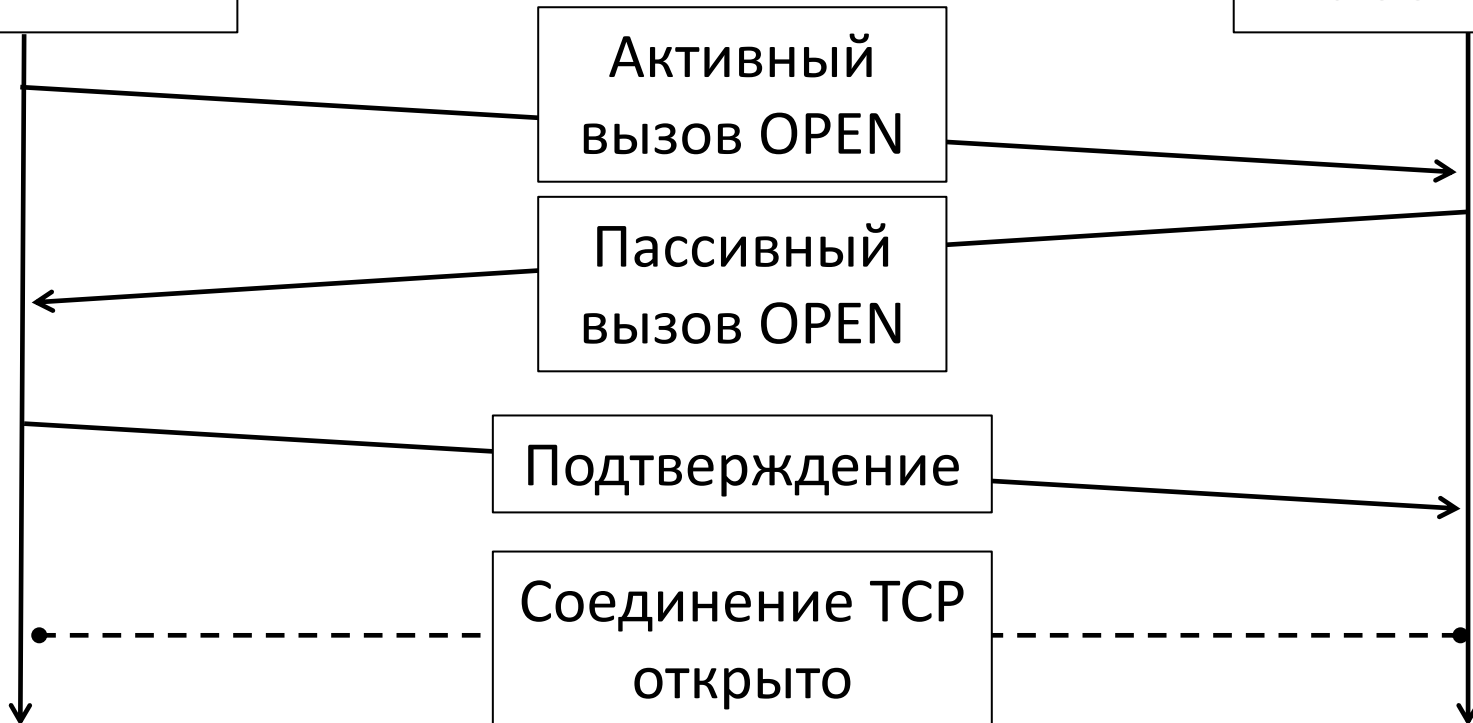
Тип OPEN активный или пассивный

Активный и пассивный запросы OPEN



Активный

Пассивный



Два активных запроса OPEN



АКТИВНЫЙ

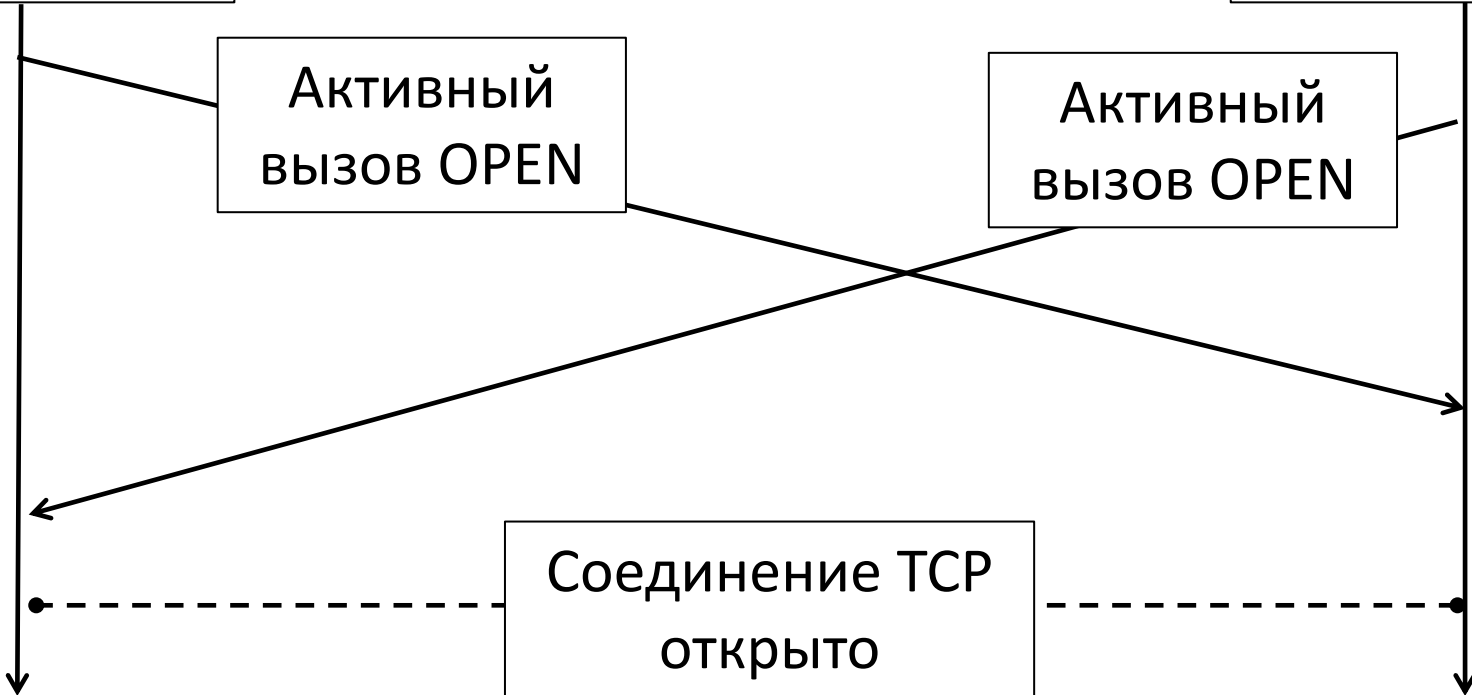


АКТИВНЫЙ

АКТИВНЫЙ
ВЫЗОВ OPEN

АКТИВНЫЙ
ВЫЗОВ OPEN

Соединение TCP
открыто



Организация соединения

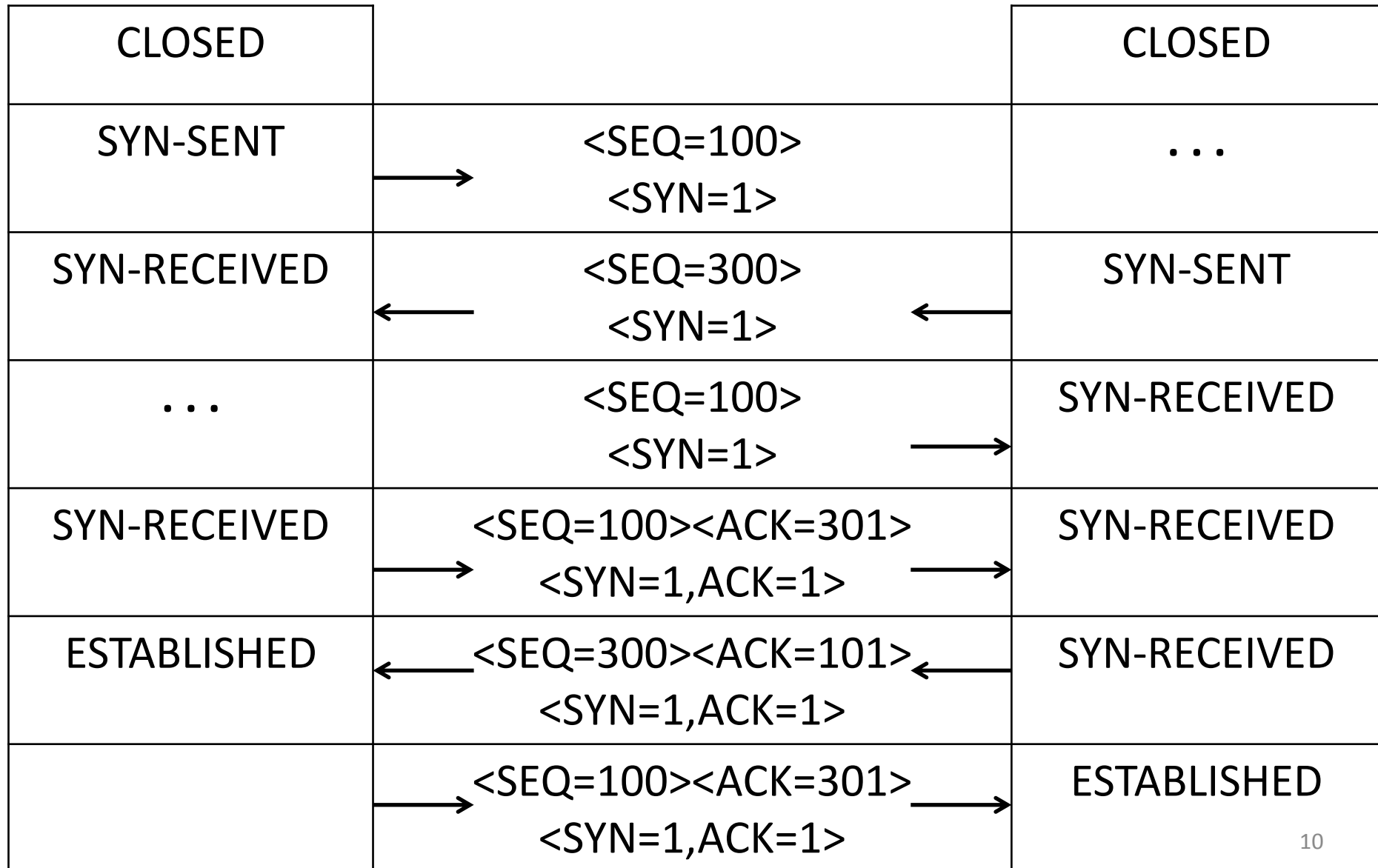
Базовое 3х этапное согласование



| Хост А. TCP | | Хост В. TCP | | |
|-------------|---|-------------------------------------|---|--------------|
| CLOSED | | LISTEN | | |
| SYN-SENT | → | <SEQ=100> <SYN=1> | → | SYN-RECEIVED |
| ESTABLISHED | ← | <SEQ=300><ACK=101> <SYN=1,ACK=1> | ← | SYN-RECEIVED |
| ESTABLISHED | → | <SEQ=101><ACK=301> <ACK=1> | → | ESTABLISHED |
| ESTABLISHED | → | <SEQ=101><ACK=301> <ACK=1><DATA> | → | ESTABLISHED |

Организация соединения

Одновременная синхронизация



Управление тайм-аутами повторной передачи

Для учета изменяющейся в широких пределах времени задержки используют

adaptive retransmission algorithm

$$RTT = a * \text{Старый_RTT} + (1 - a) * \text{Замеренный_RTT}$$

$$\text{Тайм-аут} = b * RTT$$

В первоначальной спецификации рекомендовали $b = 2$

Проблема «неоднозначности сигналов подтверждения приема» (acknowledgement ambiguity)

Алгоритм Карна

В случае потери пакетов связывание RTT с первым или последним отправленным пакетом приводит к неверной оценке RTT.

Идея. Модуль TCP не должен пересчитывать RTT на основе данных, полученных при повторной передаче сегмента. Но игнорировать потери тоже нельзя.

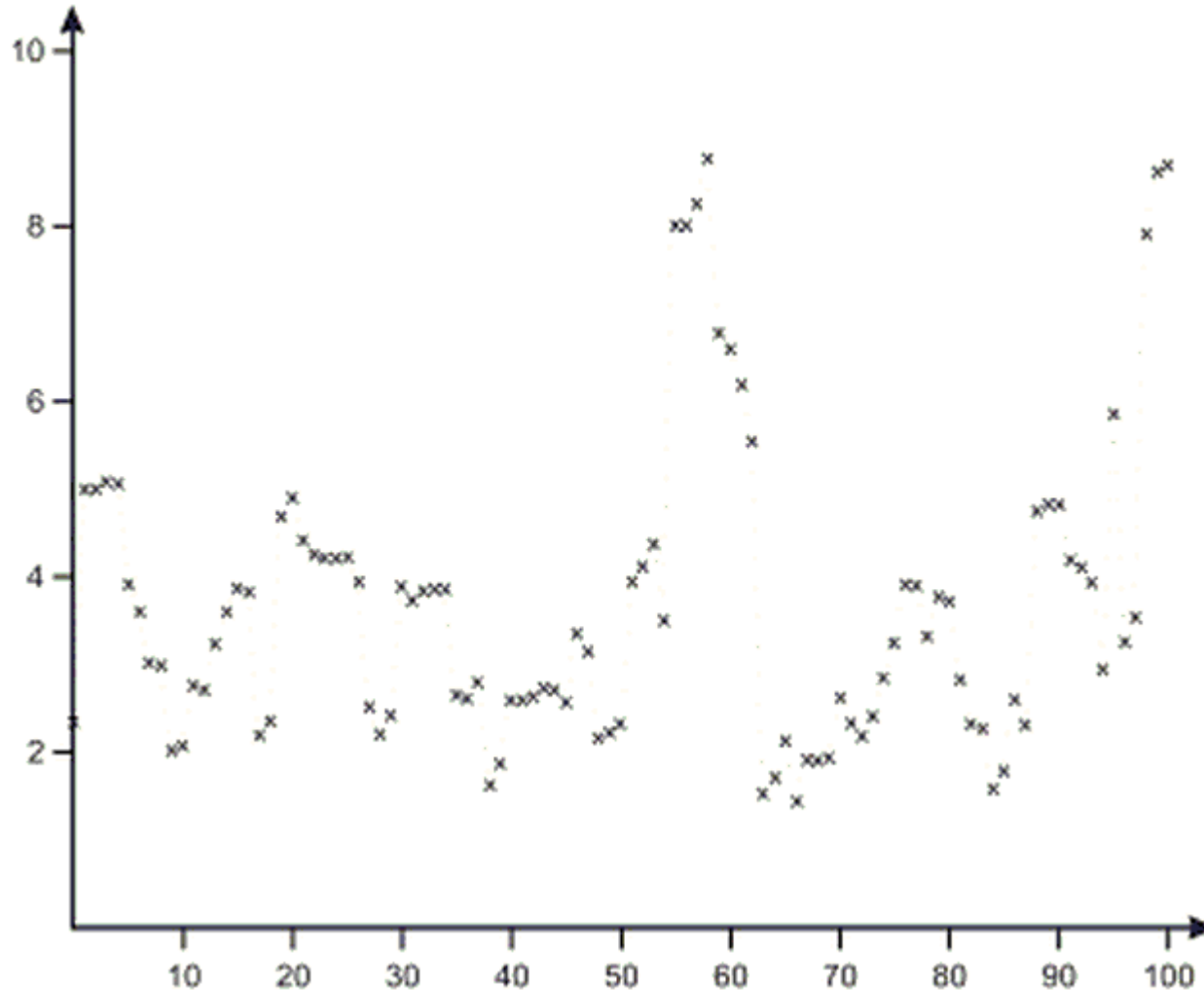
Новый_Тайм-аут = c * Тайм-аут

Рекомендуют $c = 2$

Пульсация задержек

Значения RTT для 100 последовательных дейтаграмм

По оси Oх – номера дейтаграмм, по оси Oу – время в секундах



Обработка большого разброса значений задержки

Разность = Замер – Старый_RTT

Сглаженный_RTT = Старый_RTT + $d * \text{Разность}$

Откл = Старое_Откл + $r * (|\text{Разность}| - \text{Старое_Откл})$

Тайм-аут = Сглаженный_RTT + $g * \text{Откл}$

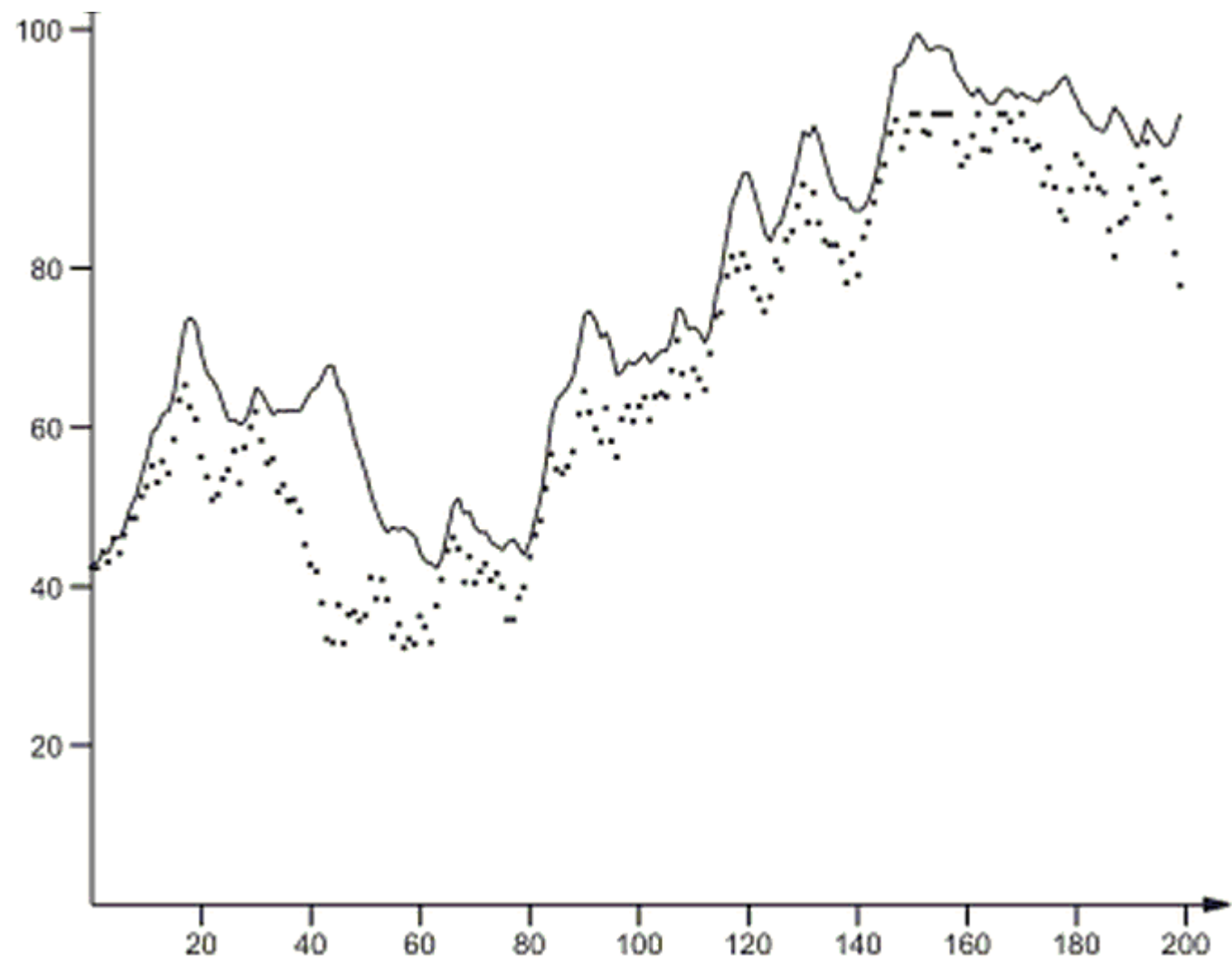
Здесь Откл – оценка среднего значения отклонения

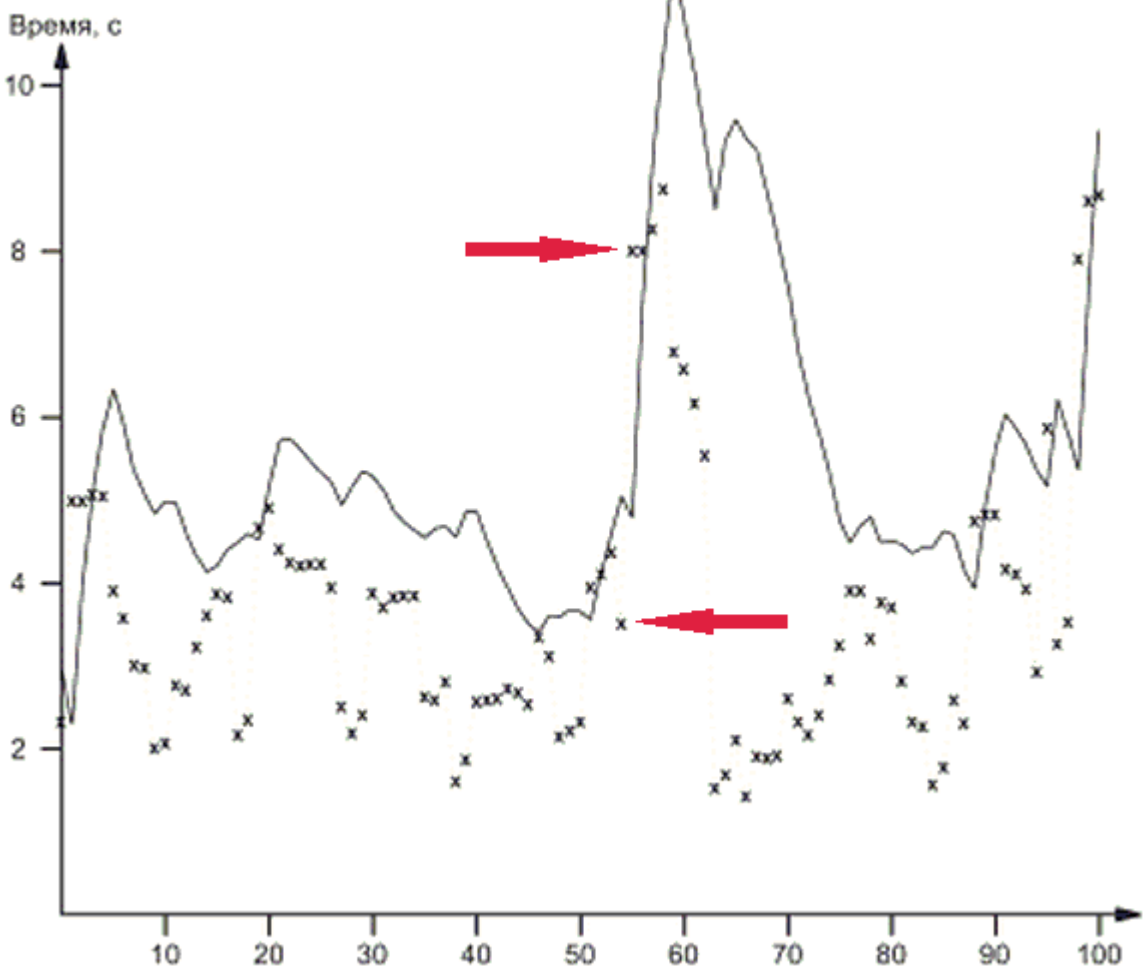
d – сила влияния новых замеров на RTT, $\in [0, 1]$

r – сила влияния новых замеров на отклонение, $\in [0, 1]$

g – сила влияния отклонения на тайм-аут

Рекомендации: $d = 1/8$, $r = 1/4$, $g = 4$.





Борьба с заторами: алгоритм ван Джекобсона

Для борьбы с перегрузкой в Сети модуль TCP использует

1. мультипликативное уменьшение
2. медленный старт

Размер окна = $\min \{ \text{Анонс получателя, Окно перегрузки} \}$

Предположение: большая часть пакетов теряется из-за перегрузки.

1. При возникновении затора:

Новое_Окно_перегрузки = $0.5 * \text{Старое_Окно_перегрузки}$

2. После преодоления затора:

Первоначальный размер Окна_перегрузки = 1 сегмент;
после получения ACK Окно_перегрузки увеличивается на 1 сегмент;
когда размер Окна_перегрузки = 0.5 прежнего размера увеличение на 1 сегмент только в случае получения подтверждения ВСЕХ сегментов.

Применение описанных мер увеличивает производительность TCP от 2 до 10^{17} раз.

Борьба с синдромом мелкого окна

Получатель:

1. Запрет на отправку анонса «мелкого» окна.

После отправки анонса на окно нулевого размера, следующий анонс откладывается до тех пор, пока размер окна получателя не достигнет $\min \{0.5 * \text{размер буфера получателя}, \text{MSS}\}$

2. Отложенные подтверждения получения.

Как правило, удержание не более 500 мс, иногда 200 мс.

Рекомендовано, без задержек подтверждать каждый второй сегмент.

Отправитель:

Алгоритм Нейгла – задержка отправки «маленьких» сегментов.

Алгоритм Нейгла

if there is new data to send

if the window size \geq MSS & available data is \geq MSS

send complete MSS segment now

else

if there is unconfirmed data still in the pipe

enquire data in the buffer until an acknowledge is received

else

send data immediately

end if

end if

end if

Алгоритм Нейгла хорошо работает при любых
задержках сети,
MSS,
скоростях работы приложения.

Согласование MSS

Фрагментация IP пакетов снижает эффективность передачи.

Если отправитель и получатель в одном сегменте сети, обычно выбирают $MSS = MTU$.

Иначе, либо $MSS = 536$ байт,
 либо сделать *discovery*.

- Часто не реализован механизм *discovery*
- Маршруты динамически меняются
- Оптимальный MSS зависит от дины заголовков канального уровня

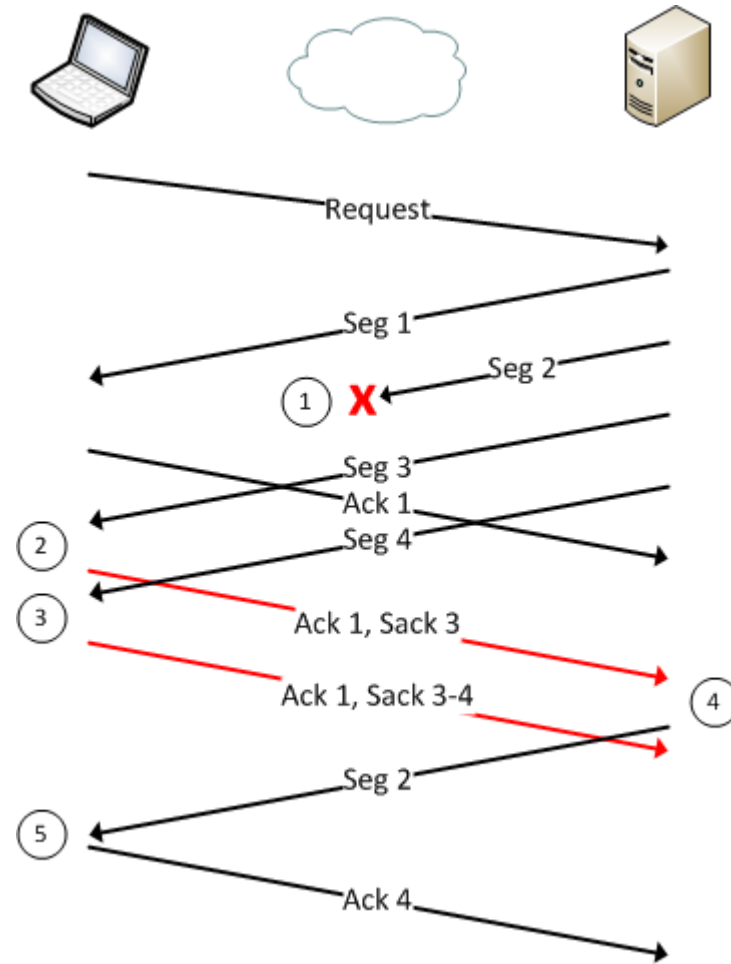
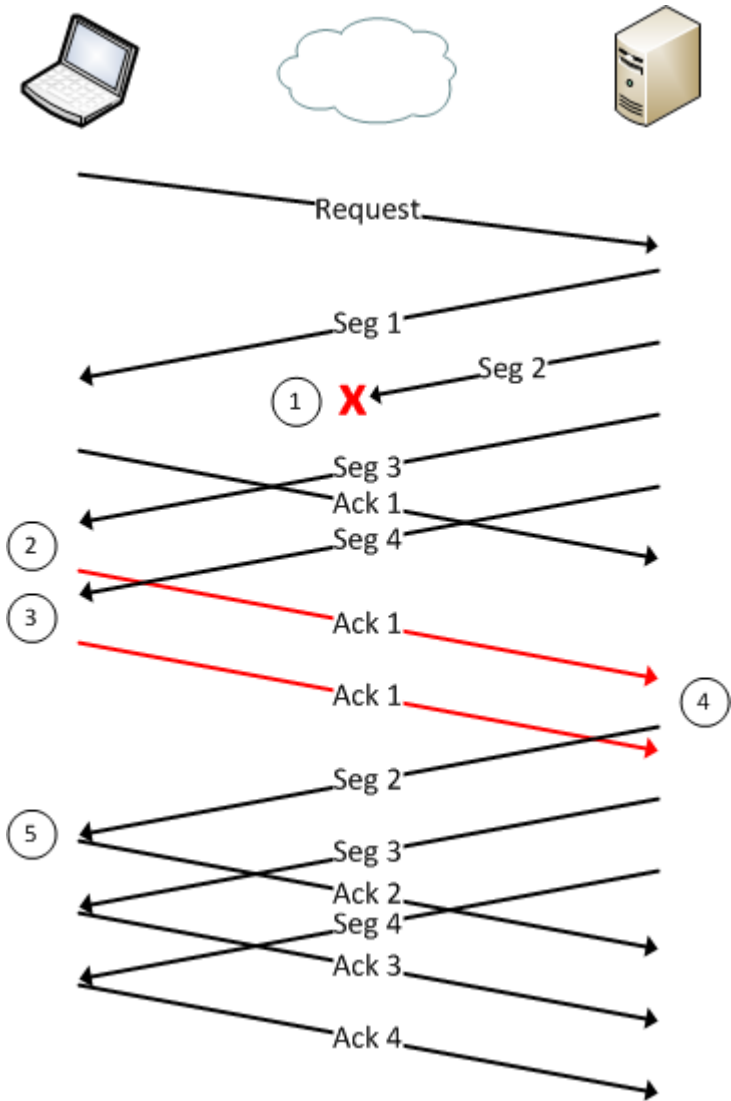
Произвольное раннее удаление на роутере

Усечение хвоста (tail drop) приводит к глобальной синхронизации – во всех соединениях модули TCP переходят к медленному старту.

RED (Random Early Drop/Discard)

TCP Selective Acknowledgments (SACK)

RFC 2018, 1996 г.



Рекомендуемая литература

Камер Дуглас Э. Сети TCP/IP. Принципы, протоколы и структура. Том 1

Исключительно качественная подача материала

<http://www.soslan.ru/tcp/tcp21.html>

<http://citforum.ru/nets/tcp/optimize01.shtml>

<http://ruteam.ru/prftungd/2365c93>