TCP (transmission control protocol)

СОДЕРЖАНИЕ

1. **Сервис**
2. **Эволюция**
3. **TCP-порт и TCP-соединение**
4. **Формат TCP-заголовка**
5. **Фазы работы**
   - «Установление соединения»
   - «Передача данных»
   - «Расторжение соединения»
6. **Управление потоком**
   - Цели
   - Алгоритм медленного старта
7. **Таймеры**

Историческая справка о протоколе TCP

- RFC-793, TRANSMISSION CONTROL PROTOCOL, Сентябрь 1981, *Jon Postel*
  - ✓ Стандарт Министерства обороны США
  - ✓ Предназначен для надежной и гарантированной доставки данных между хостами в компьютерных сетях.
  - ✓ «Мотивация. Основное внимание в RFC-793 сфокусировано на требованиях военных систем компьютерных коммуникаций, особенно вопросам устойчивости при наличии коммуникационных сбоев и доступности связи при возникновении перегрузки (насыщения)»
  - ✓ Протокол TCP основан на концепциях, впервые изложенных Серфом и Каном в работе [1]. TCP располагается в многоуровневой модели непосредственно над базовым протоколом Internet (IP) [2]. Процедуры организации соединений используют трехкратное рукопожатие (three-way hand shake [3].
- Литература, приведенная в RFC-793

[1] Cerf, V., and R. Kahn, "A Protocol for Packet Network Intercommunication", IEEE Transactions on Communications, Vol. COM-22, No. 5, pp 637-648, May **1974**.

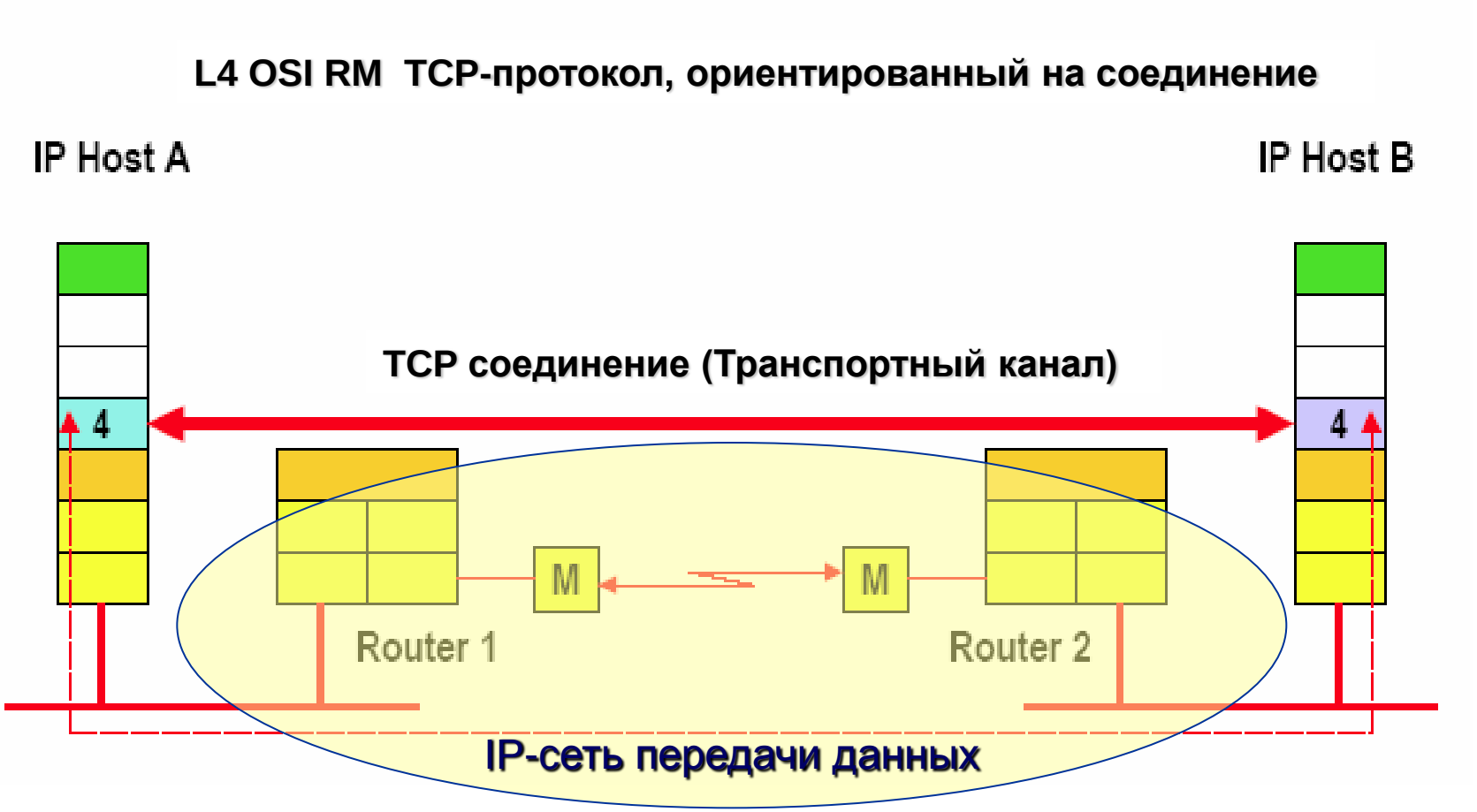
[2] Postel, J. (ed.), "Internet Protocol - DARPA Internet Program Protocol Specification", RFC 791, USC/Information Sciences Institute, September **1981**.

[3] Dalal, Y. and C. Sunshine, "Connection Management in Transport Protocols", Computer Networks, Vol. 2, No. 6, pp. 454-473, December 1978. [4] Postel, J., "Assigned Numbers", RFC 790[18], USC/Information Sciences Institute, September **1981**.

TCP –> L4 OSI RM (предоставляемый сервис)

- **Транспортный протокол**
- **Ориентированный на соединение**
- **Надежный**
  - использует ненадежный по своей природе IP-протокол
  - гарантирует безошибочный транспорт данных между процессами различных конечных систем посредством:
    - ✓ обнаружения ошибок и их исправления
    - ✓ восстановления последовательности сегментов (TCP PDU) без их дублирования и потерь
- **Двухточечный**
- **Управляет потоком и перегрузкой**
- **Поток байтовый**
- **Полный дуплекс**
- **RFC 793** (Postel, 1981)

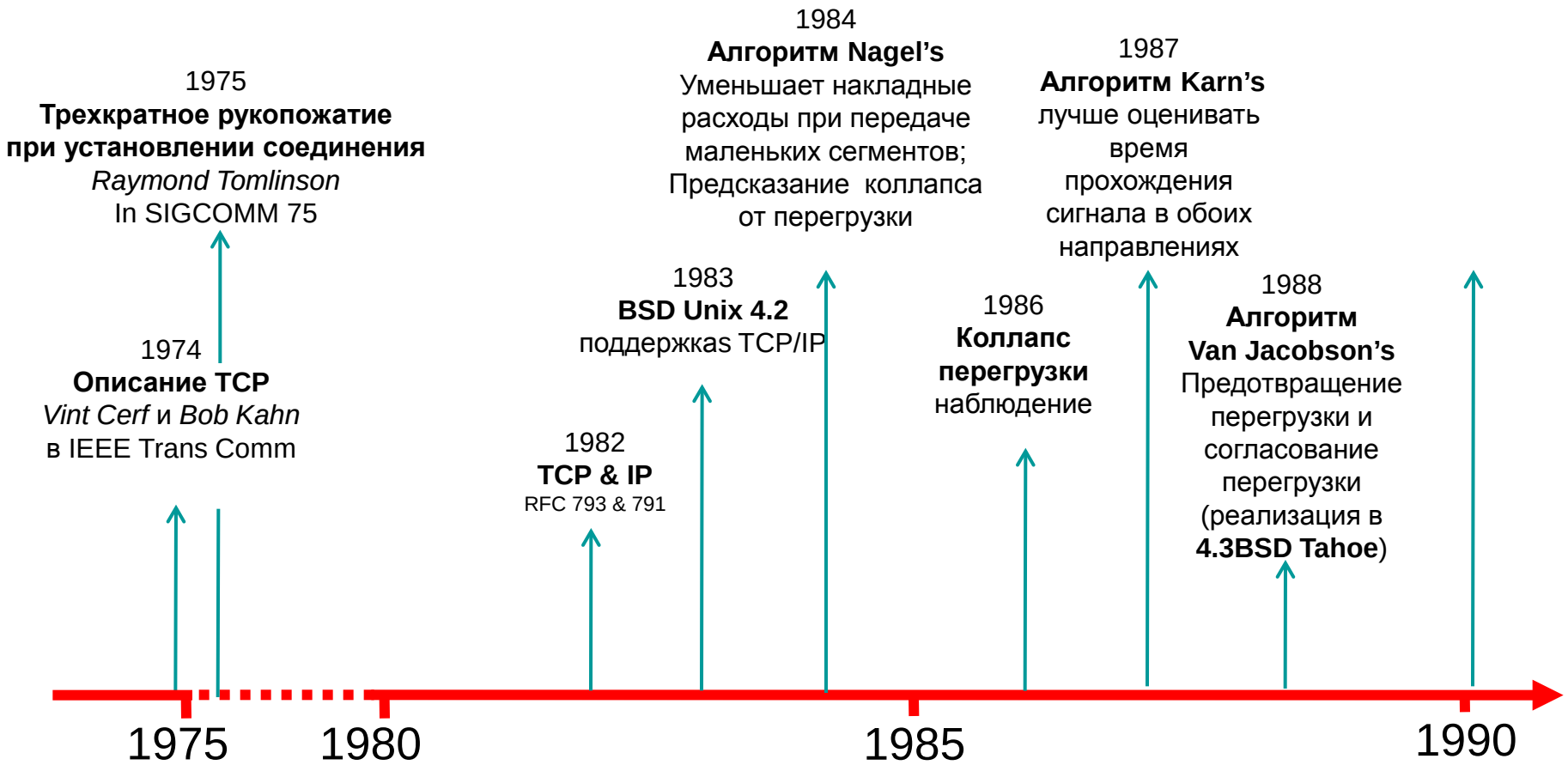
TCP –> L4 OSI RM


L4 OSI RM TCP-протокол, ориентированный на соединение
IP Host A
IP Host B
TCP соединение (Транспортный канал)
4
4
M
M
Router 1
Router 2
IP-сеть передачи данных

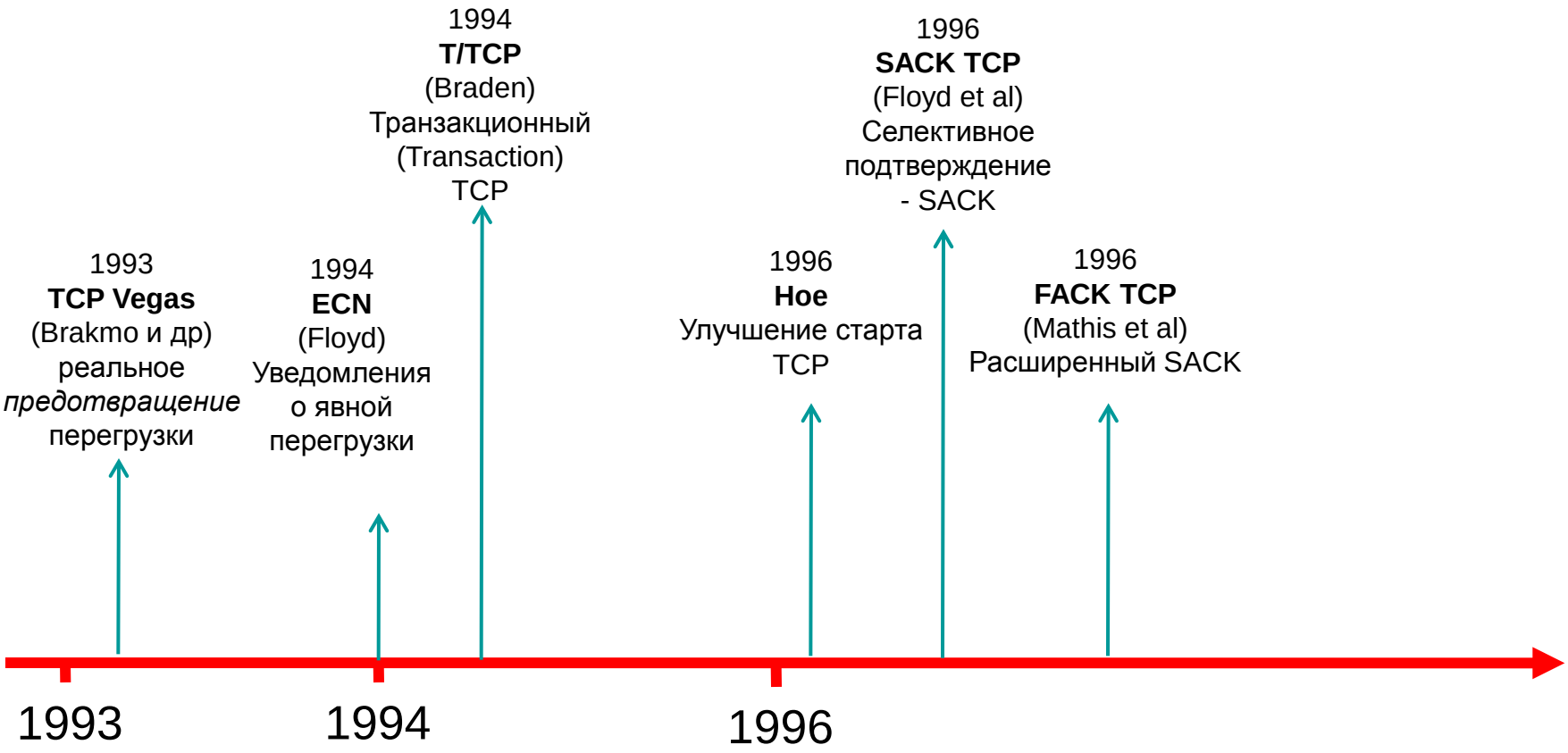
TCP PDU = сегменты -> сообщение

- **TCP протокольный блок данных (TCP PDU) называется “сегментом”**
- **Сегмент**
  - Инкапсулируется в IP-пакет
  - Идентифицируется в IP номером 6 в поле “протокол”
  - Максимальный размер сегмента (MSS) зависит от максимального размера пакета (L3) или кадра (L2) (фрагментация допускается)
- **Доставка последовательности сегментов осуществляется в виде байтового потока**
  - Под байтовым потоком в TCP понимается то, что один примитив, например, read или write может вызвать посылку адресату последовательности сегментов, которые образуют некоторый блок данных (сообщение)
- **Прикладные процессы взаимодействуют с TCP-модулем через TCP-порты**

Эволюция TCP



TCP в 1990-х годах



Фазы работы TCP

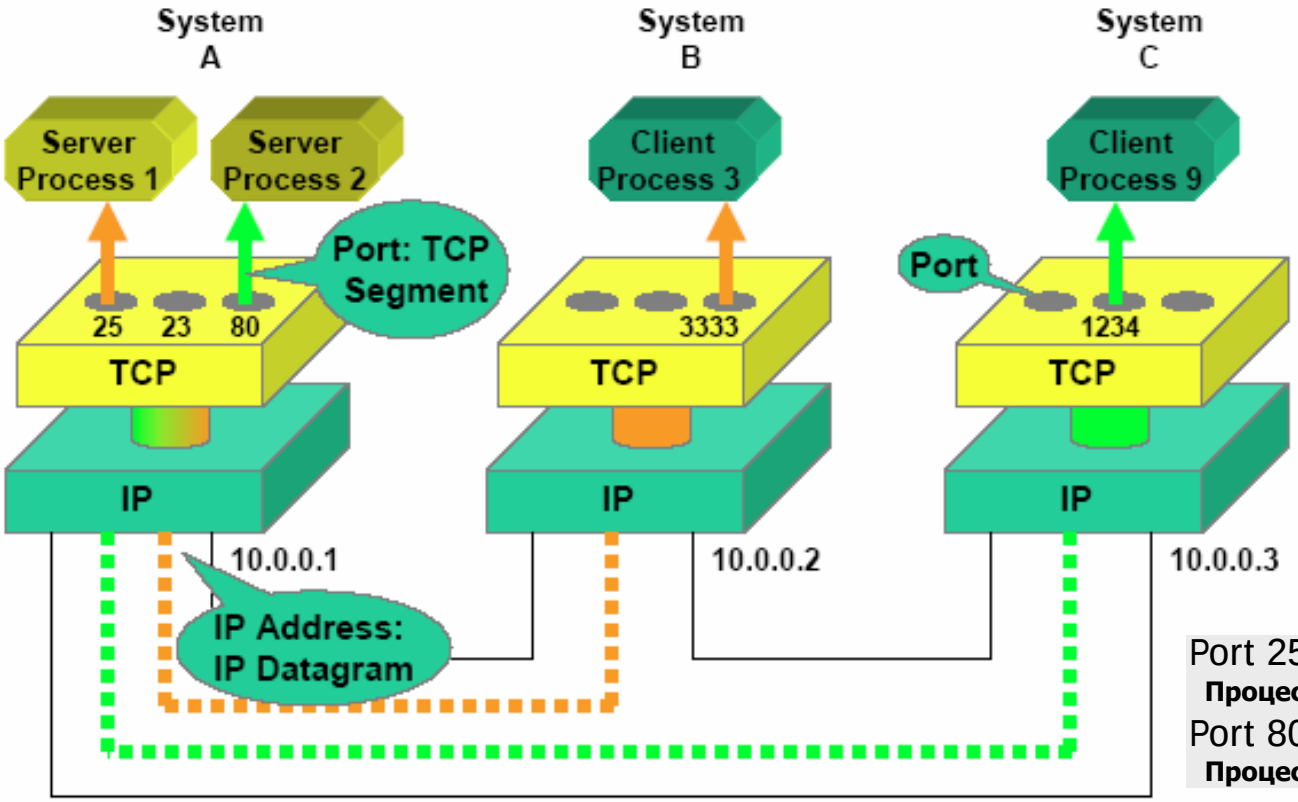
- **TCP ориентирован на соединение и использует ARQ**
  - Фаза «Установление соединения» выполняется “троекратным рукопожатием”
  - Фаза «Передача данных» - согласно механизма непрерывного ARQ (скользящее окно):
    - ✓ нумерация сегментов “с точностью до байта”
    - ✓ контрольная сумма сегмента
    - ✓ динамическое управление размером окна (размер окна адаптируется в процессе передачи сегментов к возможности сети передачи данных)
  - Фаза «Расторжение соединения» закрывает соединение, освобождая ресурсы оконечных систем

TCP гарантирует доставку сообщения

- **TCP гарантирует доставку сообщения (сообщение = последовательность сегментов)**
  - Восстанавливает сегменты, которые искажаются, теряются, дублируются или доставляются в беспорядке
  - Использует контрольные суммы сегментов для проверки их целостности проверочным суммированием (CRC)
  - Присваивает последовательный номер любому передаваемому байту данных (нумерация сегментов с точностью до байта)
  - Использует механизм положительных подтверждений от TCP-получателя о приеме им сегментов данных
- ***Вывод***
  - TCP освобождает вышележащие уровни от проблем сетей передачи данных

TCP порт и соединения

- **TCP предоставляет сервис вышележащим уровням через TCP-порты**
  - TCP-порт согласно OSI RM является TCP SAP (сервисной точкой доступа)
  - TCP посредством портов обслуживает множество прикладных процессов одновременно
  - TCP-порты мультиплексирует / демультиплексирует TCP-соединения между прикладными процессами

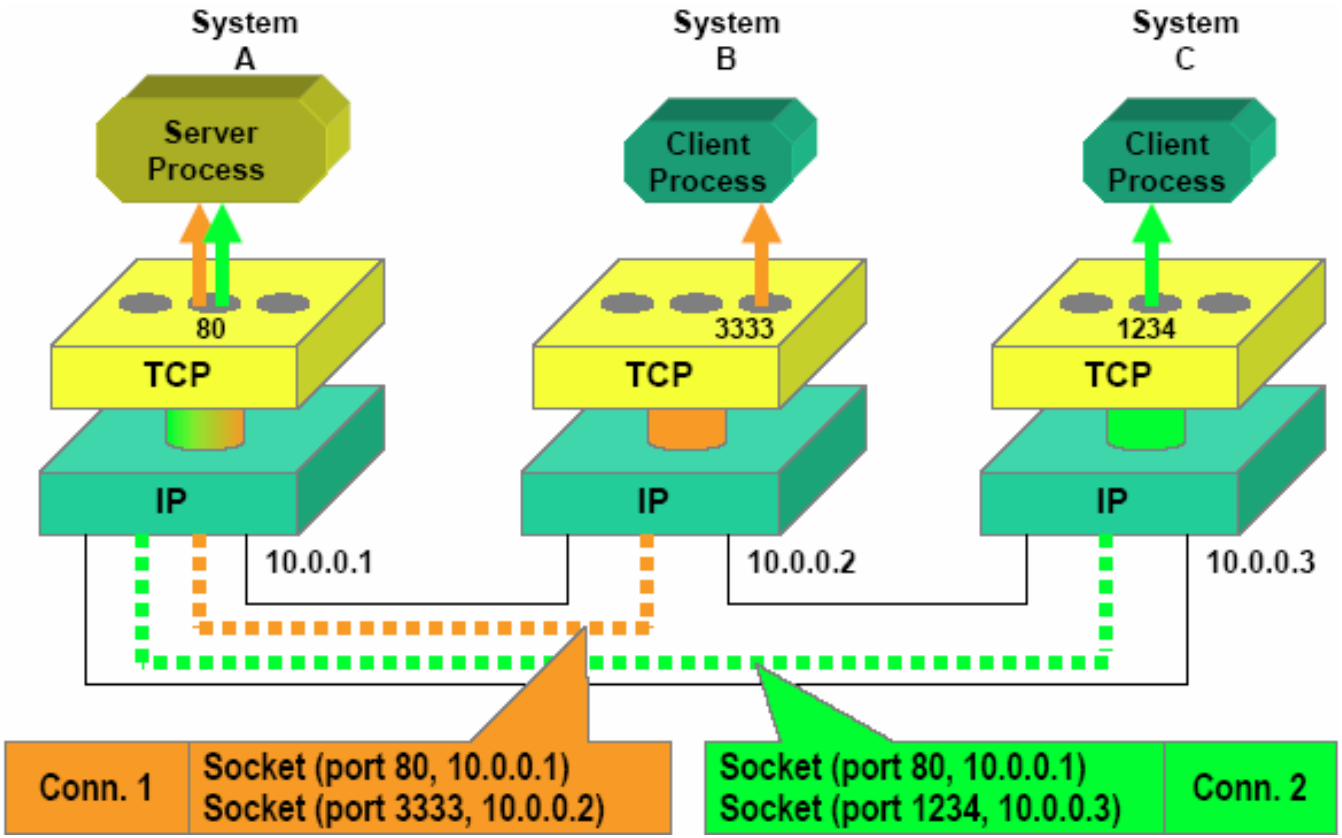


- Каждому сетевому процессу в конечной системе назначается TCP-порт
- Сетевой процесс идентифицируется номером порта

Port 25 в системе A:
Процесс 1 в системе A ↔ Процесс 3 в системе B
Port 80 в системе A:
Процесс 2 в системе A ↔ Процесс 9 в системе C

TCP сокеты и соединения

- **В среде “клиент-сервер” сервер должен поддерживать одновременно несколько сеансов с различным адресатами (клиентами)**
  - Поэтому через один порт сервера должны мультиплексироваться несколько соединений посредством гнезд (Sockets)
  - “Socket” (сокет) – это комбинация IP-адреса и номера порта

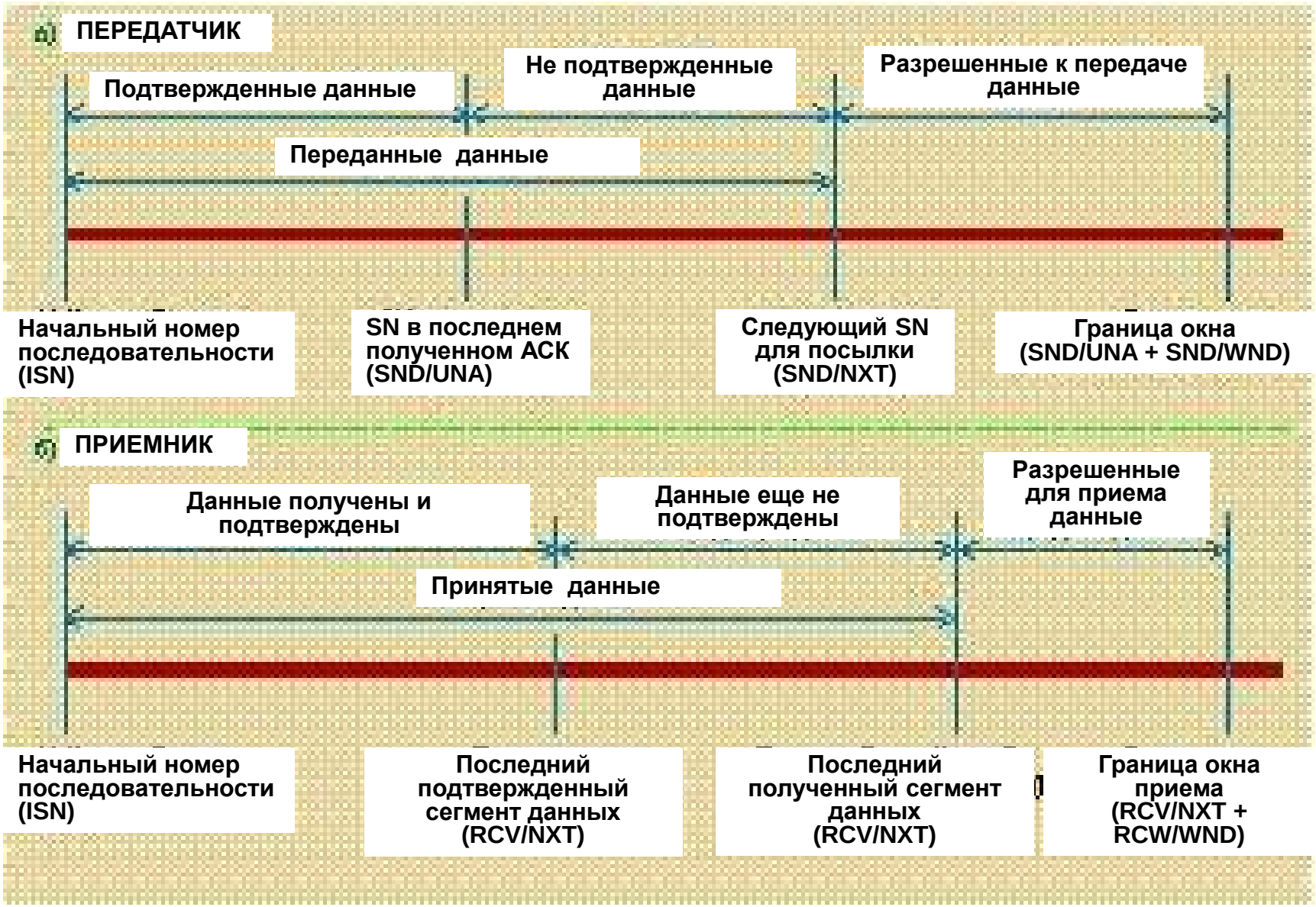


- Соединение идентифицируется парой сокетов (парой составных адресов: port и IP)
- Пара сокетов определяет конечную точку соединения OSI RM (CEP - **C**onnection **E**nd**p**oint) в сервисной точки доступа (SAP)
- Информация о соединениях хранится в TCB-таблицах

TCP сокеты и соединения

- **Информация, необходимая для формирования и поддержки соединения, каждый раз при установлении соединения создается в структуре данных, называемой блоком управления передачей (Transmission Control Block, TCB)**
- **TCB поддерживает ряд переменных, определяющих очередность отправления и получения сегментов. К ним, в частности, относятся:**
  - SND.UNA — не подтвержденные сегменты
  - SND.NXT — номер следующего сегмента
  - SND.WND — окно передаваемых сегментов
  - RCV.NXT — номер ожидаемого сегмента
- **Чередование этапов отправки и приема данных показано на рис. 2.**

TCP сокеты и соединения



TCP сокеты и соединения

- **Рассмотрим пример использования некоторых переменных**
  - Отправитель, используя переменную SND.NXT, контролирует отправку очередного сегмента из очереди
  - Получатель с помощью переменной RCV.NXT отслеживает номер следующего поступающего сегмента
  - В поле переменной SND.UNA отправитель помещает последний номер сегмента, который уже отправлен, но еще не подтвержден
  - При отсылке нового сегмента отправитель увеличивает значение своей переменной SND.NXT
  - Получатель при приеме этого сегмента увеличивает значение своей переменной RCV.NXT и посылает подтверждение. При получении подтверждения увеличивается значение переменной SND.UNA отправителя.
  - Разность величин SND.NXT и SND.UNA может служить мерой задержки сегментов в сети. Переменные изменяются на величину длины поля данных сегмента. Более подробно формат заголовка, назначение полей, флаги, возможные состояния соединения и переменные описаны в RFC 793.

TCP нумерация портов

- **“Общеизвестные” порты для известных всем сетевых приложений и сервисов (WWW, FTP, Telnet и т.д.). Диапазон 0 – 1023, контролируются IANA**
- **Порты «специфических приложений» (Lotus, Oracle, Cisco и т.д., RFC1700). Начинаются с номера 1024, регистрируются в IANA**
- **Клиентские приложения выбирают для нумерации своих портов любые числа**

TCP нумерация портов

Общеизвестные порты:

7	Echo
20	FTP (Data), File Transfer Protocol
21	FTP (Control)
23	TELNET, Terminal Emulation
25	SMTP, Simple Mail Transfer Protocol
53	DNS, Domain Name Server
69	TFTP, Trivial File Transfer Protocol
80	HTTP Hypertext Transfer Protocol
111	Sun RPC, Sun Remote Procedure Call
161	SNMP, Simple Network Management Protocol
162	SNMPTRAP

Регистрируемые порты:

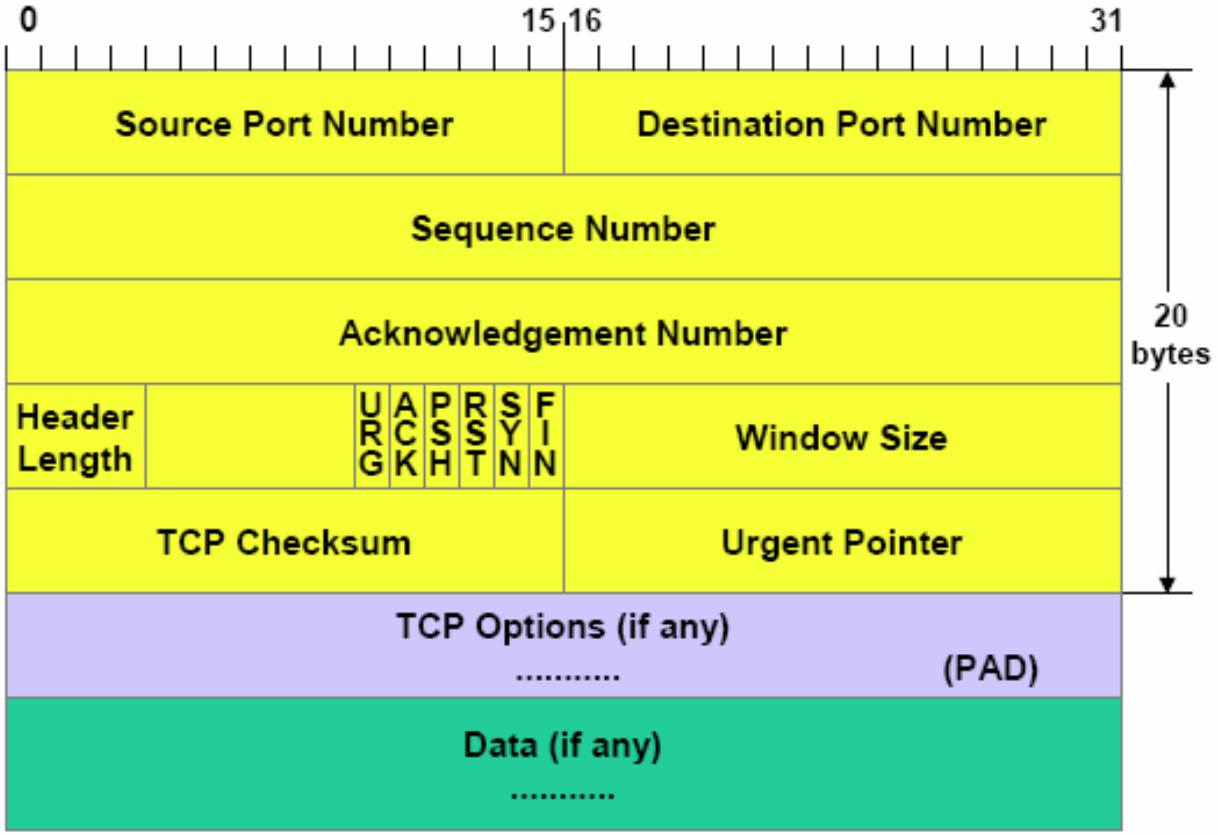
1416	Novell LU6.2
1433	Microsoft-SQL-Server
1439	Eicon X25/SNA Gateway
1527	oracle
1986	cisco license managmt
1998	cisco X.25 service (XOT)
6000	\
.....	> X Window System
6063	/

... etc.

(see RFC1700)

Формат TCP-заголовка

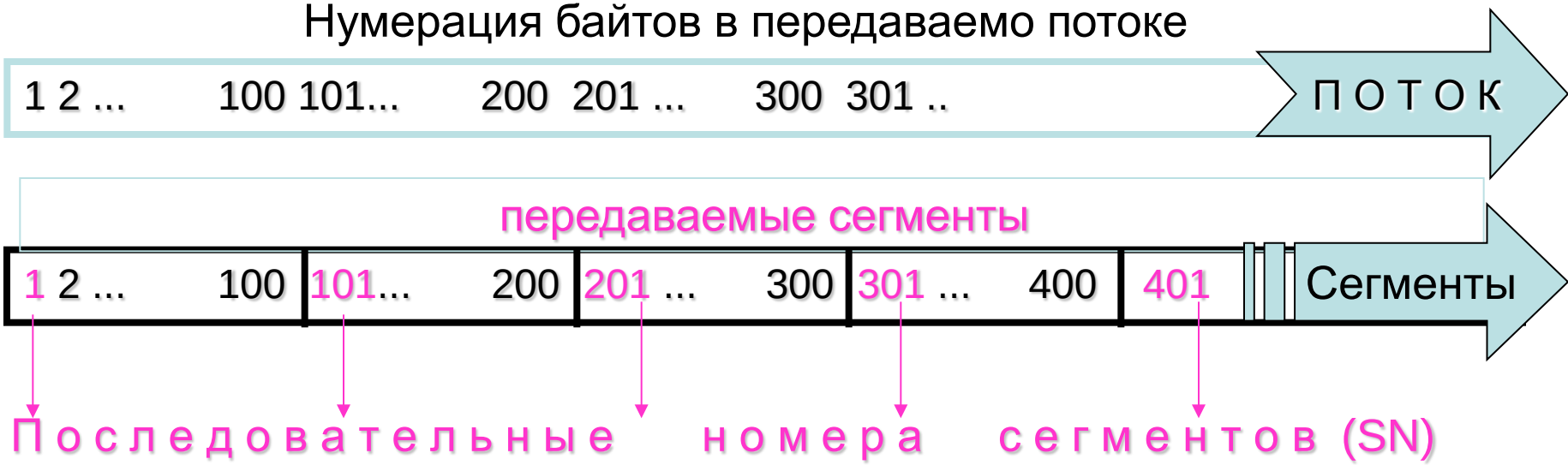
- **Source/Destination Port Number –** порт источника/приемника
- **Sequence number (SN)–** последовательный номер (сегмента)
- **Acknowledgement Number (AckN) –** подтверждаемый номер
- **Head Length –** длина заголовка в 32-разрядных словах
- **Windows Size –** размер окна в байтах при флаге ACK=1



- **Checksum –** контрольная сумма
- **Urgent Pointer –** указатель срочных данных
- **TCP Options** - опции
- **PAD** – заполнитель

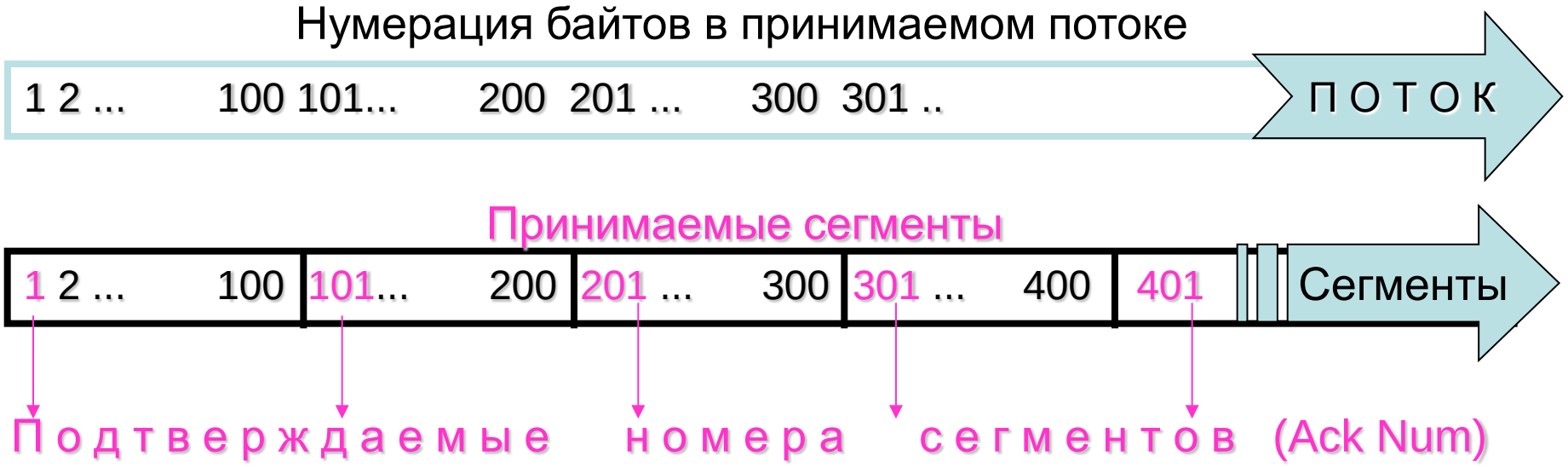
Поля TCP-заголовка

- **Sequence Number (SN) – последовательный номер сегмента**
  - Позиция первого байта этого сегмента в потоке данных (после достижения максимального значения возвращается к 0)
  - при syn=1 в этом поле код ISN (подробно дальше)
  - позволяет однозначно идентифицировать байт данных в потоке данных
  - сегменты с флагами SYN и FIN также последовательно нумеруются
    - ✓ Наименьший номер байта следует сразу за заголовком



Поля TCP-заголовка

- **Acknowledgement Number (AckN) – подтверждаемый номер сегмента**
  - положительное подтверждение корректного приема всех байт потока, с номерами меньше AckN
  - ожидаемый байта с номером AckN в потоке принимаемых данных
  - используется с флагом ACK для положительного подтверждения “всех предшествующих байт”



Поля TCP-заголовка

- **Windows Size – размер окна (в байтах, ACK=1)**
  - устанавливает источник в каждом передаваемом сегменте, чтобы сообщить о своем текущем размере “окна приема”
  - "динамическая работа с окнами" позволяет управлять потоком принимаемых данных
  - размер окна показывает число байт, которые можно передать не ожидая подтверждения
  - размер окна может изменятся на протяжении всего времени существования TCP-соединения
  - размер окна соотносится с имеющимися у TCP-передатчика ресурсами
  - размер окна используется для реализации процедур управления потоком (“медленный старт” или полная приостановка потока при размере окна равном нулю)
  - при нулевом окне приема разрешается только прием АСК-сегментов, поскольку блокируется посылка данных

Поля TCP-заголовка

- **Urgent Pointer – указатель срочных данных**
  - Используется с флагом URG=1
  - Показывает последний байт срочных данных
  - Требует немедленного реагирования на прикладном уровне
  - Примечание: исчерпывающего ответа на вопрос, что такое срочные данные, в литературе нет. Возможно разработчики хотели обеспечить передачу некоторых данных “вне основной полосы пропускания”

Поля TCP-заголовка (Флаги)

Бит	Обозначение битов (слева на право) поля *флаги*	Значение бита, если он равен 1
7	NS	
8	CWR	
9	ACE	
10	urg	Флаг важной информации, поле *Указатель важной информации* имеет смысл, если urg=1.
11	ack	Номер октета, который должен прийти следующим, правилен.
12	psh	Этот сегмент требует выполнения операции push. Получатель должен передать эти данные прикладной программе как можно быстрее.
13	rst	Прерывание связи.
14	syn	Флаг для синхронизации номеров сегментов, используется при установлении связи.
15	fin	Отправитель закончил посылку байтов.

Поля TCP-заголовка (Флаги)

- **SYN - флаг синхронизации**
  - Используется в фазе установления соединения (примитив “connect request”)
  - Если SYN=1, в поле SN содержится начальное значение нового соединения
- **FIN – флаг окончания (примитив “disconnect request”)**
  - Используется в фазе расторжения соединения
  - Означает передачу последнего байта потока
- **ACK - флаг подтверждения**
  - Если установлен, то номер ожидаемого байта потока в поле “подтверждаемый номер” и подтверждение приема всех предшествующих байт потока
- **RST – флаг сброса соединения (прерывание соединения)**
  - Если установлен, то соединение должно быть немедленно расторгнуто
  - Может быть использован для отказа попытки соединения или "уничтожения" текущего соединения

Поля TCP-заголовка (Флаги)

- **URG - флаг важной информации**
  - Извещает о нахождении в сегменте срочных данных
  - В поле “указатель срочных данных” показано место последнего байта срочных данных (вариант 1)
    - ✓ Последовательный номер последнего байта срочных данных = последовательный номер сегмента + указатель срочных данных
    - ✓ RFC793 и некоторые реализации воспринимают срочный указатель как указывающий на первый байт после срочных данных; Однако, "ведущие требования RFC1122 объявляют это ошибкой” (вариант 2)
    - ✓ Существует мнение, что “нет никакого способа указать начало срочных данных”
  - Когда модуль TCP принимает сегмент с URG=1, то уведомляет об этом приложение, которое переключается в “срочный режим” до прихода последнего байта срочных данных.
  - Пример использования: клавиша прерывания (DEL, Ctrl c) в Telnet, Rlogin и FTP

Поля TCP-заголовка (Флаги)

- **PSH – флаг «выталкивание»**
  - Указание принимающему модулю немедленно передать данные приложению
  - Сегмент должен быть отправлен вышележащему уровню немедленно без буферизации
  - TCP решает самостоятельно, когда послать данные следующему уровню. Одна стратегия состоит в накоплении данных в буфере и отправлении при достижении некоторого объема
  - Приложение, которому нужно низкое время ожидания или постоянный поток, хотело бы обойти этот буфер с флажком PSH
  - Также последний сегмент соединения хотел бы использовать это флаг
  - Сегодня часто игнорируется

Поля TCP-заголовка

- **Checksum – контрольная сумма**
  - включает: TCP-заголовок + TCP-данные + псевдо заголовок IP (12 байт)
  - псевдо заголовок IP содержит: IP-адреса источника и приемника, поле “тип протокола” и общую длину сегмента
  - это гарантирует, что не только порт, но и сокет включен в контрольную сумму
  - Включение псевдо заголовка IP в контрольную сумму позволяет TCP-уровню обнаружить ошибки, которые могут быть не опознаны IP (например, IP передает безошибочные TCP-сегменты к “неправильным” IP конечным системам

Поля TCP-заголовка

- **Опции**
  - Необязательное поле
  - Дополняется до кратного 32-бит с помощью поля PAD (заполнитель)
  - В настоящее время определены опции:
    - ✓ Конец списка опций
    - ✓ Никаких операций. Используется для заполнения поля опции до числа октетов, кратного 4
    - ✓ Максимальный размер сегмента (MSS)
    - ✓ Масштаб окна
    - ✓ Временная метка

Поля TCP-заголовка

Примеры опций

- **Вид - код опции**
- **LEN - число байт включая поля вид и LEN**

ОПЦИИ

- **0: End of Option List - конец списка опций**
  - Этот тип указывает на завершение списка опций, которое может не совпадать с концом заголовка TCP, определяемым полем Data Offset. Эта опция указывается после завершения всех опций, а не какой-то отдельной опции, но ее необходимо использовать лишь в тех случаях, когда точки завершения заголовка TCP и списка опций не совпадают. Опция определена в RFC 793 [2].С этой опцией не связано никаких данных, поэтому схеме компрессии достаточно просто закодировать присутствие опции. Однако, следует помнить о возможном присутствии после опции битов заполнения для выравнивания заголовка TCP по 4-октетной границе (биты заполнения имеют значение 0 в соответствии с документом [2]).
- **1:** No-Operation **– нет операции**
  - Эта опция может включаться между другими опциями заголовка (например, для выравнивания следующей опции по границе слова). Использование такого выравнивания на передающей стороне не гарантируется, поэтому получатель должен быть готов к обработке опций, которые начинаются не на границе слова (RFC 793 [2]). С этой опцией не связано никаких данных, поэтому схеме компрессии достаточно показать наличие опции. Это можно сделать путем обозначения наличия выравнивания и необходимости передачи информации о нем. В этом случае биты заполнения могут быть удалены.

ОПЦИИ

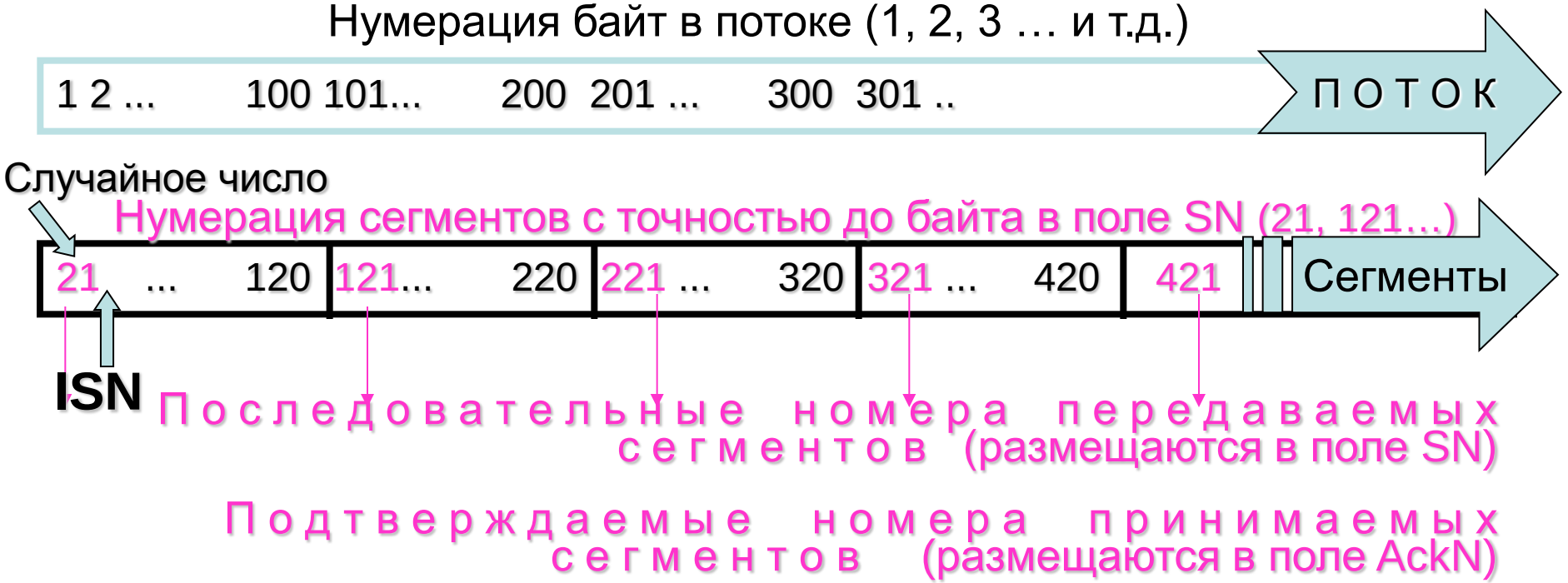
- 4 Maximum Segment Size
- 3 WSopt - Window Scale
- 2 SACK Permitted
- SACK
- 6 Echo (отменено опцией 8)
- 7 Echo Reply (отменено опцией 8)
- 8 TSopt - Time Stamp Option
- 9 Partial Order Connection Permitted
- 10 Partial Order Service Profile
- 11 CC
- 12 CC.NEW
- 13 CC.ECHO
- 14 Alternate Checksum Request

Фаза «Установление соединения»

Установление соединения

последовательная нумерация

- В этой фазе устанавливается начальный номер последовательности (ISN - initial sequence number), формируемый для передачи первого байта потока
- ISN не равен 1 или другому постоянному для TCP числу по следующим причинам:



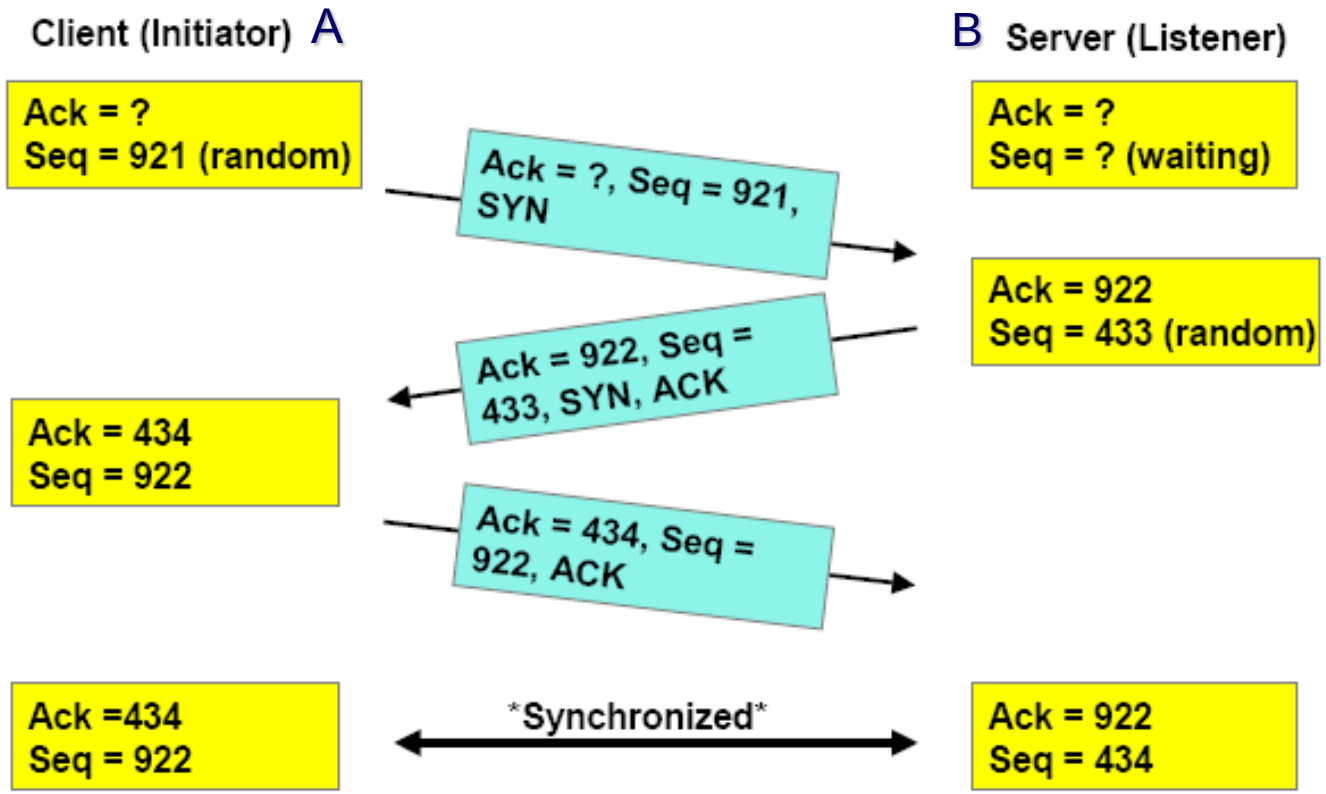
Начальный номер последовательности (ISN)?

- **Между парой устройств (парой составных адресов) возможны многократные соединения.**
  - Следовательно в сети допустимы IP-пакеты с вложенными в них TCP-сегментами уже закрытых соединений
  - Значит старые TCP-сегменты могут попасть в поток данных новых соединений
- **Для предотвращения этих случаев применяется**
  - специальный механизм генерации начального номера последовательности (ISN), который передается в поле SN совместно с флагом SYN в фазе установления соединения
  - ISN формируется из 32-разрядного счетчика времени, значение которого увеличивается на 1 каждые 4 мкс (RFC793)
  - Полный цикл счетчика 4,55 часа и предполагается, что время жизни любого сегмента в сети не превышает эту величину

Установление соединения (**трехкратное рукопожатие**)

- A → B SYN, ISN_A
- B → A ACK, ISN_A+1
- B → A SYN, ISN_B
- A → B ACK, ISN_B + 1

1. A → B SYN, ISN_A
2. B → A (SYN, ISN_B) + (ACK, ISN_A+1)
3. A → B ACK, ISN_B+1



Пояснения:

B -> A - Система B передает системе A сегмент, содержащий:

(SYN, ISN_B) означает SYN=1, SN=ISN_B

(ACK, ISN_A+1) означает ACK=1, AckN=ISN_A

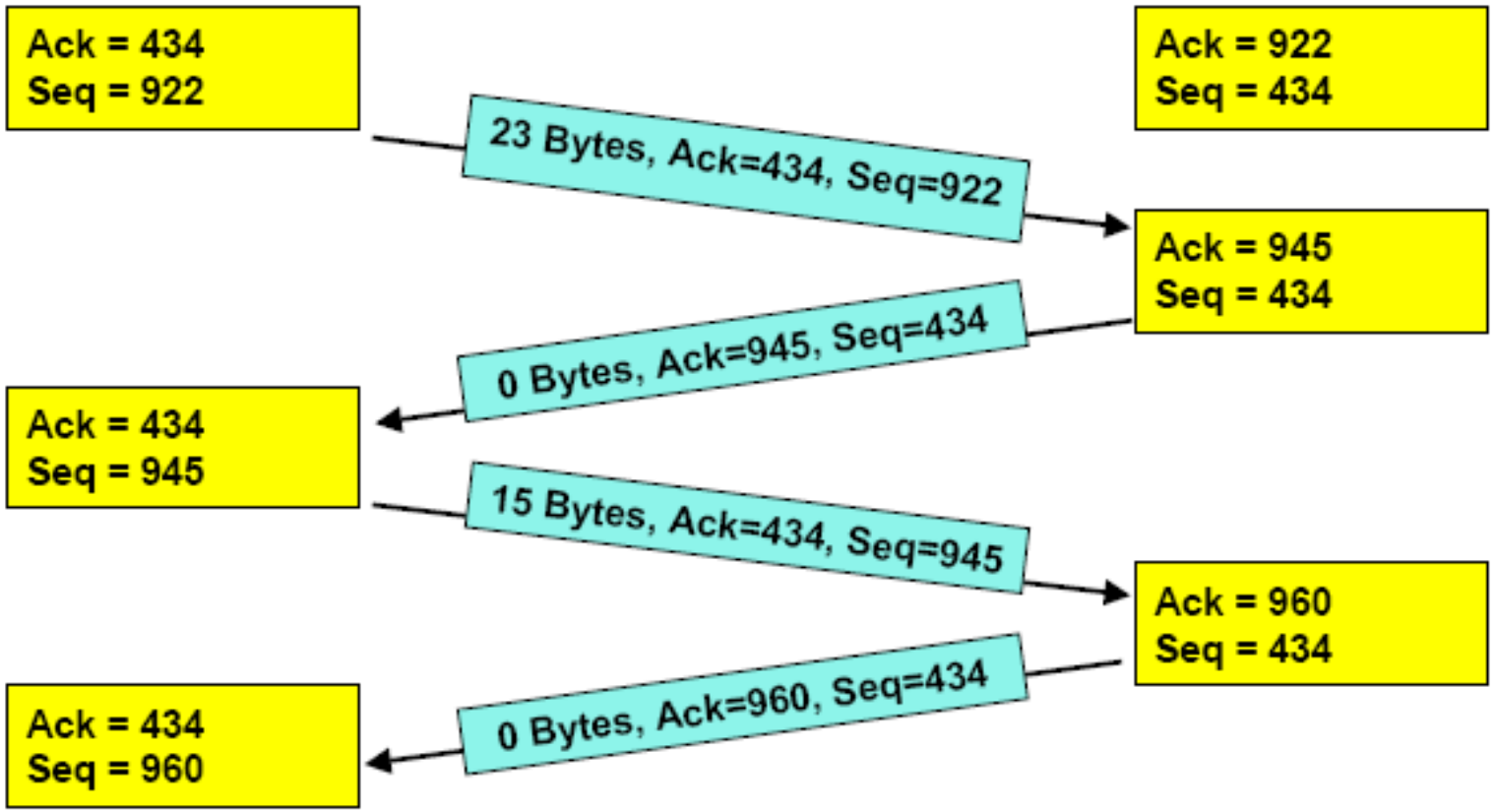
Установление соединения

максимальный размер сегмента

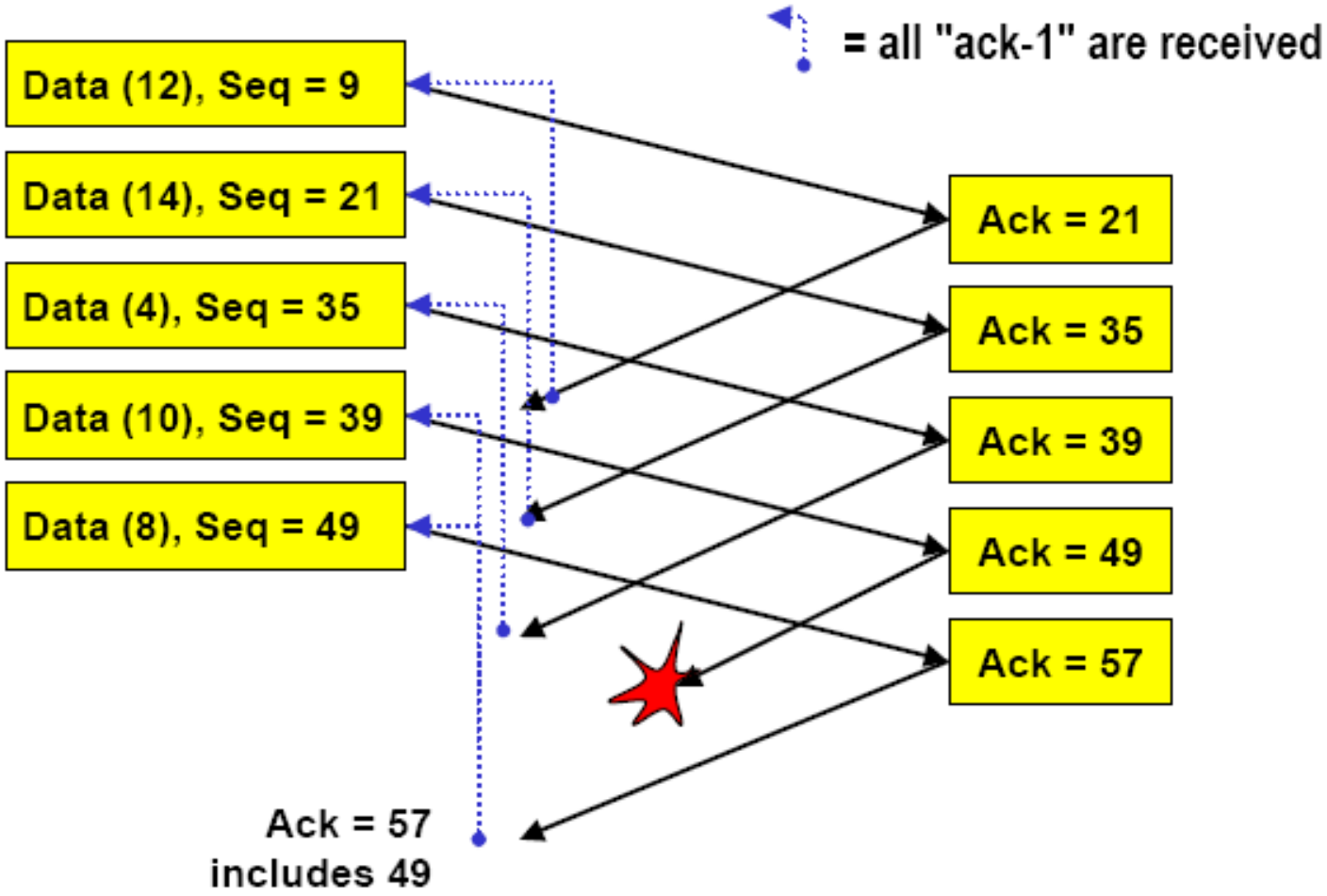
- **Максимальный размер сегмента (MSS) это самая большая порция данных, которую TCP пошлет на удаленный конец**
- **В фазе установление соединения каждой стороной объявляется свой MSS, которой она собирается принимать**
- **Если одна сторона не принимает опцию MSS от другой стороны, используется размер по умолчанию в 536 байт**
- **В этом случае, при 20-байтном IP заголовке и 20-байтном TCP заголовке, размер IP датаграммы будет составлять 576 байт**
- **В общем случае, чем больше MSS тем лучше, до тех пор пока не происходит фрагментация.**
- **Большие размеры сегмента позволяют уменьшить накладные расходы на IP и TCP заголовки**
- **может быть установлено значение MSS равное MTU исходящего интерфейса минус размер фиксированных TCP и IP заголовков. Для Ethernet MSS может достигать 1460 байт. При использовании инкапсуляции IEEE 802.3 MSS может быть до 1452 байт.**

Фаза «Передача данных»

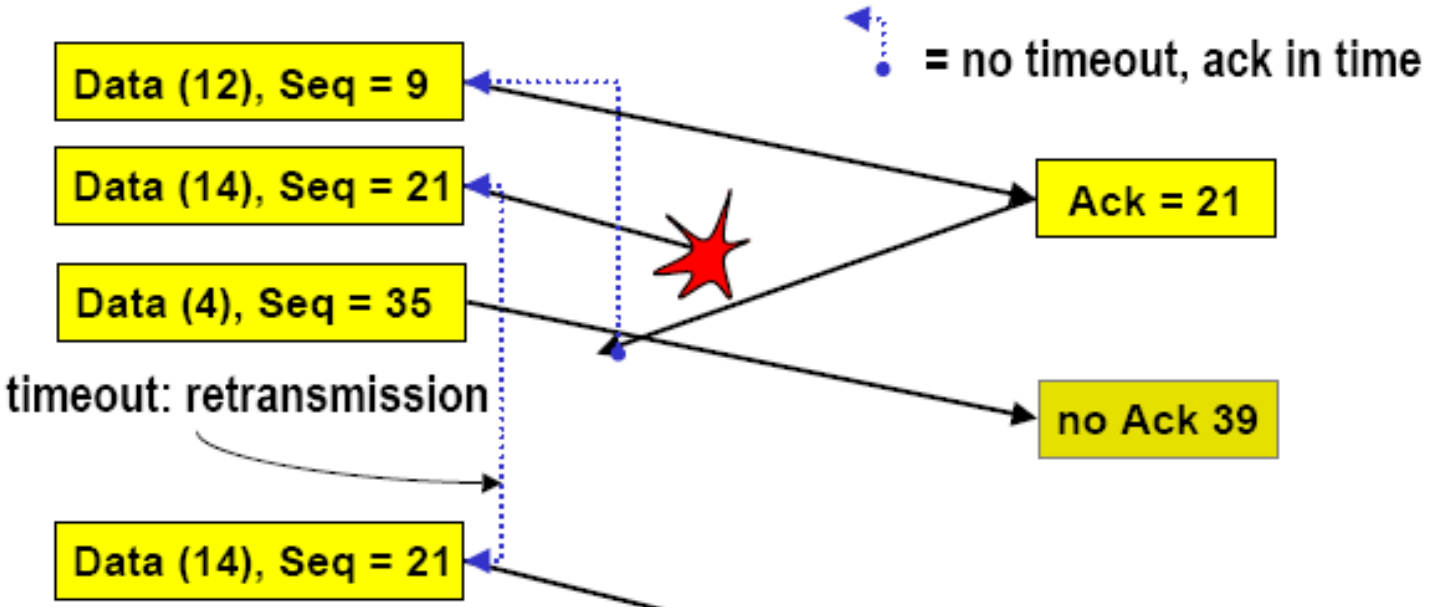
Нумерация сегментов и подтверждений



“Совокупное” подтверждение

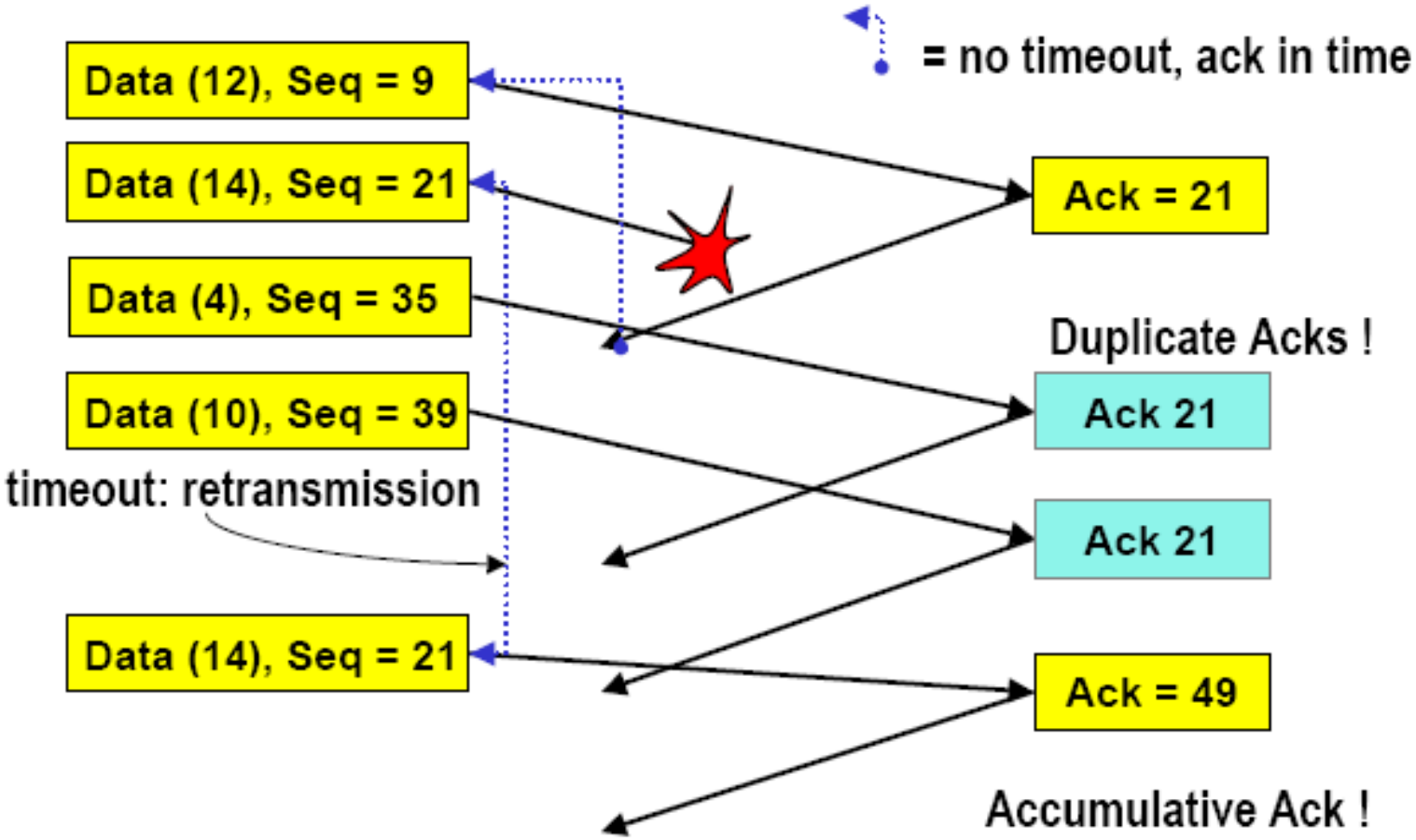


TCP дубликаты, потеря оригинала (старый TCP)

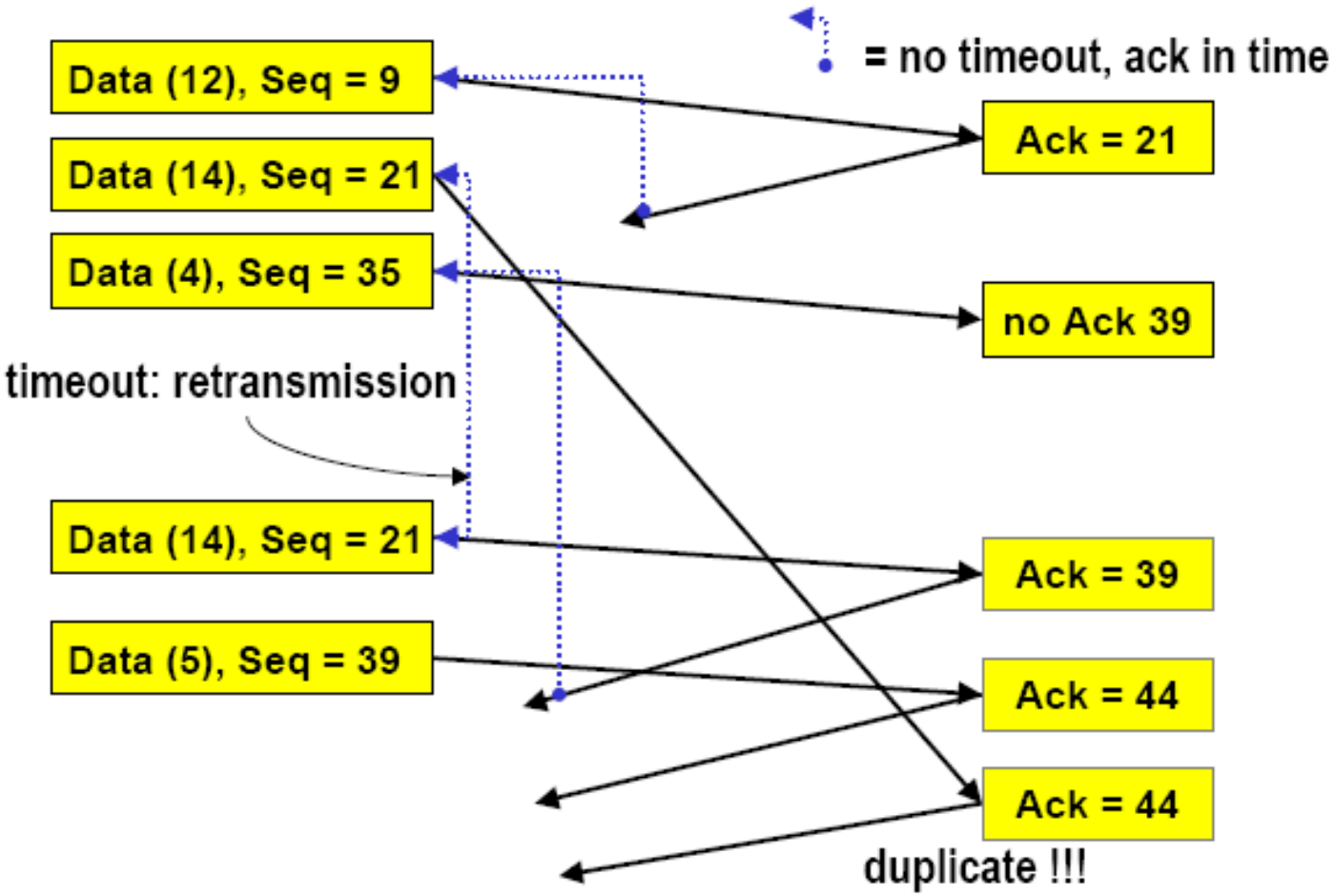


NOTE: Instead of suspending Acks, the receiver may also repeat the last valid Ack = Duplicate Ack in order to notify the sender immediately about a missing segment (hereby aiding “slow start and congestion avoidance")

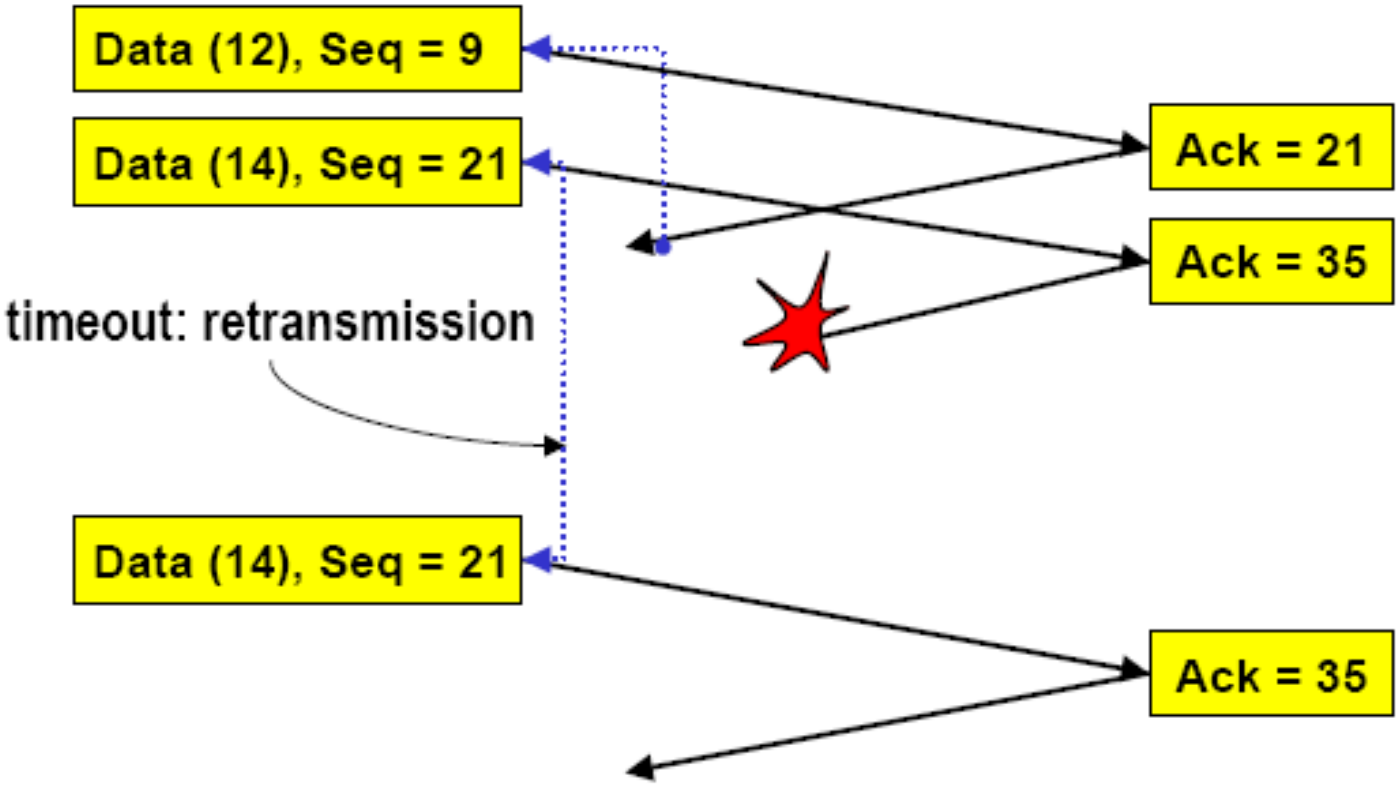
TCP-дубликаты Ack (новый TCP)



TCP-дубликаты, запоздавший оригинал



TCP-дубликаты, потеря Ack

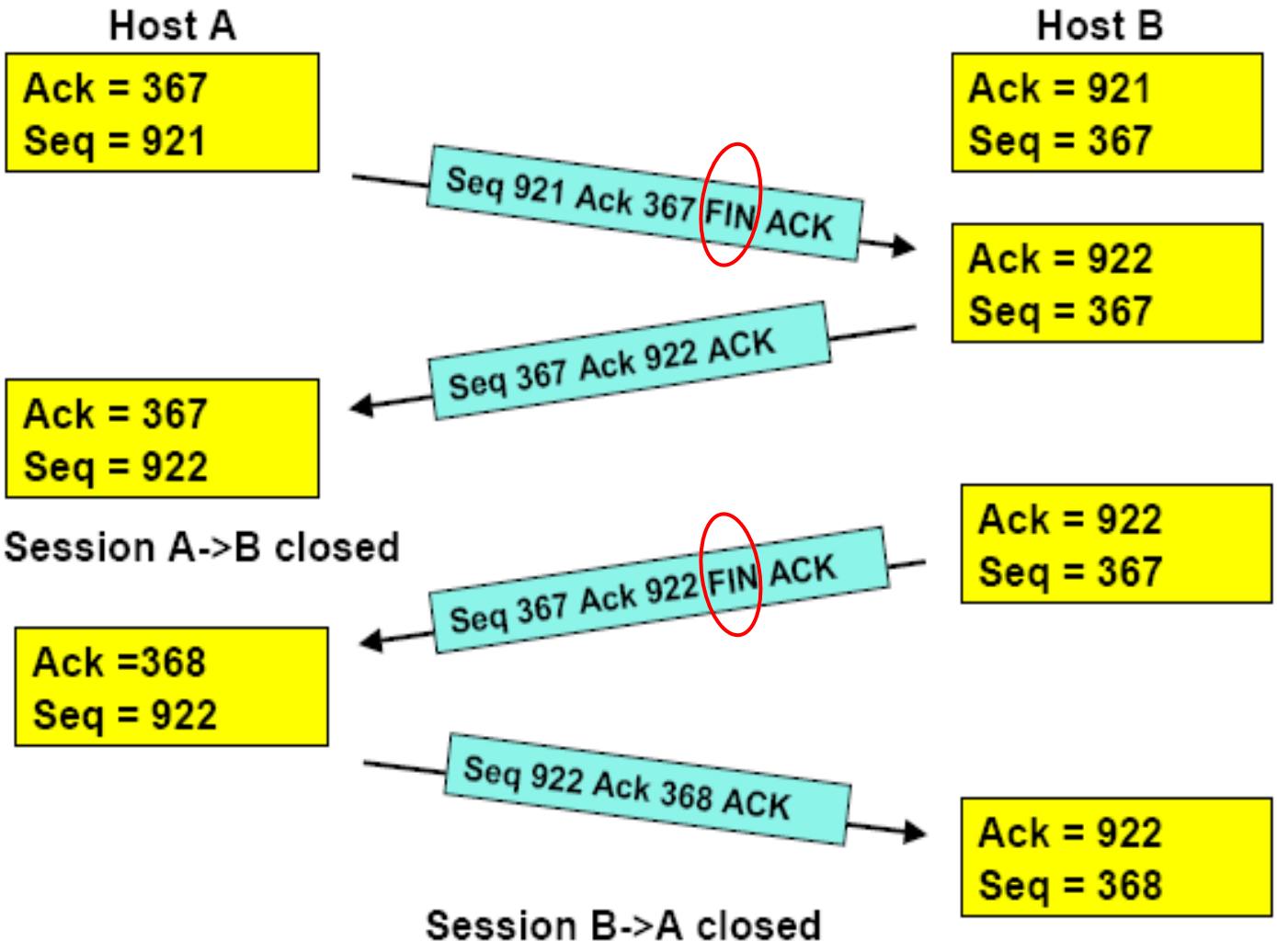


Фаза «Расторжение соединения»

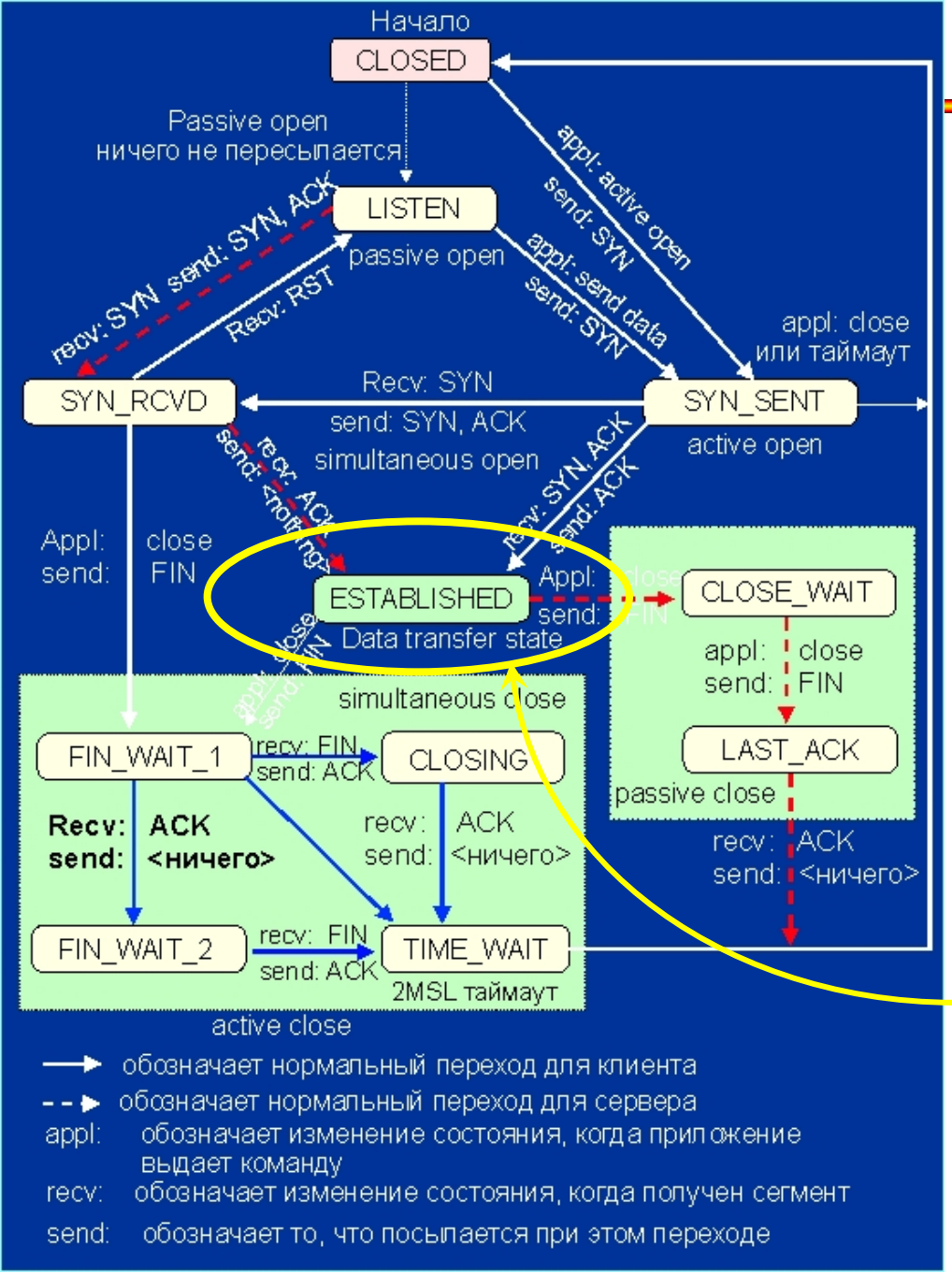
Расторжение соединения

- **Аналогично “установлению соединения”**
- **Поскольку TCP-соединение полнодуплексное, его можно рассматривать как два полудуплексных канала, каждый из которых зарывается отдельно**
- **Одна станция флагом FIN маркирует последний сегмент передаваемых данных**
- **Другая станция (партнер) подтверждает и закрывает соединение в этом направлении. При этом передача в противоположном направлении может беспрепятственно продолжаться**
- **Партнер также маркирует последний передаваемый сегмент данных флагом FIN и по получении подтверждения (ACK) соединение окончательно расторгается**
- **Обмен флагами FIN и ACK гарантирует, что обе стороны получили все байты**

Расторжение соединения



Машина состояний TCP



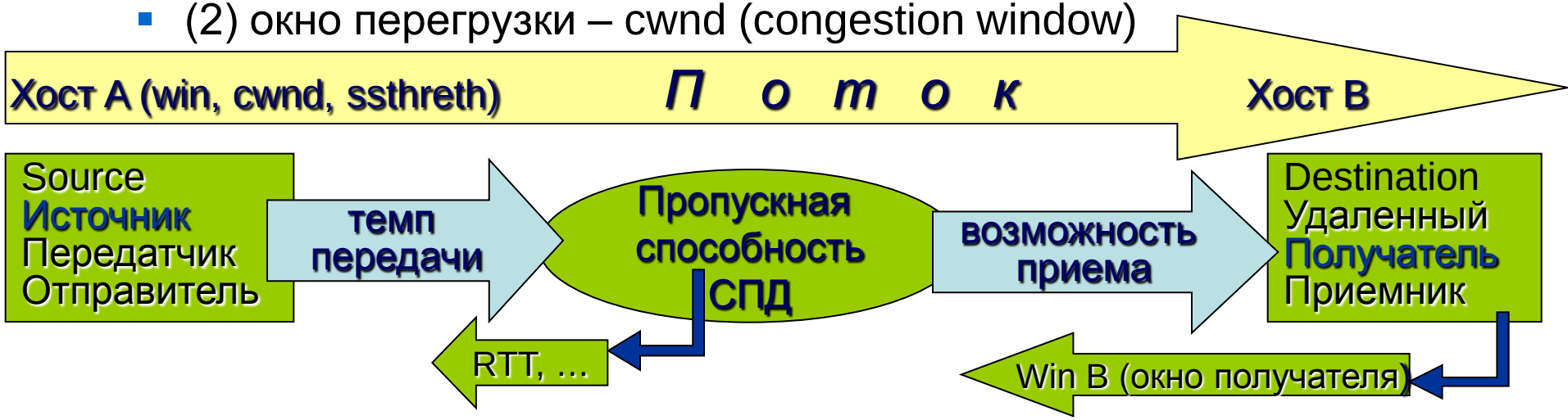
В машине состояний представлены фазы работы TCP

Не предусматривается изменения состояний при посылке или получении обычных пакетов, содержащих данные

Управление потоком

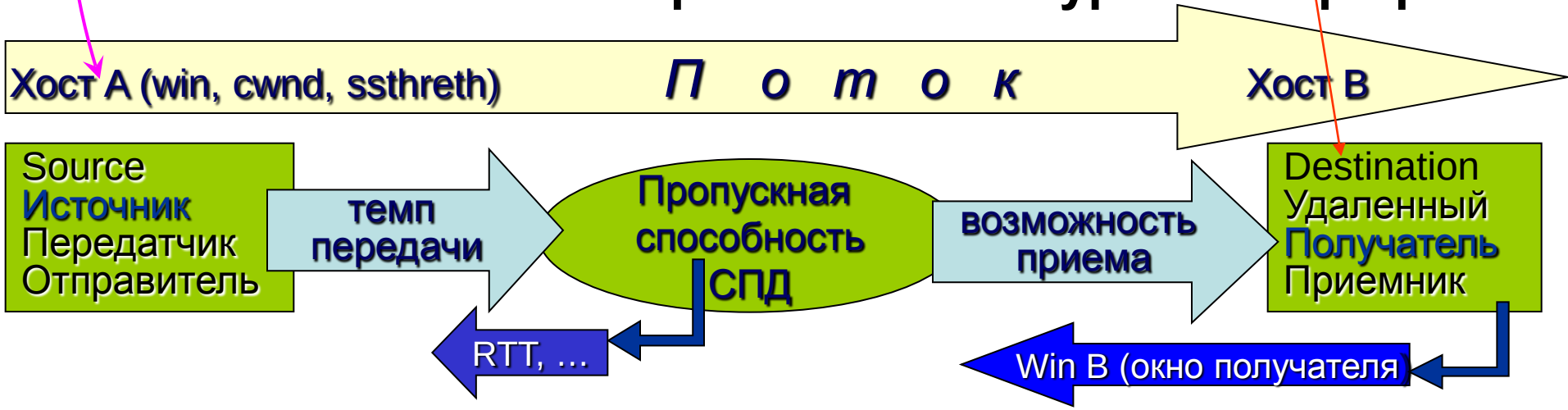
Управление потоком

- **Конечная цель регулирования трафика – установление соответствия между:**
  - (1) темпом передачи источником и возможностью приема получателем (ограниченность размера буфера или других ресурсов приемника)
  - (2) темпом передачи источником и пропускной способностью сети передачи данных (СПД)
- **С учетом этого обстоятельства каждый отправитель формирует два окна:**
  - (1) окно получателя - Win
  - (2) окно перегрузки – cwnd (congestion window)

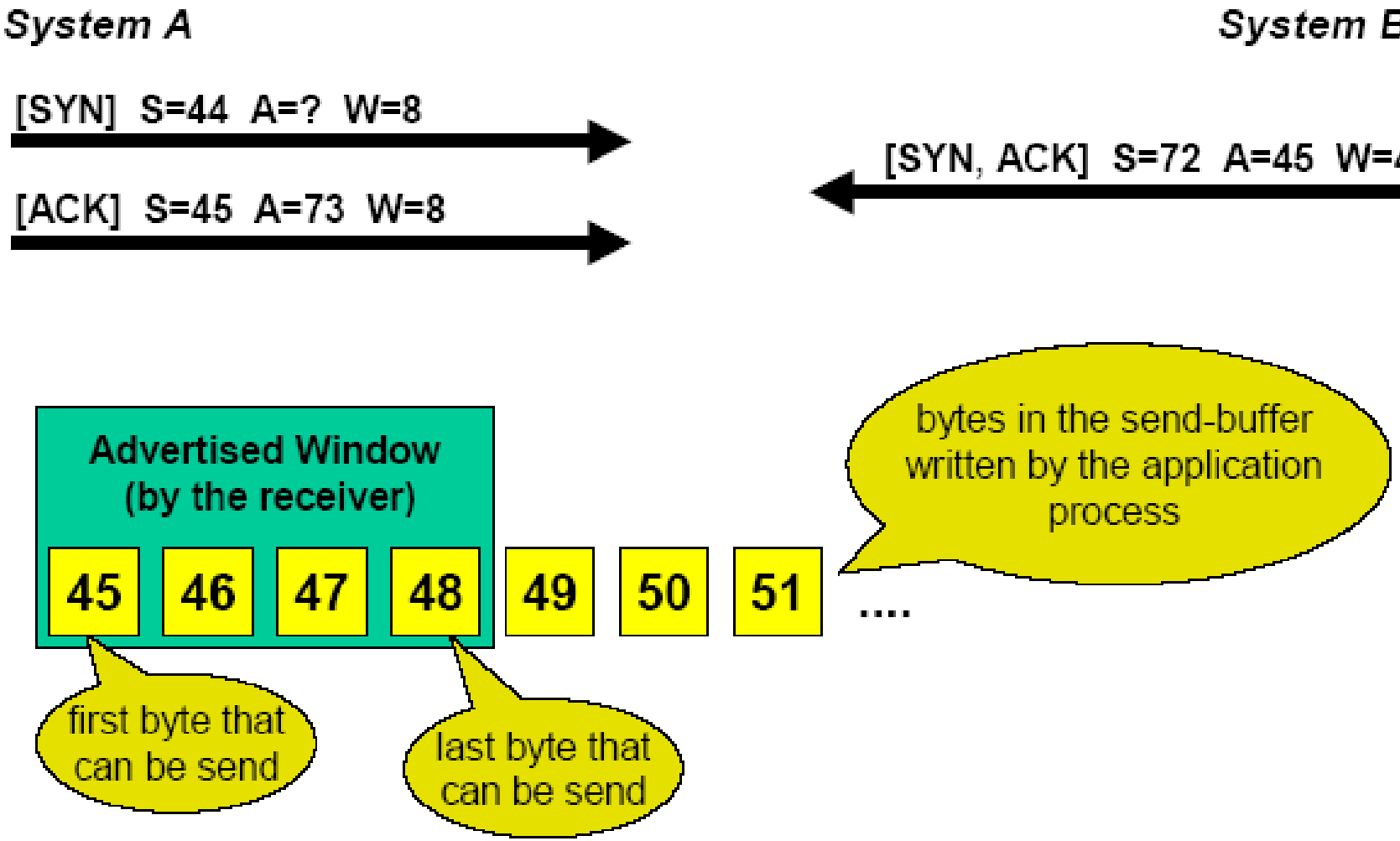


Управление потоком

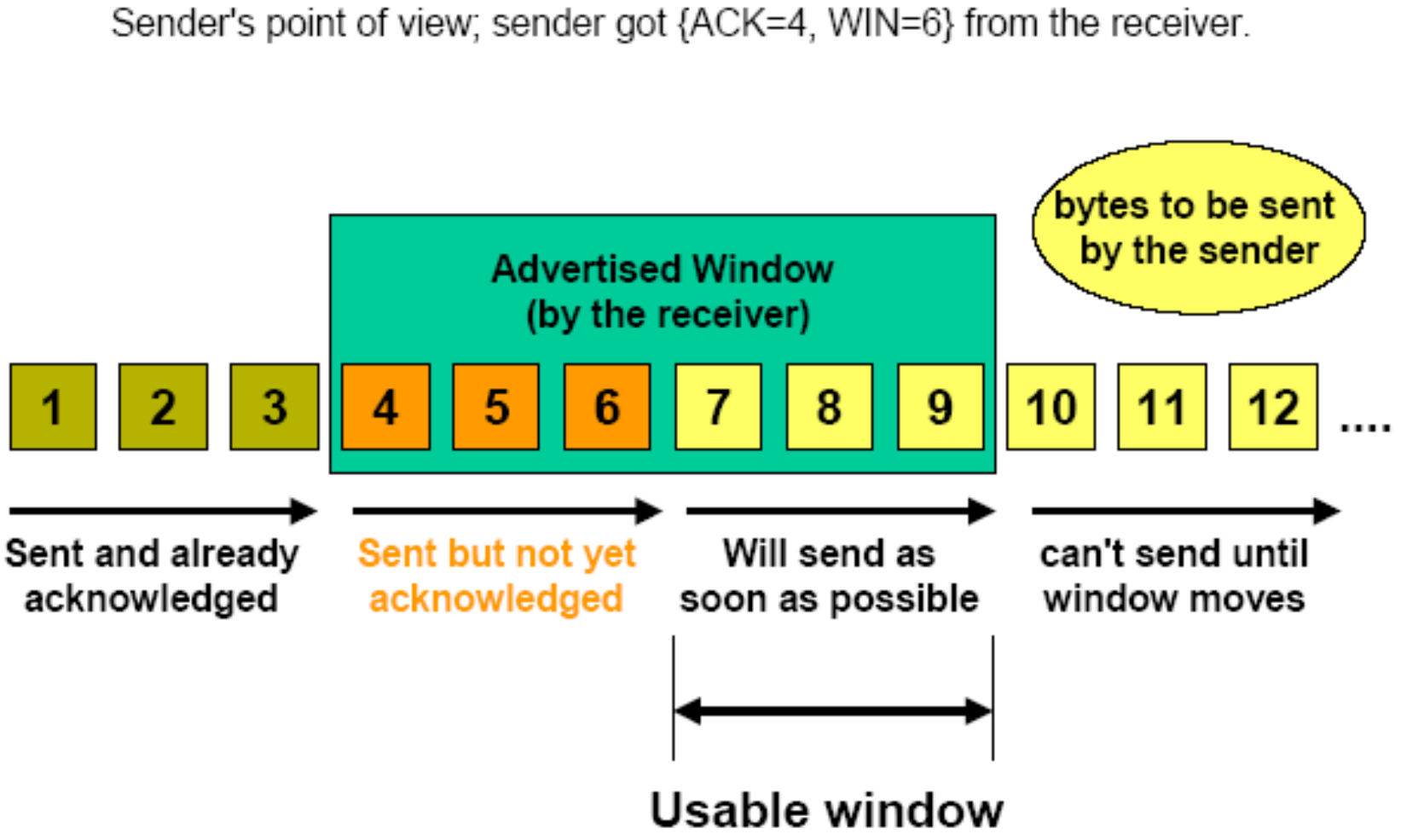
- **Подразумевается существование двух независимых процессов:**
  - контроль доставки, управляемый получателем с помощью параметра win
  - контроль перегрузки, управляемый отправителем с помощью
    - ✓ окна перегрузки cwnd (congestion window)
    - ✓ порог медленного старта- ssthreth (slow start threshold)
- **Первый процесс отслеживает заполнение входного буфера получателя**
- **Второй - регистрирует перегрузку канала, а также связанные с этим потери и понижает уровень трафика**



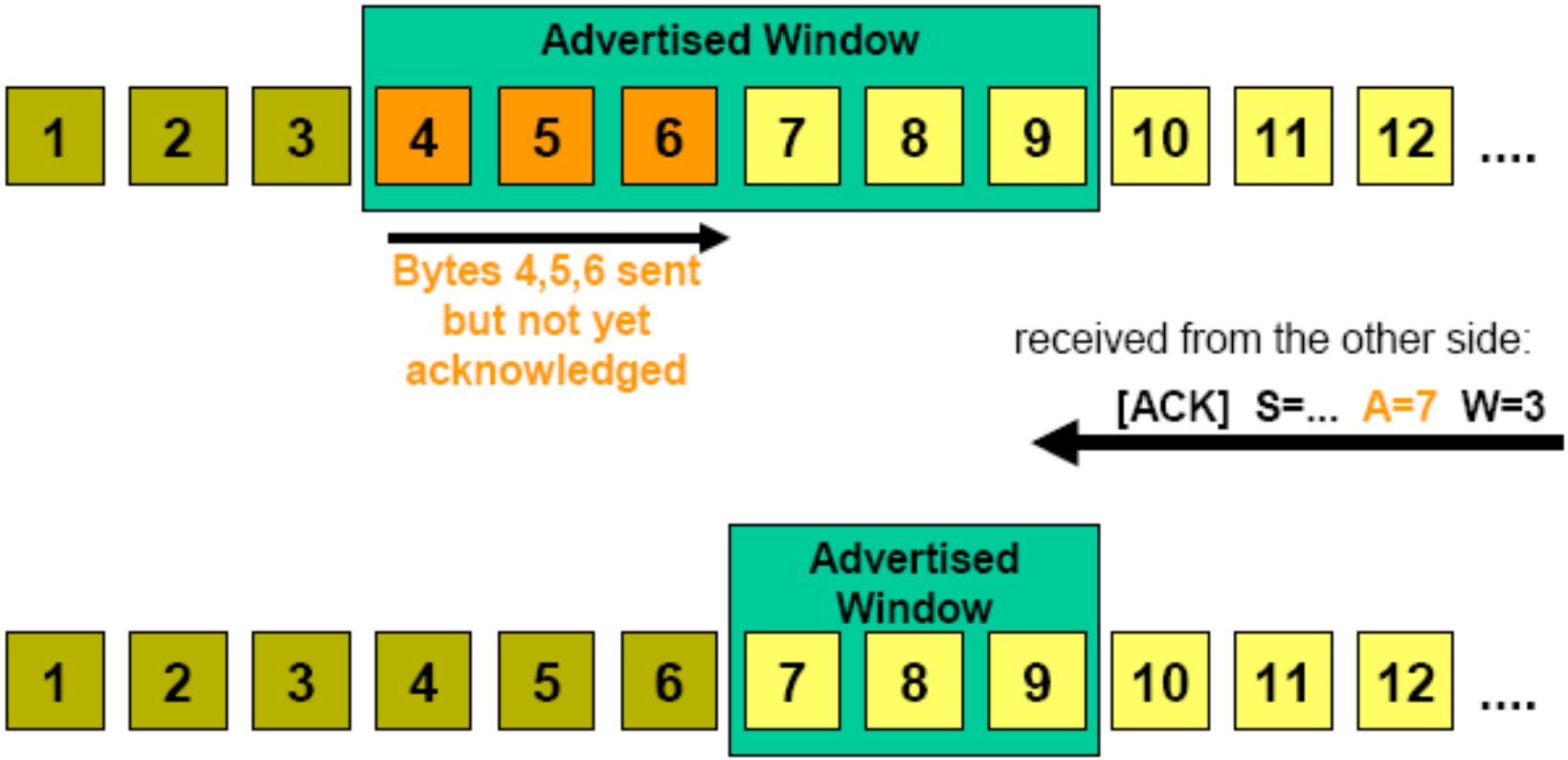
Скользящее окно (уст. соединения)



Скользящее окно (принцип)

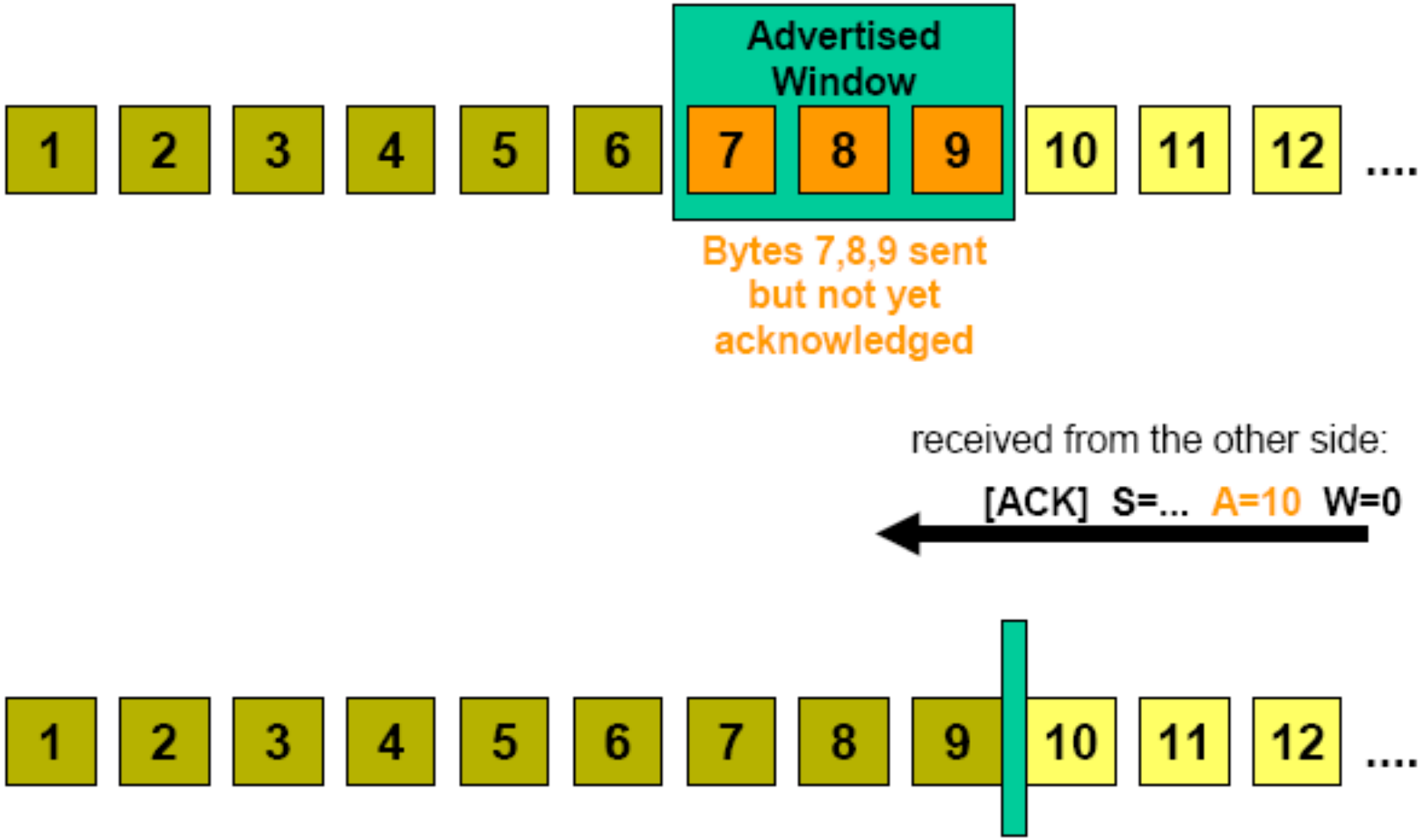


Скользящее окно (закрытие)

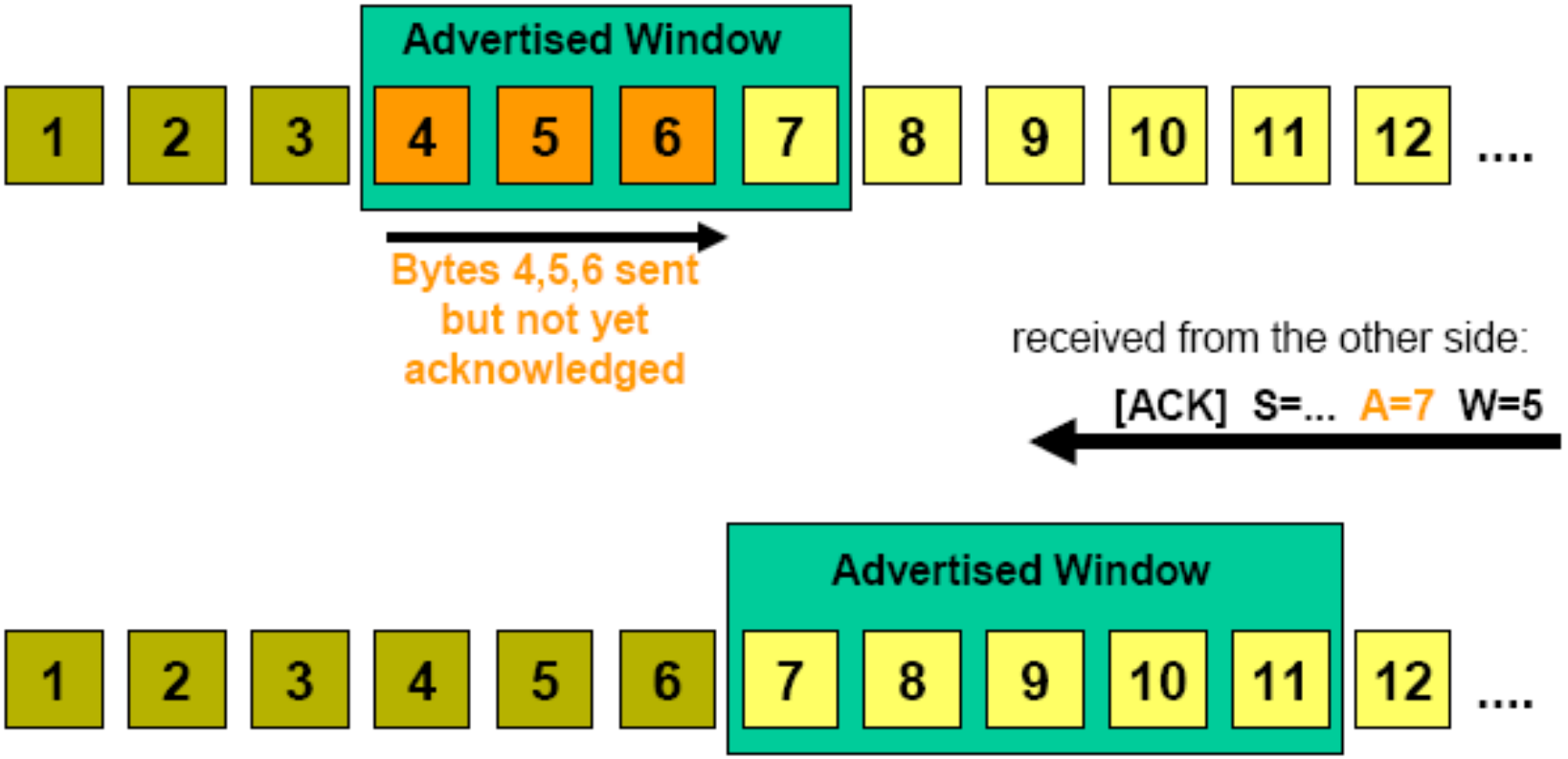


Now the sender may send bytes 7, 8, 9. The receiver didn't open the window (W=3, right edge remains constant) because of congestion. However, the remaining three bytes inside the window are already granted, so the receiver cannot move the right edge leftwards.

Скользящее окно (закрытие)



Скользящее окно (закрытие)



The receiver's application read data from the receive-buffer and acknowledged bytes 4,5,6. Free space of the receiver's buffer is indicated by a window value that makes the right edge of the window move rightwards. Now the sender may send bytes 7, 8, 9,10,11.

Эффективный размер окна

- **Для TCP размер скользящего окна (win) может измеряться в байтах или в числе сегментов**
- **Эффективный размер окна в байтах определяется соотношением:**
  - win > RTT * B

  где: B – полоса пропускания канала в бит/с

  RTT - время распространения пакета туда и обратно

  win – размер окна в байтах
- **Эффективный размер окна в числе сегментов определяется соотношением:**

  win > RTT * B / MSS

  где: MSS – максимальный размер сегмента в битах

  win – размер окна в числе сегментов

Три указателя окна передатчика

- **В TCP существует три указателя передающей стороны:**
  - ✓ Первый указатель определяет положение левого края окна, отделяя подтвержденные и не подтвержденные сегменты
  - ✓ Второй указатель отмечает правый край окна и указывает на последний сегмент, который может быть послан до получения очередного подтверждения
  - ✓ Третий указатель помечает границу внутри скользящего окна между уже посланными сегментами и теми, которые еще предстоит послать
- Если указатель 3 совпадет с указателем 2, отправитель прекращает передачу сегментов до получения хотя бы одного подтверждения

Три указателя окна приемника

- **Получатель организует аналогичные окна для контроля потока принимаемых данных**
  - Обычно получатель посылает одно подтверждение (ACK) на два полученных сегмента

Улучшения: Алгоритм Нагля

- **В telnet нажатием клавиши передается 1 байт и посылается 41-битный сегмент (без учета издержки Ethernet)**
- **Эффективность работы может быть улучшена с помощью алгоритма Нагля (Nagle, 1984; RFC-896)**
  - Нагль предложил посылать первый байт, а последующие буферизовать до прихода подтверждения получения посланного. После этого посылаются все буферизованные октеты, а запись в буфер вводимых кодов возобновляется
  - Если символы вводятся быстро, а сеть работает медленно, этот алгоритм заметно снижает загрузку канала
  - Алгоритм Нагля желательно отключить при работе в режиме X-терминала, где сигналы перемещения мышки должны пересылаться немедленно, чтобы не ввести в заблуждение пользователя относительно истинного положения маркера

Улучшения: Синдром узкого окна (Clark, 1982)

- **Ситуация: (Кларк, 1982)**
  1. данные передаются отправителем крупными блоками
  2. буфер получателя (RecvBuffer) заполнится и передающая сторона знает об этом (window=0),
  3. интерактивное приложение получателя считывает информацию побайтно
  4. получатель пошлет уведомление отправителю, разрешающее ему послать один байт (window=1)
  5. этот байт будет послан и снова заполнит до краев буфер получателя, что вызовет отправку ACK со значением window=0
  6. Далее повторяются действия с пункта 3
- **Проблема**
  - Если получатель рекламирует малые увеличения окна приема[window(i) = 0, window(i+1) = 1, window(i+2)=0, window(i+3) = 1, ….. и т.д.)]
    - ✓ передатчик посылает много маленьких сегментов
    - ✓ тратиться время на передачу заголовков этих маленьких сегментов
    - ✓ понижая коэффициент использования канала
- **Решение (Кларк, 1982)**
  - Получатель не должен рекламировать малые увеличения окна
  - Минимальный размер рекламируемого окна увеличения равен min(MSS, RecvBuffer/2, где
    - ✓ MSS (Maximum Segment Size) – максимальный размер сегмента
    - ✓ RecvBuffer – буфер приема получателя

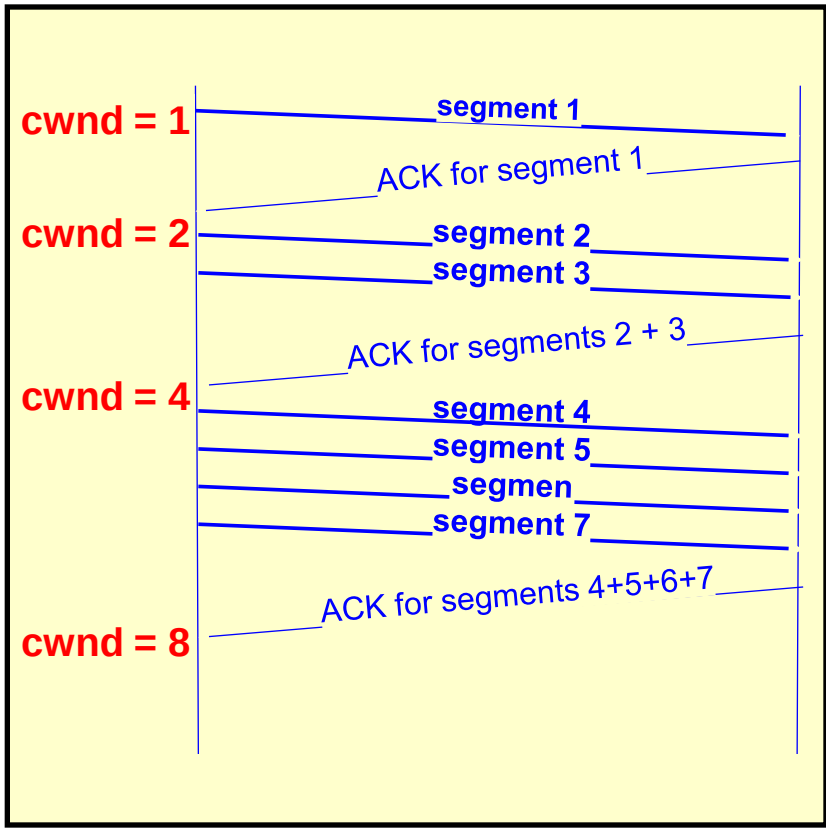
Управление перегрузкой

Параметры управление перегрузкой

- **Параметры (переменные) управления перегрузкой**
  - cwnd (congestion window) - окно перегрузки
  - rcv_win (receiver advertised window) – объявленное получателем окно
  - ssthreth (slow start threshold) - порог медленного старта
- **Отправитель никогда не пошлет больше байт, чем это задано в cwnd и объявлено получателем rcv_win**
  - win = min (rcv_win, cwnd)
- **Отправитель, при инициализации TCP соединения устанавливает**
  - cwnd = MSS (MSS – максимальный размер сегмента)
  - ssthreth=65535 байтам (2 в степени 16)
- **Если будет прислано подтверждение до истечения таймаута – удваивает окно перегрузки**
  - CWND i+1 = 2 x CWND i (медленный старт)

Медленный старт , (Джекобсон, 1988)

- **Медленный старт - “Slow Start”**
  - Каждое подтверждение увеличивает окно перегрузки на один сегмент
    - ✓ CWND i+1 = CWND i + MSS
  - Или можно сказать так: если окно перегрузки равно B сегментам и все они подтверждены, окно перегрузки возрастает на число байт, содержащихся в этих сегментах.
- **Ширина окна перегрузки последовательно удваивается**
  - пока доставка всех сегментов подтверждается
- **Рост ширины окна перегрузки имеет экспоненциальный характер**
- **Рост продолжается до тех пор, пока не наступит таймаут или окно перегрузки не сравняется с окном получателя.**
- **Эта процедура называется *медленным стартом* (Джекобсон, 1988)**

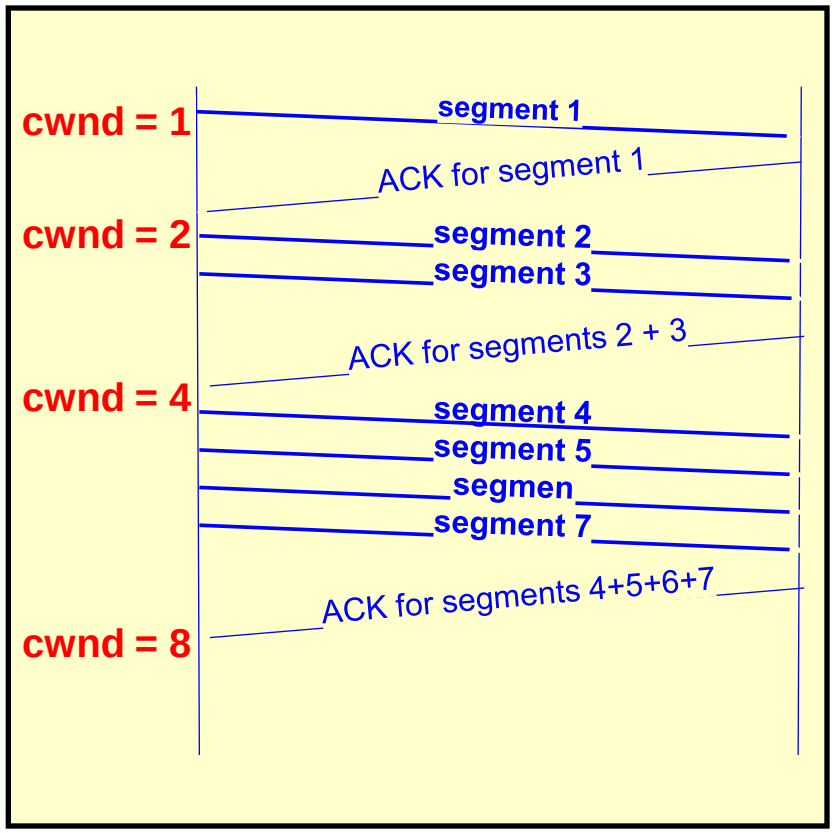


Медленный старт “Slow Start” (повтор)

- **Итак, в качестве модуля приращения cwnd используется MSS или 1**
- **Каждое подтверждение (ACK) увеличивает окно перегрузки:**
  - CWND i+1 = CWND i + MSS ,

  если размер окна задан в байтах
  - CWND i+1 = CWND i + 1 ,

  если размер окна задан в числе сегменов
- **Это и есть алгоритм медленного старта**
- **И теперь отправитель может послать, не дожидаясь ACK, уже два сегмента и т.д.**

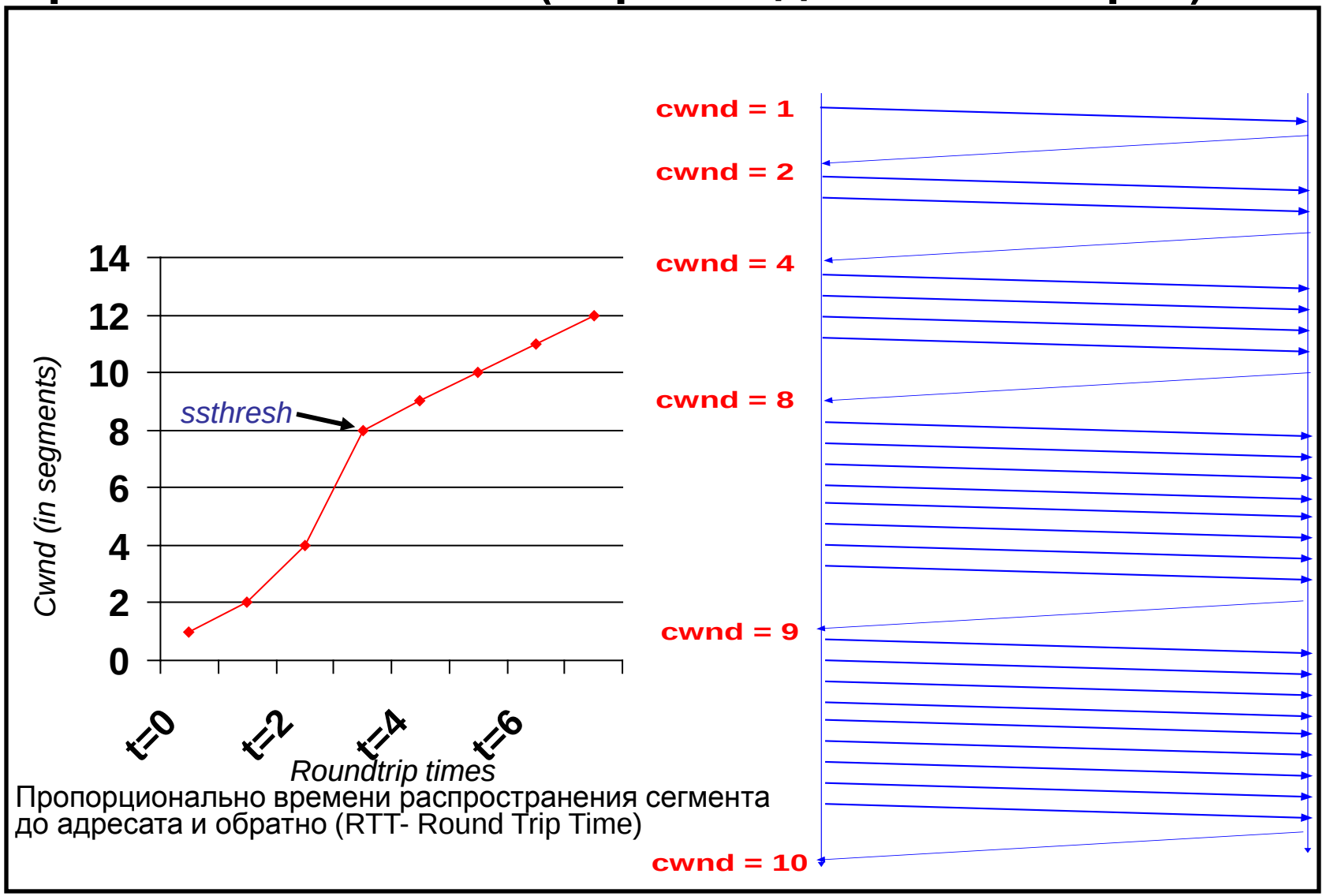


Подавление перегрузки (ssthreth)

- **Случай успешной доставки сегментов (получении ACK подтверждений)**
  - Отправитель увеличивает значение cwnd
  - Динамика процесса увеличения окна перегрузки (cwnd) зависит от величины порога медленного старта (ssthreth)
    - ✓ Если cwnd ≤ ssthreth, выполняется “медленный старт”
      - CWND i+1 = CWND i + MSS если размер окна задан в байтах
      - CWND i+1 = CWND i + 1 если размер окна задан в числе сегментов
    - ✓ Если cwnd > ssthreth выполняется процедура подавления перегрузки (cwnd увеличивается на 1)
      - если все сегменты были подтверждены, начинается с порога медленного старта ssthresh окно перегрузки увеличивается на единицу
      - Cwnd+

Пример подавления перегрузки

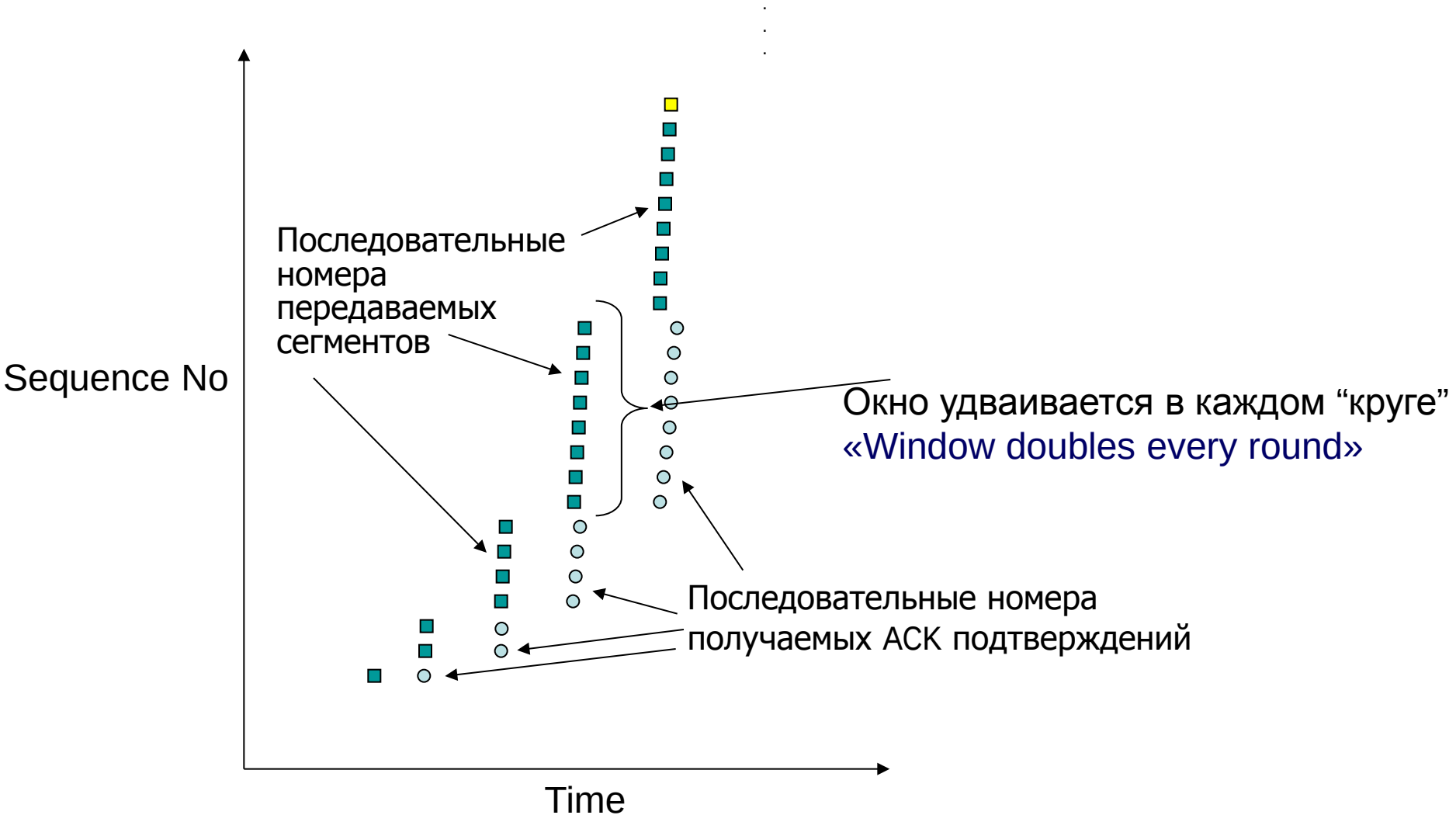
- **Примем ssthresh = 8 (порог медленного старта)**



Пропорционально времени распространения сегмента до адресата и обратно (RTT- Round Trip Time)

Медленный старт и подавление перегрузки

- **Итак, в качестве модуля приращения cwnd используется MSS или 1**



Как определяется потеря сегмента(ов)?

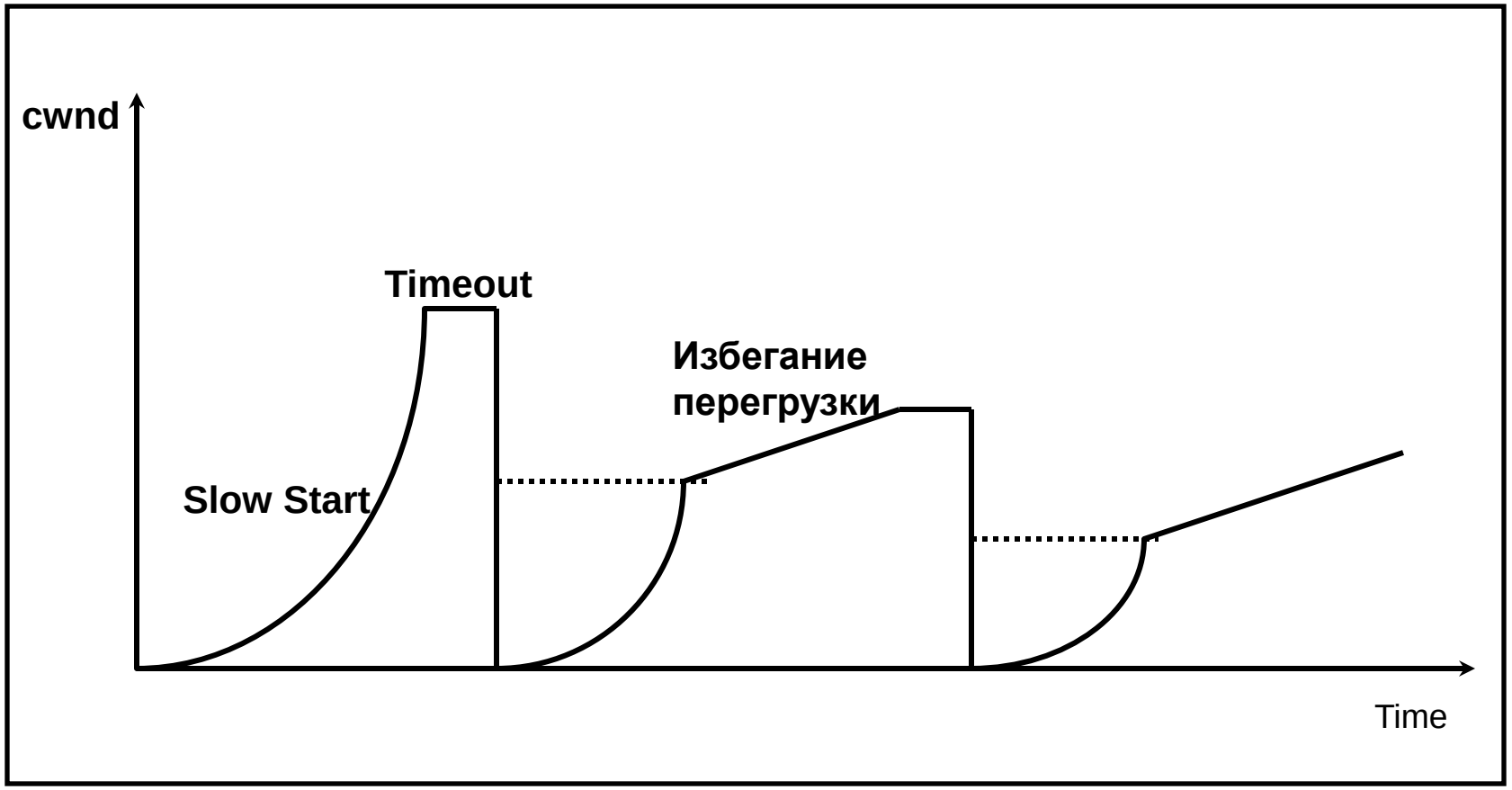
- **Причина потери пакетов - перегрузка канала**
- **Механизмы обнаружения потери пакетов отправителем**
  1. не получение вовремя подтверждения (по таймауту, RTO)
  2. получение ACK-дубликатов
- **Почему формируются получателем ACK-дубликаты?**
  - TCP требует посылки немедленного подтверждения (дублированного ACK) при обнаружении прихода сегментов с нарушением порядка следования.
    - Причиной нарушения порядка следования может быть флуктуация задержки в сети или потеря пакета.
- **Если получено три или более ACK-дубликатов, это убедительно указывает на потерю пакета и, не дожидаясь таймаута, осуществляется его повторная передача**
  - Перехода в режим медленного старта в этом случае не производится, но понижаются значения cwnd и ssthresh (почти вдвое).

Что делается в случае потери сегментов?

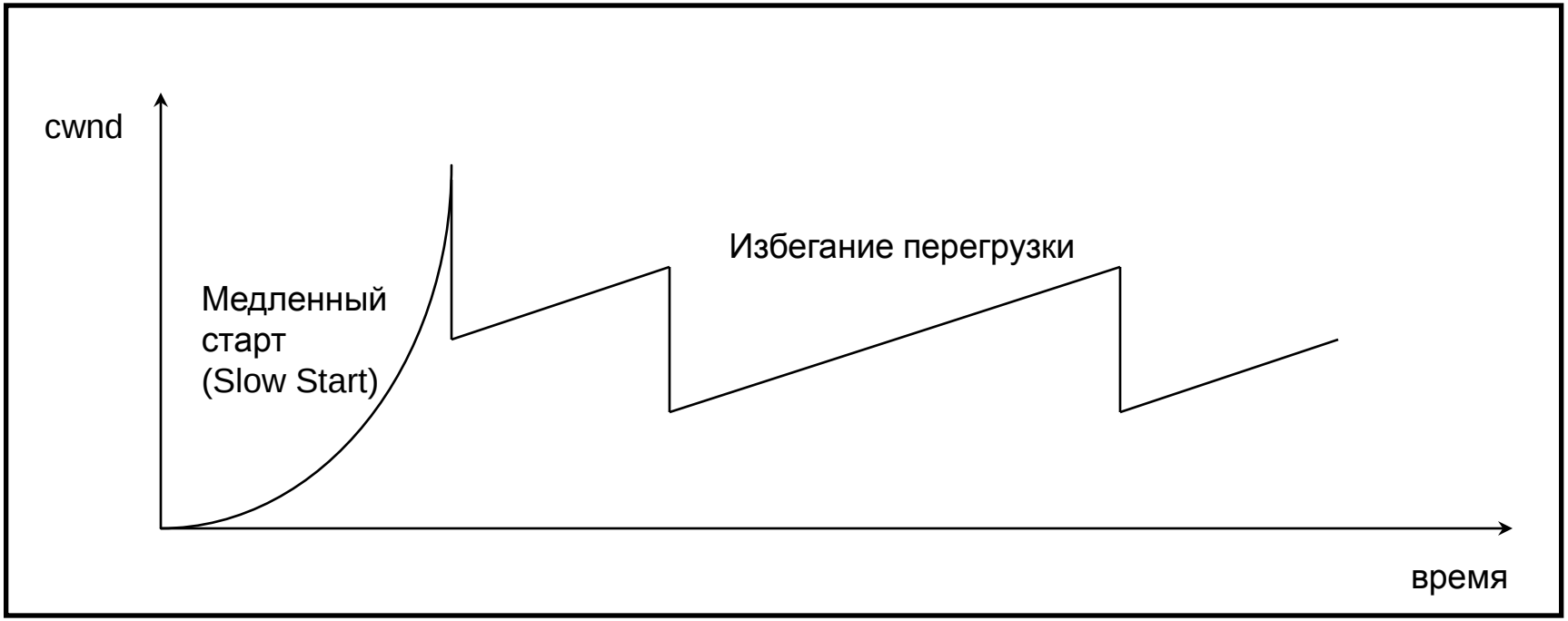
- **В случае неуспешной доставки сегмента(ов)**
  - По таймауту система переходит в режим "медленного старта" (ширина окна перегрузки cwnd делается равной 1 сегменту, а значение порога медленного старта - ssthresh делается равным двум сегментам).
    - ✓ cwnd = MSS (MSS – максимальный размер сегмента)
    - ✓ ssthresh = 2 * MSS делается равным двум сегментам).
      - ➢ Вспомним, при инициализации TCPсоединения переменная ssthresh обычно равна 65535.
  - Дублирующие ACK индицируют потерю пакета до наступления таймаута
    - ✓ В этом случае сначала меняется алгоритм приращения величины окна перегрузки cwnd (замедляется темп его роста). После прихода очередного ACK новое значение cwnd вычисляется по формуле:

    cwndi+1 = cwndi + (размер_сегмента*размер_сегмента)/cwndi + размер_сегмента/8

Медленный старт



Быстрая ретрансмиссия и быстрая регенерация



Предотвращение перегрузки (4)

- **Размер окна может стать большим → возможно приведет к ошибкам доставки (потерям IP-пакетов)**
  - тогда будет запущена (вновь) процедура “медленного старта”
  - или другой алгоритм, который определит новое, уменьшенное значение окна.
- **Окно перегрузки (cwnd) управляет потоком со стороны отправителя, блокируя возможные перегрузки и потери данных в промежуточных узлах сети**
- **Если переполнение сети не происходит, CWND становится больше окна, объявленного получателем, и именно последнее будет ограничивать поток данных в канале**

Контроль перегрузки

- **Если переполнение сети не происходит, CWND становится больше окна, объявленного получателем, и именно последнее будет ограничивать поток данных в канале**
- **Размер окна, объявленный получателем, ограничивается произведением полосы пропускания канала (бит/с) на RTT (время распространения пакета туда и обратно)**
  - window > RTT*B / MSS, где
    - ✓ B – полоса пропускания канала в бит/с,
    - ✓ MSS – максимальный размер сегмента в битах,
    - ✓ window – окно в сегментах
- **Максимально допустимый размер окна в TCP равен 65535 байт (задается размером поля)**
  - В фазе установления соединения можно ввести масштаб окна 2^n

Контроль перегрузки (по шагам 1)

- **При инициализации соединения окно перегрузки имеет ширину равную максимальному размеру сегмента, который может быть использован в данном канале. Отправитель посылает такой сегмент**
- **Если будет прислано подтверждение до истечения времени таймаута размер окна перегрузки удваивается и посылается два сегмента максимальной длины**
- **При получении подтверждения доставки каждого из сегментов окно перегрузки увеличивается на один сегмент максимальной длины**
- **Когда ширина окна перегрузки становится равной B сегментов и все B посланных сегментов получают подтверждение, окно перегрузки возрастает на число байт, содержащихся в этих сегментах**

Контроль перегрузки (по шагам 2)

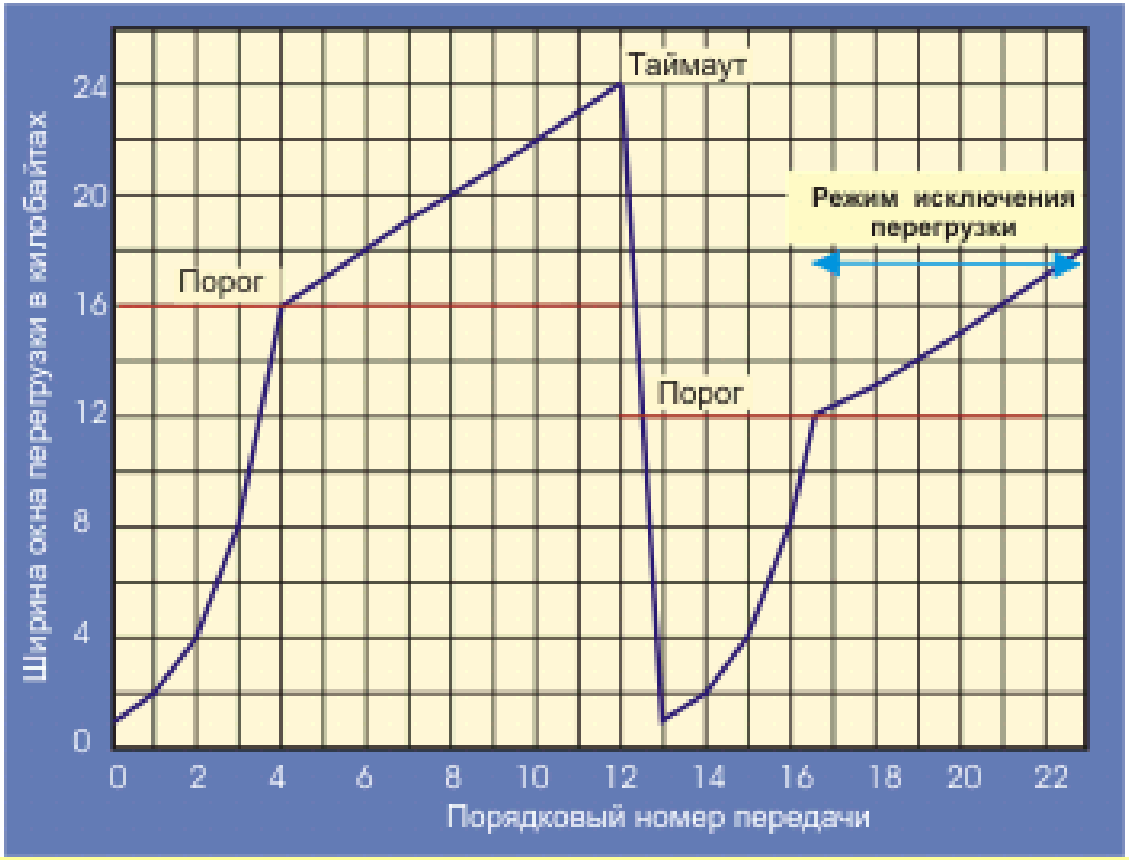
- **Таким образом, ширина окна перегрузки последовательно удваивается, пока доставка всех сегментов подтверждается**
- **Рост ширины окна перегрузки при этом имеет экспоненциальный характер**
- **Это продолжается до тех пор, пока не наступит таймаут или окно перегрузки не сравняется с окном получателя. Именно эта процедура и называется медленным стартом (Джекобсон, 1988)**

Контроль перегрузки (по шагам 3)

- **Как было сказано выше, помимо окона перегрузки и окна получателя в TCP используется третий параметр - порог (ssthresh)**
  - ssthresh – называют иногда порогом медленного старта
- **При установлении соединения ssthresh = 64 Кбайт**
- **В случае возникновения таймаута**
  - ssthresh = CWND/2,
  - CWND = MSS
- **Далее запускается процедура медленного старта, чтобы выяснить возможности канала.**
  - экспоненциальный рост cwnd осуществляется вплоть до значения ssthresh.
  - дальнейший рост cwnd происходит линейно с приращением на каждом шагу равным MSS.

Контроль перегрузки (по шагам 3)

- **Пример: MSS=1 Кбайт. Началу диаграммы соответствует установка значения ssthresh=16 Кбайт.**
- **После таймаута, который на рисунке произошел при передаче с номером 12, значение порога понижается до 12 Кбайт (=cwnd/2).**



Контроль перегрузки (по шагам 4)

- **Здесь предполагается, что MSS=1 Кбайт. Началу диаграммы соответствует установка значения ssthresh=16 Кбайт.**
- **Данная схема позволяет более точно выбрать значение cwnd. После таймаута, который на рисунке произошел при передаче с номером 12, значение порога понижается до 12 Кбайт (=cwnd/2).**
- **Ширина окна cwnd снова начинает расти от передачи к передаче, начиная с одного сегмента, вплоть до нового значения порога ssthresh=12 Кбайт.**
- **Стратегия с экспоненциальным и линейным участками изменения ширины окна переполнения позволяет несколько приблизить среднее его значение к оптимальному.**

Контроль перегрузки (по шагам 5)

- **Для локальных сетей**
  - RTT невелико
  - вероятность потери сегмента мала
  - следовательно оптимизация задания cwnd не так существенна, как в случае протяженных внешних (например, спутниковых) каналов.
- **Однако, если в локальной сети имеется фрагмент, где вероятность потерь пакетов велика, оптимизация задания cwnd**
  - Таким фрагментом может быть коммутатор/мост, один из каналов которого подключен к сегменту Fast Ethernet, а другой к обычному Ethernet на 10 Мбит/с

Контроль перегрузки (по шагам 5)

- **Если такой коммутатор/мост не снабжен системой подавления перегрузки, то каждый из сегментов/пакетов/кадров будет потерян в среднем 9 раз, прежде чем будет передан**
  - предполагается, что передача идет из сегмента FE
- **При этом cwnd будет практически все время равно MSS, что крайне неэффективно при передаче по каналам Интернет**
- **Такие потери вызовут определенные ошибки при вычислении среднего значения и дисперсии RTT, а как следствие и величин таймаутов**
- **Применение в таких местах маршрутизаторов или других приборов, способных реагировать на перегрузку посредством ICMP(4), решает эту проблему**

Контроль перегрузки (Вопрос 1)

- **Вопрос: Почему при потере сегмента CWND делается равным 1, а не CWND-1 или CWND/2 ? Ведь эффективность канала максимальна при наибольшем, возможном значении CWND**
- **Ответ: Если произошла потеря сегмента из-за переполнения буфера, оптимальное значение CWND может быть выбрано лишь при исчерпывающем знании прогноза состояния виртуального канала**
- **Постольку такая информация обычно недоступна, система переходит в режим освобождения буфера (CWND=1). Ведь если потеря была связана с началом сессии обмена с конкурирующим клиентом, операция CWND= CWND-1 проблему не решит. А потеря пакета вызовет таймаут и канал будет блокирован минимум на 1 секунду, что вызовет резкое падение скорости передачи**

Таймеры TCP

1. Таймер повторных передач (RTO- Retransmission TimeOut)
2. Таймер запросов (persist timer)
3. Таймер контроля работоспособности (keepalive)
4. 2MSL-таймер (Maximum Segment Lifetime)

RTT

- **Важным параметром, определяющим рабочие параметры таймеров, является RTT (время путешествия сегмента до адресата и обратно).**

- **TCP-агент самостоятельно измеряет RTT. Такие измерения производятся периодически и по их результатам корректируется среднее значение RTT:**

RTTm = a * RTTm + (1-a) * RTTi,

Где:

RTTi - результат очередного измерения,

RTTm - величина, полученная в результате усреднения предыдущих измерений,

a - коэффициент сглаживания, обычно равный 0.9.

Таймеры TCP

- **Для взаимного согласования операций в рамках TCP-протокола используется четыре таймера:**

1. **Таймер повторных передач (RTO- Retransmission TimeOut)**
   - Контролирует время прихода подтверждений (ACK)
   - Запускается в момент посылки сегмента
   - При получении отклика ACK до истечения времени таймера, он сбрасывается
   - Если же время таймера истекает до прихода ACK, сегмент посылается адресату повторно, а таймер перезапускается.
     - ✓ RFC-793 рекомендует устанавливать время таймаута для ретрансмиссии (повторной передачи)
     - ✓ значение RTO - Retransmission TimeOut равно
       - ➢ RTO = RTTm * b, где b равно 2
     - ✓ От корректного выбора этих параметров зависит эффективная работа каналов.
     - ✓ Так занижение времени ретрансмиссии приводит к неоправданным повторным посылкам сегментов, перегружая каналы связи
     - ✓ Для более точного выбора RTO необходимо знать дисперсию RTT

Таймеры TCP

2. Таймер запросов (persist timer)

- Контролирует размер окна при window=0
- Получатель при изменении ситуации посылает сегмент с ненулевым значением ширины окна, что позволит отправителю возобновить свою работу.
- Но если этот сегмент будет потерян, возникнет тупик, тогда каждая из сторон ждет сигнала от партнера
- Именно в этой ситуации и используется таймер запросов. По истечении времени этого таймера отправитель пошлет сегмент адресату.
- Отклик на этот сегмент будет содержать новое значение ширины окна.
- Таймер запускается каждый раз, когда получен сегмент с window=0

Таймеры TCP

3. Таймер контроля работоспособности (keepalive)

- Регистрирует факты выхода из строя или перезагрузки ЭВМ-партнеров.
- Время по умолчанию равно 2 часам.
- Keepalive-таймер не является частью TCP-спецификации.
- Полезен для выявления состояний сервера half-open при условии, что клиент отключился (например, пользователь выключил свою персональную ЭВМ, не выполнив LOGOUT).
- По истечении времени таймера клиенту посылается сегмент проверки состояния. Если в течение 75 секунд будет получен отклик, сервер повторяет запрос 10 раз с периодом 75 сек, после чего соединение разрывается. При получении любого сегмента от клиента таймер сбрасывается и запускается вновь.

Таймеры TCP

4. 2MSL-таймер (Maximum Segment Lifetime)

- Контролирует время пребывания канала в состоянии TIME_WAIT.
- Выдержка таймера по умолчанию равно 2 мин (FIN_WAIT-таймер) и RFC-793.
- Таймер запускается при выполнении процедуры active close в момент посылки последнего ACK

Развитие TCP

- **TCP является основным транспортным протоколом, попытки усовершенствовать его предпринимаются, начиная с 1992 года (RFC-1323, Якобсон, Браден и Борман). Целью этих усовершенствований служит повышение эффективности и пропускной способности канала, а также обеспечение безопасности. При этом рассматриваются следующие возможности:**
  - увеличение MTU (максимальный передаваемый блок данных);
  - расширение окна за пределы 65535 байт;
  - исключение "трех-сегментного" процесса установления связи и "четырехсегментного" ее прерывания (T/TCP, RFC-1644);
  - совершенствование механизма измерения RTT
  - оптимизация отслеживания CWND

MTU

- **Оптимальный выбор MTU позволяет минимизировать или исключить фрагментацию (и последующую сборку) сегментов.**
- **Верхняя граница на MTU налагается значением MSS (максимальный размер сегмента).**
- **Разумно находить и запоминать оптимальные значения MTU для каждого конкретного маршрута.**
- **Так как в современных системах используются динамические протоколы маршрутизации, поиск оптимального MTU рекомендуется повторять каждые 10 мин (RFC-1191).**

Развитие TCP http://book.itep.ru/4/44/tcp.htm

- **TCP-reno**
  - NewReno
- **TCP Vegas**
- **TCP-Tahoe**
- **TCP Westwood**

ECN (Explicit Congestion Notification)

В других лекциях для НИРС

ЛИТЕРАТУРА

1. **Семенов Ю.А. (ГНЦ ИТЭФ). Book.itep.ru**