



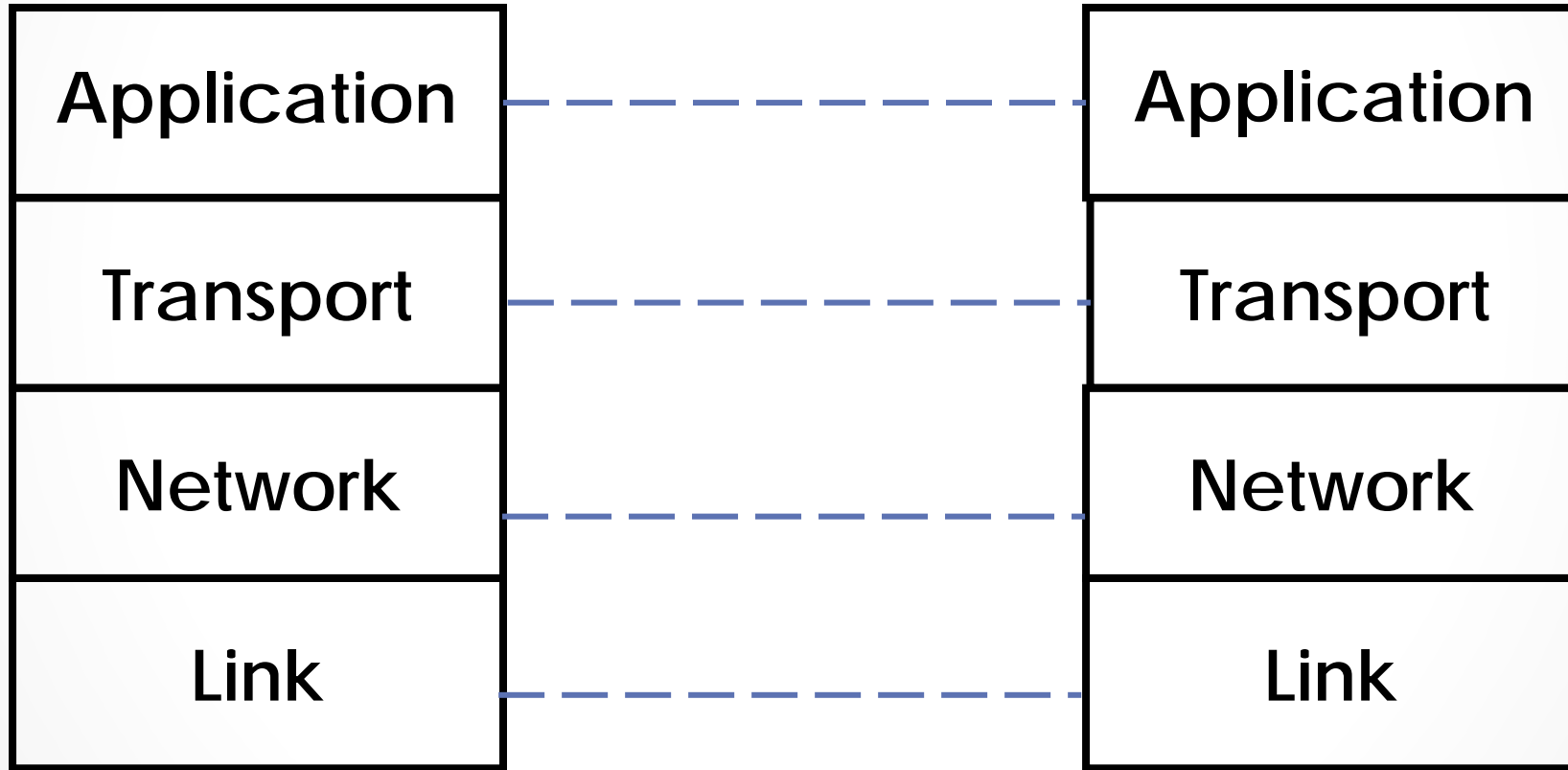
Интернет: модель TCP/UDP сервисов и транспортных протоколов

Введение в компьютерные сети
чл.-корр. РАН, проф. Смелянский Р.Л.

Кафедра АСВК ф-т ВМК МГУ



4-х уровневая модель Интернета



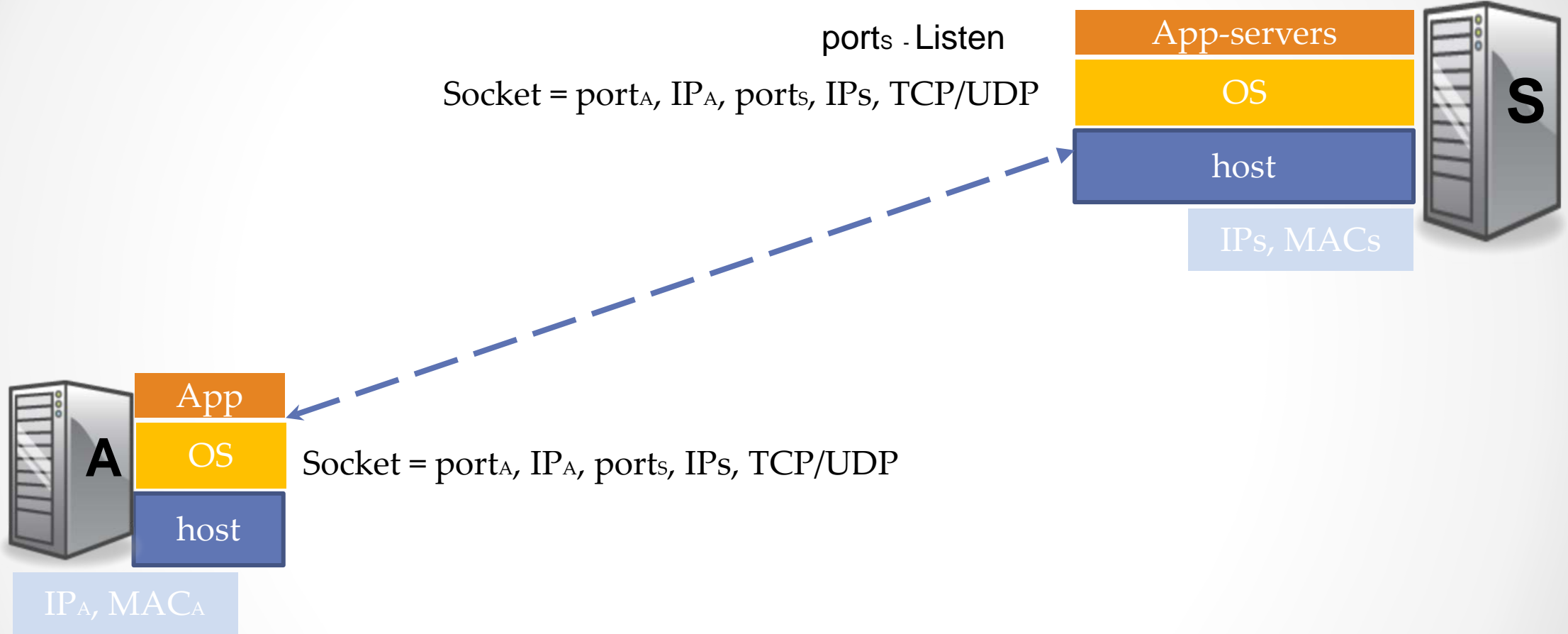


Адресация

- *Адресация на транспортном уровне*
- *Адресация на сетевом уровне*
- *Адресация на канальном уровне (Ethernet)*
- *Протоколы преобразования адресов*
- *Что такое протокол?*

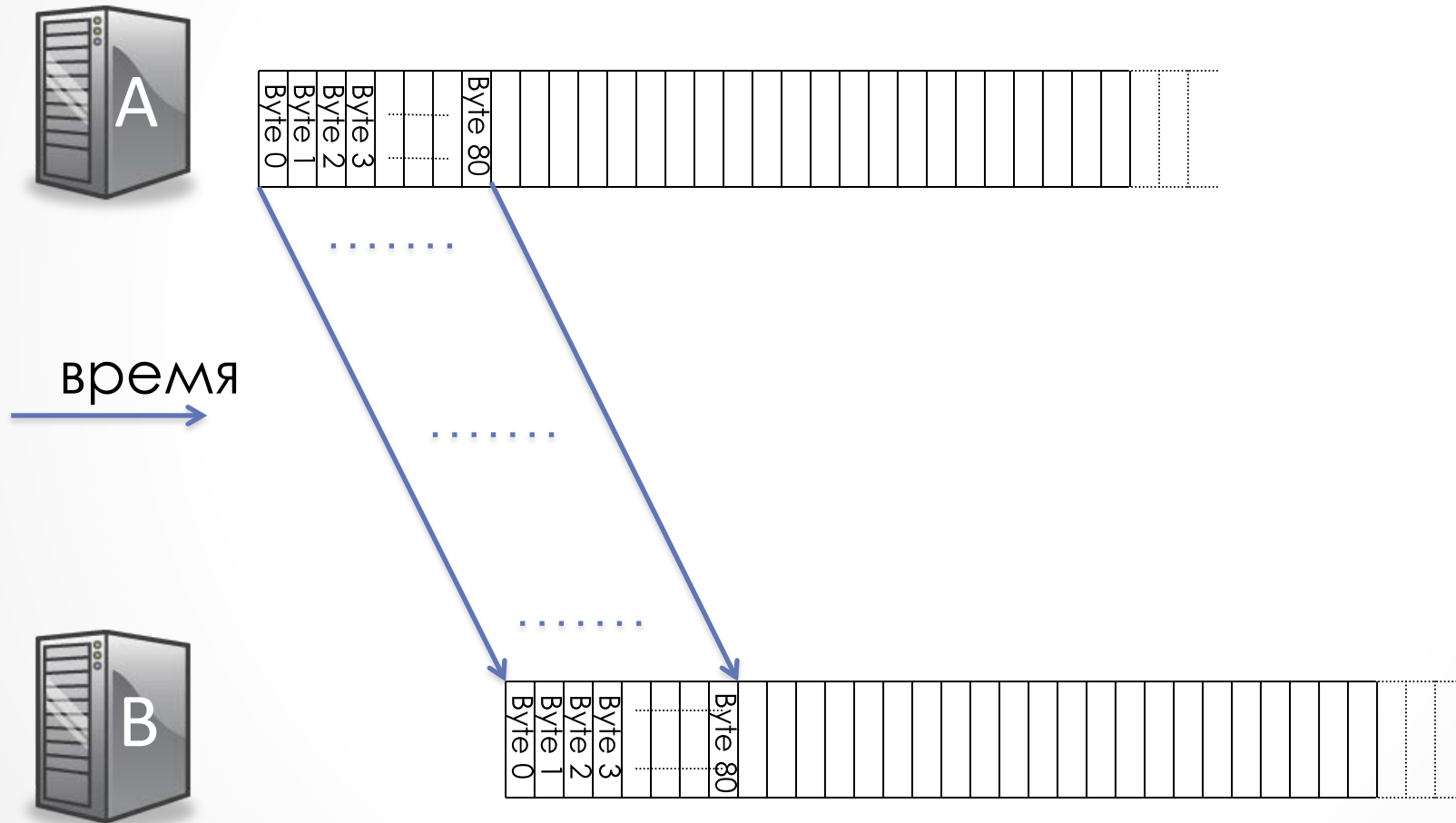


Соединение на прикладном уровне



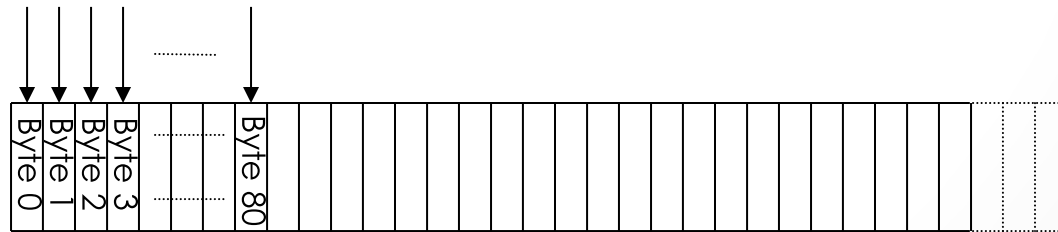
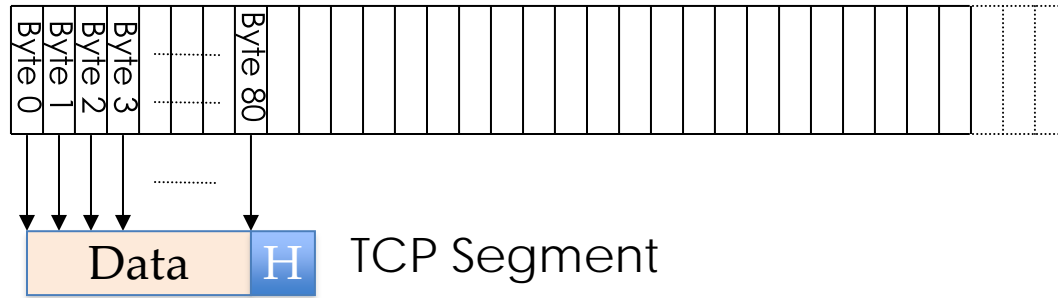


ТСР «поток байтов»



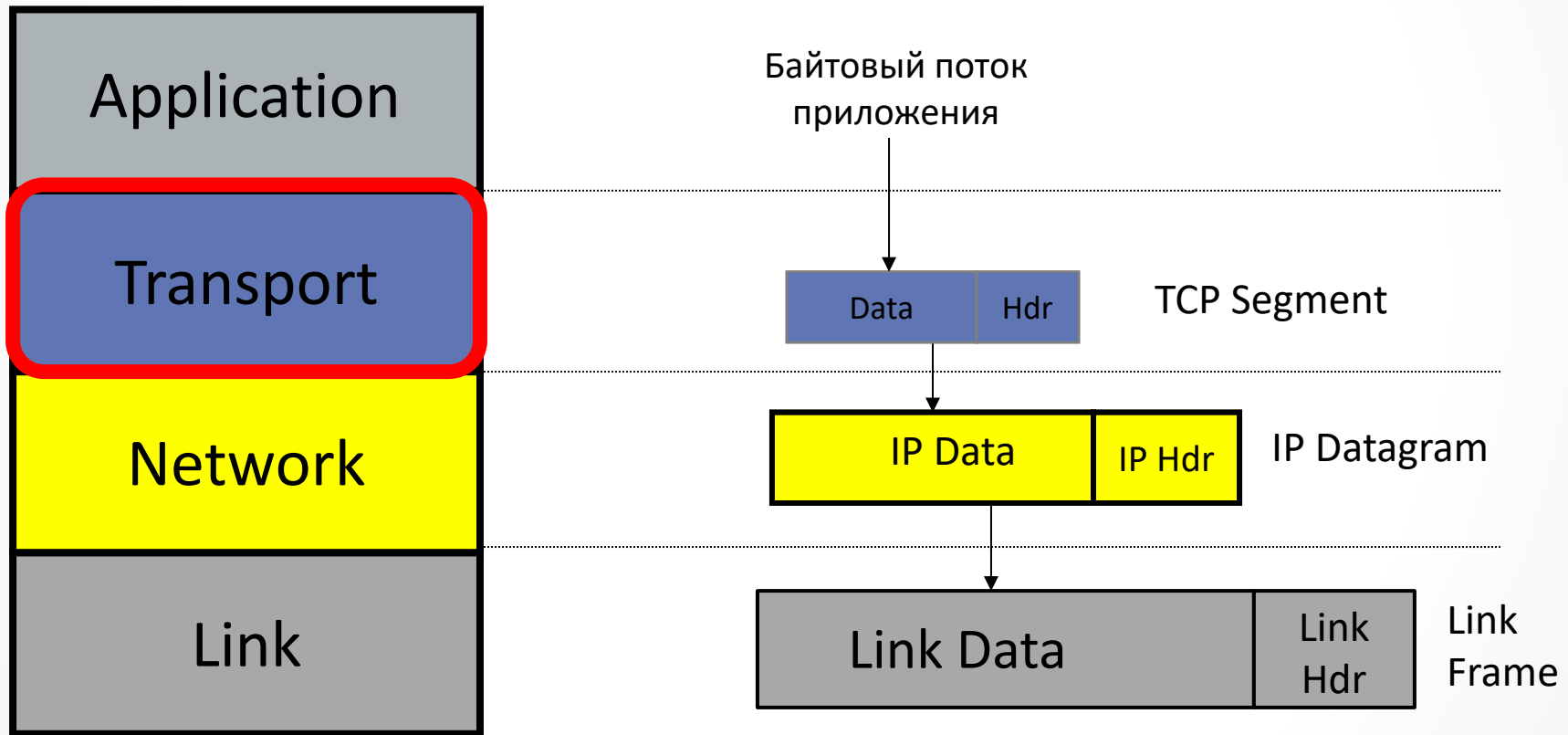


...Эмулируемый ТСР сегментами



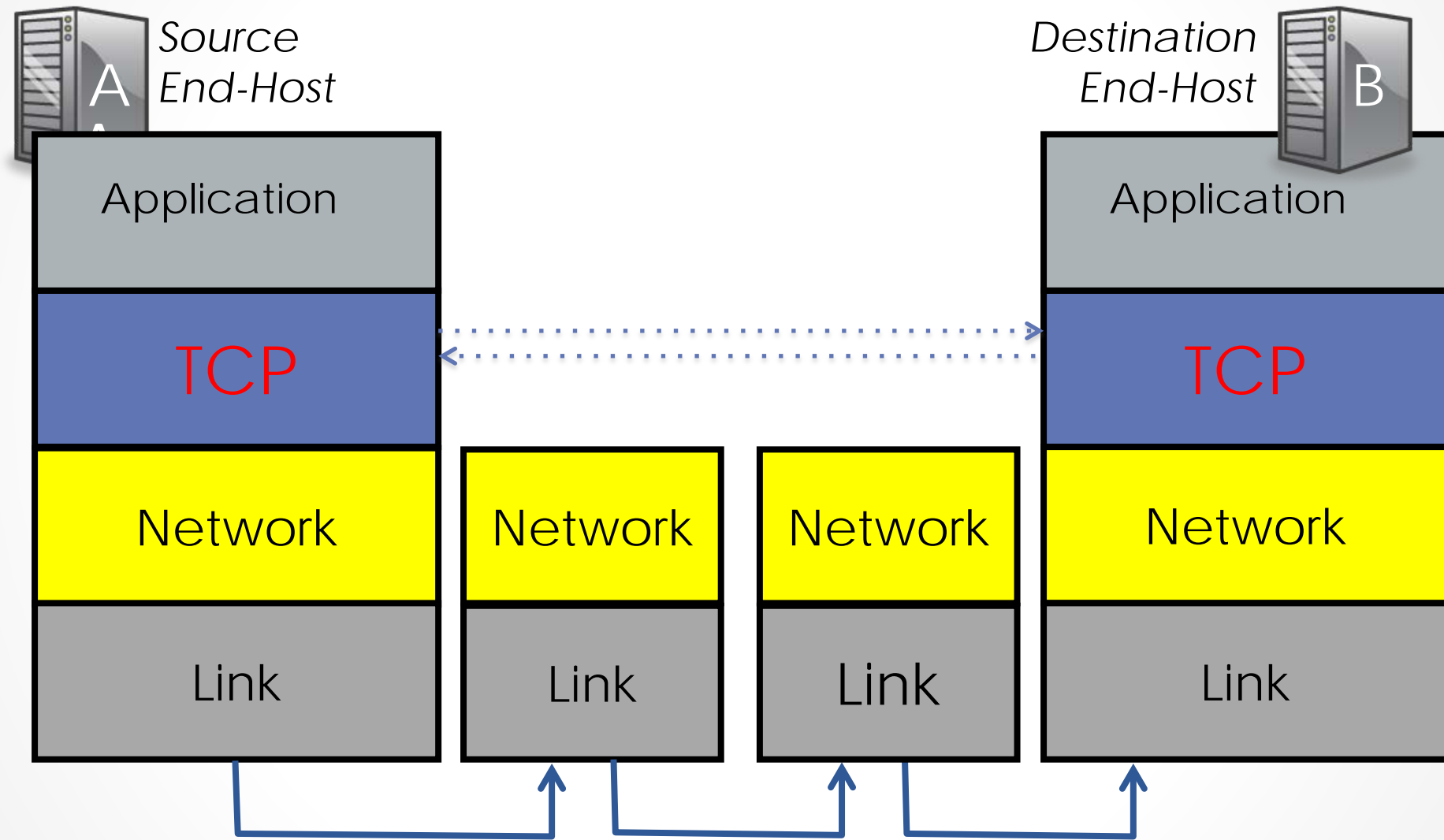


Transmission Control Protocol (TCP)





Взаимодействие ТСР уровней





Проблемы

Надежность взаимодействия достигается благодаря наличию согласованных состояний на концах соединения

- Проблема: как установить соединение (установить согласованные состояния)?
- Проблема: как разорвать соединение (установить согласованные состояния, освободить порты, очереди и т.д.) ?
- Проблема: как согласовано и надежно поддерживать номера принятых и отправленных байтов?
- Проблема: как согласовать скорости отправителя и получателя?
- Как обнаруживать и управлять перегрузками?



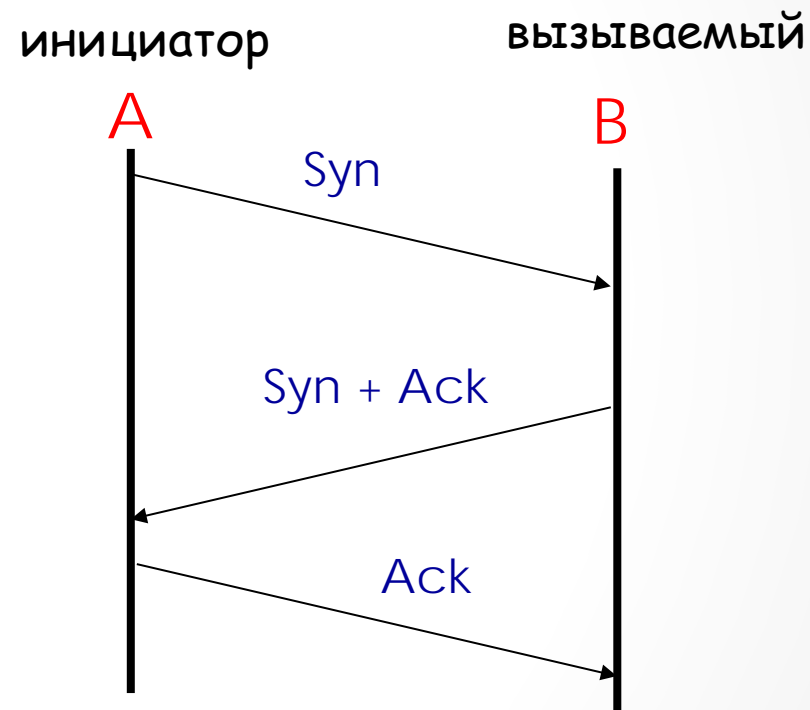
Установка соединения: 3-х кратное рукопожатие

Инициатор шлет 1-й пакет
SYN + sequence number

Вызываемый отвечает SYN + sequence number
(соединение в обратном направлении)
ACK для SYN инициатора

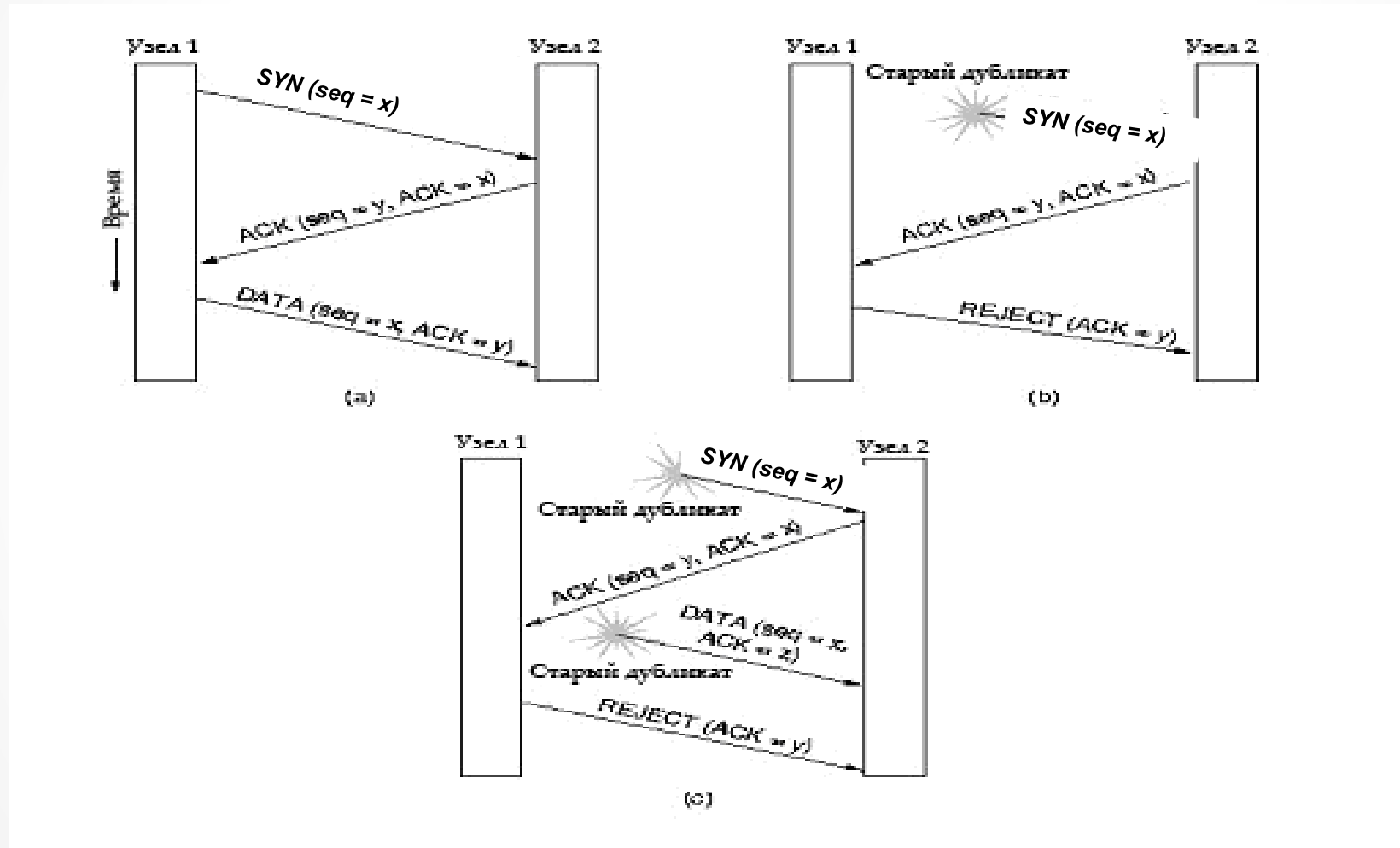
Инициатор отвечает
ACK на SYN вызываемого

- Также поддерживается одновременное открытие
 - Два SYN передаются друг другу
 - Каждая сторона шлет ACK другой стороне



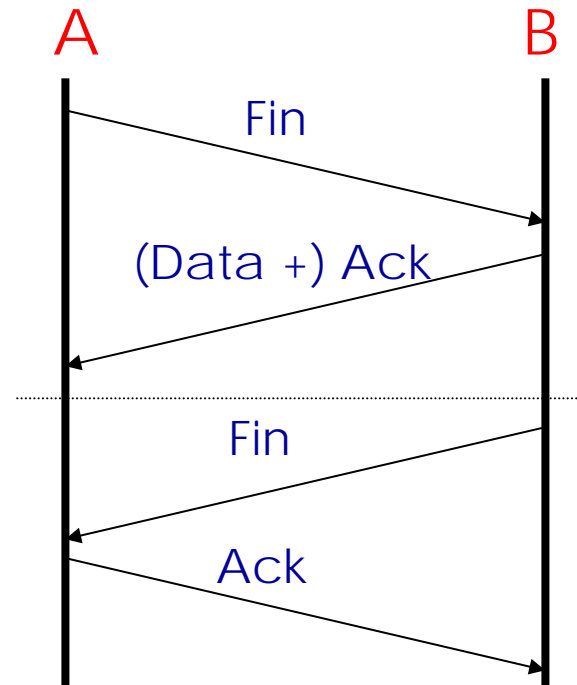


Установка соединения (ошибки)





Разрыв соединения

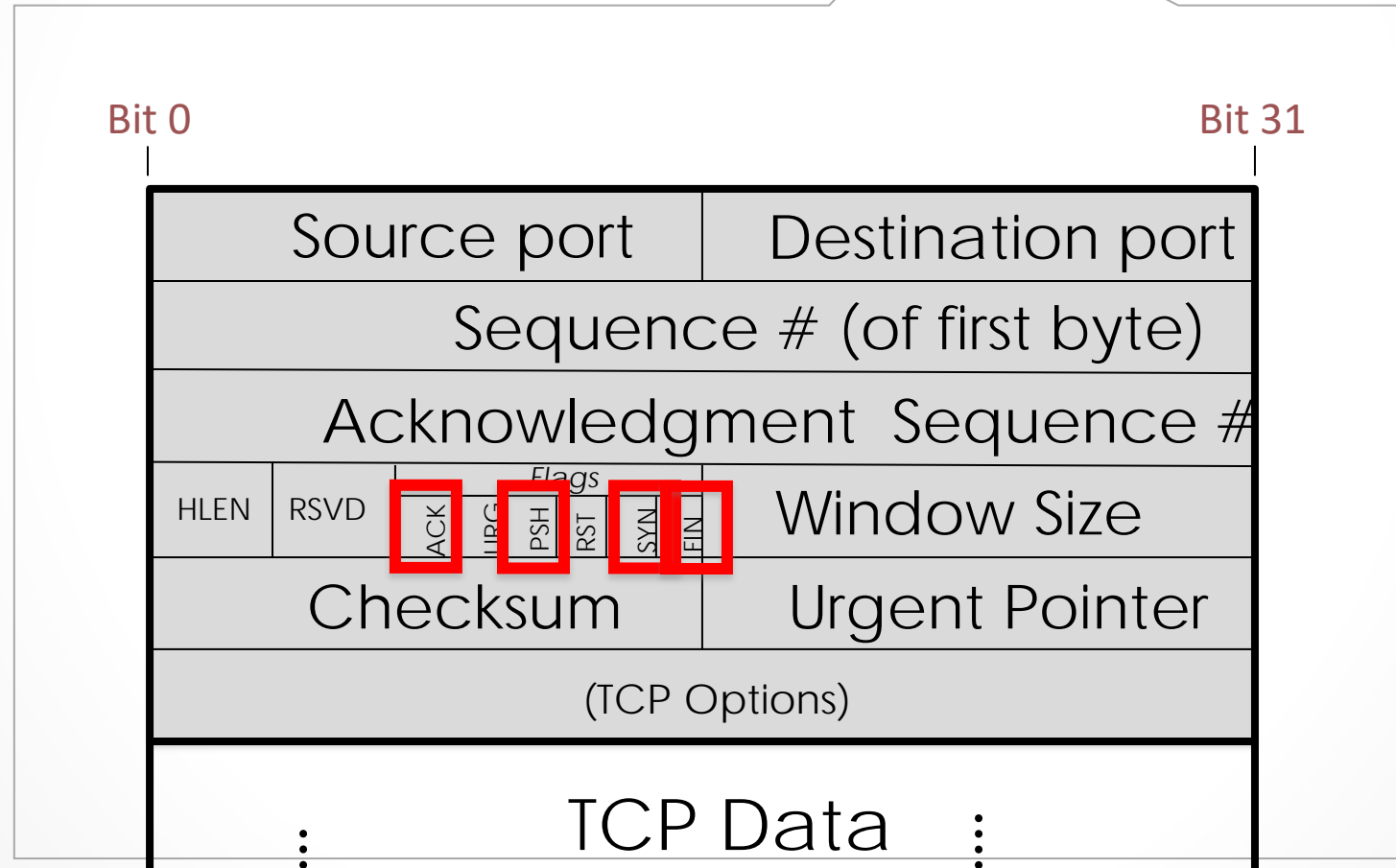
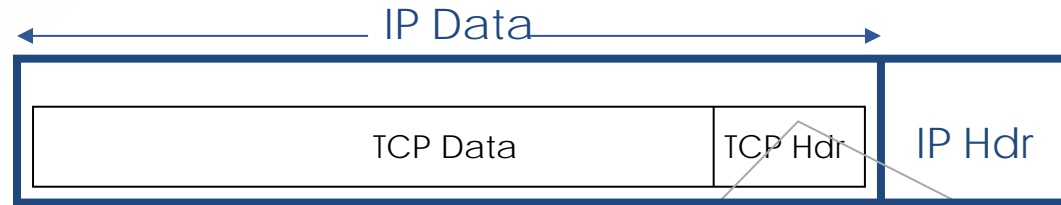




Интернет: модель TCP/UDP сервисов (продолжение)



Формат TCP сегмента



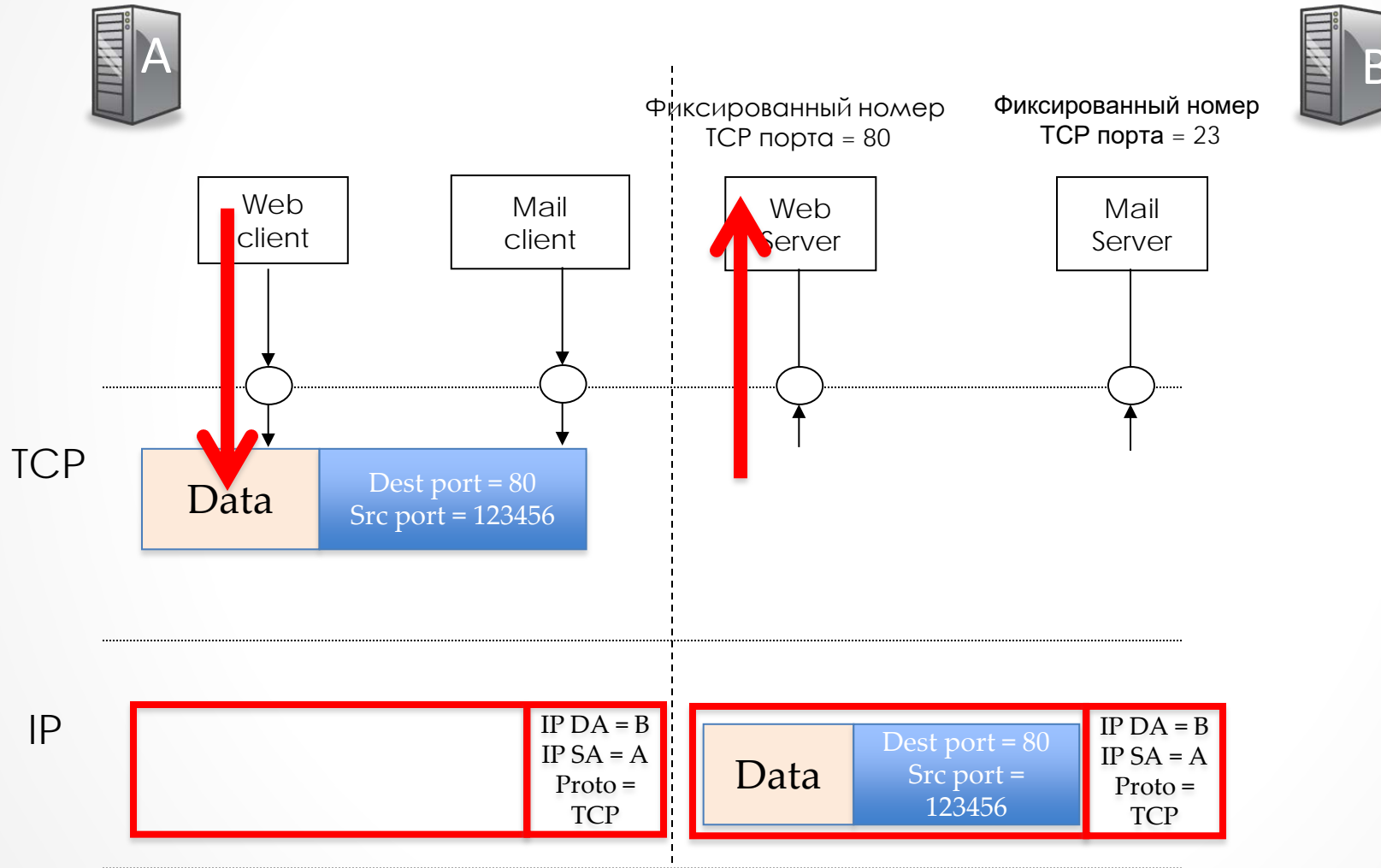


Модель ТСП сервиса

Свойство	Поведение
Поток байт	Сервис доставки байт
Надежная доставка	<ol style="list-style-type: none">1. АСК подтверждает доставку2. Контрольная сумма выявляет ошибки передачи (охватывает весь сегмент)3. Последовательные номера позволяют обнаружить пропущенные данные4. Управление потоком предотвращает получателя от «захлебывания»
Сохранение последовательности	Данные доставляют приложению в том же порядке, в каком они передавались
С установкой соединения	Процедура 3-х кратного рукопожатия
Управление перегрузкой	Управление перегрузками в сети



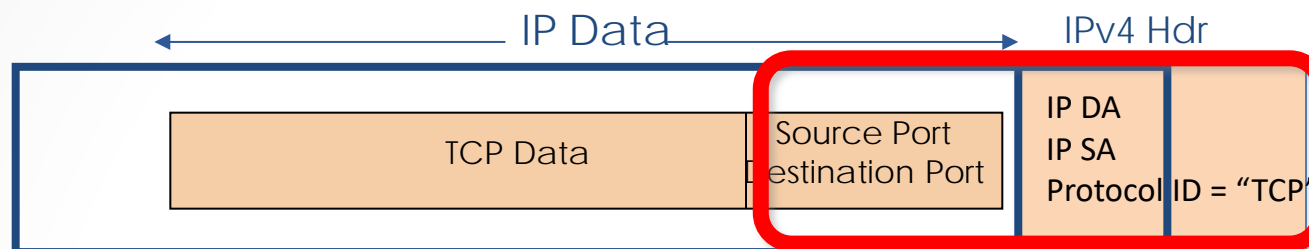
TCP: распределение по портам



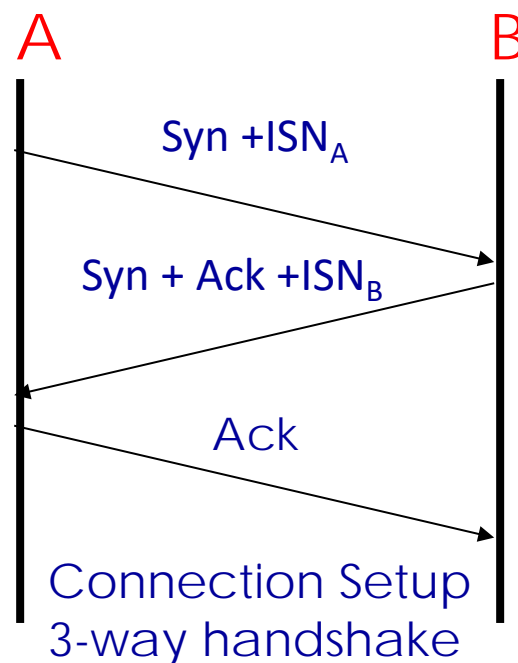


Уникальное ID для TCP соединения

104-бит уникальный ID
(во всем Internet)

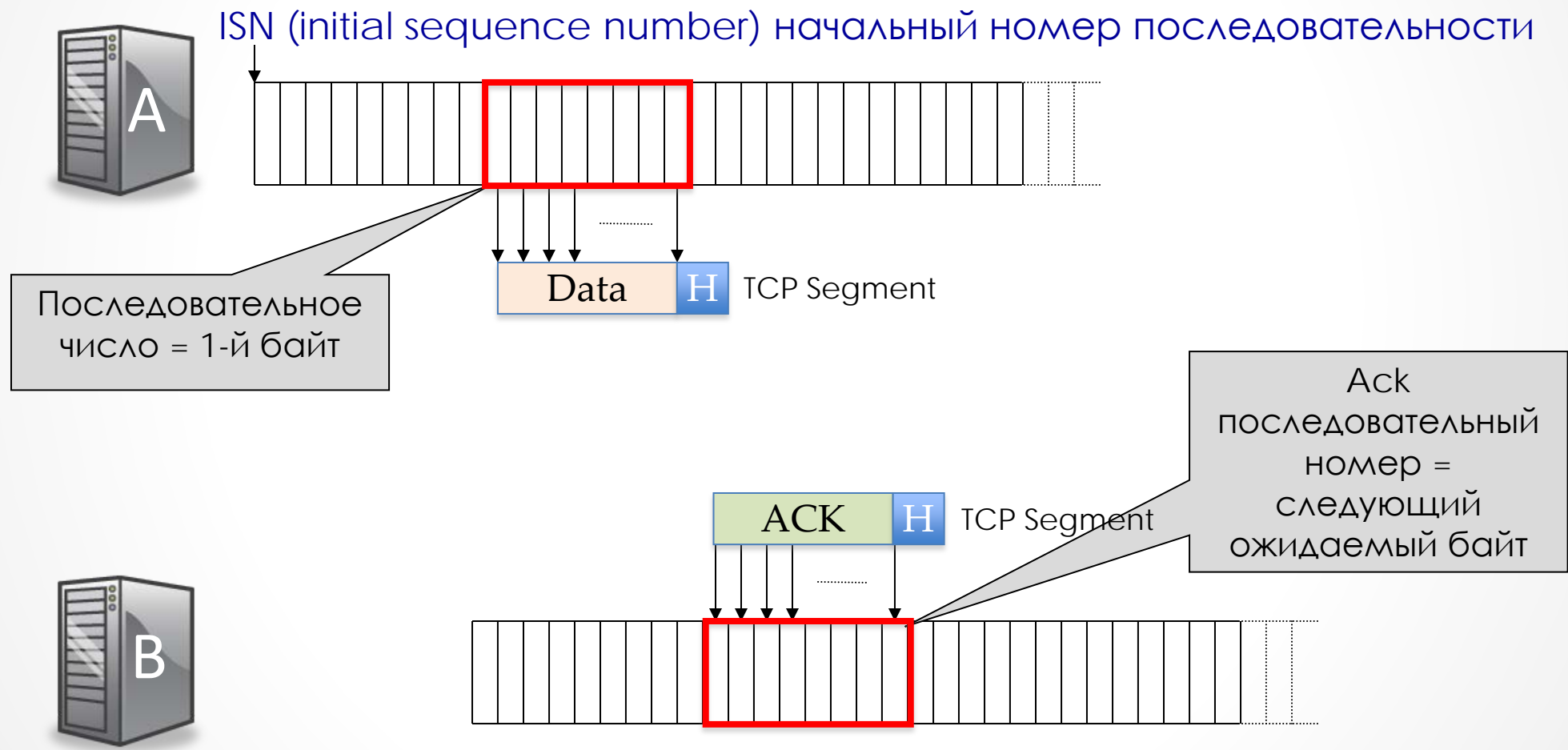


1. Хост А увеличивает номер source port для КАЖДОГО нового соединения
2. TCP выбирает ISN чтобы избежать пересечения с предыдущим соединением с тем же ID





Последовательные номера





Состояние транспортного соединения

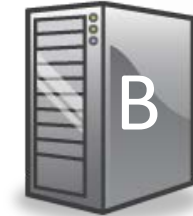
Application -client



TCP agent

S ₀	получен не подтвержден
S ₁	получен подтвержден
S ₂	подтвержден не передан App
S ₄	подтвержден и передан App

www-server



TCP agent

S ₀	отправлен/не подтвержден
S ₁	отправлен/подтвержден



Безопасное закрытие сокета

- *Проблемы с закрытым сокетом*
 - *Что будет если последний ACK будет потерян?*
 - *Что будет если та же пара портов будет сразу повторно использована для нового соединения?*
- *Решение: инициатор разрыва устанавливает Time Wait*
 - *Инициатор разрыва шлет FIN*
 - *Не закрывает сокет в течении 2 MSL (Maximum Segment Lifetime)*
- *Проблемы на стороне сервера:*
 - *Работа OS замедляется, т.к. очень много сокетов в состоянии Time Wait*
 - *Уязвимость: в состоянии Time Wait можно послать RST и переустановить параметры сокета*



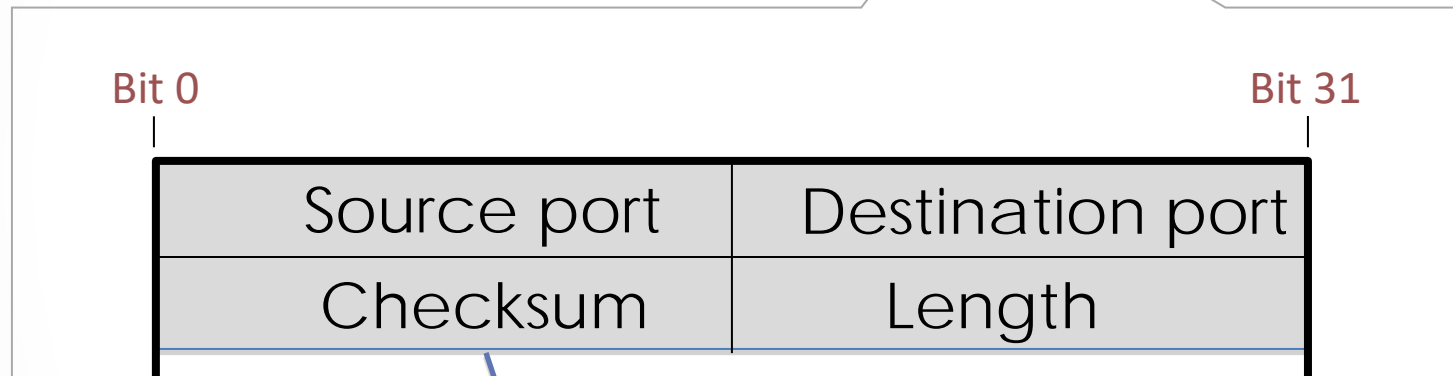
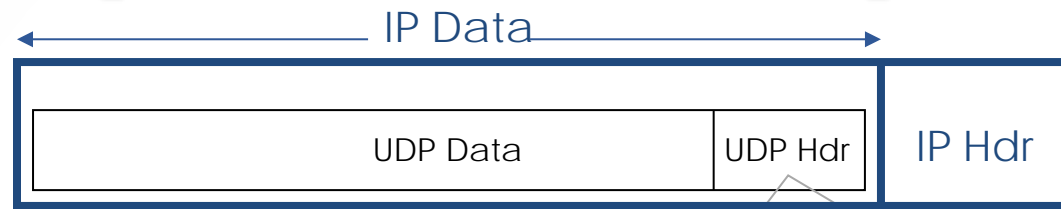
ТСР – что еще осталось

У нас остались не рассмотренными

- Как управлять потоком пакетов и сегментов?*
- Как должно быть организована повторная передача потерянных сегментов?*
- Послали сегмент, как долго ждать подтверждения получения?*
- Что делать если в сети начались «пробки»?*



Формат UDP дейтаграммы



Поля псевдозаголовка,
защищаемые контрольной суммой (если она не 0)

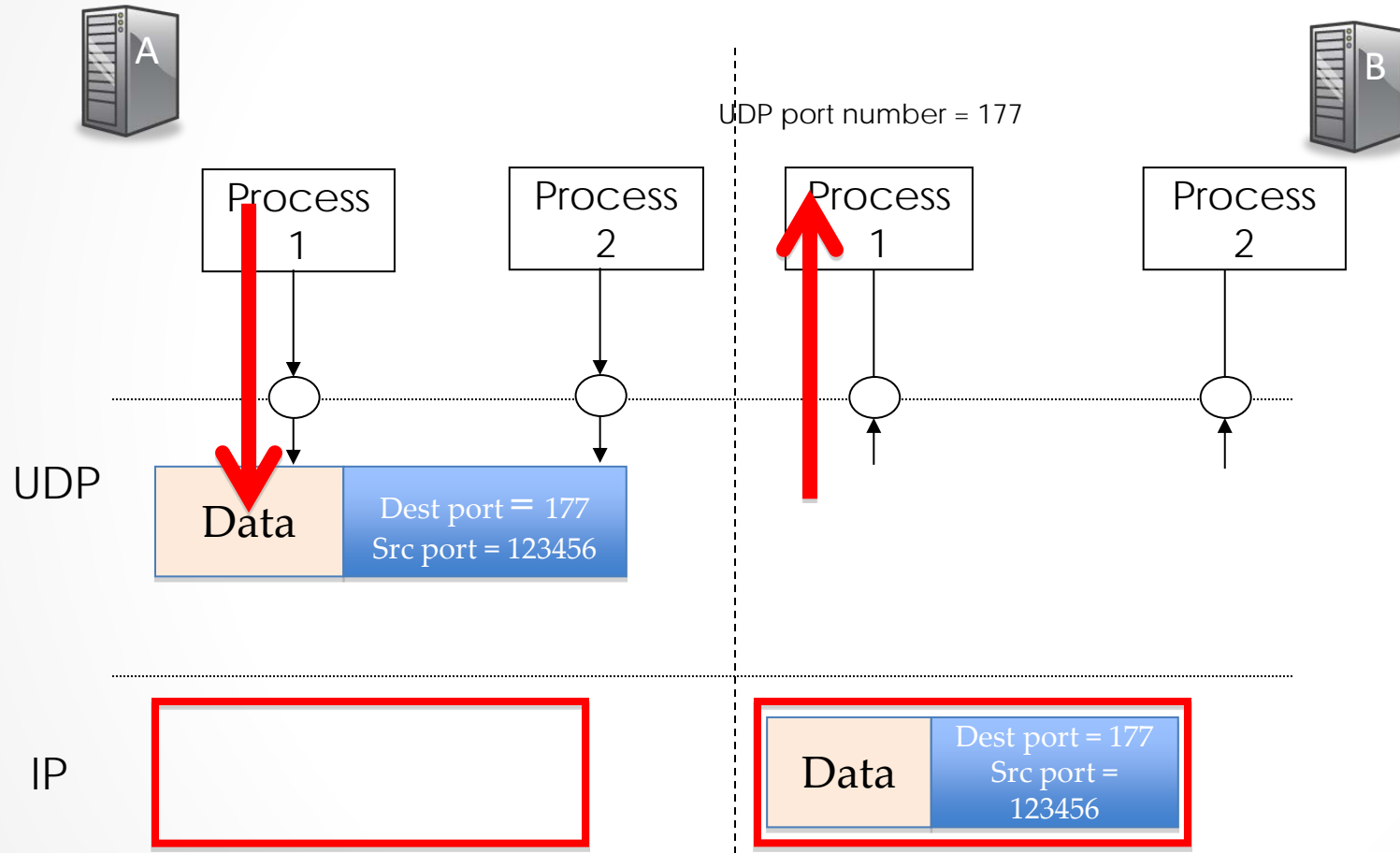
IPv4: IP DA, IP SA, Protocol ID for UDP (17), UDP data+hdr length

UDP: source port, destination port, length

UDP: data



UDP: распределение по портам





Модель UDP сервиса

Свойство	Поведение
<i>Сервис без соединения</i>	<i>Соединение не устанавливается Пакеты могут приходить в произвольном порядке</i>
<i>Дейтаграммы самодостаточны</i>	
<i>Ненадежная доставка</i>	<ol style="list-style-type: none"><i>Отсутствуют подтверждения (Аск)</i><i>Контрольная сумма охватывает только заголовков</i><i>Отсутствуют средства выявления пропущенных данных или данных, нарушающих порядок отправки.</i><i>Нет управления потоком</i>



Заключение

- TCP предоставляет сервис, сохраняющий порядок байтов потока, обеспечивающий надежную доставку
- UDP обеспечивает простую доставку дейтаграмм между приложениями.



Интернет: Обнаружение ошибок передачи

(Смелянский Компьютерные сети. т.1, стр. 116-122)

Введение в компьютерные сети

проф. Смелянский Р.Л.
Лаборатория Вычислительных комплексов
ф-т ВМК МГУ



Обнаружение и исправление ошибок

(см. учебник т.1 стр.116-122)

- *Ошибки единичные и групповые (блочные)*
- *Коды с обнаружением ошибок*
 - *кодослово*
 - *расстояние Хемминга*
- *Коды исправляющие ошибки*
0000000000, 0000011111, 1111100000, 1111111111
- $$(n+1)2^m \leq 2^n; (m+r+1) \leq 2^r$$



Коды с исправлением ошибок

- Код Хемминга для единичных ошибок
 - разряды кодослова нумеруют слева направо, начиная с 1;
 - все биты, номера которых есть степень 2 (1,2,4,8,16 и т.д.) - контрольные, остальные - биты сообщения;
 - каждый контрольный бит отвечает за четность группы битов, включая себя. Один и тот же бит может относиться к разным группам. Значение бита сообщения определяется по значениям контрольных битов. Чтобы определить какие контрольные биты контролируют бит в позиции k надо представить значение k по степеням двойки. Например, $11 = 1 + 2 + 8$, $39 = 1 + 2 + 4 + 32$.



Код Хемминга для исправления одиночных ошибок

Char.	ASCII	Check bits
H	1001000	00110010000
a	1100001	10111001001
m	1101101	11101010101
m	1101101	11101010101
i	1101001	01101011001
n	1101110	01101010110
g	1100111	11111001111
	0100000	10011000000
c	1100011	11111000011
o	1101111	00101011111
d	1100100	11111001100
e	1100101	00111000101

Order of bit transmission



Коды обнаруживающие ошибки

- Групповые ошибки
- Биты четности не позволяют эффективно бороться с групповыми ошибками
- Иногда перепослать дешевле, чем исправить
- CRC код (Cyclic Redundancy Code)
 - строка 110001 представляет полином $x^5+x^4+x^0$
 - арифметика выполняется по модулю 2



Избыточные циклические коды (CRC)

- Отправитель и получатель договариваются о конкретном образующем полиноме $G(x)$ степени r (коэффициенты при старшем члене и при младшем члене должны быть равны 1).
- Для вычисления контрольной суммы блока из m бит надо чтобы обязательно $m > r$.
- Добавить контрольную сумму к передаваемому блоку, рассматриваемому как полином $M(x)$ так, чтобы передаваемый блок с контрольной суммой был кратен $G(x)$. Когда получатель получает блок с контрольной суммой, он делит его на $G(x)$. Если есть остаток, то были ошибки при передаче.



Избыточные циклические коды

- Алгоритм вычисления контрольной суммы:
 - Добавить r нулей в конец блока так, что он теперь содержит $t+r$ разрядов и соответствует полиному $x^r M(x)$;
 - Разделить по модулю 2 полином $x^r M(x)$ на $G(x)$;
 - Вычесть остаток (длина которого всегда не более r разрядов) из строки, соответствующей $x^r M(x)$, по модулю 2. Результат и есть блок с контрольной суммой (назовем его $T(x)$).



Избыточные циклические коды

$(T(x) + E(x))$, где $E(x)$ – многочлен, где 1 – инвертированные при передаче биты

$$(T(x) + E(x)) / G(x) = E(x) / G(x) \quad \text{остаток}$$

$E(x) = x^r$ - единичная ошибка, где r – ошибочный бит

$E(x) = x^i + x^j = x^j (1 + x)^{i-j}$ - две изолированные ошибки

$x^{15} + x^{14} + 1$ не является делителем для $x^k + 1$ для k от 1 до 32 768

Дома показать, что никакой многочлен с нечетным количеством членов не делится на $(x + 1)$



Коды обнаруживающие ошибки

- Существует три международных стандарта на вид $G(x)$:
 - $CRC-12$ $= x^{12} + x^{11} + x^3 + x^2 + x + 1$
 - $CRC-16$ $= x^{16} + x^{15} + x^2 + 1$
 - $CRC-CCITT$ $= x^{16} + x^{12} + x^5 + 1$
- $CRC-12$ используется для передачи символов из 6 разрядов. Два остальных - для 8 разрядных. $CRC-16$ и $CRC-CCITT$ ловят одиночные, двойные ошибки, групповые ошибки длины не более 16 и нечетное число изолированных ошибок с вероятностью 99,997%.



Контрольная сумма в IP пакете

- *IP, UDP и TCP используют один и тот же алгоритм комплементарной контрольной суммы:*
 - *Установить поле checksum= 0*
 - *Сложить все 16 разрядные слова в пакете (в IP пакете суммируется только заголовок)*
 - *Установить разряд четности*
 - *Контрольная сумма должны быть такой чтобы сумма всего пакета, включая контрольную сумму была бы 0xffff*
- *Основное достоинство - простота*
- *Недостаток - слабая защита от ошибок (только одиночные ошибки).*



Схемы обнаружения ошибок

Заключение

- *Контрольная сумма добавляется в IP, TCP, UDP пакеты*
 - *Быстро, дешево*
 - *Неустойчиво*
 - *L2 - защищает весь кадр*
 - *L3 - только заголовок*
 - *L4 - часть заголовка и тело*
- *CRC коды используются в Ethernet кадрах*
 - *Дороже чем контрольная сумма*
 - *Устойчивы к двукратным ошибкам, групповым ошибкам и ошибкам четности*
- *Использование кодов с обнаружением ошибок (бит четности)*



Средства описания протоколов: Диаграммы переходов для конечных автоматов

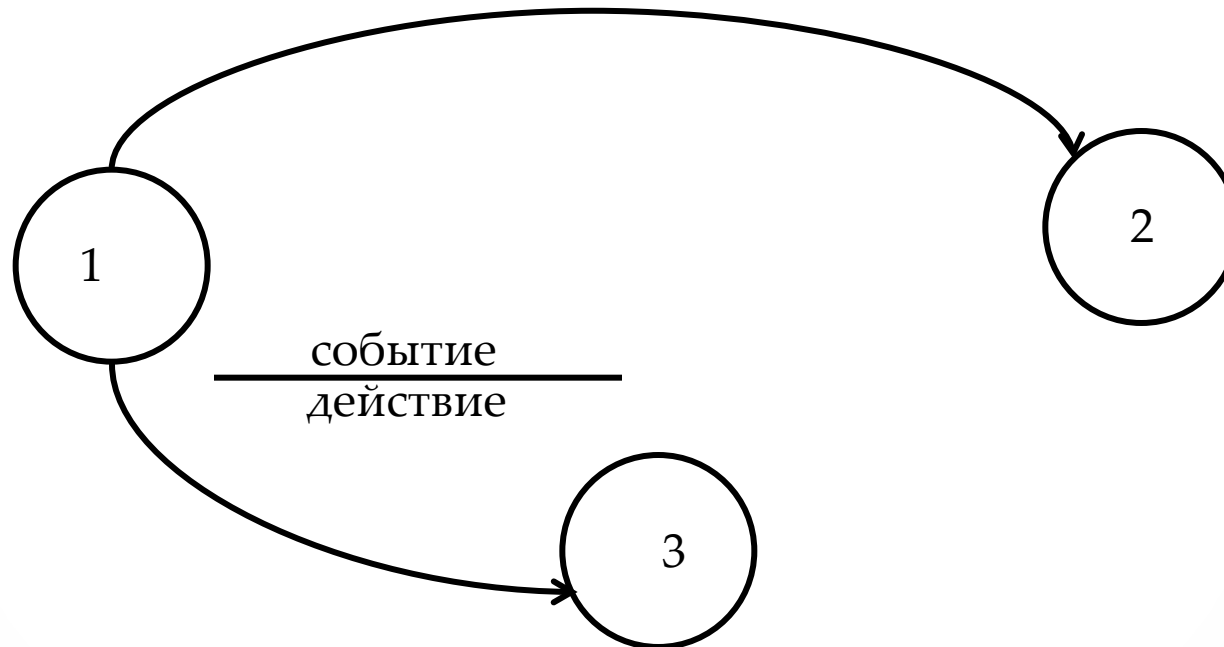


Конечный автомат

$\langle A_{in}; A_{out}; S; Q: aq \rightarrow bq'; s^* \rangle$

Событие, вызывающее переход

Действие при изменении состояния





Пример: HTTP запрос на ТСР клиенте

Idle

Open

Rqst

Rsp



Пример: HTTP запрос

Idle

Open

Rqst

Rsp



Интернет: управление ПОТОКОМ

Введение в компьютерные сети
чл.-корр. РАН Смелянский Р.Л.
Кафедра АСВК
ф-т ВМК МГУ



Управление потоком

- *Не посылать пакетов больше, чем может принять получатель*
- *Есть обратная связь между отправителем и получателем*
- *Два основных подхода:*
 - *Stop and Wait*
 - *Скользящее окно (Sliding Window)*



Управление потоком (stop and wait)

- *В одно и то же время передают не более одного пакета*
- *Sender отправляет пакет*
- *Receiver посылает пакет с ask, когда получает пакет данных*
- *Получив ask, sender шлет новый пакет с данными*
- *По time_out, sender повторно посылает пакет с данными*
- *Счетчик на 1 бит позволяет выявлять дублирование*



Управление ПОТОКОМ (т.1 стр. 122-128)

```

#define MAX_PKT 1024          /* determines packet size in bytes */

typedef enum {false, true} boolean; /* boolean type */
typedef unsigned int seq_nr;        /* sequence or ack numbers */
typedef struct {unsigned char data[MAX_PKT]} packet; /* packet definition */
typedef enum {data, ack, nak} frame_kind; /* frame_kind definition */

typedef struct {
    frame_kind kind;          /* frames are transported in this layer */
    seq_nr seq;              /* what kind of a frame is it? */
    seq_nr ack;              /* sequence number */
    packet info;             /* acknowledgement number */
} frame;                    /* the network layer packet */

/* Wait for an event to happen; return its type in event. */
void wait_for_event(event_type *event);

/* Fetch a packet from the network layer for transmission on the channel. */
void from_network_layer(packet *p);

/* Deliver information from an inbound frame to the network layer. */
void to_network_layer(packet *p);

/* Go get an inbound frame from the physical layer and copy it to r. */
void from_physical_layer(frame *r);

/* Pass the frame to the physical layer for transmission. */
void to_physical_layer(frame *s);

/* Start the clock running and enable the timeout event. */
void start_timer(seq_nr k);

/* Stop the clock and disable the timeout event. */
void stop_timer(seq_nr k);

/* Start an auxiliary timer and enable the ack_timeout event. */
void start_ack_timer(void);

/* Stop the auxiliary timer and disable the ack_timeout event. */
void stop_ack_timer(void);

/* Allow the network layer to cause a network_layer_ready event. */
void enable_network_layer(void);

/* Forbid the network layer from causing a network_layer_ready event. */
void disable_network_layer(void);

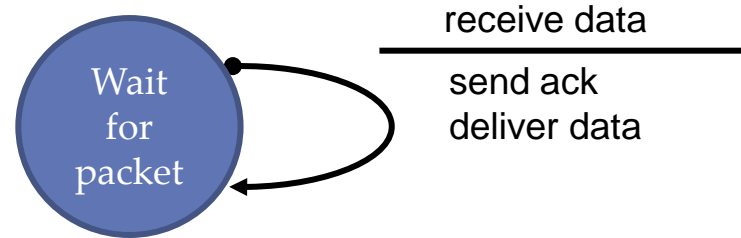
/* Macro inc is expanded in-line: Increment k circularly. */
#define inc(k) if (k < MAX_SEQ) k = k + 1; else k = 0

```

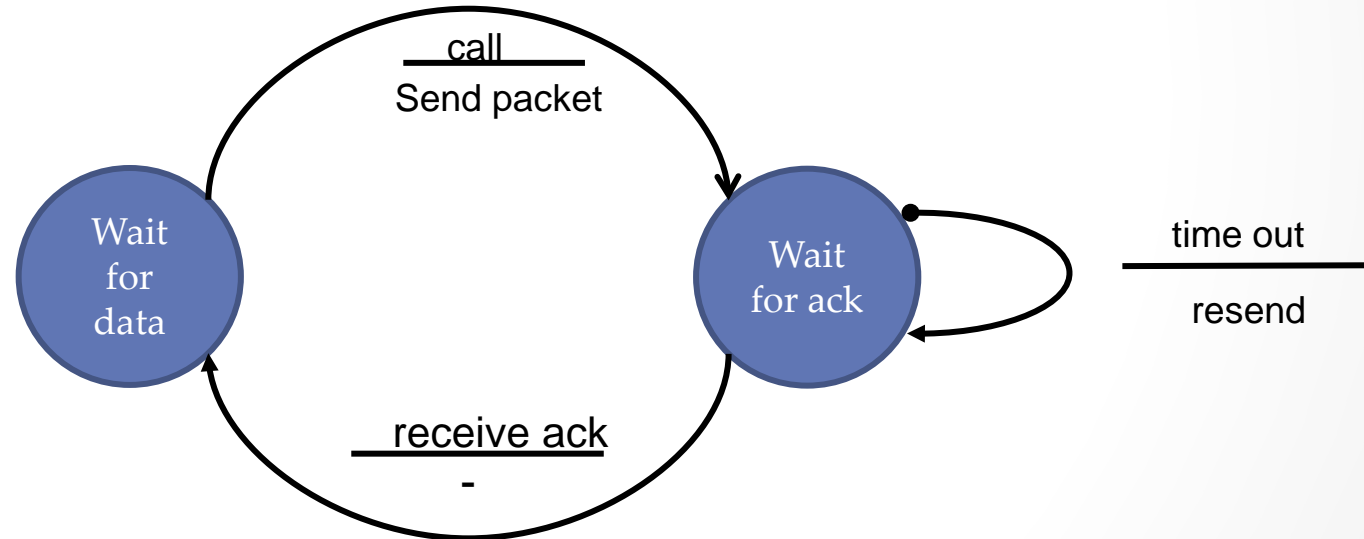


Управление потоком (Stop and Wait)

Receiver FSM



Sender FSM





/* Protocol 1 (utopia) provides for data transmission in one direction only, from sender to receiver. The communication channel is assumed to be error free, and the receiver is assumed to be able to process all the input infinitely fast. Consequently, the sender just sits in a loop pumping data out onto the line as fast as it can. */

```
typedef enum {frame_arrival} event_type;
#include "protocol.h"

void sender1(void)
{
    frame s;          /* buffer for an outbound frame */
    packet buffer;    /* buffer for an outbound packet */

    while (true) {
        from_network_layer(&buffer);    /* go get something to send */
        s.info = buffer;                /* copy it into s for transmission */
        to_physical_layer(&s);         /* send it on its way */
    }
    /* Tomorrow, and tomorrow, and tomorrow,
       Creeps in this petty pace from day to day
       To the last syllable of recorded time
       - Macbeth, V, v */
}

void receiver1(void)
{
    frame r;
    event_type event;    /* filled in by wait, but not used here */

    while (true) {
        wait_for_event(&event); /* only possibility is frame_arrival */
        from_physical_layer(&r); /* go get the inbound frame */
        to_network_layer(&r.info); /* pass the data to the network layer */
    }
}
```




/* Protocol 2 (stop-and-wait) also provides for a one-directional flow of data from sender to receiver. The communication channel is once again assumed to be error free, as in protocol 1. However, this time, the receiver has only a finite buffer capacity and a finite processing speed, so the protocol must explicitly prevent the sender from flooding the receiver with data faster than it can be handled. */

```
typedef enum {frame_arrival} event_type;
#include "protocol.h"

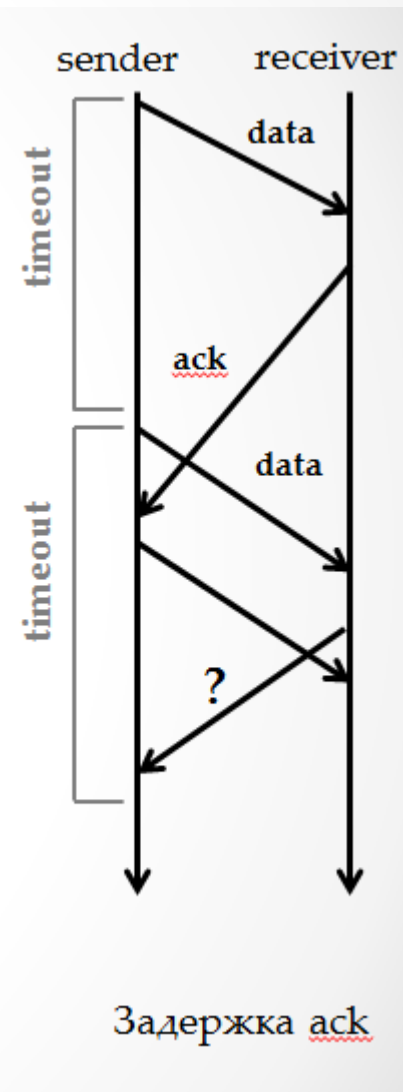
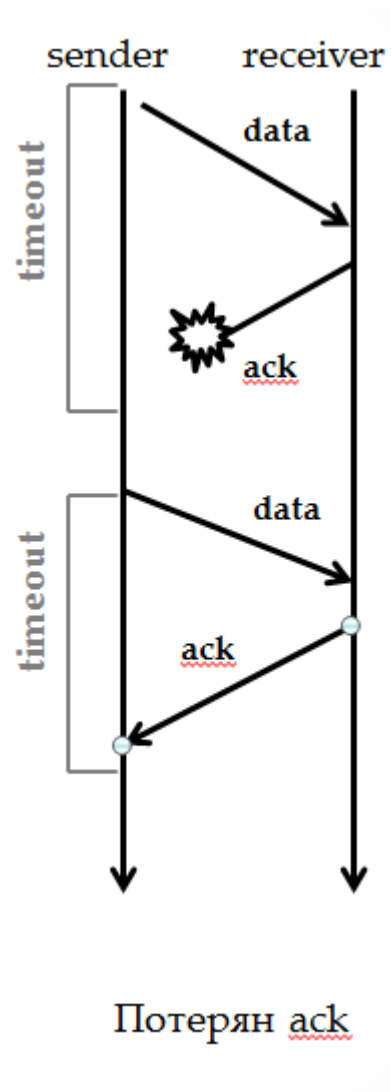
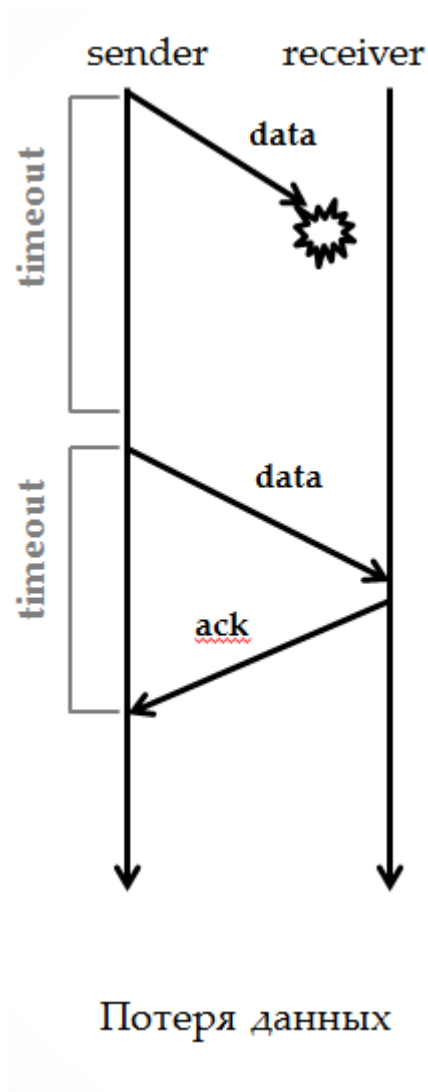
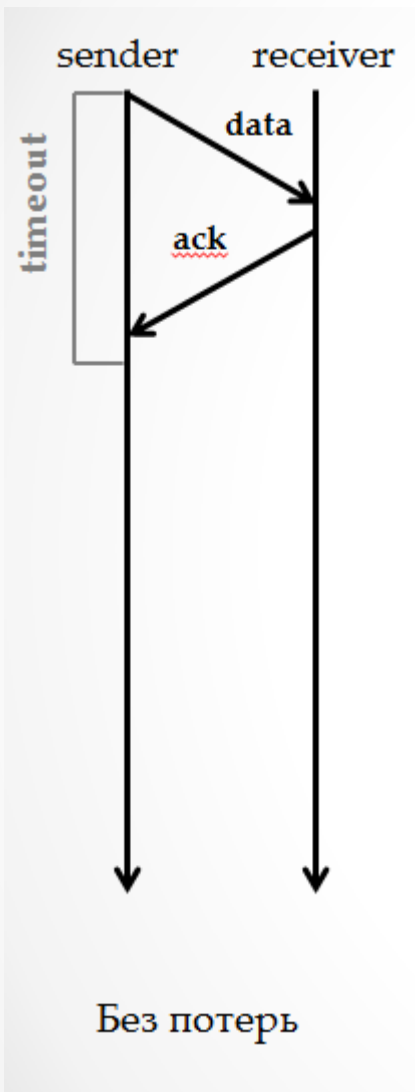
void sender2(void)
{
    frame s;                /* buffer for an outbound frame */
    packet buffer;          /* buffer for an outbound packet */
    event_type event;       /* frame_arrival is the only possibility */

    while (true) {
        from_network_layer(&buffer);          /* go get something to send */
        s.info = buffer;                      /* copy it into s for transmission */
        to_physical_layer(&s);               /* bye bye little frame */
        wait_for_event(&event);              /* wait event = ack V time_out the go ahead */
    }
}

void receiver2(void)
{
    frame r, s;              /* buffers for frames */
    event_type event;       /* frame_arrival is the only possibility */
    while (true) {
        wait_for_event(&event);          /* send a packet with ack rival */
        from_physical_layer(&r);         /* go get the inbound frame */
        to_network_layer(&r.info);      /* pass the data to the network layer */
        to_physical_layer(&s);          /* send a dummy frame to awaken sender */
    }
}
```



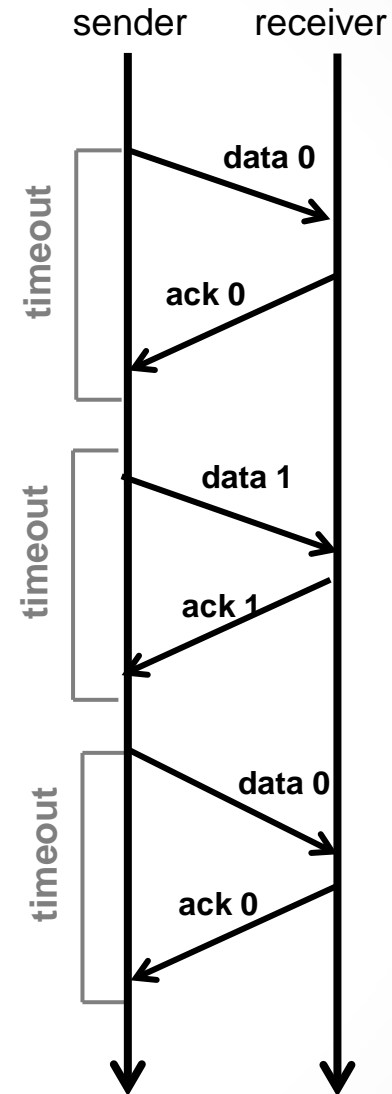
Управление потоком





Дублирование

- *Счетчик на 1 бит в данных и уведомлении позволяет отличать новые данные от дубликатов*
- *Будем предполагать*
 - *Сама сеть не размножает пакеты*
 - *Запаздывание пакетов гарантированно не более одного time_out*

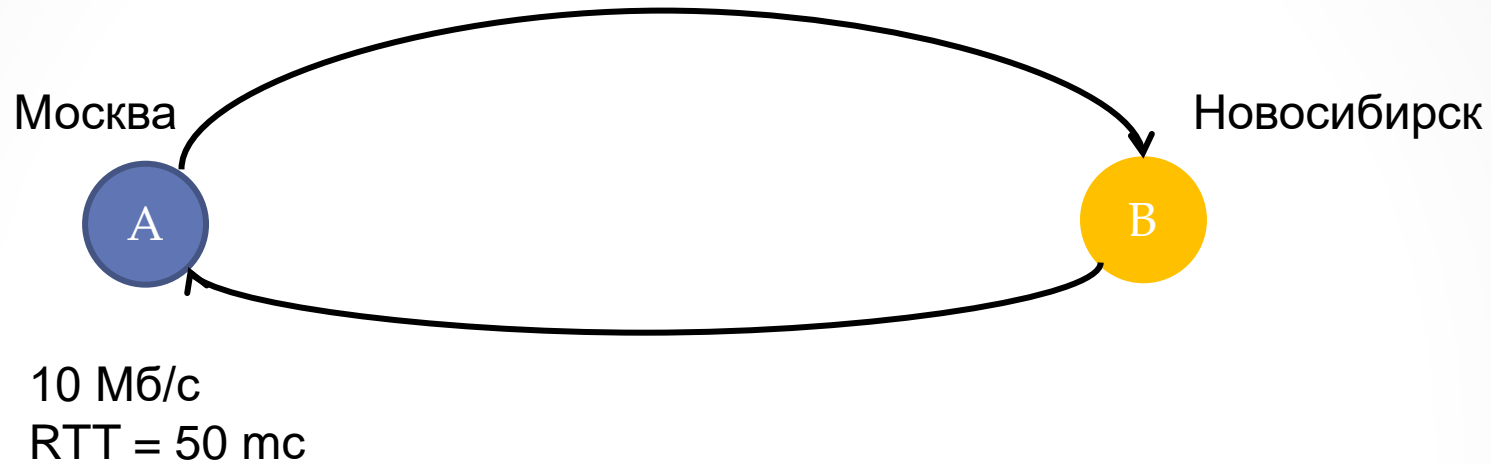




Интернет: управление ПОТОКОМ (Sliding Window)



Проблема



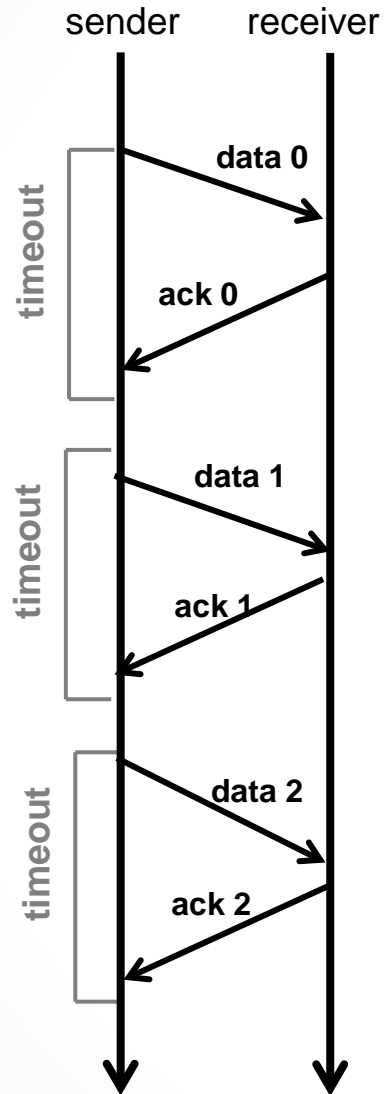
Макс. пропускная способность 10Мб/с
RTT - 50мс

Сделаем обобщение S&W протокола:

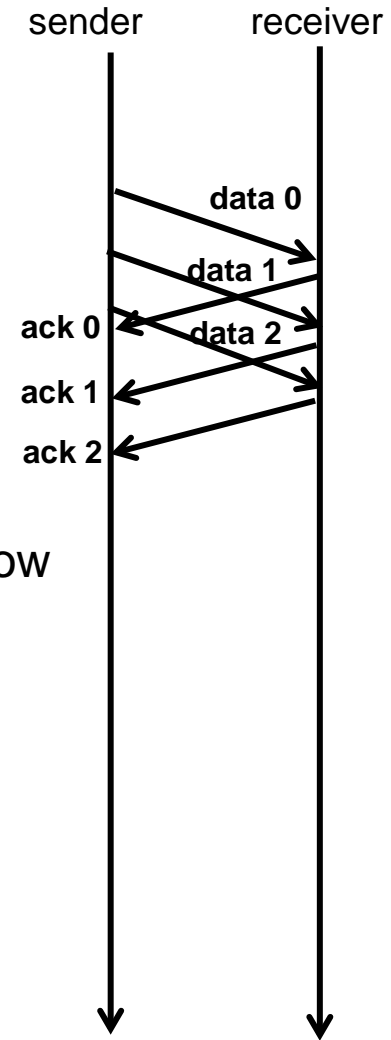
- Разрешаем использовать сразу несколько неподтвержденных сегментов
- Максимальное число таких сегментов - окно
- Можем плотно «забить» канал



Пример



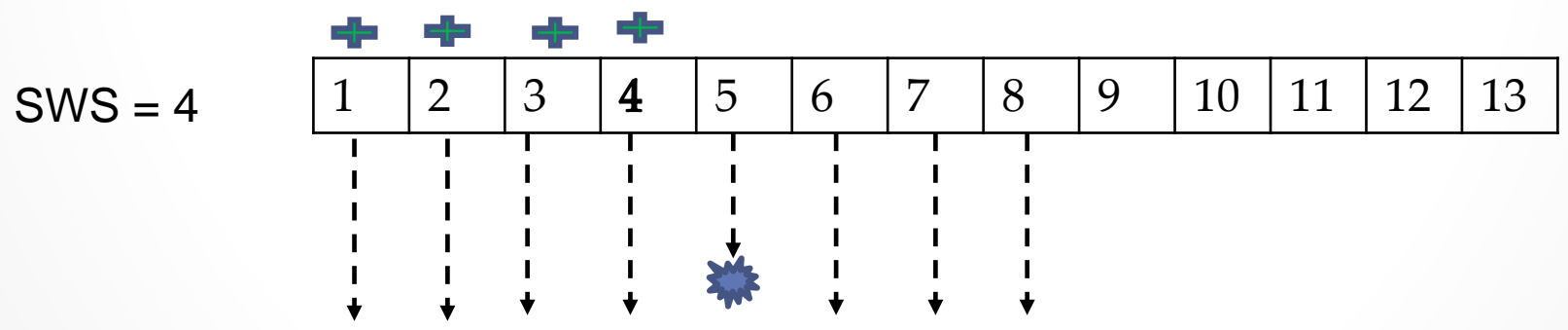
Sliding Window





SW Sender

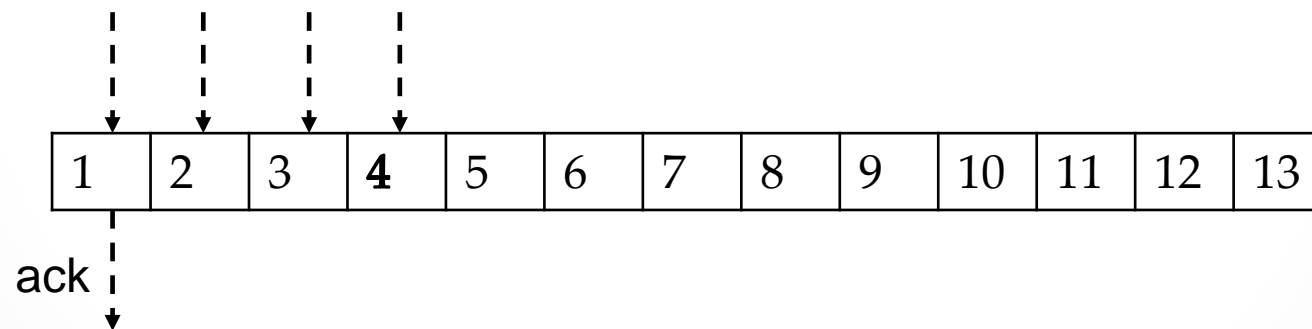
- У каждого сегмента есть последовательный номер
- Поддерживаются 3 переменных
 - Размер окна отправки (SWS)
 - Последнее полученное от получателя подтверждение (LAR)
 - Последний отправленный сегмент (LSS)
- Всегда $(LSS - LAR) \leq SWS$
- LAR возрастает при каждом новом подтверждении
- Буфер на SWS сегментов





SW Receiver

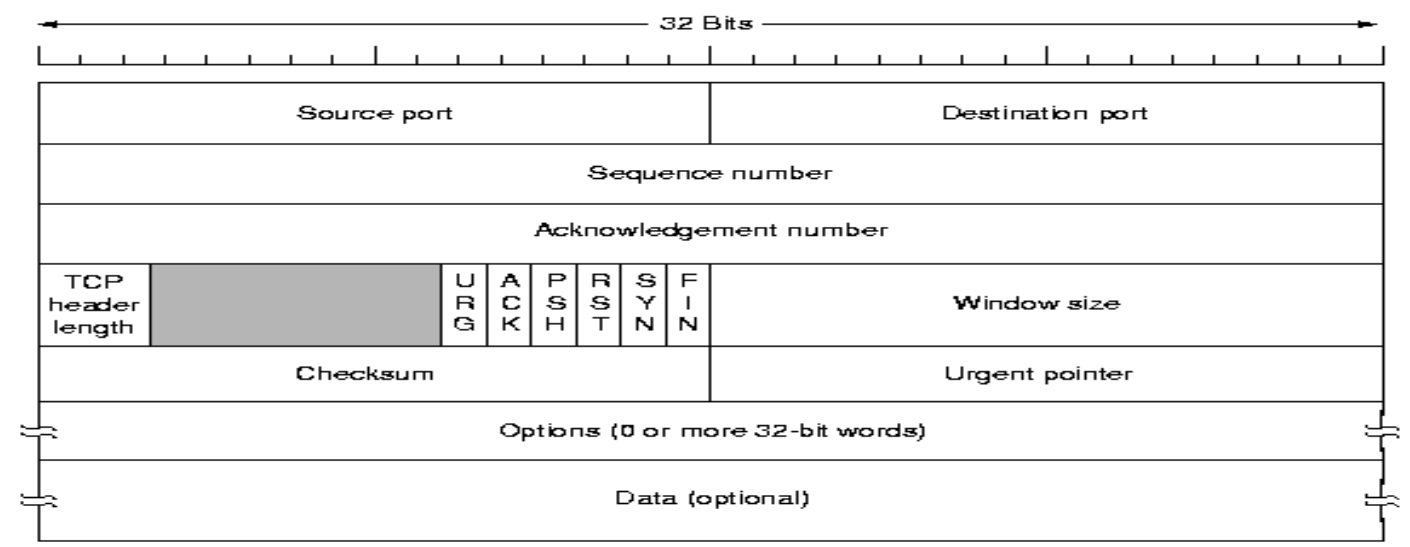
- **Поддерживаются 3 переменных**
 - Размер окна получения (*SWR*)
 - Наибольший допустимый номер сегмента (*LAS*)
 - Последний полученный сегмент (*LRS*)
- **Всегда $(LAS - LRS) \leq SWR$**
- **Если номер полученного сегмента $< LAS$, то шли подтверждение**
 - **Накопительный ack:** если получены 1,2,3,5 - подтверждаем 3
 - **TCP** в уведомлении подтверждает номер ожидаемого сегмента (т.е. 4 в предыдущем примере)





Управление потоком в TCP

- Receiver информирует о размере SWR через поле Window в TCP заголовке
- Sender может посылать данные с номерами не больше LAR+ window





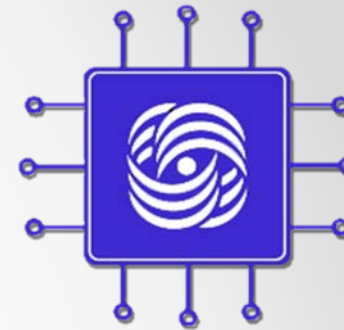
SWR, SWS и пространство последовательных номеров

- $SWR \geq 1, SWS \geq 1, SWS \geq SWR$
- Если $SWR=1$, "go back N" протокол, то $SWS+1$ последовательных номеров будут неподтвержденными
- Если $SWR=SWS$, нужно $2 SWS$ последовательных номеров
- В общем случае нужно $SWR + SWS$ номеров
 - SWR пакеты в неизвестном состоянии (аск могут быть утеряны)
 - SWS пакеты в пути, но не должны переполнять пространство последовательных номеров



Управление потоком с SW

- *Допускает в окне только пакеты в пути, т.е. неподтвержденные*
- *Как только пришло уведомление, окно сдвигается*
- *Необходимое пространство последовательных номеров зависит от размера окна (поле window)*
- *Необходимо согласовать размеры SWS и SWR*



Интернет: модель ISMP сервисов

Введение в компьютерные сети
чл.-корр. РАН, проф. Смелянский Р.Л.
Кафедра АСВК
ф-т ВМК МГУ

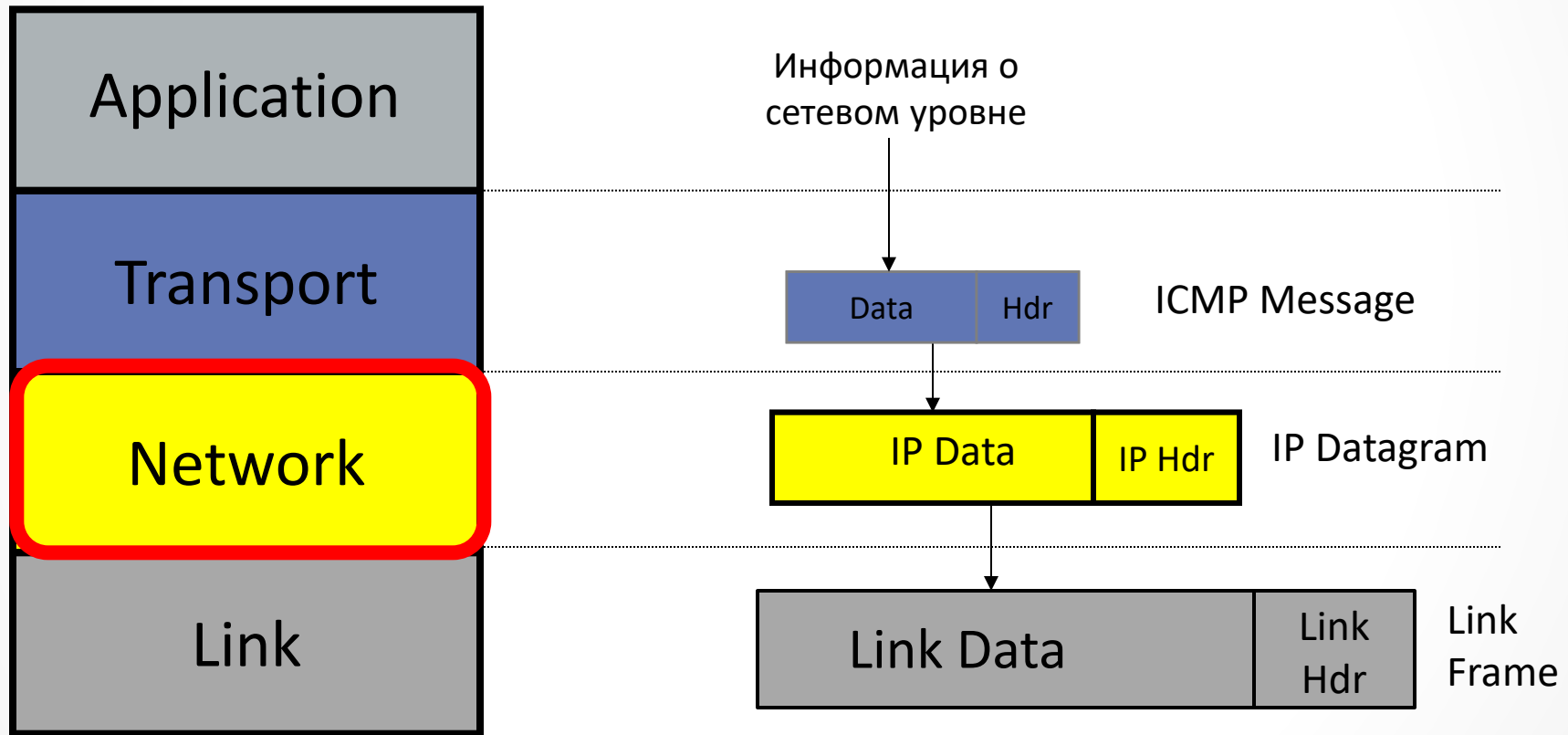


ICMP протокол

- Предназначен для поддержки IP протокола
- IP протокол
 - Создает IP дейтаграммы
 - Передает их hop-by-hop
 - Формирует таблицы маршрутизации
 - Алгоритмы распространения таких таблиц
- ICMP протокол
 - Обеспечивает коммуникацию между сетевыми уровнями хостов и маршрутизаторов
 - Сообщения об ошибках
 - Диагностика проблем

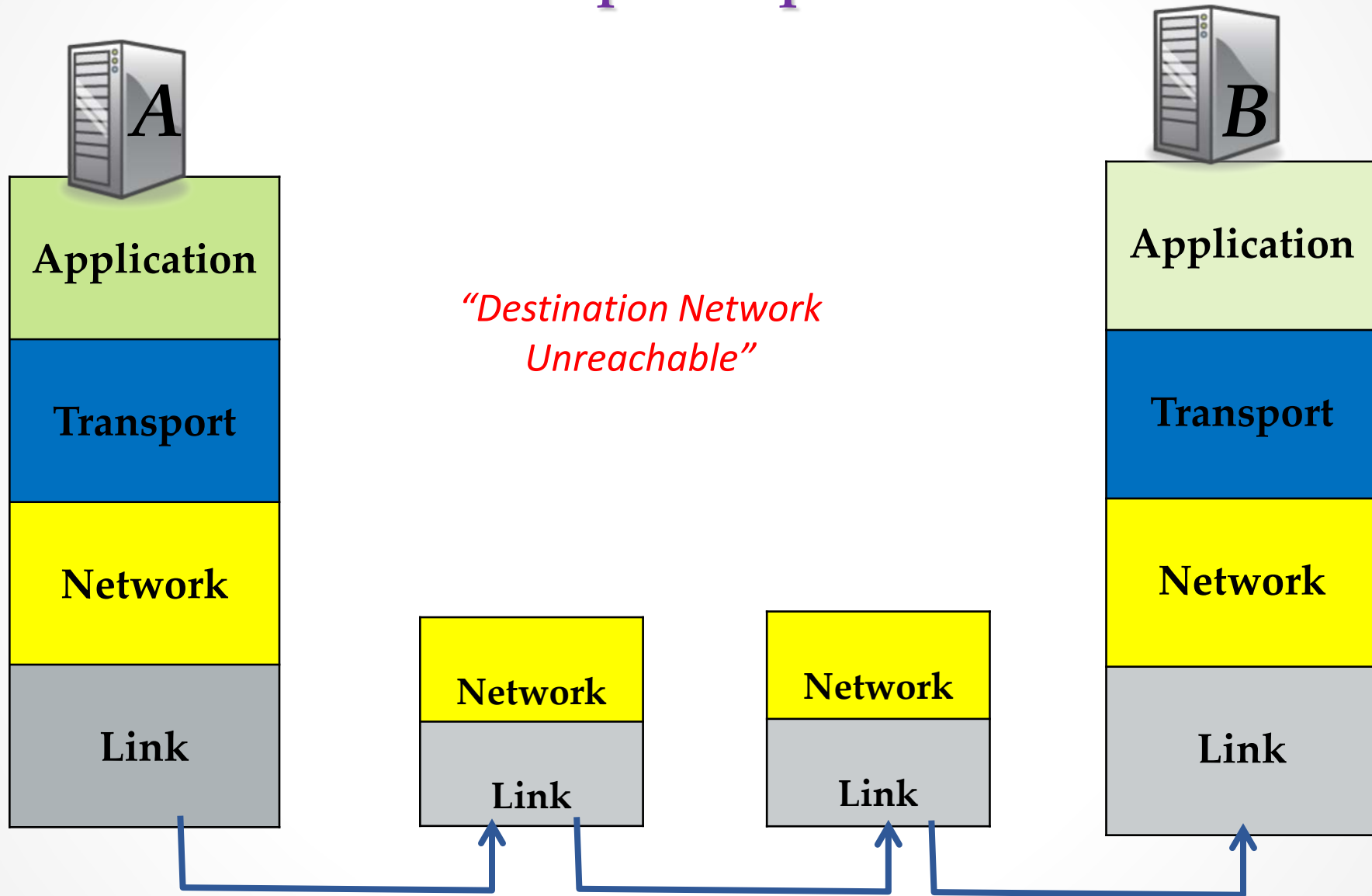


ICMP работает над сетевым уровнем





Пример





Модель ISMR сервиса

Свойство	Поведение
<i>Сообщение о состоянии</i>	<i>Самодостаточное сообщение об ошибке или исключительном состоянии</i>
<i>Ненадежный</i>	<i>Сервис без соединения и подтверждения</i>



Примеры типов и кодов ICMP сообщений

ICMP тип	ICMP код	Назначение
0	0	Echo Reply (ping)
3	0	Destination Network Unreachable
3	1	Destination Host Unreachable
3	3	Destination Port Unreachable
8	0	Echo Request (ping)
11	0	TTL Expired (traceroute)

RFC 792



Структура заголовка ICMP сообщения



Заголовок ICMP-сообщения состоит из 8 байт:

- тип (1 байт) – числовой идентификатор типа сообщения:
 - 0 или 8, где 0 - **ICMP reply** (ответ), 8 - **ICMP request** (запрос);
- код (1 байт) – числовой идентификатор, точно определяющий тип ошибки
- контрольная сумма (2 байта) – вычисляется для всего ICMP-сообщения
- Оставшиеся 4 байта и поле данных зависит от значений полей типа и кода.



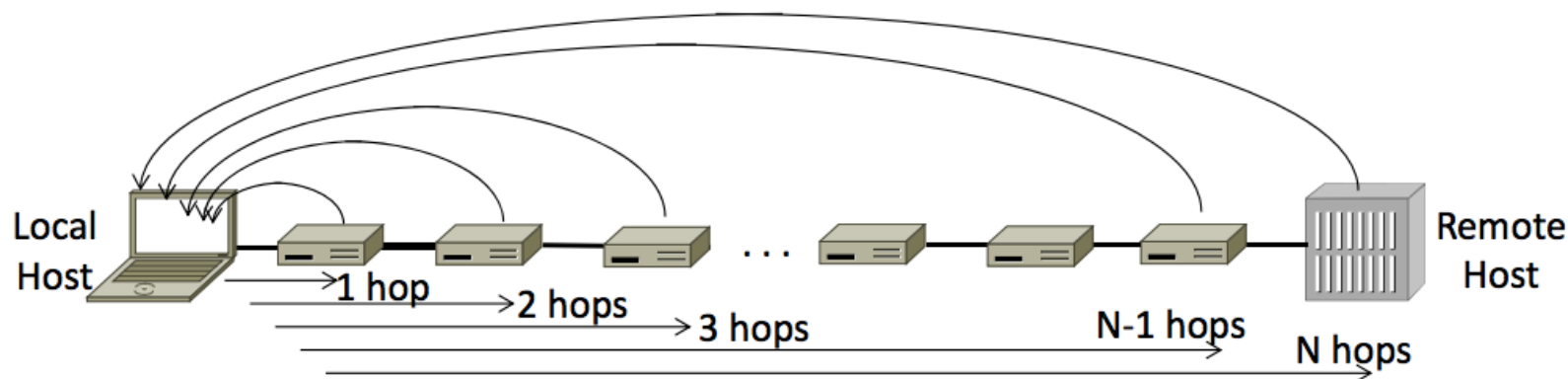
ping

- Команда ping использует сообщения
 - эхо запроса (Echo Request) и
 - эхоответа (Echo Reply) протокола ICMP
- Используется для диагностики работоспособности сети.
- Пример диагностики сети:
 - `ping 127.0.0.1` (проверка работы адреса замыкания на себя)
 - `ping <local ip>` (проверка связи с ip адресом локального компьютера)
 - `ping <default gateway>` (проверка связи со шлюзом по умолчанию)
 - `ping <remote ip>` (проверка связи с удаленным узлом)
- Возможные ответы команды ping
 - Получен обычный echo-ответ
 - Echo-ответ от запрашиваемого узла не был получен
 - Получено сообщение о недостижимости узла-получателя
 - Получено сообщение о невозможности фрагментации
 - Получен неизвестный пакет



traceroute / tracert

- Отправляет пробные пакеты с TTL=1, увеличивая значение счетчика на каждой итерации
- Сообщения об ошибках ICMP идентифицируют узлы маршрута





Заключение

- ICMP предоставляет информацию о сетевом уровне хостам и маршрутизаторам
- ICMP работает над IP и относится к транспортному уровню
- ping и traceroute реализованы с помощью ICMP



Демо

- Как узнать свой локальный IP?
 - Выход в командную строку (cmd)
 - Ipconfig /all
- Как узнать свой внешний IP?
 - <http://ip-address.ru/>
- Как меняется внешний IP, в зависимости от точки выхода во внешнюю сеть.
 - Переключиться на WiFi модем телефона и все повторяем
- Как узнать какому домену и к какой Автономной Системе принадлежит выделенный внешний IP?
 - <http://whatismyipaddress.com/ip-lookup>
 - <https://www.nic.ru/whois/>
 - <https://apps.db.ripe.net/search/query.html>
- Как узнать доступность хоста по его имени?
 - Выход в командную строку (cmd)
 - Ping имя или IP
- Как узнать маршрут доступа к хосту по имени?
 - tracert имя или IP
- Препарирование протоколов – Wireshark.