Архитектура компьютерных сетей

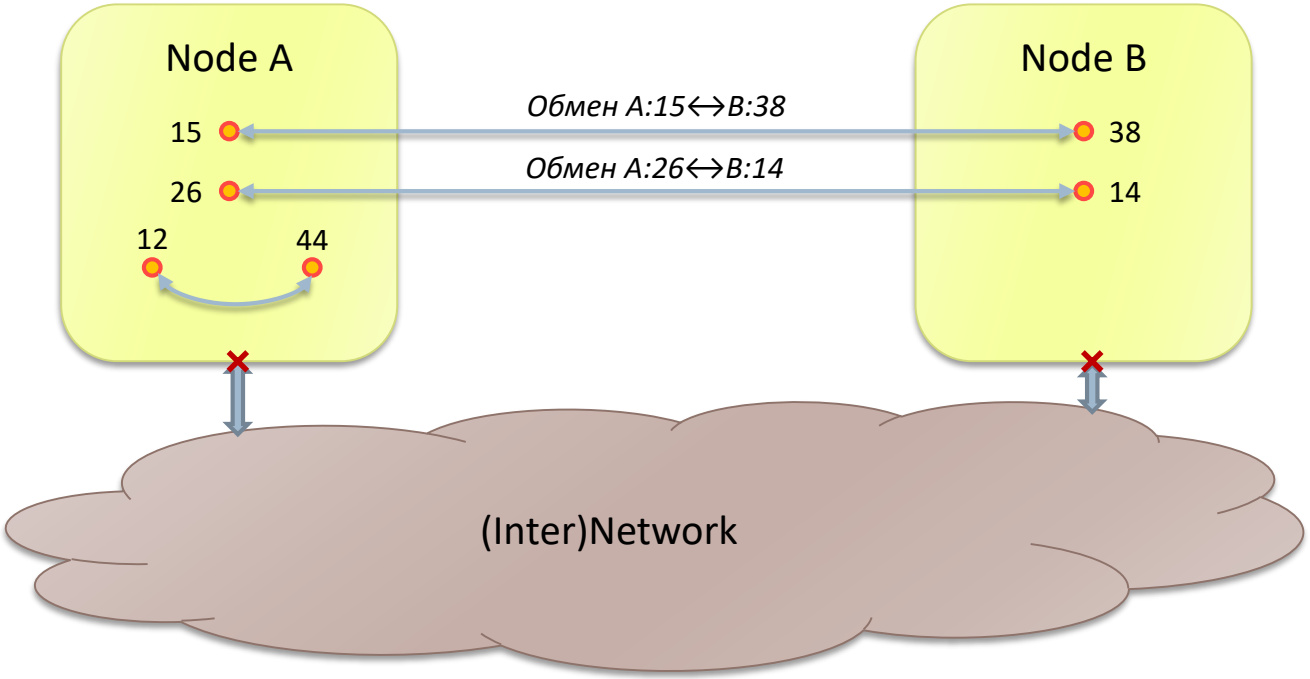
Останков Александр Иванович

План курса

1. Введение в компьютерные сети
2. Основные методы построения СПД
3. Архитектура Internet Protocol Suite (TCP/IP)
4. Архитектура модулей физического уровня
5. Технологии беспроводных сетей
6. Архитектура модулей канального уровня
7. **Протоколы транспортного уровня**
    - 5.1. Назначение и основные функции транспортного уровня
    - 5.2. Режимы обмена данными и протоколы транспортного уровня
    - 5.3. Идентификация SAP
    - 5.4. Протокол UDP
    - 5.5. Протокол TCP
8. Технологии WWW

Назначение транспортного уровня

Протокол сетевого уровня (IP) обеспечивает передачу пакетов от одного узла к другому. Протокол транспортного уровня должен обеспечивать передачу данных от одной SAP к другой SAP



SAP внутри узла может быть создано много (динамически). Транспортный модуль должен знать и идентифицировать все «свои» SAP, открытые в данный момент.

Функции транспортного уровня

- Предоставлять удобный API для управления SAP (socket-ами): создания, закрытия, определения режимов работы, присвоения адресов, установки и ожидания установки соединений и т.п.
- Обеспечивать параллельное одновременное функционирование множества SAP в пределах одного узла и при этом не путать данные разных SAP
- Реализовывать API для обмена данными приложений через SAP (socket-ы) в определенном режиме: дейтаграммном, потокоориентированном и др.
- Разбивать поток данных приложений на сегменты для передачи и выполнять сборку потока данных из полученных сегментов при приеме
- Восстанавливать сегменты, потерянные в процессе передачи по сети (*)
- Обеспечивать прием данных приложений в том же порядке, в котором они были переданы (*)
- Управлять темпом передачи данных для согласования скорости передачи данных со скоростью их приема/обработки (*)
- Принимать меры по уменьшению вероятности перегрузки трафиком сетевой инфраструктуры (network congestion avoidance) (*)

Режимы обмена данными между SAP

Дейтаграммный:

- По сети передается совокупность независимых сообщений (дейтаграмм)
- Гарантируется целостность сообщения
- Не гарантируется доставка, сохранение порядка сообщений
- Размер сообщения ограничен (64 кбайт)
- Не требуется установки соединения
- Через один socket возможен как двухсторонний обмен, так и групповые режимы (broadcast, multicast)

Потокоориентированный:

- Организуется передача по сети **потоков** (упорядоченных последовательностей) произвольных байтов
- Гарантируется доставка байтов и сохранение их порядка в потоке
- Не гарантируется сохранение границ между отправляемыми порциями
- Объем передаваемых данных не ограничен (определяется отправителем)
- Перед обменом данными необходимо устанавливать соединение
- Обмен возможен только между двумя SAP, входящими в соединение

Протоколы транспортного уровня IPS

В рамках **IPS** определены два основных транспортных протокола:

- дейтаграммный сервис – **UDP** (User Datagram Protocol)
- потокоориентированный сервис – **TCP** (Transmission Control Protocol)

Они обеспечивают различный сервис и взаимно дополняют друг друга:

- ✓ **UDP** это легковесный протокол (всего 8 байт накладных расходов), не требующий установки соединения и способный обеспечивать быструю передачу между несколькими взаимодействующими SAP
- ✓ С другой стороны **UDP** это ненадежный протокол с ограниченным размером PDU, поэтому использующие его прикладные программы должны сами реализовывать методы восстановление потерянных данных
- ✓ **TCP** это комплексный протокол реализующий множество сложных функций (ARQ, управление потоком, предотвращение перегрузок и т.п.) и гарантирующий доставку данных ценой увеличения накладных расходов и времени
- ✓ При работе с **TCP** прикладная программа должна реализовывать логику установки соединения, а также метод упаковки сообщений в потоке байтов

Идентификация SAP внутри узла

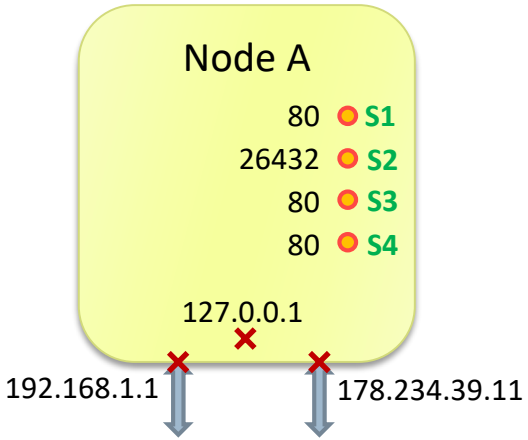
Поскольку на одном узле одновременно может функционировать множество SAP, соответствующие протокольные модули (TCP или UDP) используют для их идентификации *16 битные целые числа*, называемые **номерами портов**.

Полный (абсолютный) адрес SAP представляет собой:

<Proto>:<IP-addr>: <port>

Например:

Адрес **S1**: **UDP:0.0.0.0:80**
Адрес **S2**: **TCP:0.0.0.0:26432**
Адрес **S3**: **TCP:192.168.1.1:80**
Адрес **S4**: **TCP:127.0.0.1:80**



Перед началом использования SAP его адрес задается:

- явным присвоением путем вызова функции socket-API ***bind(…)***
- автоматическим назначением свободного номера порта (ephemeral port)

Система контролирует корректность присвоения и однозначность адресов SAP. Например, попытка присвоения ***UDP:178.234.39.11:80*** приведет к ошибке *«Address already in use»*, а ***TCP: 10.54.0.12:8080*** – к ошибке *«Cannot assign requested address»*

Сценарий использования UDP

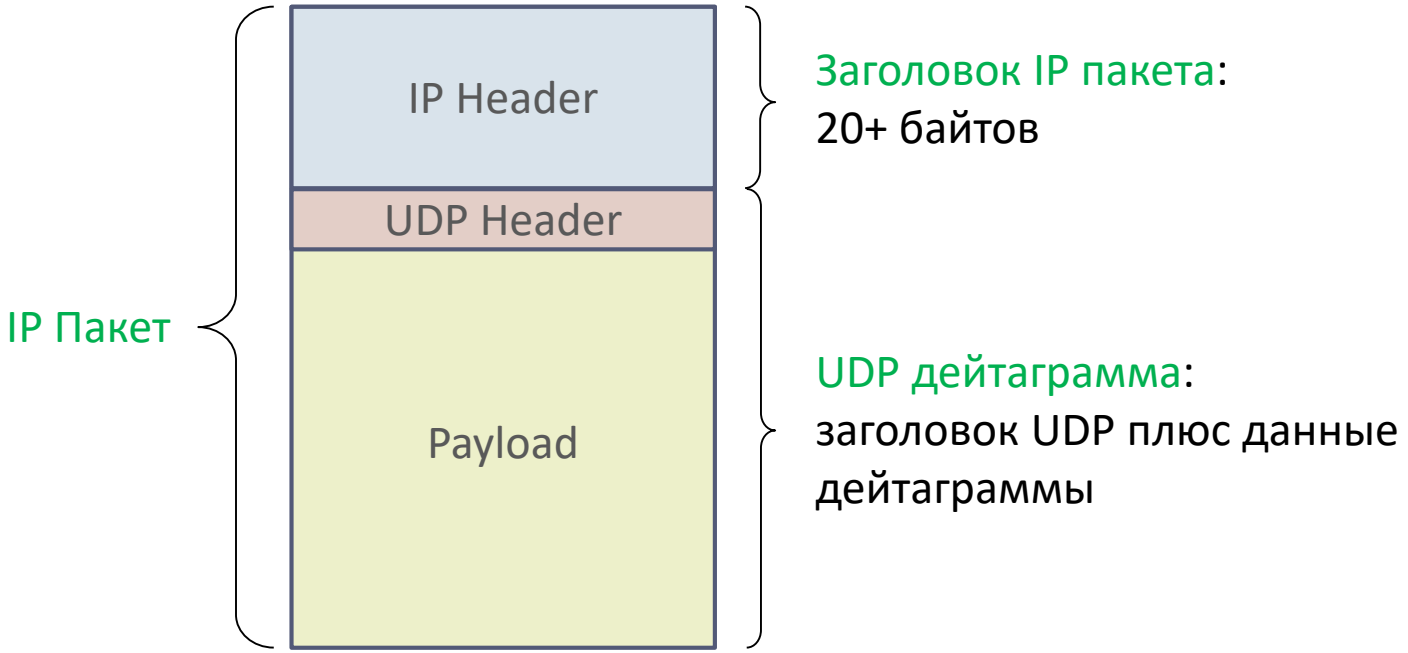
```
s = socket(                          /* Создание непоименованного socket */
      AF_INET,                       /* - socket относится к IPS */
      SOCK_DGRAM,                    /* - режим обмена дейтаграммный */
      PROTO_UDP);                    /* - протокол UDP */

 bind(s,                             /* Присвоение адреса socket-у */
      &saddr, sizeof(saddr));        /* sockaddr_in saddr; (IP+ном.порта) */

sendto(s,                            /* Отправить дейтаграмму */
      &sndbuf, msglen,               /* - буфер сообщения, длина в байтах */
      0,                             /* - поле options (не используется) */
      &dstaddr, sizeof(dstaddr));    /* - sockaddr_in с адресом получателя */

rcvbytes = recvfrom(s,               /* Принять дейтаграмму */
      &rcvbuf, sizeof(rcvbuf),       /* - буфер для приема данных */
      0,                             /* - поле options (не используется) */
      &fromaddr, &fromaddrlen);      /* - буфер, куда будет записан адрес
                                        отправителя и его длина */
```

Структура UDP пакета



UDP заголовок:

(16 бит) **Source Port –** порт отправителя	*(16 бит)* **Destination Port –** порт получателя
(16 бит) **Общая длина дейтаграммы в байтах**	*(16 бит)* Контрольная сумма дейтаграммы

Сценарий использования TCP

Клиент:

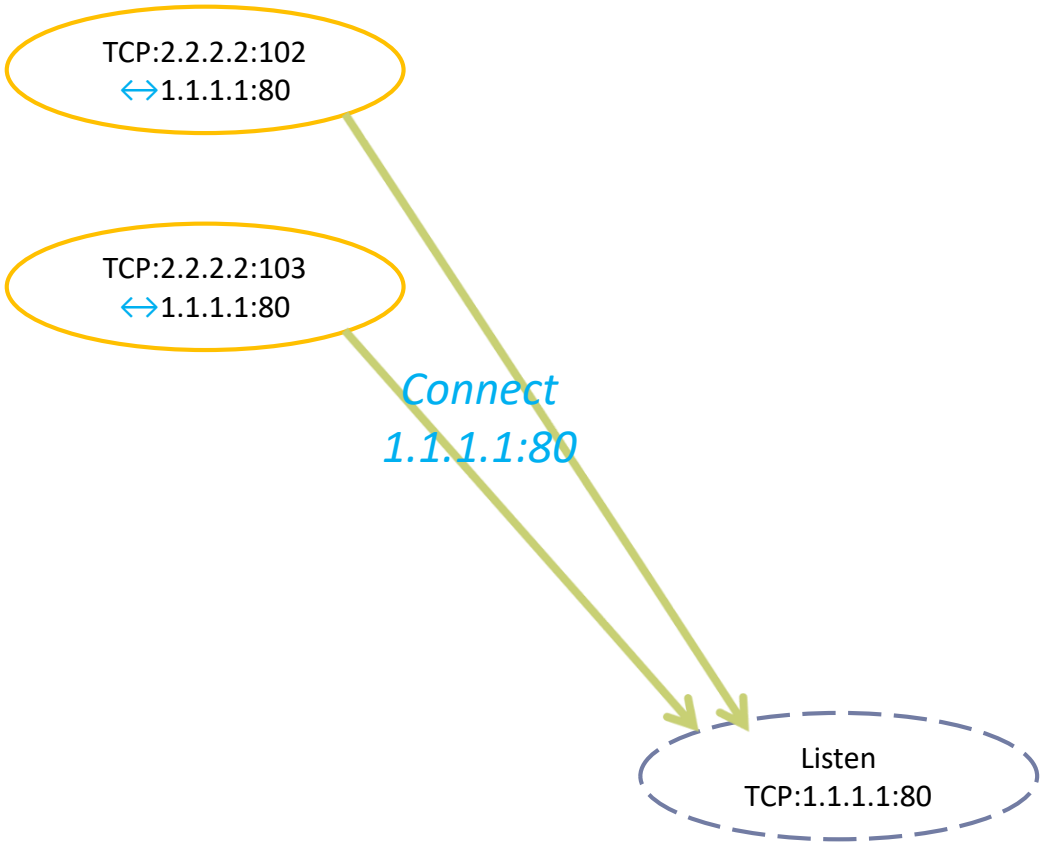
s=socket(AF_INET, SOCK_STREAM,0) – создать socket
bind(s,…) – как правило, не обязательно
connect(s, &tgtaddr, sizeof(tgtaddr)) – установить соединение с сервером, адрес которого указан в tgtaddr (IP-адрес и номер порта)
Perform_dialog(s) – начать диалог с сервером

Клиенту необходимо знать адрес сервера, с которым предстоит установить соединение: **IP-адрес и номер порта**.

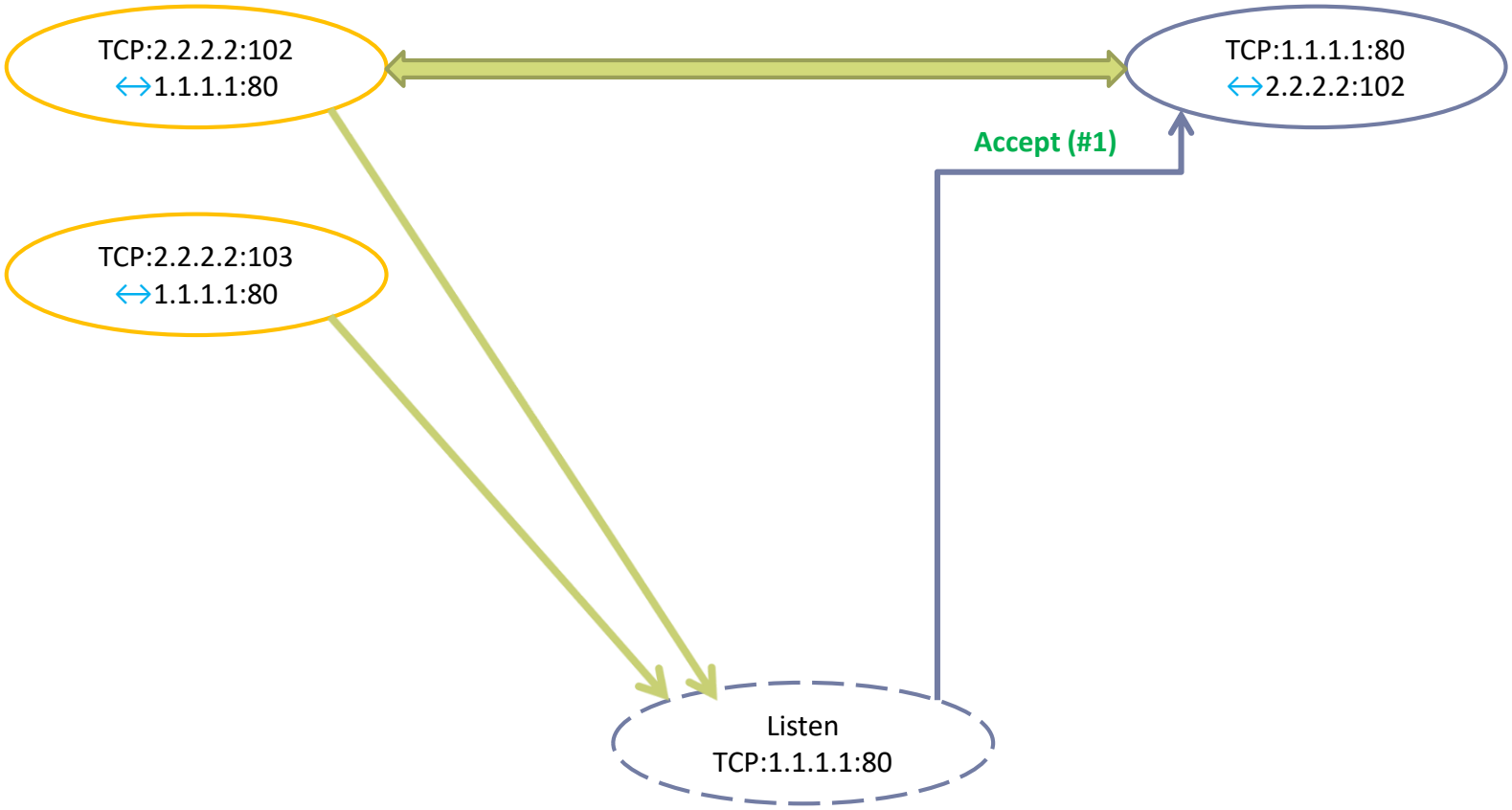
Сервер:

s=socket(AF_INET, SOCK_STREAM,0) – создать socket
bind(s,…) – назначить адрес socket-у (как правило, необходимо)
listen(s,…) – пометить socket как пассивный
while 1 – бесконечный цикл приема соединений
s1 = ***accept(s,….)*** - ожидание прихода очередного клиента
Fork_dialog(s1) - запуск диалога в параллельном режиме

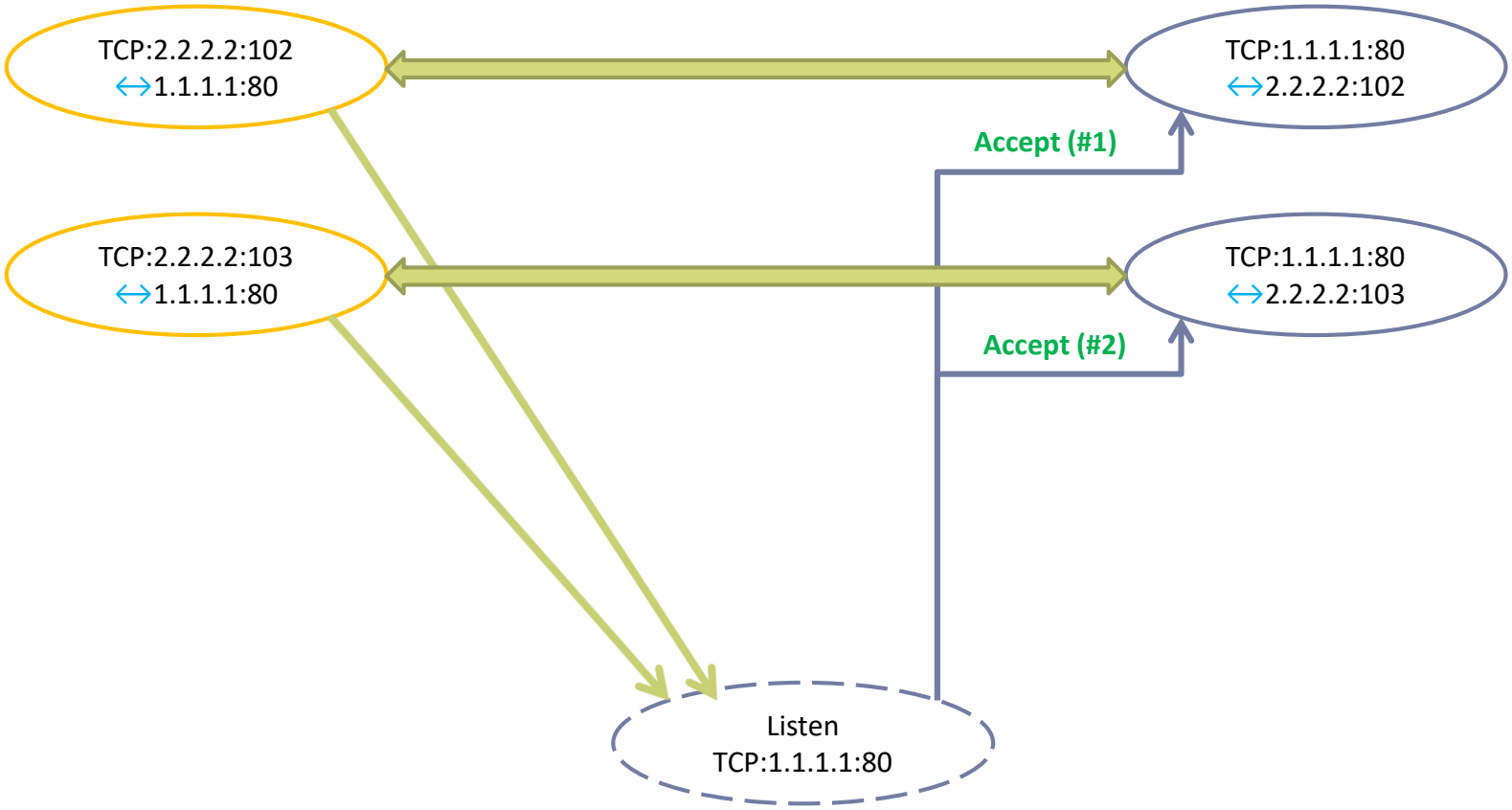
Магия функции Accept



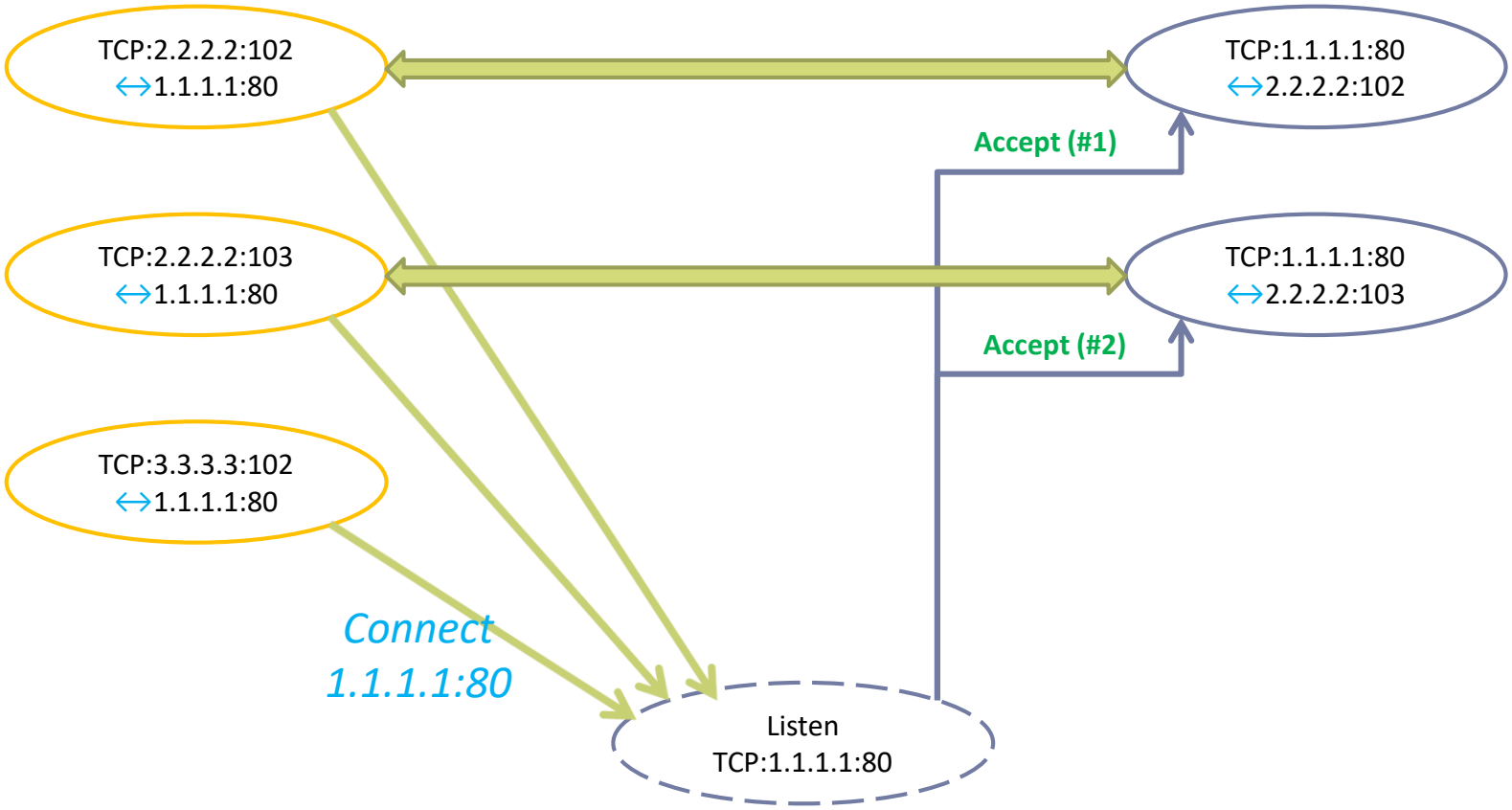
Магия функции Accept



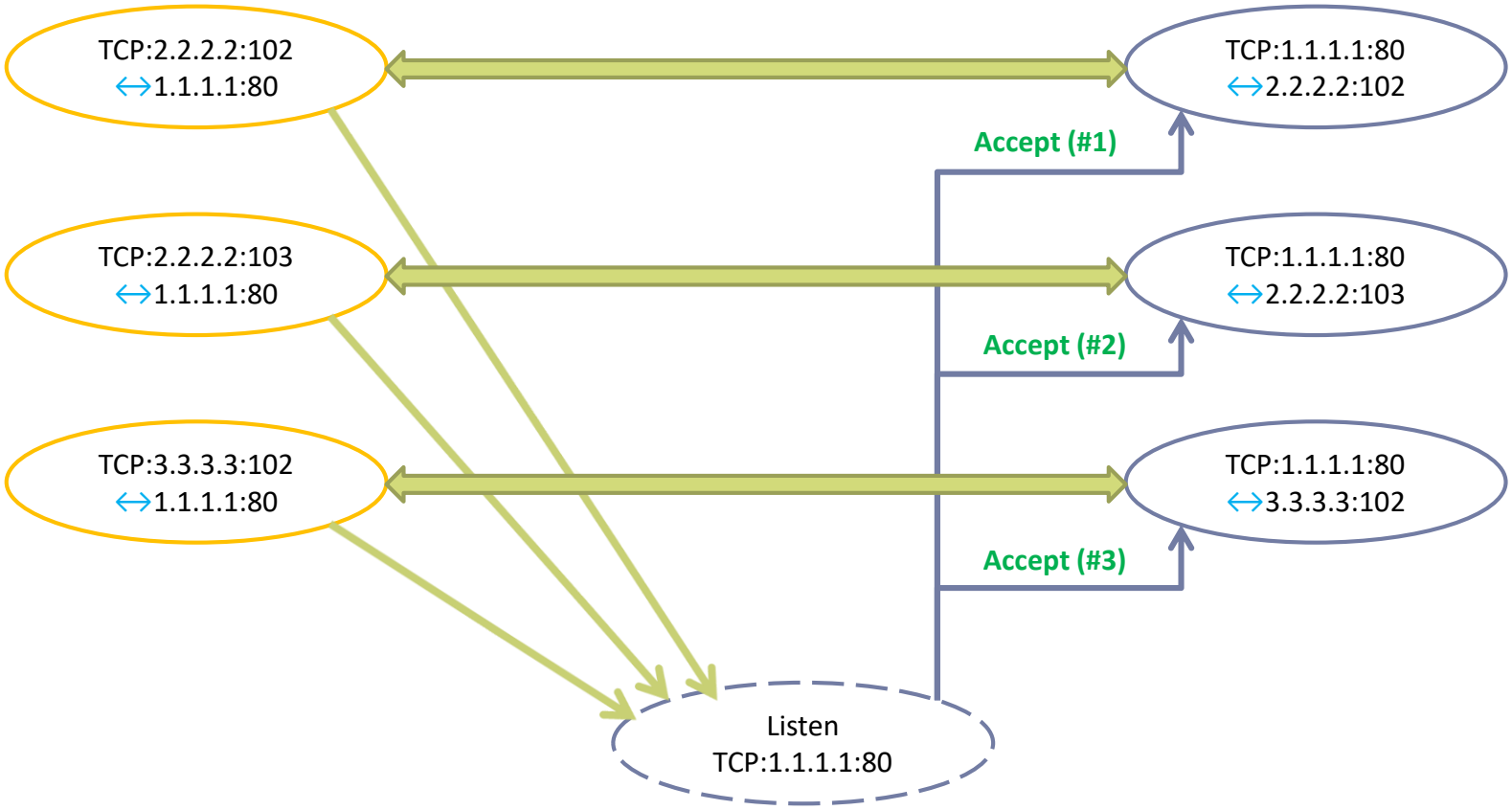
Магия функции Accept



Магия функции Accept



Магия функции Accept

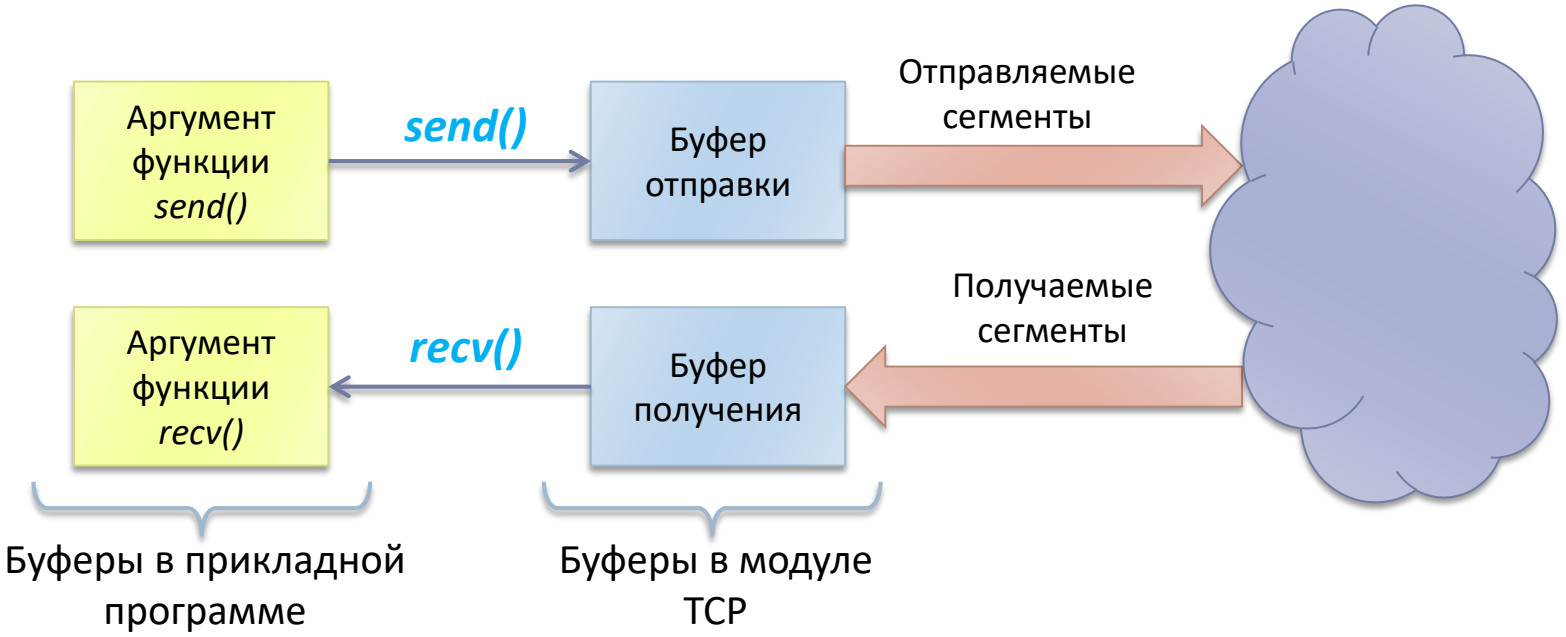


Обмен данными в TCP

Имеется несколько функций отправки и приема данных через открытый socket:

- ✓ прием данных: *recv, recvfrom, recvmsg, read*
- ✓ отправка данных: *send, sendto, sendmsg, sendfile, write*

Однако все они используют подход **двойной асинхронной буферизации**



Эффекты двойной буферизации в TCP

- ✓ Send лишь переписывает данные из программного буфера в буфер TCP, где могут накапливаться порции из нескольких последовательных Send.
- ✓ В какой момент отправлять по сети очередной сегмент данных решает TCP. Как правило, TCP пытается дождаться чтобы в буфере был накоплен полный сегмент. Отправка неполного (укороченного) сегмента может производиться только после истечения таймаута (~0,1 сек). Кроме того на точный момент отправки влияет еще масса факторов (управление потоком, восстановление ошибок, алгоритм Nagle и др.)
- ✓ Отправляемый сегмент может содержать лишь какую-либо часть порции данных Send сцепленную с порциями других Send.
- ✓ Read читает то количество байтов, которое уже находится в приемном буфере TCP, в том числе это может быть часть сегмента, которую «не дочитали» при прошлом Read.
- ✓ При отсутствии данных в буфере, функция Read либо блокируется в ожидании прихода данных, либо немедленно возвращает управление с количеством прочитанных байтов 0.
- ✓ Аналогично функция Send может блокироваться при отсутствии места в буфере TCP

Структура заголовка TCP сегмента

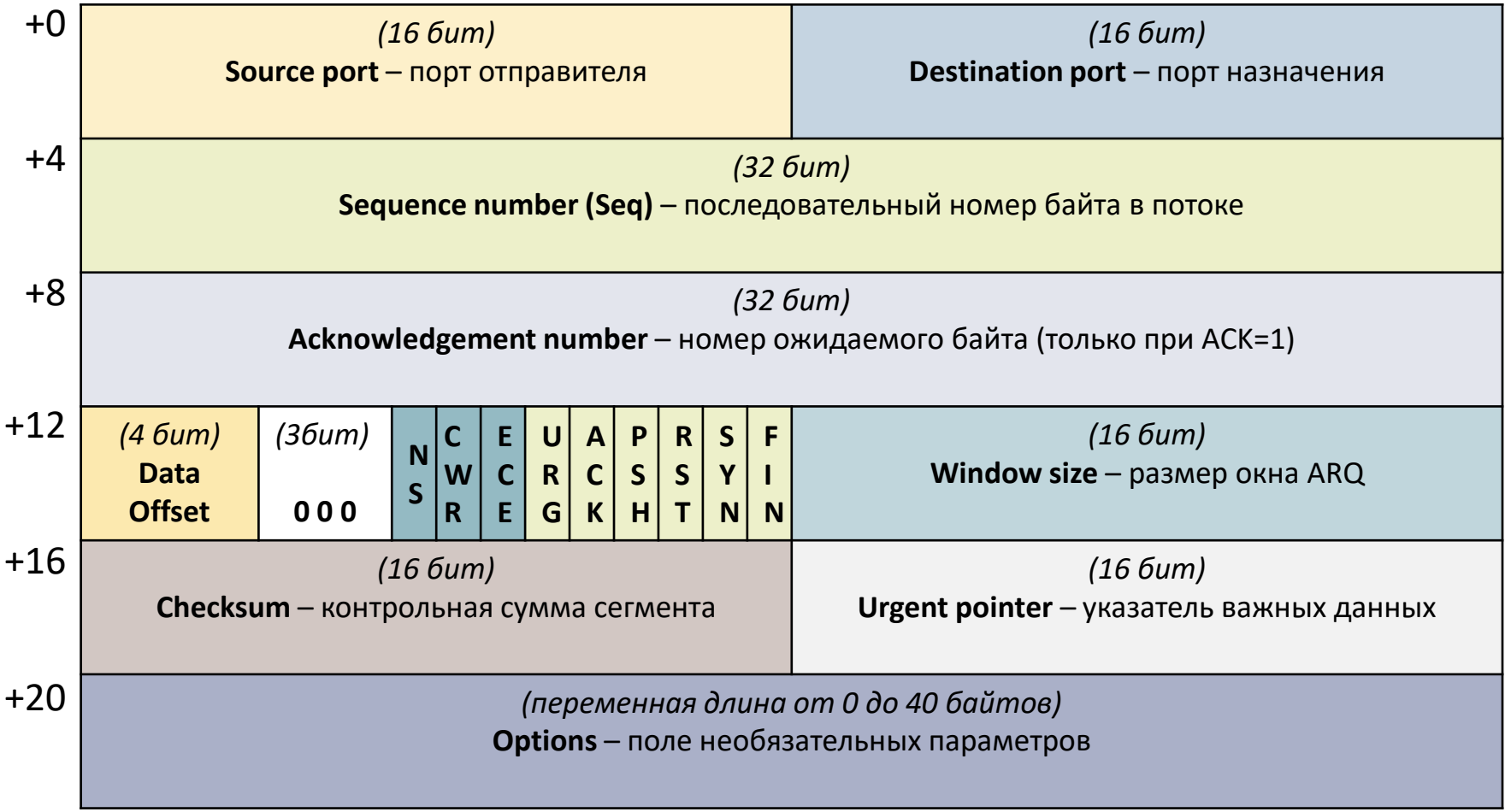
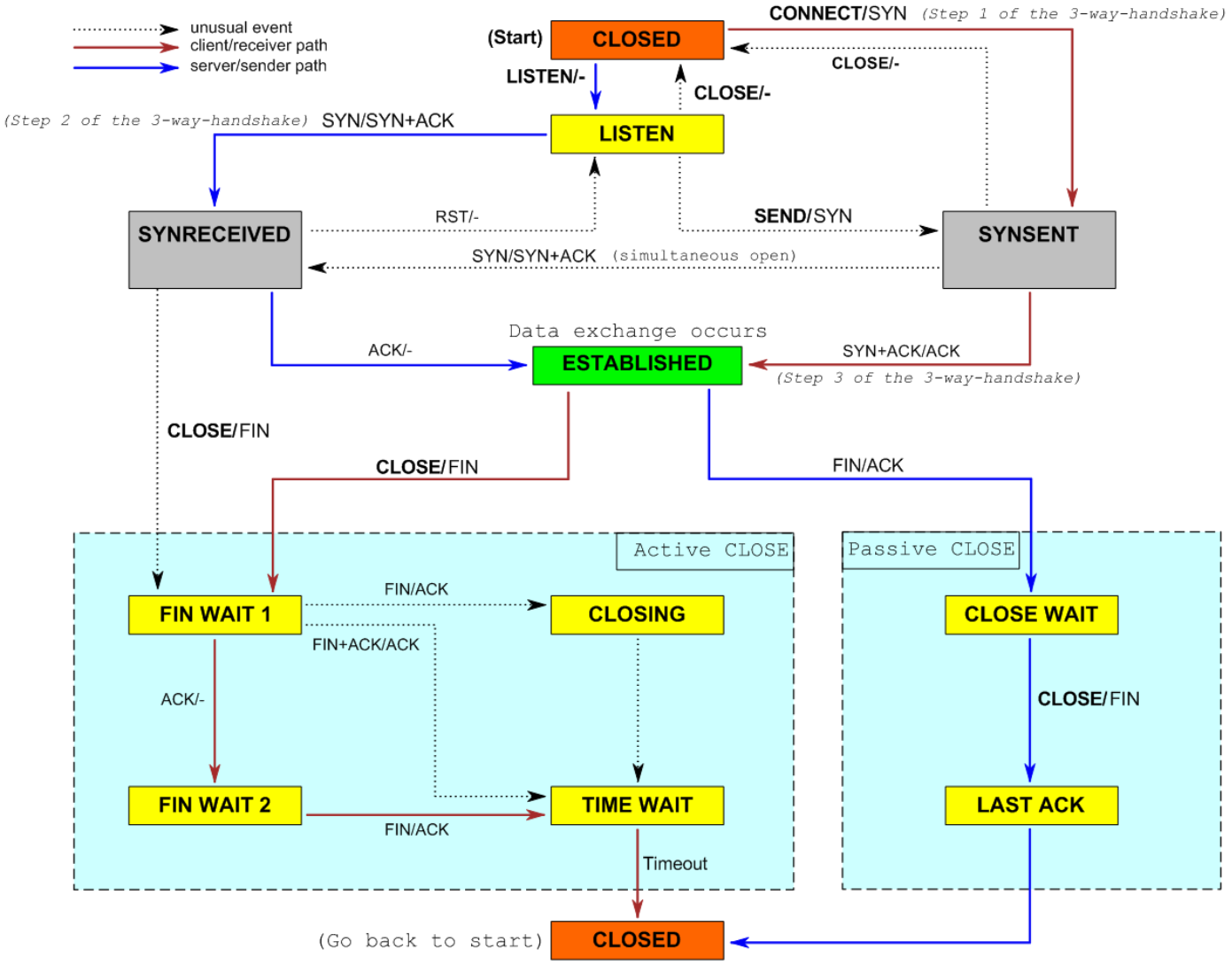
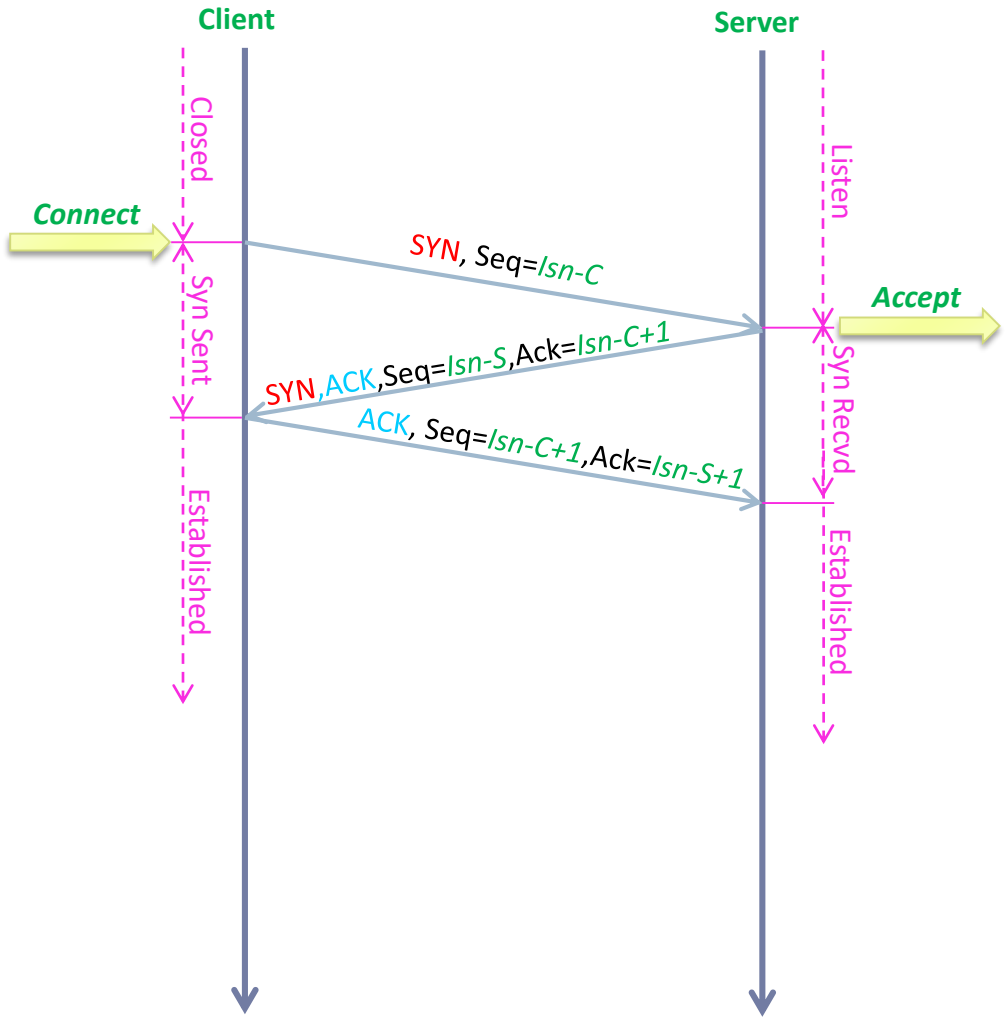


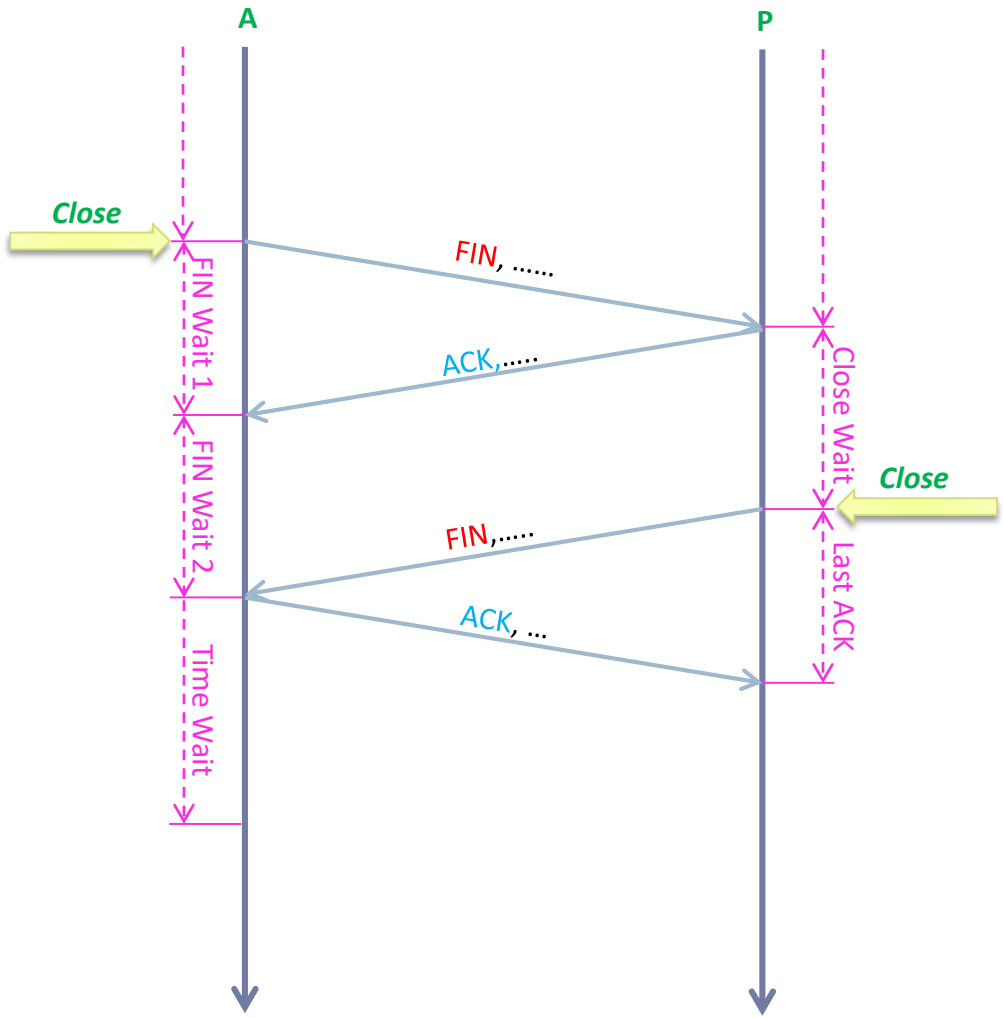
Диаграмма состояния TCP модуля



Установка TCP соединения



Завершение TCP соединения



Завершение TCP соединения