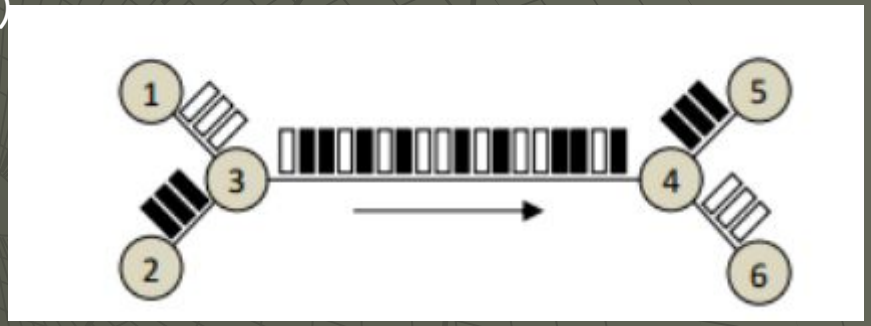
# Лекция №3. Протоколы передачи данных.

## Многоуровневая модель протоколов

1. Понятие протокола
2. Семиуровневая модель протоколов.
3. Проблемы сети, решаемые с помощью работы протоколов

# Зачем данные, передаваемые по сети, делятся на пакеты?

В современных сетях пересылаемые данные делятся на пакеты. Дело в том, что чаще всего одна линия связи используется для обмена данными между несколькими узлами. Если передавать большие файлы целиком, то получится, что сеть будет заблокирована, пока не закончится передача файла. Кроме того, в этом случае при сбое весь файл нужно передавать заново, э то увеличивает нагрузку на сеть. Если передавать отдельные пакеты, время ожидания сокращается до времени передачи одного пакета (это доли секунды), нагрузка на линию связи становится более равномерной. По сети одновременно передаются пакеты, принадлежащие нескольким файлам. На рисунке по одной линии связи (между узлами 3 и 4) одновременно выполняется передача данных от узла 2 к узлу 5 (пакеты обозначены черными прямоугольниками) и от узла 1 к узлу 6 (белые прямоугольники)




Вместе с каждым пакетом передается его контрольная сумма число, найденное по специальному алгоритму и зависящее от всех данных пакета. Узел-приемник рассчитывает контрольную сумму полученного блока данных и, если она не сходится с контрольной суммой, указанной в пакете, фиксируется ошибка и этот пакет (а не весь файл!) передается, как правило, еще раз.

# *Почему размер пакета не должен быть очень маленьким?*

- Казалось бы, чем меньше размер пакета, тем лучше. Однако это не так, потому что любой пакет кроме «полезных» данных содержит служебную информацию: адреса отправителя и получателя, контрольную сумму. Поэтому в каждом случае есть некоторый оптимальный размер пакета, который зависит от многих условий (например, от уровня помех, количества компьютеров в сети, передаваемых данных и т.д.). Чаще всего для обмена данными в локальных сетях и в Интернете используются пакеты размером не более 1,5 Кб

# Понятие протокола

- Главная цель, которая преследуется при соединении компьютеров в сеть - это возможность использования ресурсов каждого компьютера всеми пользователями сети. Для того, чтобы реализовать эту возможность, компьютеры, подсоединенные к сети, должны иметь необходимые для этого средства взаимодействия с другими компьютерами сети. Задача разделения сетевых ресурсов является сложной, она включает в себя решение множества проблем - выбор способа адресации компьютеров и согласование электрических сигналов при установление электрической связи, обеспечение надежной передачи данных и обработка сообщений об ошибках, формирование отправляемых и интерпретация полученных сообщений, а также много других не менее важных задач.

# Понятие протокола

- Основное аппаратное обеспечение связи включает устройства, позволяющие передавать данные из одной точки сети в другую. Однако применения для связи только аппаратных средств недостаточно. Все участники обмена данными должны согласовывать между собой определенный набор правил, который применяется при обмене сообщениями (например, используемый язык и правила, определяющие порядок отправки сообщений). Эти правила можно сравнить с правилами ведения международных переговоров. Дипломаты называют подобные соглашения протоколом. Этот термин применяется также к компьютерной связи, а набор правил, который определяет формат сообщений и соответствующие действия, необходимые для отправки каждого сообщения, называют **сетевым протоколом** или **протоколом компьютерной связи**. Программное обеспечение, которое реализует такие правила, называют **программным обеспечением протокола**.

*Соглашение, определяющее формат и смысл сообщений, которыми обменива ются компьютеры, называется протоколом* связи. *Прикладные программы, которые используют ресурсы сети, не взаимодействуют непосредственно с сетевым аппаратным обеспечением. Приложения взаимодействуют с программным обеспечением протокола, которое при обмене данными выполняет правила соответствующие.*

# Работа протоколов

Протоколы реализуются не только программно-аппаратными средствами компьютеров, но и коммуникационными устройствами. Действительно, в общем случае связь компьютеров в сети осуществляется не напрямую - "компьютер-компьютер", а через различные коммуникационные устройства такие, например, как концентраторы, коммутаторы или маршрутизаторы. В зависимости от типа устройства, в нем должны быть встроены средства, реализующие некоторый набор сетевых протоколов.

# Работа протоколов

- Вместо применения единственного гигантского протокола, который определял бы все операции для всех возможных форм связи, было решено разделить задачу связи на отдельные подзадачи и разработать отдельный протокол для каждой подзадачи. Это позволяет упростить проектирование, анализ, реализацию и проверку каждого протокола. Как будет показано ниже, разделение программного обеспечения связи на несколько протоколов способствует повышению гибкости, поскольку дает возможность использовать по мере необходимости отдельные протоколы.
- Разделение правил работы сети на отдельные протоколы должно быть выполнено очень тщательно, чтобы система связи была надежной и эффективной. В целях устранения дублирования каждый протокол должен решать только ту часть общей проблемы, которая не относится к сфере действия других протоколов. Для обеспечения эффективной реализации протоколы должны быть спроектированы так, чтобы они могли обмениваться структурами данных и информацией. И наконец, вся совокупность совместно действующих протоколов должна успешно действовать при возможных отказах аппаратных средств или в других исключительных случаях.

# *Наборы/семейства протоколов*

- Для обеспечения успешной совместной работы протоколы проектируются и раз рабатываются по единому плану, в виде полных, согласованных комплектов, называемых *наборами* или *семействами протоколов.* Каждый протокол в наборе решает одну часть проблемы; вместе они решают всю задачу. Более того, весь набор спроектирован с учетом обеспечения эффективного взаимодействия между протоколами.

# Стек

- **Стек** (англ. stack — стопка; читается стэк) — абстрактный тип данных, представляющий собой список элементов, организованных по принципу LIFO (англ. last in — first out, «последним пришёл — первым вышел»). Чаще всего принцип работы **стека** сравнивают со стопкой тарелок: чтобы взять вторую сверху, нужно снять верхнюю.

# Семиуровневая модель протоколов

- Из того, что протокол является соглашением, принятым двумя взаимодействующими объектами, в данном случае двумя работающими в сети компьютерами, совсем не следует, что он обязательно представляет собой стандарт.
- Международная Организация по Стандартам (International Standards Organization, ISO) разработала модель, которая четко определяет различные уровни взаимодействия систем, дает им стандартные имена и указывает, какую работу должен делать каждый уровень. Эта модель называется моделью взаимодействия открытых систем (Open System Interconnection, OSI) или моделью ISO/OSI.

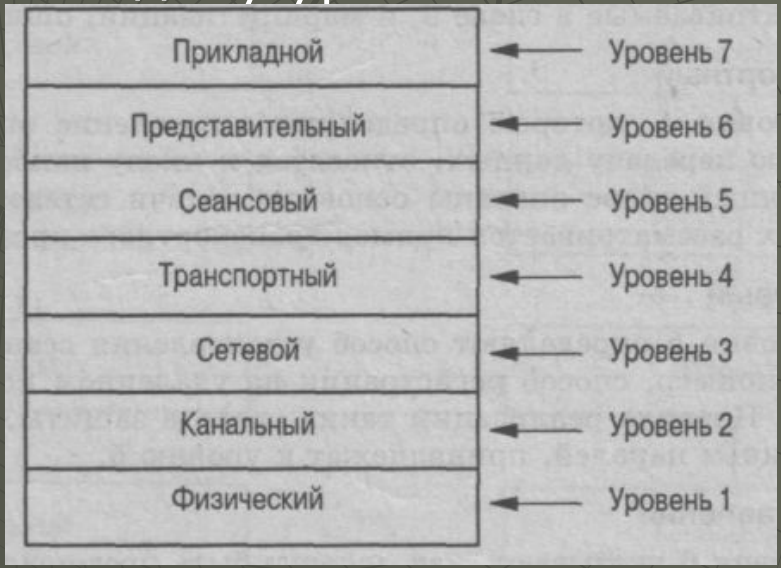
# Семиуровневая модель протоколов

- В модели OSI взаимодействие делится на семь уровней или слоев. Каждый уровень имеет дело с одним определенным аспектом взаимодействия. Таким образом, проблема взаимодействия декомпозирована на 7 частных проблем, каждая из которых может быть решена независимо от других.

# Семиуровневая модель протоколов

# Общий план проекта протокола

Разработчики протоколов должны вначале выделить отдельные подзадачи общей задачи связи и спланировать весь набор протоколов. Для этого применяется: ряд инструментальных средств. По сути многоуровневая модель описывает один из способов разделения общей задачи связи на отдельные подзадачи, называемые в этой модели *уровнями.* На ее основе можно разработать набор протоколов, определяя отдельные протоколы, которые соответствуют каждому уровню.



# Прикладной уровень

**Прикладной уровень** (уровень приложений) — верхний уровень модели, обеспечивающий взаимодействие пользовательских приложений с сетью:

- позволяет приложениям использовать сетевые службы:
  - удалённый доступ к файлам и базам данных,
  - пересылка электронной почты;
- отвечает за передачу служебной информации;
- предоставляет приложениям информацию об ошибках;
- формирует запросы к уровню представления.

# Представительский уровень

- **Представительский уровень** обеспечивает преобразование протоколов и кодирование/декодирование данных. Запросы приложений, полученные с прикладного уровня, на уровне представления преобразуются в формат для передачи по сети, а полученные из сети данные преобразуются в формат приложений. На этом уровне может осуществляться сжатие/распаковка или шифрование/дешифрование, а также перенаправление запросов другому сетевому ресурсу, если они не могут быть обработаны локально.
- Уровень представлений обычно представляет собой промежуточный протокол для преобразования информации из соседних уровней. Это позволяет осуществлять обмен между приложениями на разнородных компьютерных системах прозрачным для приложений образом. Уровень представлений обеспечивает форматирование и преобразование кода. Форматирование кода используется для того, чтобы гарантировать приложению поступление информации для обработки, которая имела бы для него смысл. При необходимости этот уровень может выполнять перевод из одного формата данных в другой.
- Уровень представлений имеет дело не только с форматами и представлением данных, Таким образом, уровень 6 обеспечивает организацию данных при их пересылке.
- Другой функцией, выполняемой на уровне представлений, является шифрование данных, которое применяется в тех случаях, когда необходимо защитить передаваемую информацию от доступа несанкционированными получателями. Чтобы решить эту задачу, процессы и коды, находящиеся на уровне представлений, должны

# Сеансовый уровень

- **Сеансовый уровень** модели обеспечивает поддержание сеанса связи, позволяя приложениям взаимодействовать между собой длительное время. Уровень управляет созданием/завершением сеанса, обменом информацией, синхронизацией задач, определением права на передачу данных и поддержанием сеанса в периоды неактивности приложений.
- Хорошим примером будут служить аудио и видеоконференции.

# Транспортный уровень

- **Транспортный уровень** модели предназначен для обеспечения надёжной передачи данных от отправителя к получателю: функции передачи данных, гарантированная доставка в пункт назначения нескольких пакетов данных в надлежащей последовательности.

# Сетевой уровень

- **Сетевой уровень** модели предназначен для определения пути передачи данных. Определение кратчайших маршрутов, коммутацию и маршрутизацию, отслеживание неполадок и «заторов» в сети.
- Протоколы сетевого уровня маршрутизируют данные от источника к получателю. Работающие на этом уровне устройства (маршрутизаторы) условно называют устройствами третьего уровня (по номеру уровня в модели OSI).

# Канальный уровень

- **Канальный уровень** предназначен для обеспечения взаимодействия сетей на физическом уровне и контроля за ошибками, которые могут возникнуть. Полученные с физического уровня данные, представленные в битах, он упаковывает в кадры, проверяет их на целостность и, если нужно, исправляет ошибки (формирует повторный запрос поврежденного кадра) и отправляет на сетевой уровень. На этом уровне работают коммутаторы, мосты и другие устройства.

# Физический уровень

**Физический уровень** — нижний уровень модели, который определяет метод передачи данных, представленных в двоичном виде, от одного устройства (компьютера) к другому. Осуществляют передачу электрических или оптических сигналов в кабель или в радиоэфир. На этом уровне также работают концентраторы, повторители сигнала.

- Функции физического уровня реализуются на всех устройствах, подключенных к сети. Физический уровень определяет такие виды сред передачи данных как оптоволокно, витая пара, коаксиальный кабель, спутниковый канал передач данных и т. п.

# Многоуровневое программное обеспечение

- Если протоколы разрабатываются в соответствии с многоуровневой моделью, то результирующее программное обеспечение протокола приобретает многоуровневую организацию. Программное обеспечение протокола на каждом компьютере подразделяется на модули, и каждый модуль соответствует определенному уровню. Многоуровневая организация определяет взаимодействие между модулями: при передаче или приеме данных программным обеспечением протокола каждый модуль взаимодействует только с модулем уровня, расположенного непосредственно над его уровнем или под его уровнем. Поэтому исходящие данные проходят через каждый уровень сверху вниз, а входящие данные — снизу вверх. Этот принцип проиллюстрирован на рис.2.

# Многоуровневое программное обеспечение

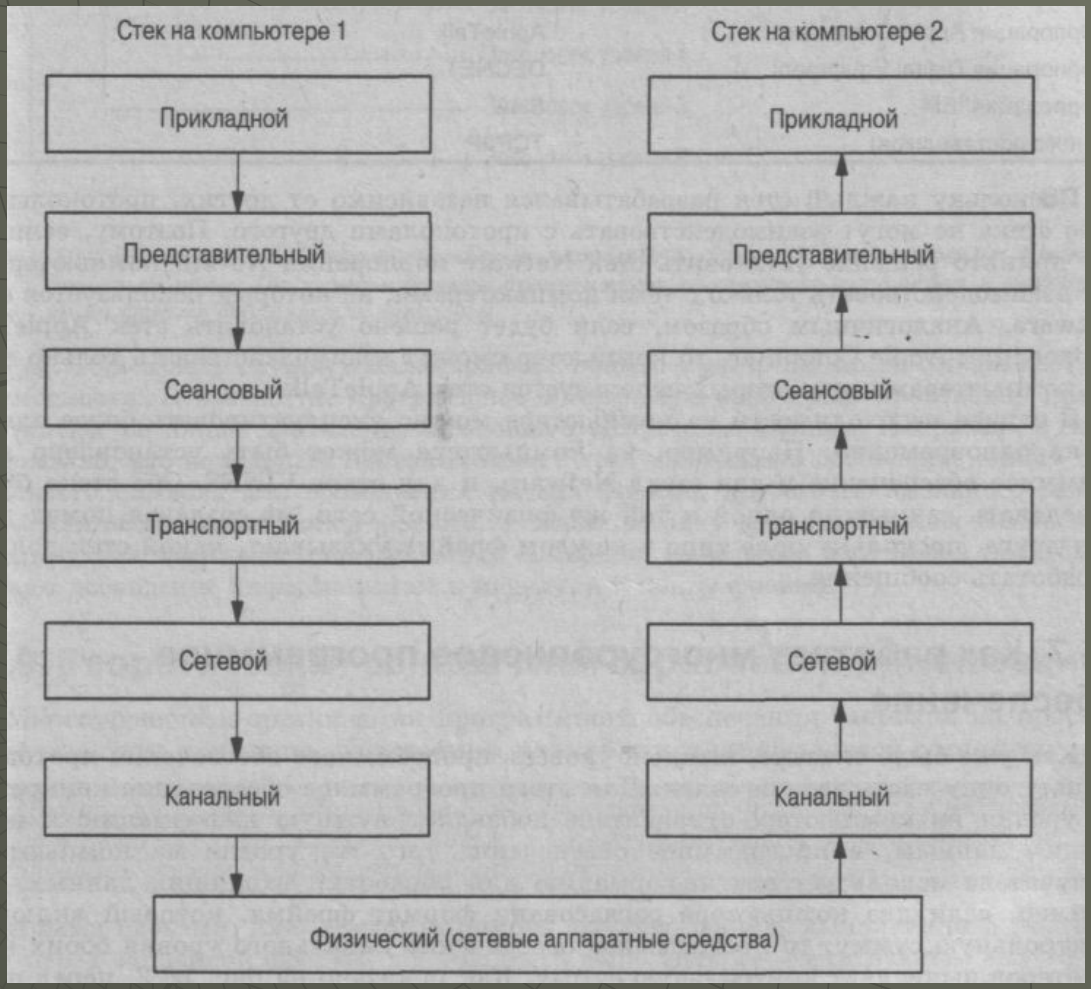



*Рис.2. Схема прохождения данных, передаваемых по сети из приложения одного компьютера в приложение другого компьютера*

# Многоуровневое программное обеспечение

- Любой протокол модели OSI должен взаимодействовать либо с протоколами своего уровня, либо с протоколами на единицу выше и/или ниже своего уровня. Взаимодействия с протоколами своего уровня называются горизонтальными, а с уровнями на единицу выше или ниже — вертикальными. Любой протокол модели OSI может выполнять только функции своего уровня и не может выполнять функций другого уровня.
- Как уже было сказано, каждый уровень программного обеспечения протокола решает одну часть задачи связи. Для этого программное обеспечение конкретного уровня на компьютере-отправителе добавляет нужную информацию к исходящим данным, а программное обеспечение того же уровня на компьютере-получателе использует эту информацию для обработки входящих данных.

# Вложенные заголовки

- Обычно каждый уровень помещает дополнительную информацию в заголовок перед передачей данных на более низкий уровень. Поэтому фрейм, проходящий по сети, содержит ряд вложенных заголовков (рис.3).

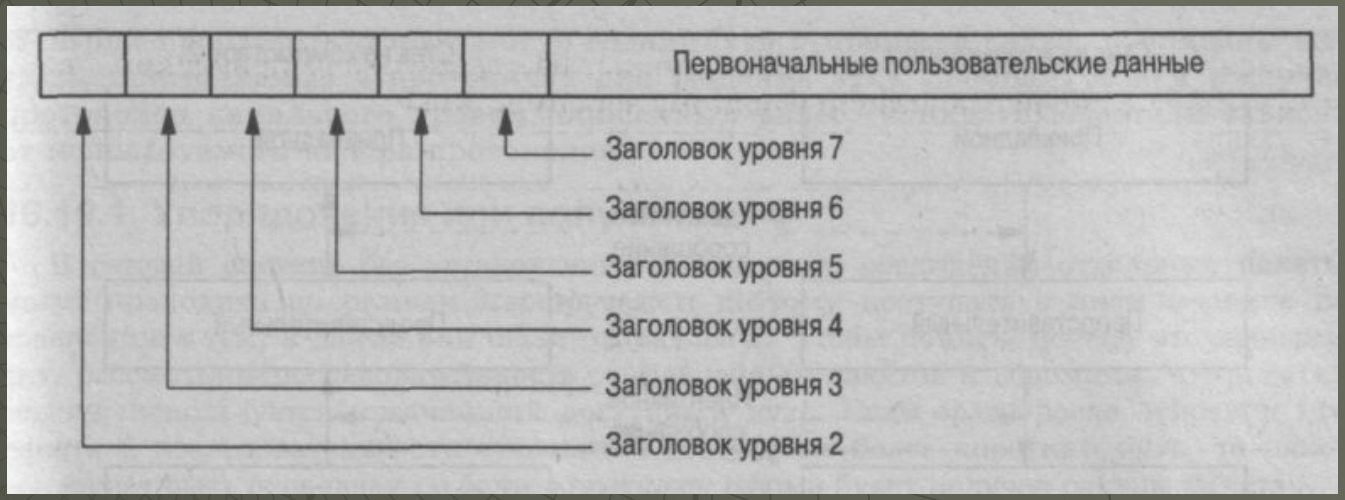



Рис. 3. Вложенные заголовки протокола, которые присутствуют во фрейме, проходящем по сети. На каждом уровне программного обеспечения протокола к исходящему фрейму добавляется заголовок

# Теоретическая основа многоуровневой организации

- Многоуровневая организация программного обеспечения основана на простом теоретическом принципе, называемом *принципом многоуровневой организации.*
- *Программное обеспечение уровня N компьютера-получателя должно принимать точно такое же сообщение, которое было передано программным обес печением уровня* N *компьютера-отправителя.*
- Иными словами, для каждого прямого преобразования, выполненного протоколом перед отправкой фрейма, должно быть выполнено соответствующее ему обратное преобразование при получении фрейма. Если конкретный уровень протокола компьютера-отправителя добавляет к фрейму заголовок, то протокол соответст вующего уровня компьютера-получателя должен удалить этот заголовок. Если протокол одного из уровней шифрует фрейм перед передачей, протокол соответст вующего уровня компьютера-получателя должен дешифровать этот фрейм.

# Теоретическая основа многоуровневой организации

- Многоуровневая организация — мощный метод, упрощающей проектирование и проверку протокола. Многоуровневая организация исключает возможность вносить в программное обеспечение одного уровня протокола такие изменения, которые не соответствуют требованиям других уровней. Поэтому программное обеспечение передачи и приема для каждого уровня можно проектировать, реализовывать и проверять независимо от других уровней. На рис. 4 показано, как принцип многоуровневой организации применяется в разных точках стека протокола

# Многоуровневая организация

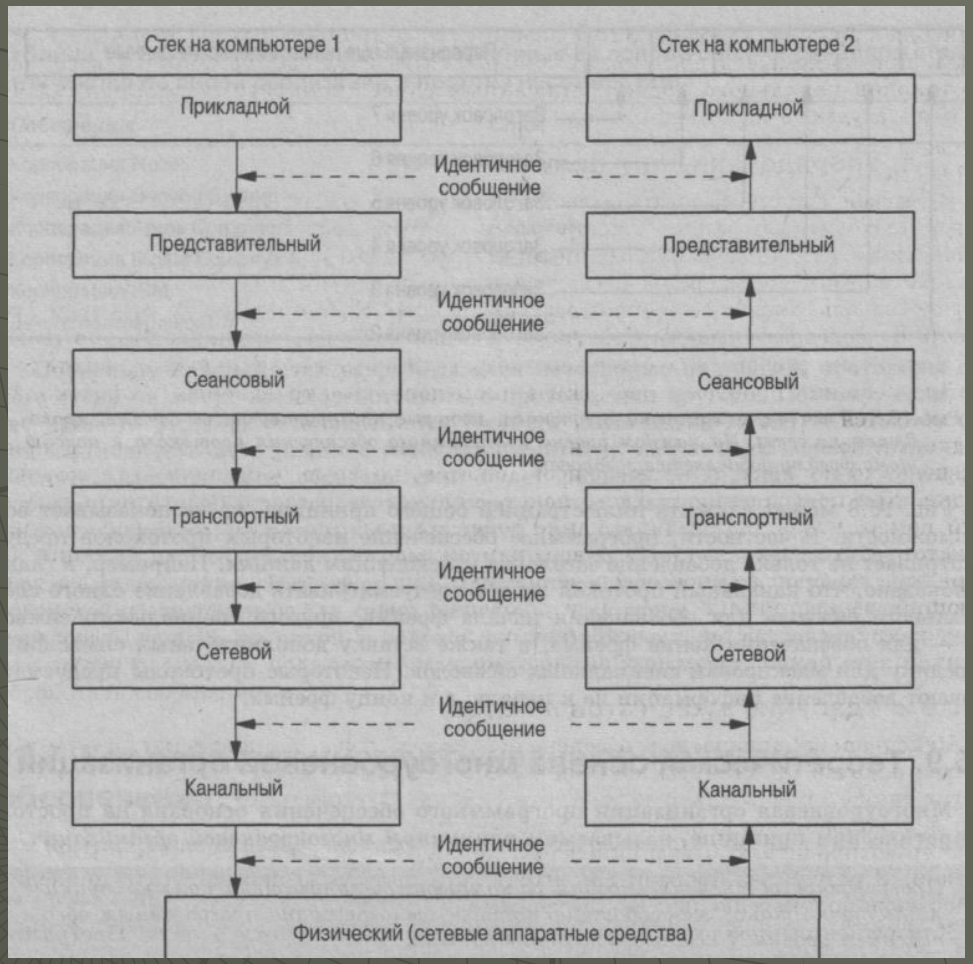



*Рис.4. Принцип многоуровневой организации, применяемый на всех уровнях модели ISO. Если программное обеспечение протокола компьютера-отправителя изменяет сообщение, то для устранения этого изменения в программном обеспечении соответствующего протокола компьютера-получателя должно быть выполнено обратное изменение*

# Проблемы сети

- При передаче пакетов данных иногда возникают проблемы (возникновение коллизий, потеря данных и т.п.) и с помощью протоколов они решаются.
- Некоторые протоколы выполняют намного больше, чем просто обнаружение ошибок: они устраняют или обходят ошибки. В частности, в транспортных протоколах используется ряд инструментальных средств по устранению некоторых из наиболее сложных проблем связи.
- Рассмотрим ряд проблем, которые могут возникнуть в процессе связи, и методы, применяемые в протоколах для решения этих проблем.

# Упорядочение при получении

- В сетевой системе без установления логического соединения отдельные пакеты могут проходить по разным маршрутам и поэтому поступать в ином порядке по сравнению с тем, в каком они были отправлены. Чтобы понять, почему это происходит, рассмотрим последовательность отправленных пакетов и вспомним, что в сетях обычно используется кратчайший доступный путь. Если сразу после отправки i-го пакета в последовательности становится доступным более короткий путь, то пакет i-1 может быть отправлен по более короткому пути и будет получен раньше пакета *i.*

# Упорядочение при получении

- Для решения проблемы доставки в ином порядке в **транспортных протоколах** применяется метод упорядочения. Отправитель добавляет к каждому пакету порядковый номер. Получатель хранит порядковый номер последнего пакета, прибывшего по порядку, а также список пакетов, которые поступили не в том по рядке. При поступлении каждого пакета получатель рассматривает его порядковый номер, чтобы определить, какие действия должны быть выполнены с этим пакетом. Если пакет является очередным ожидаемым пакетом (т.е. поступил по порядку), то программное обеспечение протокола доставляет этот пакет на следующий вышестоящий уровень и проверяет свой список, чтобы определить, можно ли также доставить недостающие пакеты. Если пакет поступил вне очереди, то программное обеспечение протокола добавляет этот пакет к списку.

# Удаление дубликатов пакетов

- Неисправное аппаратное обеспечение может вызвать появление повторяющихся пакетов. Дубликаты пакетов часто возникают в распределенных сетях, но могут также появляться и в локальных сетях. Например, неисправный трансивер в локальной сети, может сообщить компьютеру-отправителю о возникновении коллизии даже в случае нормальной передачи. В результате отправитель выполняет повторную передачу, что приводит к доставке получателю двух копий фрейма.
- Для решения проблемы дублирования применяется *упорядочение.* Программное обеспечение получателя обнаруживает дубликаты, проверяя порядковые номера поступающих пакетов. Если пакет с каким-то порядковым номером уже был успешно доставлен или какой-либо порядковый номер совпадает с номером одного из пакетов, ожидающих в очереди поступления недостающих пакетов, то программное обеспечение отбрасывает этот дубликат.

# Повторная передача потерянных пакетов

- Потеря пакетов — одна из основных проблем компьютерных сетей, поскольку ошибки при передаче могут вызвать искажение битов, в результате чего весь фрейм становится недействительным. При обнаружении получателем подобных искажений отбрасывается весь фрейм.
- Для обеспечения надежной передачи (т.е. передачи без потерь) в протоколах используется метод подтверждения с повторной передачей. Если фрейм поступает в неизменном виде, то программное обеспечение протокола получателя передает отправителю небольшое сообщение об успешном приеме. Это сообщение называется *подтверждением,* (для него используется также обозначение АСК, сокращение от acknowledgement). Отправитель отвечает за успешную доставку пакетов. После передачи пакета программное обеспечение протокола отправителя запускает таймер. Если подтверждение поступает до истечения установленного времени, то программное обеспечение отменяет отсчет по таймеру. Если же установленный тайм-аут истекает до поступления подтверждения, то программное обеспечение отправляет еще одну копию пакета и снова запускает таймер. Отправка второй копии называется *повторной передачей.*
- Повторная передача не позволяет решить проблему, если в результате отказа оборудования сетевое соединение разорвано или произошел отказ компьютера-получателя. Поэтому в протоколах, применяющих повторную отправку сообщений, обычно устанавливается максимальное число повторных передач. По достижении установленного предела программное обеспечение протокола

# Предотвращение воздействия посторонних пакетов

- Одна из причин задержки в системе коммутации пакетов вызвана применением промежуточного накопления. Пакет, поступивший в коммутатор пакетов, помещается в очередь. Если пакеты поступают быстрее, чем может перенаправить коммутатор, очередь ожидающих пакетов будет большой и задержка может оказаться чрезмерной. Чрезмерные задержки могут привести к появлению ошибок, связанных с воздействием задержавшихся в очереди пакетов.

# Например, рассмотрим такую последовательность событий

- Два компьютера согласовывают между собой сеанс связи в 13:00.
- Один компьютер отправляет последовательность из десяти пакетов на другой.
- В результате сбоя аппаратного обеспечения пакет 3 задерживается.
- Для устранения нарушения передачи данных изменяются маршруты.
- Программное обеспечение протокола компьютера-отправителя повторно передает пакет 3, и он вместе с остальными пакетами передается без ошибок.
- В 13:05 оба компьютера снова согласовывают между собой сеанс обмена данными.
- После прибытия второго пакета поступает задержанная копия пакета 3, принадлежащая к предыдущему сеансу связи.
- Поступает пакет 3, принадлежащий ко второму сеансу связи.

К сожалению, если протокол не был тщательно спроектирован, то пакет из предыдущего сеанса связи может быть принят в следующем сеансе связи, а правильный пакет отброшен как дубликат.

# Предотвращение воздействия посторонних пакетов

- Посторонние пакеты могут также появляться при передаче управляющих пакетов. Например, в протоколах часто применяется передача специальных управляющих пакетов для прекращения сеанса обмена данными. Получение копии запроса на разрыв связи из предыдущего сеанса может заставить программное обеспечение протокола преждевременно прервать текущий сеанс.
- Для предотвращения воздействия пакетов, принадлежащих к другим сеансам, в протоколах предусматривается обозначение каждого сеанса уникальным идентификатором (например, с указанием времени установления сеанса), и этим уникальным идентификатором обозначается каждый пакет. Программное обеспечение протокола отбрасывает все поступившие пакеты, которые содержат неправильный идентификатор. Идентификатор не должен использоваться повторно до истечения достаточно большого интервала времени (например, нескольких часов).

# Предотвращение переполнения данными

- Не все компьютеры работают с одинаковой скоростью. Если компьютер-отправитель передает данные по сети быстрее, чем может обработать компьютер-получатель, возникает переполнение данными, что приводит к их потере.
- Для устранения этой проблемы применяется несколько методов. Эти методы известны под общим названием *управления потоком данных.* Простейшей формой управления потоком данных является система передачи с остановками, в которой отправитель ожидает разрешения на передачу каждого пакета. Когда получатель готов к приему следующего пакета, он отправляет управляющее сообщение, обычно в форме подтверждения.
- Хотя такие протоколы передачи предотвращают переполнение данными, они могут привести к крайне неэффективному использованию пропускной способности сети.

# Предотвращение переполнения данными

Рассмотрим сеть, которая предназначена для передачи пакетов размером 1000 октетов, имеет пропускную способность 2 Мбит/с и характеризуется задержкой 50 миллисекунд. Сетевое аппаратное обеспечение может передавать данные с одного компьютера на другой со скоростью 2 Мбит/с. Однако после передачи пакета отправитель должен ждать 100 мс перед отправкой следующего пакета (так как 50 мс требуется для передачи пакета получателю и 50 мс для передачи подтверждения отправителю). Поэтому максимальная частота, с которой могут передаваться данные при использовании метода передачи с остановками, не превышает одного пакета через каждые 100 миллисекунд. После пересчета скорости передачи в битах оказывается, что максимальная скорость, которая может быть достигнута при передаче с остановками, составляет 80000 бит в секунду, что составляет всего 4% пропускной способности аппаратных средств.

# Предотвращение переполнения данными

- Для достижения более высокой производительности передачи данных в про токолах применяется метод управления потоком данных, называемый *скользящим окном.* Отправитель и получатель запрограммированы на использование *окна постоянного размера.* Так называется максимальный объем данных, который может быть передан до получения подтверждения. Например, отправитель и получатель могут согласовать между собой размер окна, равный четырем пакетам. Отправитель приступает к отправке данных, выбирает из памяти данные для заполнения первого окна и передает фреймы. Если нужна дополнительная надежность, отправитель сохраняет копии фреймов на тот случай, если потребуется повторная передача. Получатель должен иметь наготове буферное пространство для получения всего окна. Если пакет поступает по порядку, то компьютер-получатель передает пакет приложению-получателю и отправляет отправителю подтверждение. Получив подтверждение, отправитель отбрасывает копию подтвержденного пакета и передает следующий пакет. На рис. 5 показано, почему такой механизм называется

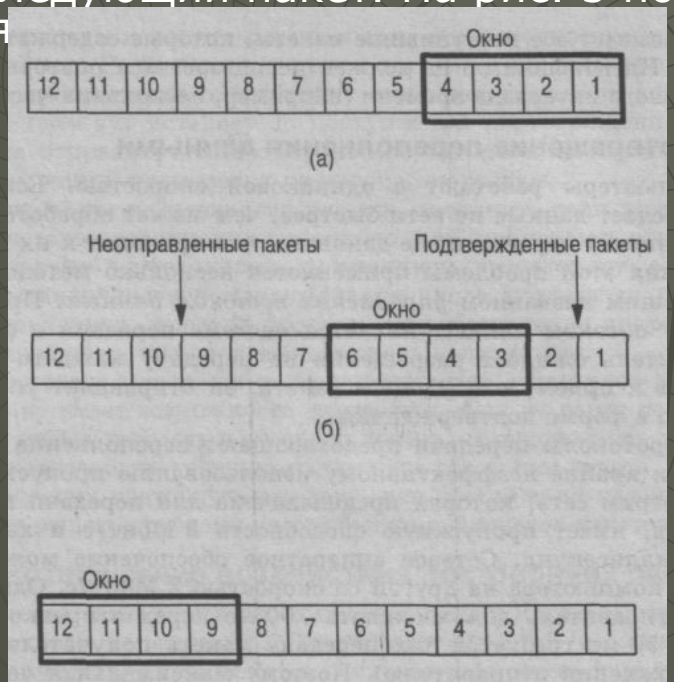



*Рис.5. Окно на 4 пакета, которое скользит по исходящим данным. Окно показано (а) в начале передачи, (б) после подтверждения получения двух пакетов и (в) после подтверждения получения восьми пакетов. Отправитель может передать сразу все пакеты, над которыми находится окно*

# Предотвращение переполнения данными

- На рис.6а показана последовательность операций передачи при использовании протокола передачи с остановками. Отправив пакет, программное обеспечение протокола ожидает подтверждения перед отправкой другого пакета. Если задержка, связанная с прохождением одного пакета по сети, равна *N,* то общее время, необходимое для передачи четырех пакетов, составляет *8N*.

- На рис.6б показана последовательность операций передачи при использовании скользящего окна. Этот протокол предусматривает отправку всех пакетов окна до перехода в состояние ожидания. На рисунке показана небольшая за держка между передачами каждого следующего пакета, как бывает в реальной ситуации. Хотя эта задержка может быть меньше, чем показано, передача всех пакетов никогда не происходит одновременно, поскольку для того, чтобы аппа ратное обеспечение завершило передачу пакета, прервало работу процессора и приступило к передаче следующего пакета, всегда требуется какое-то время (обычно несколько микросекунд). Поэтому общее время, необходимое для передачи четырех пакетов, составляет 2N+£, где £ обозначает небольшую задержку.

# Предотвращение переполнения данными

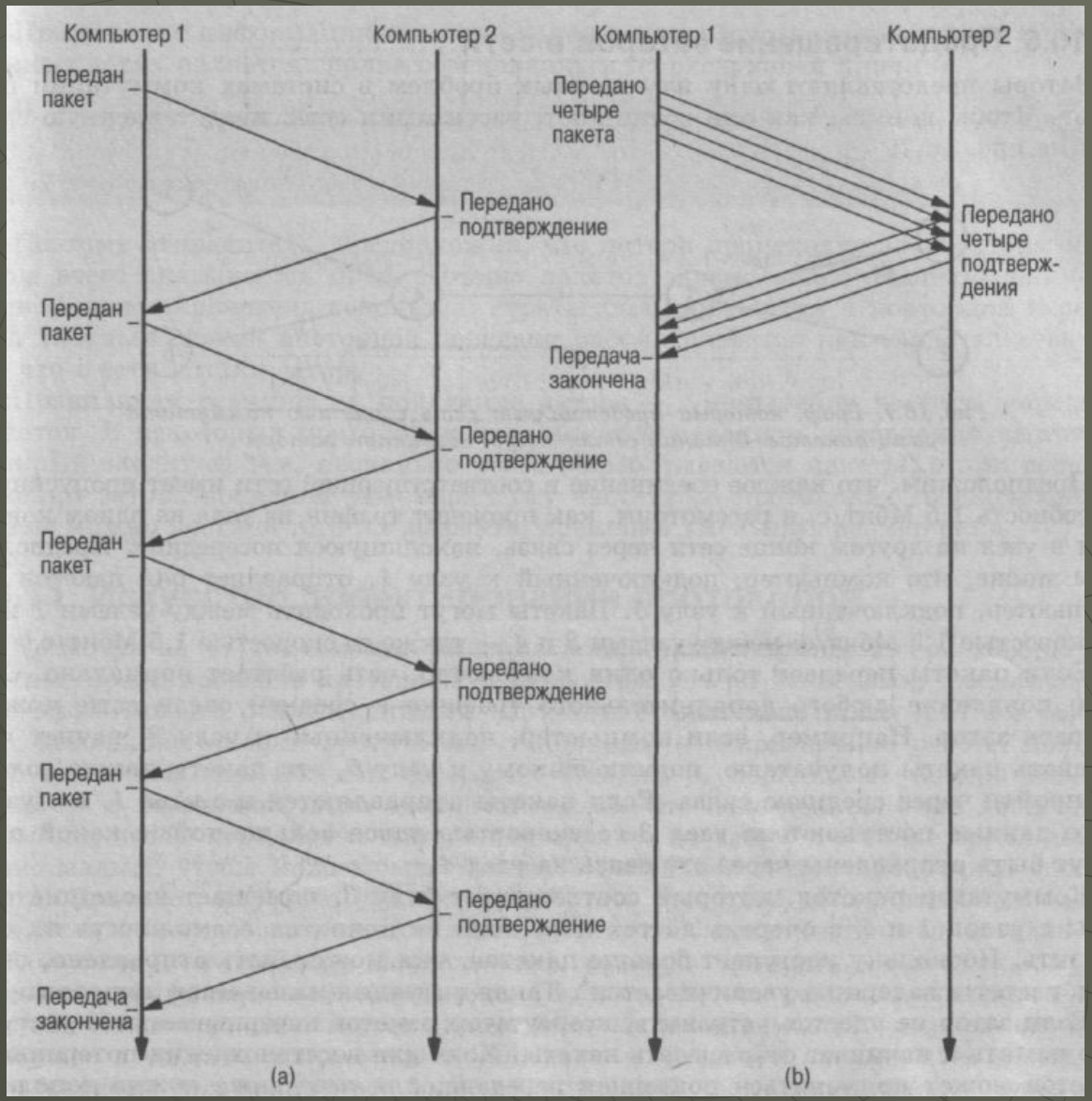



*Рис.6. Сообщения, необходимые для передачи последовательности из четырех пакетов с использованием (а) управления потоком данных с помощью метода передачи с остановками и (б) с помощью скользящего окна на 4 пакета. Ось времени направлена сверху вниз, а стрелка обозначает сообщение, переданное с одного компьютера на другой*

- Чтобы понять эффективность применения скользящего окна, представьте себе продолжительный сеанс связи, в котором должно быть передано много пакетов. В подобных случаях общее время передачи настолько велико, что задержку *£* можно проигнорировать. Чтобы оценить преимущества скользящего окна, рассмотрим сеть с высокой пропускной способностью и большой задержкой (например, спутниковый канал). В такой сети, применение протокола со скользящим окном позволяет увеличить производительность на коэффициент, который намного превышает 1. Потенциальное повышение производительности можно представить формулой:

$$Tw = Tg * W$$

Здесь Tw — производительность, которая может быть достигнута при использовании протокола со скользящим окном, *Tg* — производительность, которая может быть достигнута с использованием протокола передачи с остановками, а *W* – размер окна. Приведенное выше уравнение позволяет понять, почему протокол со скользящим окном обеспечивает примерно в четыре раза большую производительность по сравнению с протоколом передачи с остановками. Безусловно, производительность нельзя повышать беспредельно просто путем увеличения размера окна. Верхнюю границу налагает пропускная способность сети: биты не могут передаваться быстрее, чем позволяет соответствующее аппаратное обеспечение. Поэтому с учетом этого уравнение должно быть записано следующим образом:

$$Tw = \min (B, Tg * W)$$

Здесь B — пропускная способность используемого оборудования.

# Предотвращение заторов в сети

- Заторы представляют одну из основных проблем в системах коммутации па кетов. Чтобы понять, как они возникают, рассмотрим сеть, представленную графом на рис.7.

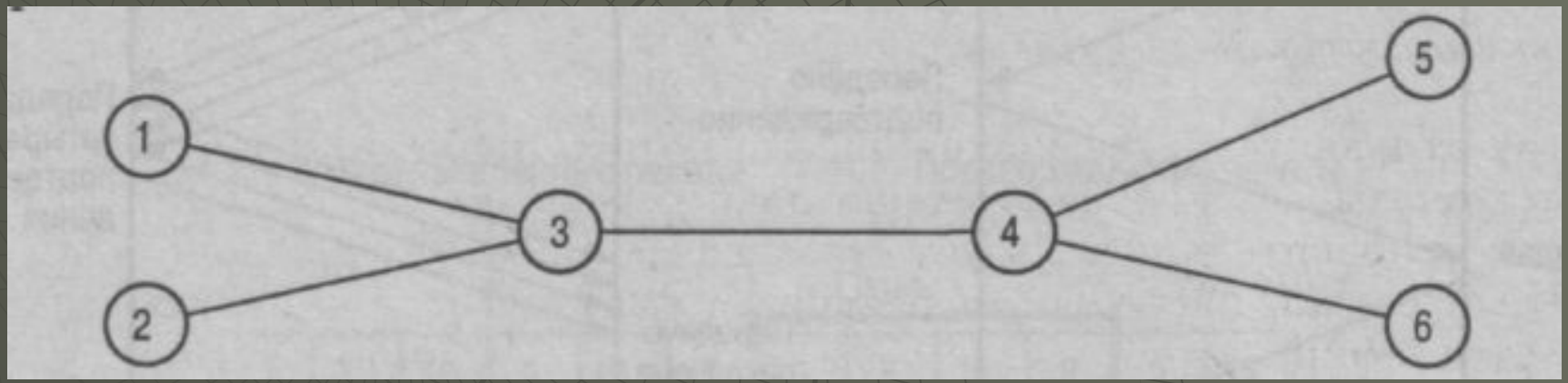
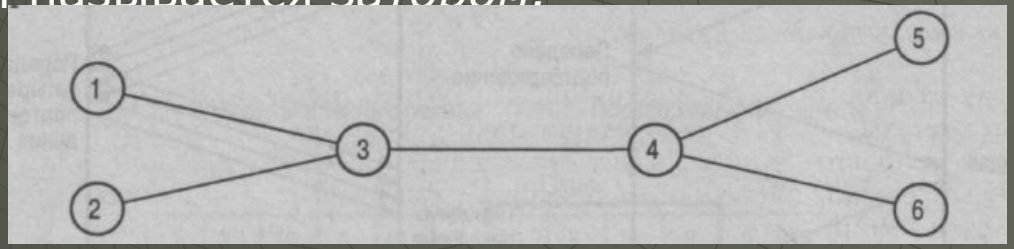



*Рис.7. Граф, который представляет сеть с шестью коммутаторами пакетов. В таких сетях могут возникать заторы*

# Предотвращение заторов в сети

- Предположим, что каждое соединение в соответствующей сети имеет пропускную способность 1,5 Мбит/с, и рассмотрим, как проходит трафик из узла на одном конце сети в узел на другом конце сети через связь, находящуюся посередине. Предположим также, что компьютер, подключенный к узлу *1,* отправляет ряд пакетов на компьютер, подключенный к узлу 5. Пакеты могут проходить между узлами *1* и 3 со скоростью 1,5 Мбит/с; между узлами *3* и *4* — также со скоростью 1,5 Мбит/с.
- Если пакеты передает только один компьютер, сеть работает нормально. Однако появление любого дополнительного трафика в средней связи сети может вызвать затор. Например, если компьютер, подключенный к узлу *2,* начнет передавать пакеты получателю, подключенному к узлу *6,* эти пакеты также должны пройти через среднюю связь. Если пакеты отправляются и с узла *7*, и с узла *2,* то данные поступают на узел *3* со скоростью вдвое больше той, с какой они могут быть отправлены через эту связь на узел *4.*
- Коммутатор пакетов, который соответствует узлу 3, помещает входящие пакеты с узлов *1* и 2 в очередь до тех пор, пока не появится возможность их отправить. Поскольку поступает больше пакетов, чем может быть отправлено, очередь растет и задержка увеличивается. Такая ситуация называется *затором.*



# Предотвращение заторов в сети

- Если затор не удается устранить, коммутатор пакетов исчерпывает всю доступную память и начинает отбрасывать пакеты. Хотя для восстановления потерянных пакетов может применяться повторная передача, для нее также нужно дополнительное время. Более того, если такая ситуация сохраняется, вся сеть может стать неработоспособной. Это называется *выходом сети из строя в связи с затором.* В протоколах предусматривается наблюдение за сетью и быстрое реагирование на первые признаки затора. Для этого могут применяется передача коммутатором предупреждений отправителю пакетов при возникновении затора.
- Схема реализуется либо путем применения в коммутаторах специального сообщения, которое передается отправителю при возникновении затора.

# Предотвращение заторов в сети

- Применение информации о потере пакетов для распознавания затора в современных сетях является вполне обоснованным по следующей причине.

*Аппаратное обеспечение современных сетей функционирует, как правило, бесперебойно; потеря пакетов чаще всего возникает в результате затора, а не сбоя аппаратного обеспечения.*

- Поэтому отправитель, предположив, что потери происходят из-за затора сети, чаще всего оказывается прав. Потерю пакетов можно легко оценить количественно, если отправитель использует стратегию с тайм-аутом и повторной передачей. Каждый случай повторной передачи рассматривается как свидетельство того, что в сети возник затор.
- Правильная реакция на появление затора — уменьшение частоты передачи пакетов. В некоторых протоколах используется механизм управления частотой, который следит за тем, насколько часто вырабатываются пакеты, а при возникновении затора временно уменьшает эту частоту. В протоколах со скользящим окном можно достичь того же эффекта, временно уменьшив размер окна.

# Выводы

- Для того, чтобы люди могли полноценно общаться, нужно, чтобы они говорили на одном языке. Эта аналогия действует и для компьютерных систем, где вместо слова «язык» используется понятие протокол. Протокол – это набор правил и соглашений, определяющих порядок обмена данными.

- Кроме аппаратного обеспечения, в сетевых системах используется сложное программное обеспечение, которое управляет обменом данными. Прикладные программы и пользователи чаще всего обращаются в своей работе не к сетевому аппаратному обеспечению, а к программному обеспечению протокола.

# Выводы

- Многоуровневая организация — основное средство, позволяющее проектировщикам успешно преодолеть сложности, связанные с разработкой протокола. Многоуровневая организация предусматривает разделение сложной задачи связи на отдельные подзадачи и позволяет проектировщику решать каждую подзадачу отдельно. В лекции описана семиуровневая справочная модель.

- Основой разработки многоуровневых проектов является теоретический принцип, известный под названием *принципа многоуровневой организации.* Принцип многоуровневой организации гласит, что на *N-ом* уровне программного обеспечения получателя должно применяться преобразование, обратное преобразованию, которое применялось на *N-ом*

# Выводы

- Протокол соответствует многоуровневой модели, на основе которой он был спроектирован. Каждому уровню соответствует программный модуль; совокупность модулей называют *стеком.* Исходящие данные проходят сверху вниз по уровням стека на компьютере-отправителе и снизу вверх по уровням стека на компьютере-получателе.

- В протоколах используется несколько основных методов решения проблем связи: для восстановления исходного порядка передачи пакетов и устранения дубликатов применяется упорядочение; для предотвращения потери пакетов используется подтверждение и повторная передача; для предотвращения воздействия посторонних пакетов применяются уникальные идентификаторы сеансов; для управления потоком данных применяется передача с остановками или механизм скользящего окна, а для предотвращения заторов в сети уменьшается частота выработки пакетов. Основным преимуществом использования скользящего окна является повышение производительности: механизм скользящего окна, исключающий передачу отправителем объема данных, который не может быть своевременно обработан получателем, можно настроить так, что будет использоваться вся доступная пропускная