Принципы работы сетей передачи данных

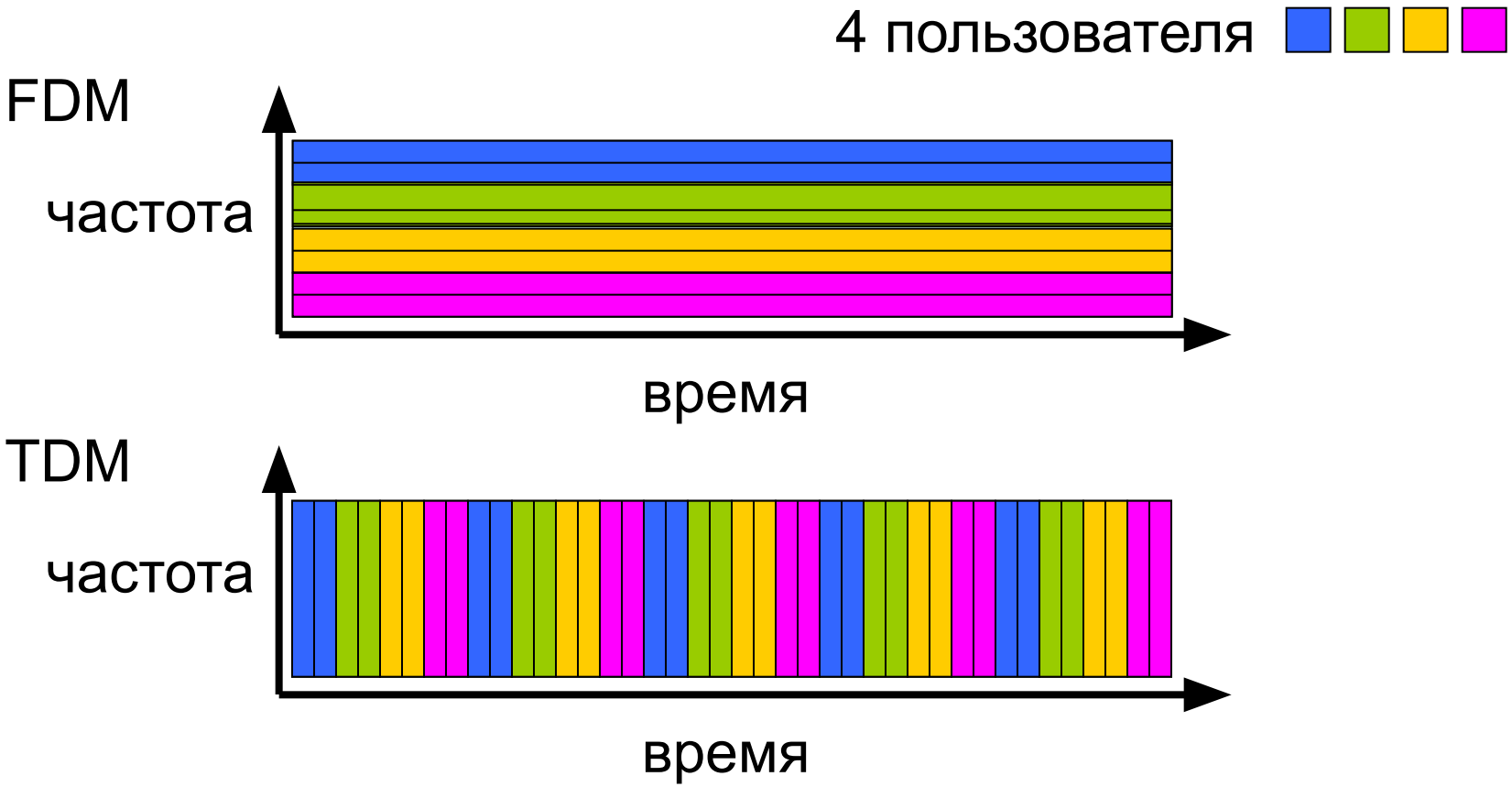
использованы материалы из
курса “Data Networks” 6.423
курса“Principles of Digital Communication” 6.450
и “Computer Networks” курса 6.829
Open Courseware, MIT

Методы коммутации

- **Коммутация линий связи**
  - **Выделенные ресурсы**
- **Коммутация пакетов**
  - **Ресурсы в общем пользовании**
  - **Виртуальные каналы**
  - **Датаграммы**

Сквозной тракт

- Передатчик – среда – приемник
- Среда: кабели, витая пара, оптоволокно, эфир
- Трансиверы подключаются к среде через разъемы или антенны
- Мультиплексирование: использование одного тракта для нескольких одновременных сеансов связи



Коммутация линий связи

- Каждой сессии выделяется фиксированная часть емкости каждого линка на всем его протяжении
  - Выделенные ресурсы
  - Фиксированный маршрут (path)
  - Если вся емкость израсходована, звонки блокируются, как в телефонной сети
- Преимущества коммутации линий
  - Фиксированные задержки
  - Гарантированная непрерывная доставка сигналов, сообщений
- Недостатки
  - Каналы не используются, когда сессия «молчит»
  - Неэффективно для неравномерного («взрывного») трафика
  - Работает обычно с фиксированной скоростью передачи (напр., 64 Kbps)
    - Трудно обспечить поддержку переменной скорости передачи

Проблемы использования коммутации линий для передачи данных

- При передаче данных **duty factor** обычно низок **(bursty** трафик**),**
  - **(message transmission time)/(message interarrival time) << 1**
  - Или**: (message arrival rate) * (message transmission time) << 1**
- Скорость передачи данных (rate), выделенная сессии, должна быть достаточно высокой для удовлетворения требований по задержке. Линия не используется, когда сессия «молчит»

- Если линия связи дорогая, коммутация линий связи неэкономична из-за необходимости удовлетворения требований по задержке или в связи со «взрывным» трафиком

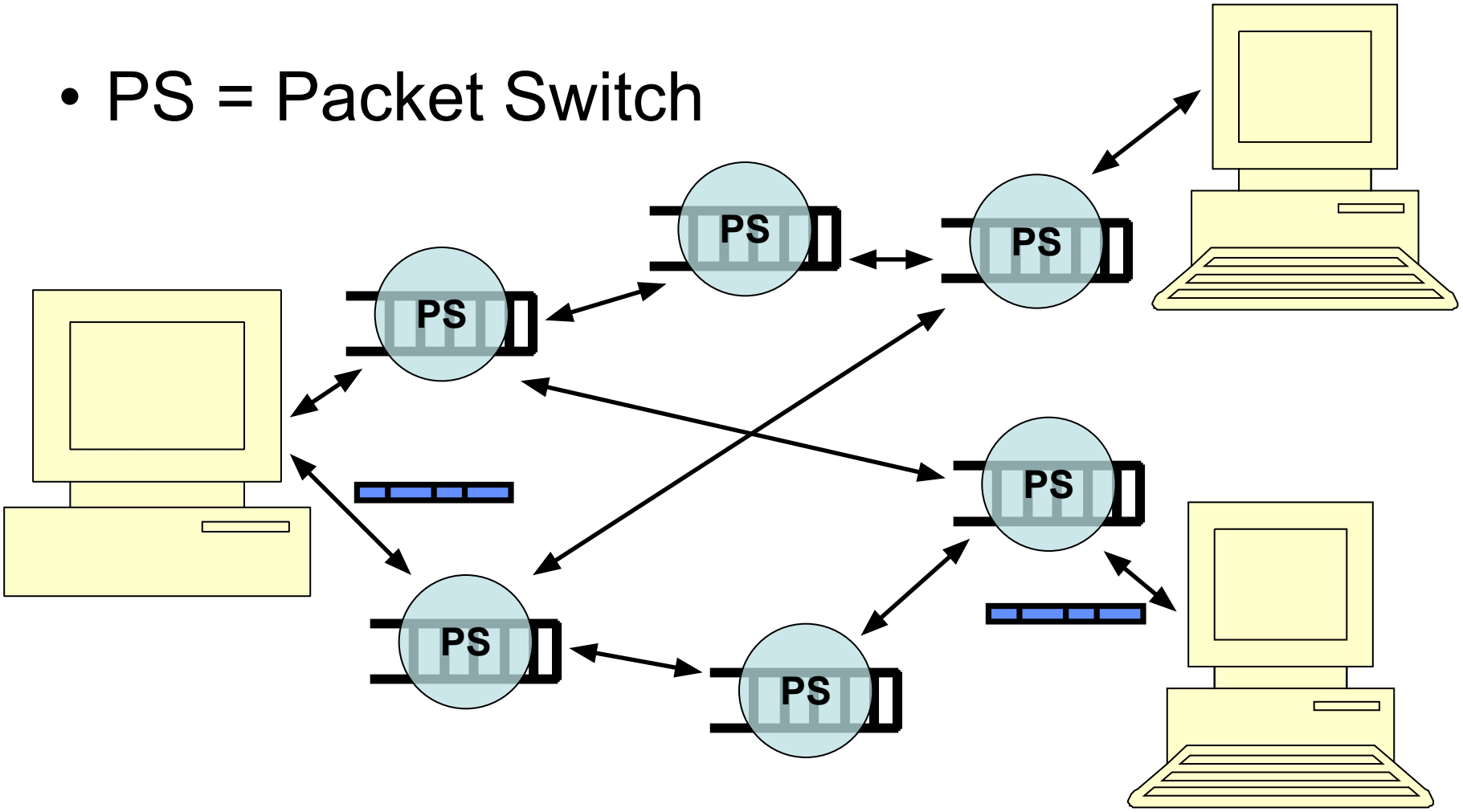
- Техника коммутации линий связи к тому же требует настройки канала для звонка (call set-up time), во время настройки ресурсы простаивают. Если сообщения много короче времени настройки канала для звонка, то коммутация линий связи опять же неэкономична (или даже неосуществима практически)
  - **Большая проблема в высокоскоростных сетях**

Пример системы с коммутацией линий связи

- **L = длина сообщения**
- **λ = скорость прихода сообщений (rate)**
- **R = скорость канала (channel rate) в битах/сек**
- **X = задержка передачи сообщения = L/R**
  - **R должна быть достаточно большой (X мала)**
  - **Bursty traffic => λX << 1 => низкий кпд**
- **Пример**
  - **L = 1000 байт (8000 бит)**
  - **λ = 1 сообщение/сек**
  - **X < 0.1 сек (требование к задержке) => R > 8000/0.1 = 80,000 bps**
- **КПД = 8000/80000 = 10%**
- **Выход – коммутация пакетов**

Сети с коммутацией пакетов

- PS = Packet Switch



Коммутация пакетов

- Коммутация пакетов с использованием **датаграмм**
  - Маршрут выбирается для каждого пакета индивидуально
  - Разные пакеты могут пойти по разным маршрутам
  - Пакеты могут приходить в пункт назначения не по порядку
  - Пример: **IP (The Internet Protocol)**
- Коммутация пакетов с созданием виртуальных каналов (**Virtual Circuit**)
  - Все пакеты в данной сессии идут по одному пути
  - Маршрут выбирается в начале сессии
  - Пакеты помечаются номером виртуального канала **(VC#)**, обозначающим маршрут
  - Номер **VC** должен быть уникальным для данного линка, но может меняться при переходе с линка на линк
- **Пусть мы настраиваем соединения в сетке (mesh) с тысячью узлов (в принципе, для каждой пары узлов). Требование уникальности VC-номеров приводит к необходимости генерировать и хранить до миллиона VC-номеров в каждом узле**
  - Пример: **ATM (Asynchronous transfer mode)**

Сравнение методов коммутации

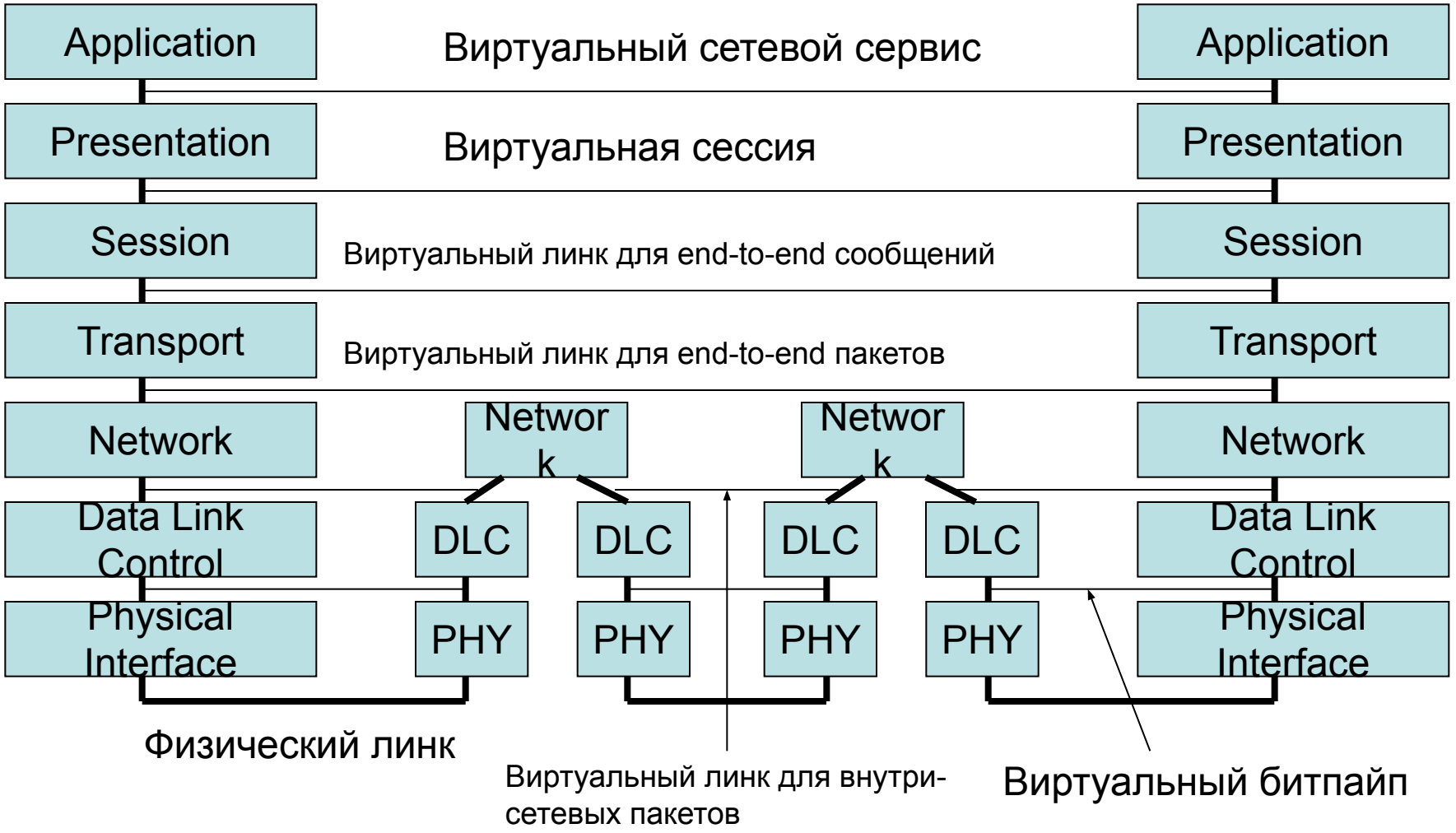
- **Преимущества packet switching**
  - Эффективность для «взрывного» трафика
  - Легкость выделения полосы по требованию для меняющихся скоростей обмена данными
- **Недостатки packet switching**
- – Колеблющиеся задержки
- – Трудно гарантировать **QoS (**вместо этого **Best-effort service)**
- – Пакеты могут приходить не по порядку

Метод коммутации	**Сетевой сервис**
Circuit switching =>	**Синхронный (напр., голосовой)**
Packet switching =>	**Асинхронный (напр., данные)**
Virtual circuits =>	**Connection oriented**
Datagram =>	**Connectionless**

Функции QoS в сетях с коммутацией пакетов

- Гарантировать ширину полосы
- Гарантировать задержки
- Гарантировать разброс задержек
- Нормировать процент потерянных пакетов

Семиуровневая модель OSI/ISO



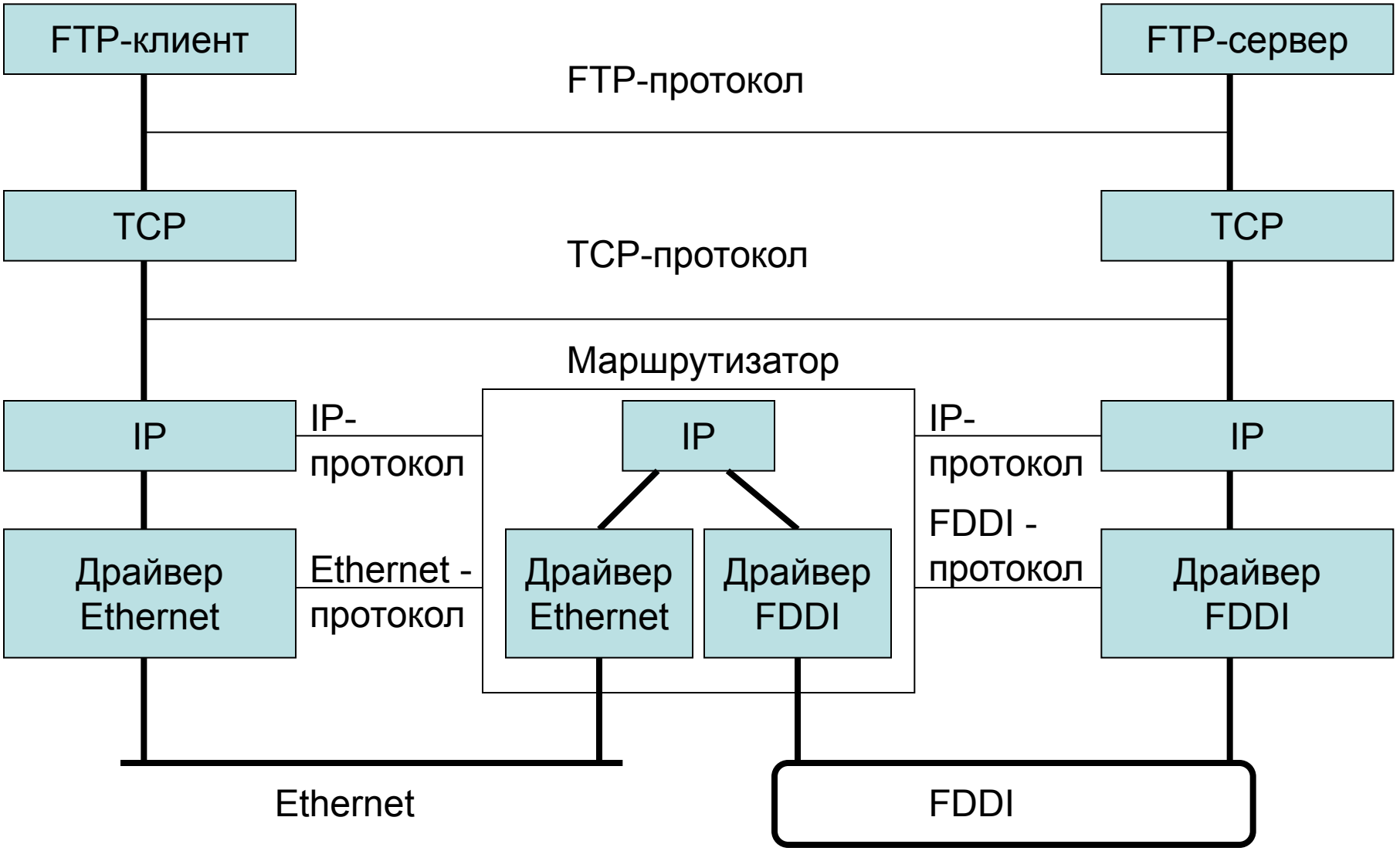
Семиуровневая модель OSI/ISO

- Presentation : кодировка, encryption/decryption, сжатие
- Session : directory, права доступа, биллинг
- Сетевой слой предоставляет транспортному слою «end-to-end»-ный пакетный пайп
- Транспортный слой предоставляет «end-to-end»-ный сервис доставки виртуальных сообщений
  - разбивает сообщения на пакеты и пересобирает в пакеты размера приемлемого для сетевого уровня
  - мультиплексирует сессии
  - отслеживает ошибки и отказы
  - end-to-end flow control
- Сетевой слой маршрутизирует пакеты на соответствующий DLC, а в узле назначения – на транспортный слой. Для маршрутизации добавляет свой заголовок в пакеты, пришедшие из транспортного уровня. Каждый узел содержит один модуль сетевого слоя плюс по одному модулю DLC на каждый линк.
- DLC : framing (начало/конец пакетов), обнаружение ошибок, повторная пересылка пакетов / протокол Automatic Repeat reQuest (ARQ).
- Физический слой
  - задержки распространения 3.3 мсек для LEO, 125 мсек для GEO, мксек для Ethernet
  - ошибки передачи, независимые от бита к биту в модели, на деле bursty

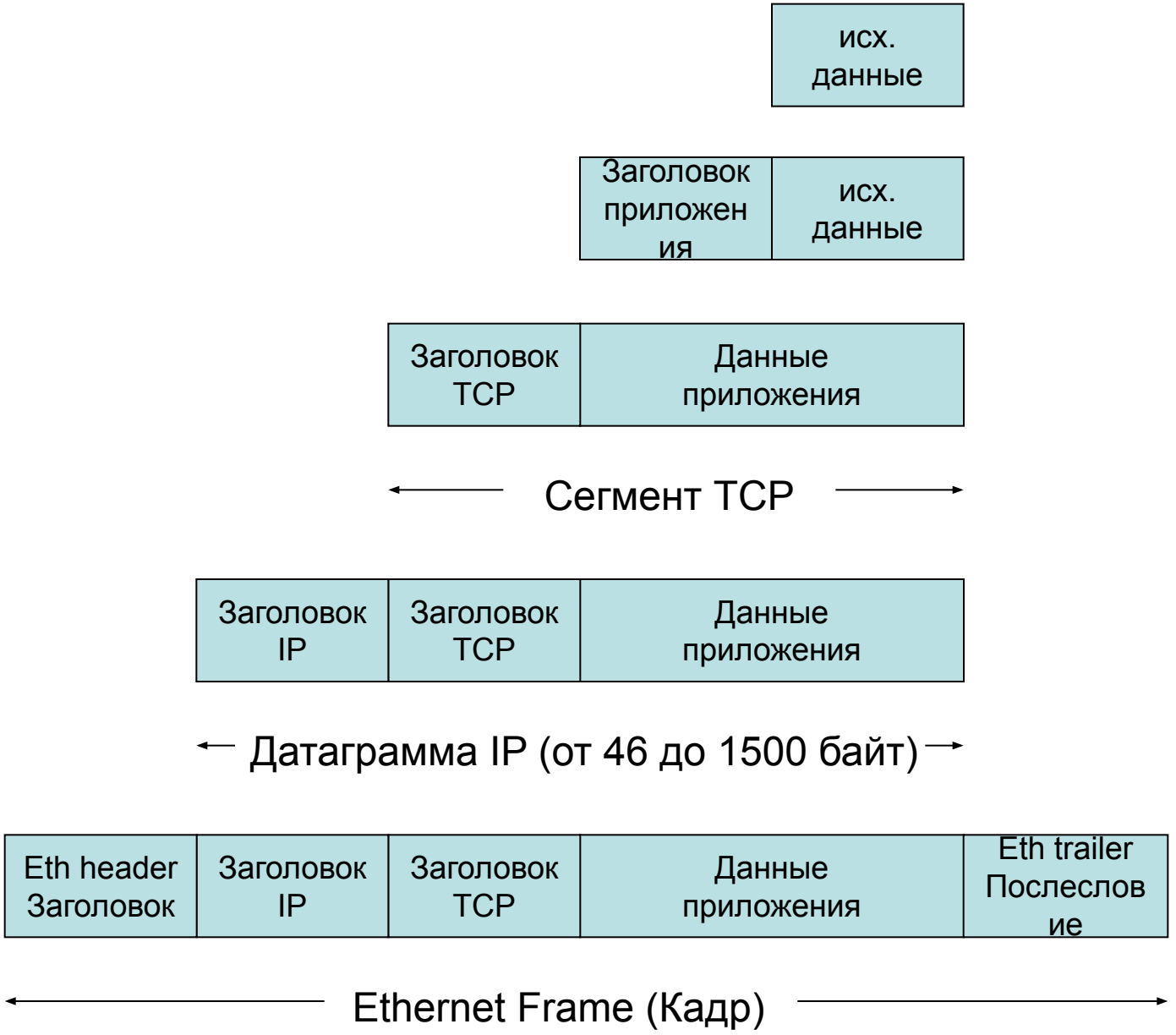
Internetworking Layer

- Расширение модели OSI/ISO: слой между сетевым и транспортным в системах, объединяющих разнородные сети
- Шлюз между различными сетями, для которых он выглядит как транспортный слой
- Отвечает за маршрутизацию и управление потоками между сетями, поэтому для транспортного слоя он выглядит как сетевой слой
- В Интернет эти функции выполняет Internet Protocol

TCP/IP

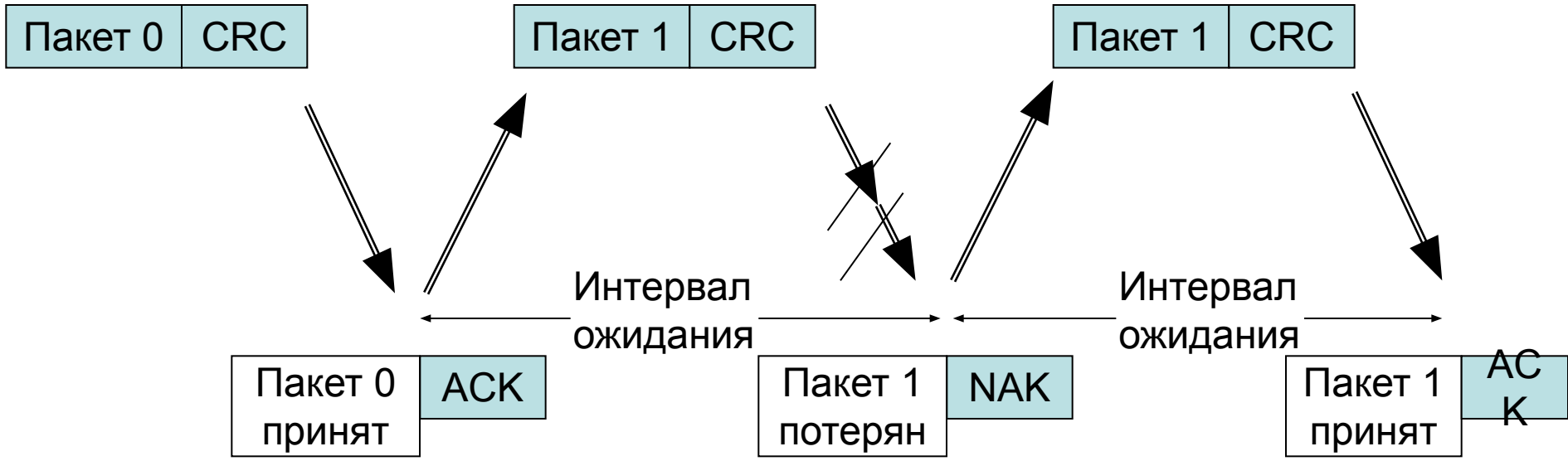

FTP-клиент
FTP-сервер
FTP-протокол
TCP
TCP
TCP-протокол
Маршрутизатор
IP
IP-
протокол
IP
IP-
протокол
IP
FDDI -
протокол
Драйвер
Ethernet
Ethernet -
протокол
Драйвер
Ethernet
Драйвер
FDDI
Драйвер
FDDI
Ethernet
FDDI

Инкапсуляция



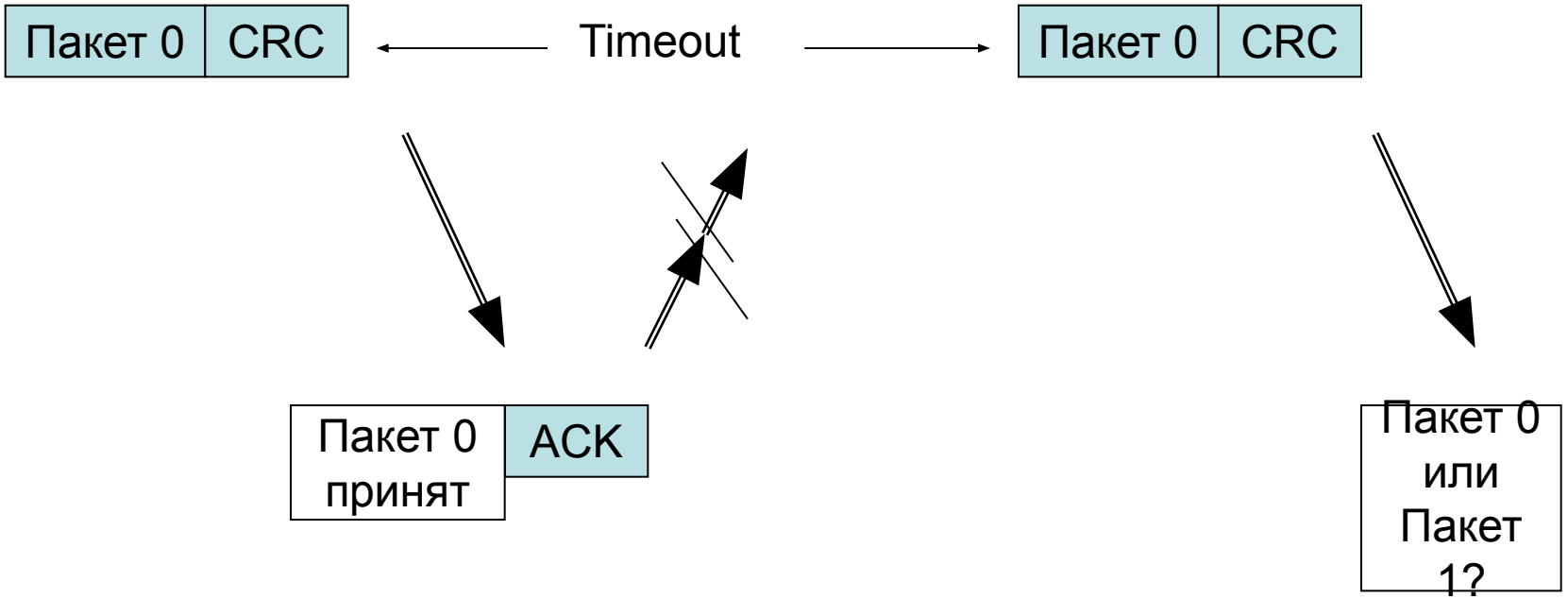
Automatic Repeat Request, ARQ

- Три схемы осуществления обмена данными с возобновлением нормального потока после обнаружения ошибки
  - Stop-n-Wait
  - Go Back N
  - Selective Repeat



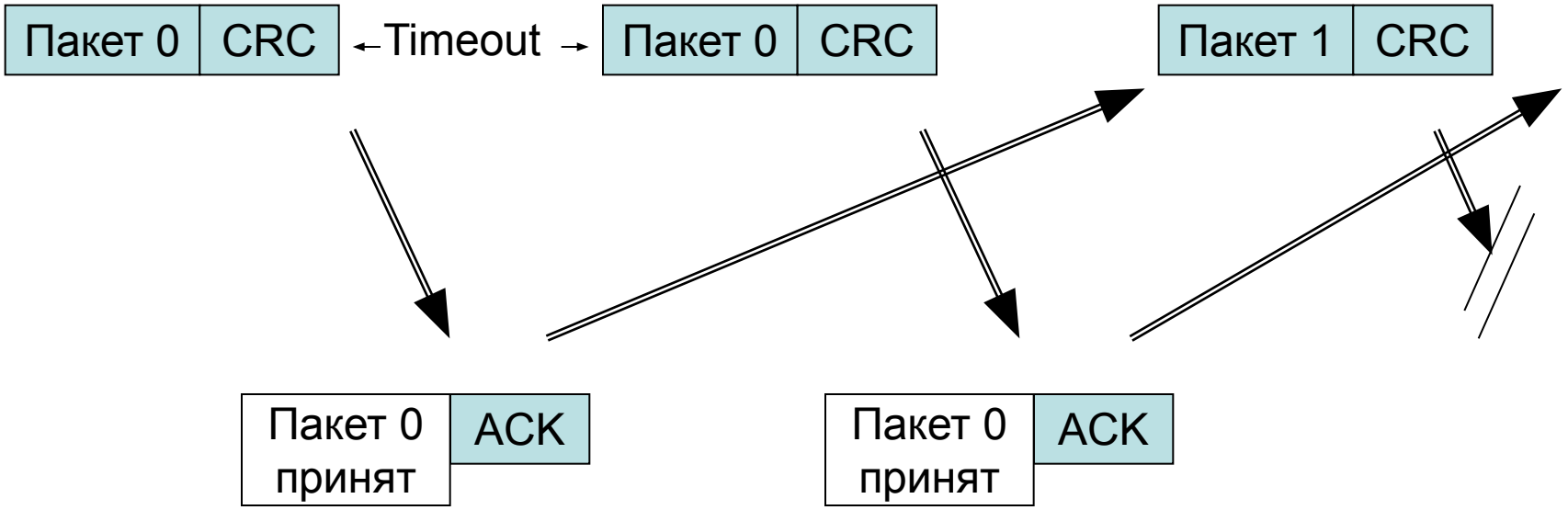
- Stop-n-Wait: необходимы также таймауты на передатчике, т.к. могут быть потеряны не только пакеты данных, но и пакеты ACK/NAK.

Stop-n-Wait 1



- Из-за возможности потерь пакетов ACK/NAK, необходимо нумеровать пакеты. Также порядок прихода датаграмм может нарушаться при маршрутизации пакетов по различным путям

Stop-n-Wait 2



- Вместо отправки пакетов двух типов (ACK и NAK), пусть получатель посылает пакет подтверждения с номером ожидаемого пакета сообщения. Можно высылать всего один бит (номер по модулю 2).
- Эта схема работает, если пакеты в линке идут по порядку (first-come first-sent), ошибки всегда обнаруживаются, и система правильно инициализирована.

Алгоритм Stop and Wait

- На узле отправителя (начальный номер пакета сообщения SN=0) :
  - 1) Вновь поступившему (от верхнего слоя) пакету назначить SN
  - 2) Передать SN в кадре с номером SN
  - 3) Дождаться безошибочного кадра от получателя
    - i. Если кадр от получателя содержит RN>SN в поле номер запрашиваемого кадра, присвоить переменной SN новое значение = RN и перейти к шагу 1
    - ii. Если превышено время ожидания (таймаут), перейти к шагу 2
- На узле получателя (начальный номер пакета запроса RN=0) :
  - 1) При получении безошибочного кадра от отправителя с SN=RN, передать на верхний слой полученный пакет и инкрементировать RN.
  - 2) В некоторый момент в установленный интервал времени, после получения безошибочного кадра от отправителя, послать отправителю кадр, содержащий RN в поле номера запрашиваемого кадра.

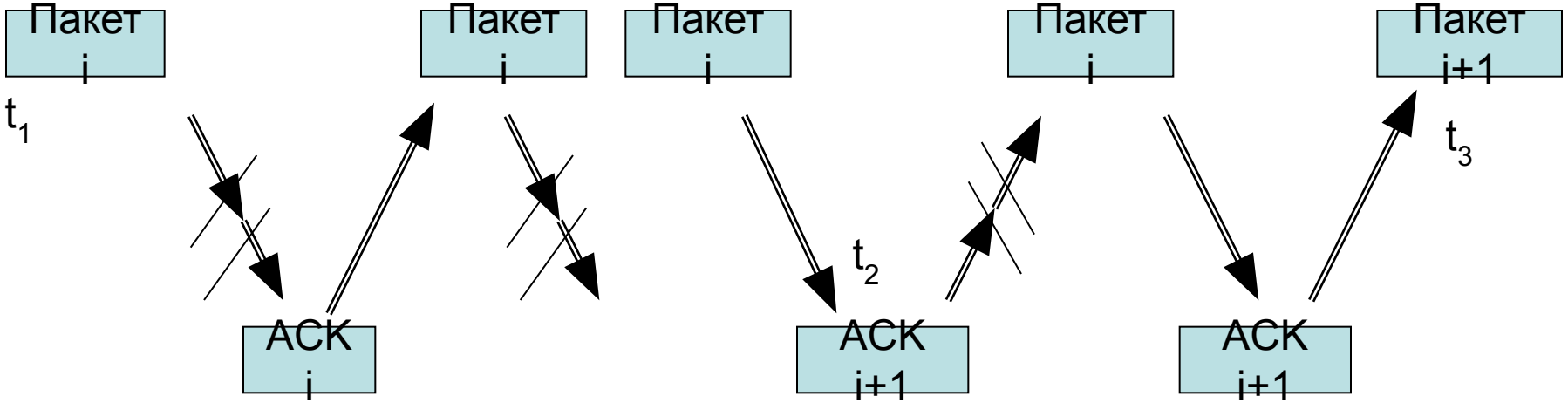
Корректность протокола Stop and Wait

- Начальные условия:
  - В линке нет кадров от предыдущих сессий
  - SN=0; RN=0
  - Все возникающие ошибки детектируются (CRC это не гарантирует)
  - Задержка кадров варьируется произвольно
  - Кадры могут теряться
  - Вероятность правильного приема каждого кадра ограничена снизу ненулевым значением $p>0$.
- Доказательство корректности состоит в доказательстве
  - ГАРАНТИРОВАННОСТИ (safety) : пакеты доставляются строго по порядку и не происходит *размножения* пакетов (каждый пакет доставляется в единственном числе)
  - и ЖИВУЧЕСТИ (liveness) : в конце концов, все пакеты доставляются (нет пропущенных пакетов)

Доказательство свойства SAFETY

- Так как изначально на линке нет пакетов, пакет 0 – первый пакет, принятый получателем. Это единственный пакет, которому присваивается SN=0. Если от отправителя дошел хотя бы один пакет, пакет, о котором идет речь, выслан отправителем. Он принимается, RN устанавливается в 1 (единицу).
- По индукции, если от отправителя доставлены все пакеты вплоть до (n-1)-го (включая n-1-ый), переменная RN устанавливается в значение n при доставке (n-1)-го пакета, и только n-ный может быть доставлен следующим – любой с SN < n не будет принят, следовательно, невозможно размножение пакетов, и каждый принятый пакет имеет номер строго больший, чем номера любых ранее доставленных пакетов (упорядоченность).
- Корольларий SAFETY: из причинности [пакет не может быть принят до того, как был хотя бы раз отправлен], к моменту первой попытки отправки (n-1)-го пакета он не мог быть принят, поэтому RN < n, и, поскольку SN = n-1 в момент отправки (n-1)-го пакета, RN ≤ SN в момент первой попытки отправки любого пакета.

Доказательство свойства LIVENESS



- t_1 – исходная попытка отправителя послать пакет i.
- t_2 – момент корректной доставки получателю пакета i и инкрементирования RN до значения i+1
- t_3 – момент инкрементирования SN до значения i+1

Докажем, что $t_1 < t_2 < t_3 < \infty$, из чего => LIVENESS

Доказательство свойства LIVENESS

1) Пусть SN(t), RN(t) – значения SN и RN в момент t
2) Алгоритм таков, что SN(t), RN(t) – неубывающие функции t, $SN(t) \le RN(t)$ в любой момент времени.
3) Поскольку до момента t_1 попыток отправить пакет i не делалось, из доказательства safety следует, $RN(t_1) \le i$ и $SN(t_1) = i \Rightarrow RN(t_1) \le SN(t_1)$
4) В момент t_1 одновременно $SN(t_1) \le RN(t_1)$ (из алгоритма) и $RN(t_1) \le SN(t_1)$ (из safety, верно только в моменты первых попыток отправки пакетов); $\Rightarrow SN(t_1) = RN(t_1) = i$.
5) RN инкрементируется в момент t_2, а SN – в момент t_3. Что будет, если наступит t_3 раньше t_2? Поскольку, вследствие пункта 4, начиная с t_1 и вплоть до следующего события выполняется равенство $SN(t_1) = RN(t_1)$, а в момент t_3 инкрементируется SN, на какое-то время окажется, что SN < RN, а это противоречит пункту 2. Следовательно, $t_2 \le t_3$.
6) Отправитель предпринимает попытки отправить пакет i вплоть до момента времени t_3, следовательно, и до момента t_2, когда этот пакет корректно принят получателем. Поскольку вероятность правильного приема каждого кадра ограничена снизу ненулевым значением p>0, и интервалы между повторами также ограничены, поэтому момент t_2 наступает в конечный момент времени.
7) Аналогично доказывается, что момент времени t_3 также конечен.

Однобитовые SN и RN

- Кадры идут по линку строго по порядку
- При типе данных целое переменных SN и RN имеем либо SN=RN (между t_1 и t_2), либо SN=RN-1 (между t_2 и t_3).
- Благодаря упорядоченности кадров, достаточно однобитовых переменных SN и RN (целые SN и RN по модулю 2):
  - RN = 0 и SN = 1 или RN =1 и SN = 0 (принятый пакет – старый пакет)
  - RN = 0 и SN = 0 или RN =1 и SN = 1 (принятый пакет – новый пакет)

Эффективность протокола Stop and Wait в отсутствии ошибок

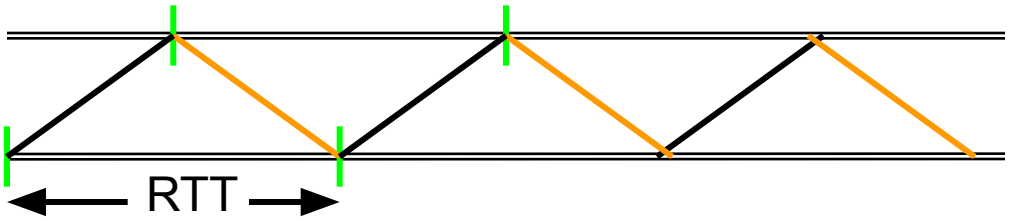
- T – полное время между отправкой пакета i и приемом подтверждения на этот пакет
- D_{TP} – время передачи пакета
- D_P – задержка распространения при передаче пакета и подтверждения
- D_{ACK} – время передачи подтверждения
- Эффективность = D_{TP}/T = $= D_{TP}/(D_{TP} + 2D_P + D_{ACK})$

Эффективность протокола Stop and Wait в присутствии ошибок

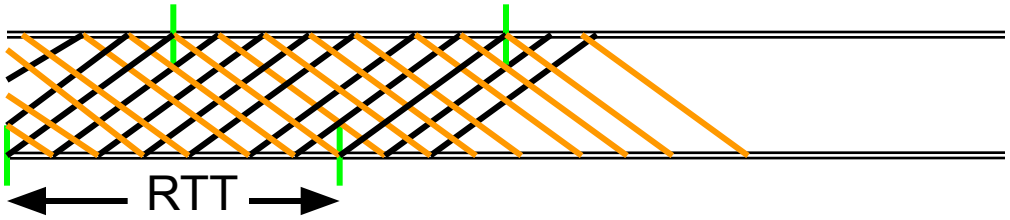
- P – вероятность ошибки при передаче пакета или подтверждения
- Среднее время передачи пакета увеличивается на величину timeout*P/(1-P)
- Эффективность = D_{TP}/T = $D_{TP}/(D_{TP} + 2D_P + D_{ACK} + timeout*P/(1-P))$
- Полудуплекс: $timeout = D_{TP} + 2D_P + D_{ACK}$
- Полный дуплекс: $timeout = D_{TP}$

Go Back N ARQ (Sliding Window)

- На отправителе "окно" из N пакетов, которые можно отправить, не дожидаясь подтверждений
- Номера пакетов в окне идут от последнего значения RN, полученного в подтверждениях от получателя (обозначается SN_{min}) до SN_{min}+N-1
- Исчерпав пакеты из окна, или в случае таймаута, отправитель вновь посылает пакет SN_{min} – пакет с наименьшим номером, на который еще не поступило подтверждение
- Пусть SN_{max} – номер пакета, который на данном шаге поступит в окно из вышележащего сетевого слоя (т.е., следующий пакет, который должен быть послан на линк)



- SAW



- Go Back N (sliding window)

Алгоритм Go Back N на стороне отправителя

- $SN_{min} = 0$; $SN_{max} = 0$
- Цикл
  - Если $SN_{max} < SN_{min} + N$ (еще не все окно целиком выслано на линк) отправить пакет SN_{max} ; $SN_{max} = SN_{max} + 1$;
  - Если от получателя приходит подтверждение с номером $RN > SN_{min}$ $SN_{min} = RN$;
  - Если $SN_{min} < SN_{max}$ (есть еще неподтвержденные пакеты) и отправитель не имеет новых пакетов к отправке, выбрать какой-нибудь пакет между SN_{min} и SN_{max} и повторно отправить его
- Есть две причины невозможности отправить новый пакет: ничего нового с верхнего слоя, или окно заполнено до отказа ($SN_{max} = SN_{min} + N$)
- Для определенности, в отсутствие новых пакетов выбираем для повторной отправки последний отправленный пакет

Алгоритм Go Back N на стороне получателя

- RN = 0
- Цикл
  - Если приходит пакет с SN = RN, принять пакет и инкрементировать RN;
- Регулярно высылать подтверждения с текущим RN . Во многих реализациях Data Link Control Layer это происходит, когда поступает пакет с другой стороны линка
- Получатель отвергает любые пакеты с SN не равным RN.

Selective Repeat Protocol (SRP)

- Попытки предпринимаются повторной отправки только фактически потерянных пакетов (из-за ошибок)
  - Получатель должен уметь получать пакеты не по порядку
  - Поскольку в верхний слой пакеты надо выдавать упорядоченными, получатель должен так или иначе буферизовать пакеты
- Запросы на повторную пересылку
  - Неявные
    - Получатель подтверждает каждый хороший пакет, неподтвержденные по истечение таймаута пакеты считаются потерянными или ошибочными
    - Именно этот подход должен использоваться, чтобы гарантировать, что все пакеты в конце концов будут доставлены
  - Явные
    - Явный NAK (selective reject) может затребовать повторной отправки конкретного пакета. Этот подход может ускорить повторную отправку, но не является неизбежно необходимым
- На практике используются оба подхода

Очереди

- В сетях с коммутацией пакетов случайными могут быть времена прихода пакетов и продолжительности пакетов
- На физическом уровне мы озабочены величиной Bit Error Rate, на сетевом главная забота – задержки
  - Сколь долго пакет находится в буфере?
  - Как велики должны быть размеры буферов?
- В сетях с коммутацией линий связи с помощью теории очередей исследуется вероятность блокировки звонка

Распределения случайных переменных в сетевых приложениях

- Средняя скорость поступления пакетов λ пакетов в секунду (распределение Пуассона)
- На протяжении малого интервала времени Δt
  - $P(0 \text{ пакетов}) = 1 - \lambda \cdot \Delta t + o(\Delta t)$
  - $P(1 \text{ пакет}) = \lambda \cdot \Delta t + o(\Delta t^2)$
  - $P(2 \text{ пакета}) = (\lambda \cdot \Delta t)^2/2 + o(\Delta t^3)$
- Точная формула распределения:
    - $P(n \text{ пакетов за время } T) = (1/n!)(\lambda \cdot T)^n \cdot \exp(-\lambda \cdot T)$
- Плотность вероятности распределения количества событий в заданном интервале времени определяется распределением Пуассона
- Плотность вероятности распределения времени ожидания наступления в точности n событий определяется распределением Эрланга
    - $f(t; n, \lambda) = (1/(n-1)!) \cdot \lambda^n \cdot T^{n-1} \cdot \exp(-\lambda \cdot T)$

Теорема Литтла для пакетов в сети

- N = среднее кол-во пакетов в системе
- T = среднее время, которое пакет проводит в системе
- λ = скорость (rate) поступления пакетов в системе (не обязательно Пуассоновское распределение)

- Теорема Литтла: $N = \lambda T$
  - Может применяться ко всей системе или к любой ее части
  - Наводненные пакетами системы -> большие задержки

Применения теоремы Литтла

1. Передатчик: D_{TP} = время передачи пакета
   - Среднее кол-во пакетов на передатчике = $\lambda D_{TP} = \rho$ = к.п.д линка (использование линка)
2. Линия передачи: D_p = задержка распространения
   - Среднее кол-во пакетов "в полете" = λD_p
3. Буфер: D_q = средняя задержка в очереди
   - Среднее кол-во пакетов в буфере = $N_q = \lambda D_q$
4. Передатчик + буфер
   - Среднее кол-во пакетов в буфере = $\rho + N_q$

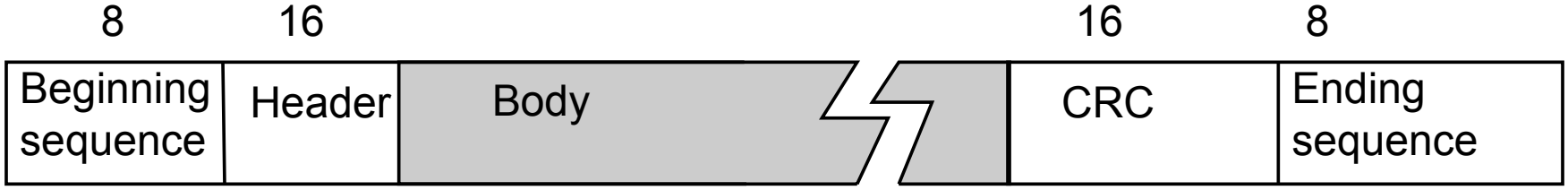
Модели очередей

- Система обозначений для классификации очередей:
- A/B/S/K/N/Disc
  - A – распределение поступающих запросов (в сетях с коммутацией пакетов это распределение времени между поступлением пакетов в узел)
  - B – распределение времен обслуживания запросов (распределение времен нахождения пакета в очереди)
  - S – количество серверов
  - K – емкость системы
  - N – популяция «клиентов»
  - Disc – принятая дисциплина обслуживания очереди (FIFO, LIFO, SFQ)
- В нотации с тремя первыми термами (A/B/S) емкость системы неограничена ($K=\infty$), популяция клиентов неограничена ($N=\infty$), FIFO.
- Стандартные обозначения для распределений, которые могут встретиться в позициях A или B:
  - M : Экспоненциальное распределение (Марковские цепи)
  - Ek : Распределение Эрланга с k фазами
  - D : Вырожденное (детерминированное) распределение (константа)
  - G : General distribution (произвольное)
  - PH : Phase-type distribution

Очереди системы с одним сервером

- В буфер поступает λ пакетов в секунду
- Сервер обслуживает μ пакетов в секунду, время обслуживания = 1/μ
- M/M/1 : Пуассоновское распределение поступающих пакетов, экспоненциальное распределение времен обслуживания
- M/G/1 : Пуассоновское распределение поступающих пакетов, общий случай распределения времен обслуживания
- M/D/1 : Пуассоновское распределение поступающих пакетов, детерминированное распределение времен обслуживания (фиксированное значение времени обслуживания)

Frames



- В начале и конце кадра вставляются флаги, например, 01111110. Если в данных есть шесть последовательных единиц, после пятой единицы при передаче вставляется лишний нуль, при приеме после пяти последовательных единиц убирается обязательно имеющийся там нуль. Если встречается шесть единиц подряд, то это либо флаг конца сообщения, либо ошибка. В худшем случае длина сообщения увеличивается на 20% (данные состоят из одних единиц). Такой способ раскадровки используется в Point-to-Point Protocol и в High-level Data Link Control (HDLC)
- BISYNC (BInary SYNchronous Communication) – Вставка спецсимволов: SYN, SOH, STX, ETX, DLE
- Счетчик длины передаваемой последовательности
- В SONET подсчитываются кадровые импульсы. Каждый кадр имеет фиксированную продолжительность, 125 мксек, или 810 байт для STS-1. STS-*n* (STS-1 = 51.84 Mbps)

Обнаружение/исправление ошибок

- Простой и двумерный биты четности
- Код Хэмминга:
  - Дистанция Хэмминга равна количеству положений битов, в которых любые два кодовых слова (из набора) различаются.
  - 0000, 0011, 1001, 1100, 1010, 0101, 0110, 1111 (дистанция = 2)
  - 000, 111 (дистанция = 3)
- 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–32: $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$
  - Обнаруживаются все однобитовые ошибки, если коэффициенты при x^k и x^0 ненулевые.
  - Обнаруживается любое нечетное кол-во ошибок, если C(x) содержит множитель (x + 1)
  - Обнаруживается любая выбросная (‘burst’) ошибка (т.е., последовательность идущих непрерывно один за другим ошибочных битов), для которых длина ‘выброса’ меньше k бит
  - Многие выбросные (‘burst’) ошибки длиннее k бит также могут быть обнаружены
- Коды Рида-Соломона (CD, дальняя космическая связь)

Теория вероятности (дискретное пространство событий)

- Вероятностное пространство– набор элементов (sample points), или исходов испытаний
- Подмножества вероятностного пространства называются события (events)
- До опыта возможен любой исход, опыт приводит к реализации только одного исхода
- Событие определяет набор возможных исходов опыта (потеря пакета – потерян любой пакет, столкновение частиц – столкнулась любая пара частиц)
- Исход полностью описывает результат опыта, событие – лишь частично
- Вероятность – мера в вероятностном пространстве, принимает значения между 0 и 1, мера всего пространства единица (1)
- Если количество исходов бесконечно, вероятность отдельных исходов равна 0, ненулевые вероятности назначаются более сложным событиям

Теория вероятности (дискретное пространство событий) II

- При генерации последовательностей символов ограниченного алфавита, вероятность генерации на n-ом шаге символа α есть событие X_n , множество всех последовательностей с символом α в n-ом положении.
- Вероятность этого события $Pr(X_n = \alpha)$ обозначается $p_{Xn}(\alpha)$. $\Sigma_\alpha p_{Xn}(\alpha)=1$. X_n называется случайная величина (chance variable).
- Случайная переменная (random variable) R есть случайная величина, для которой отображение производится из подмножества (или всего множества) вещественных чисел
- Если подмножество исходных значений ***R*** конечное (или счетное), R называется дискретной случайной переменной, и вероятность исходов описывается функцией вероятности $p_R(r)=Pr(R=r)$ для всех возможных r из ***R***.
- Если подмножество исходных значений ***R*** несчетное, вероятность описывается функцией распределения $Pr(R\leq r)$ (Cumulative Distribution Function). Во многих случаях можно ввести плотность функции распределения $f_R(r)=dPr(R\leq r)/dr$ (Probability Density Function)

Теория вероятности (дискретное пространство событий) III

- Случайная величина детерминирована, если $p_R(r)=1$ для какого-либо r. Тогда для других r, $p_R(r)=0$.
- Дискретная случайная величина равновероятна, если $p_R(r)=1/M$, а M – мощность множества исходных значений (конечная величина в данном случае). Для генерации последовательности символов, это количество символов в алфавите.
- Ожидание или среднее (усредненное) значение случайной переменной R есть

$$\mathrm{E}[R] = \sum_{r \in \Re} p_R(r) r = \overline{R}$$

- Ожидания аддитивны: $E[R_1]+E[R_2]=E[R_1+R_2]$

Теория вероятности (дискретное пространство событий) IV

- Декартово (прямое) произведение двух дискретных алфавитов А и Б есть дискретный алфавит (множество всех упорядоченных пар букв, первая из одного, вторая из другого алфавитов)
- Функция вероятности $p_{AB}(a,b)$ определена для всех a € A и b € Б, называется совместная функция вероятности (joint pmf).

$$p_Q(q) = \sum_{r \in \Re} p_{RQ}(r,q)$$

- Условная pmf определяется так: $p_{Q|R}(q \mid r) = \frac{p_{RQ}(r,q)}{p_R(r)}$

- Истинная pmf называется маргинальной. Две случайных величины независимы, если их совместная функция вероятности факторизуется в произведение соответствующих маргинальных. Ожидание произведения двух независимых случайных величин равно произведению ожиданий каждой из них

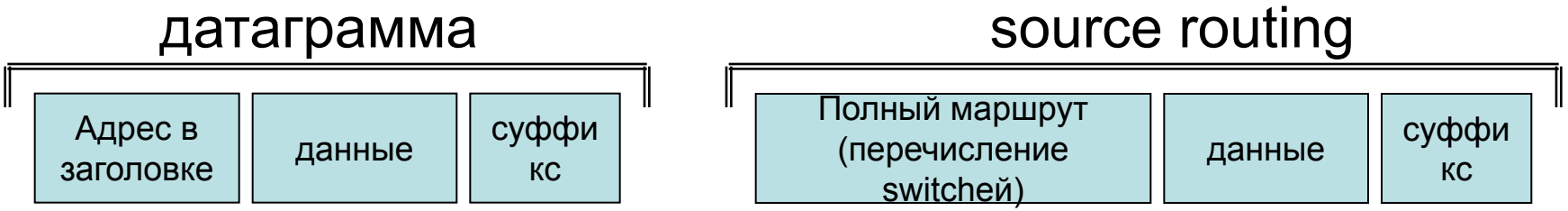
Теория вероятности (дискретное пространство событий) V

- Моменты: $E[R^n]$
- Первый момент – среднее
- R – E[R] называется флуктуация, ее среднее = 0
- Второй момент флуктуации называется вариация $\sigma_R^2 = E[R^2] - (E[R])^2$

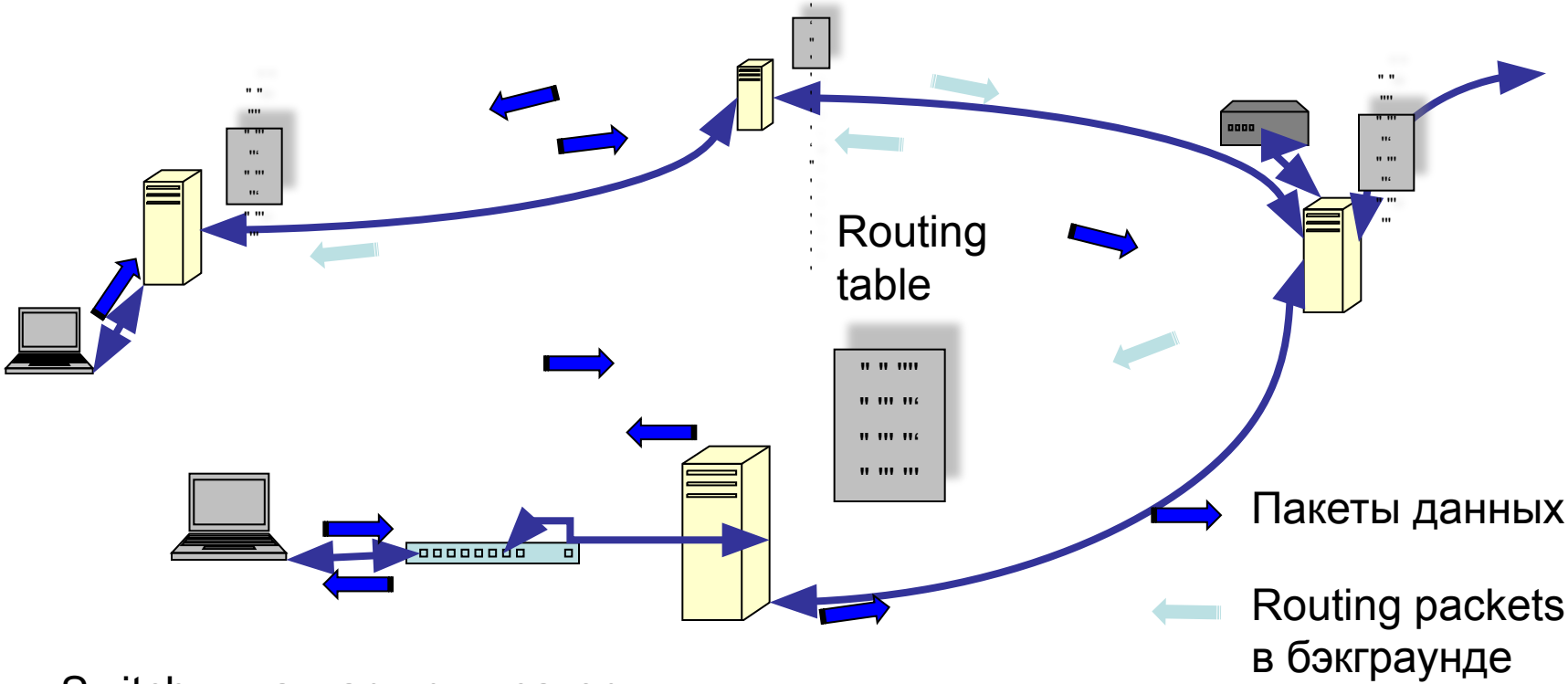
Store-and-forward

- Switch: специализированный компьютер. К switch-у подключают несколько линков. ПО switch-а анализирует пакет, пришедший по одному линку, и отправляет его по другому линку. Называется store-and-forward technique (три вида).

1) Датаграммами называются пакеты сервисов негарантированной доставки (UDP, IP). Маршрутизаторы переправляют пакеты согласно адресу в заголовке и маршрутам, хранящимся в маршрутизаторе.
2) В заголовке может содержаться полный маршрут пакета до пункта назначения. Работа switch-а проще, чем работа маршрутизатора. Эта техника применяется в LAN.
3) Виртуальные цепи, virtual circuits, используют при коммутации заголовки пакетов. С другой стороны, перед сессией выполняется процедура настройки виртуального канала, как при circuit switching

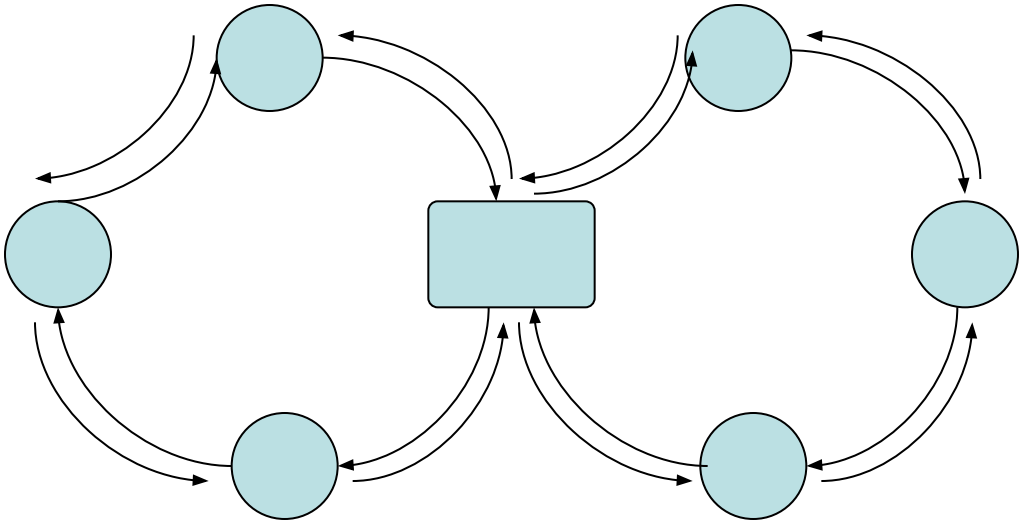


Маршрутизаторы



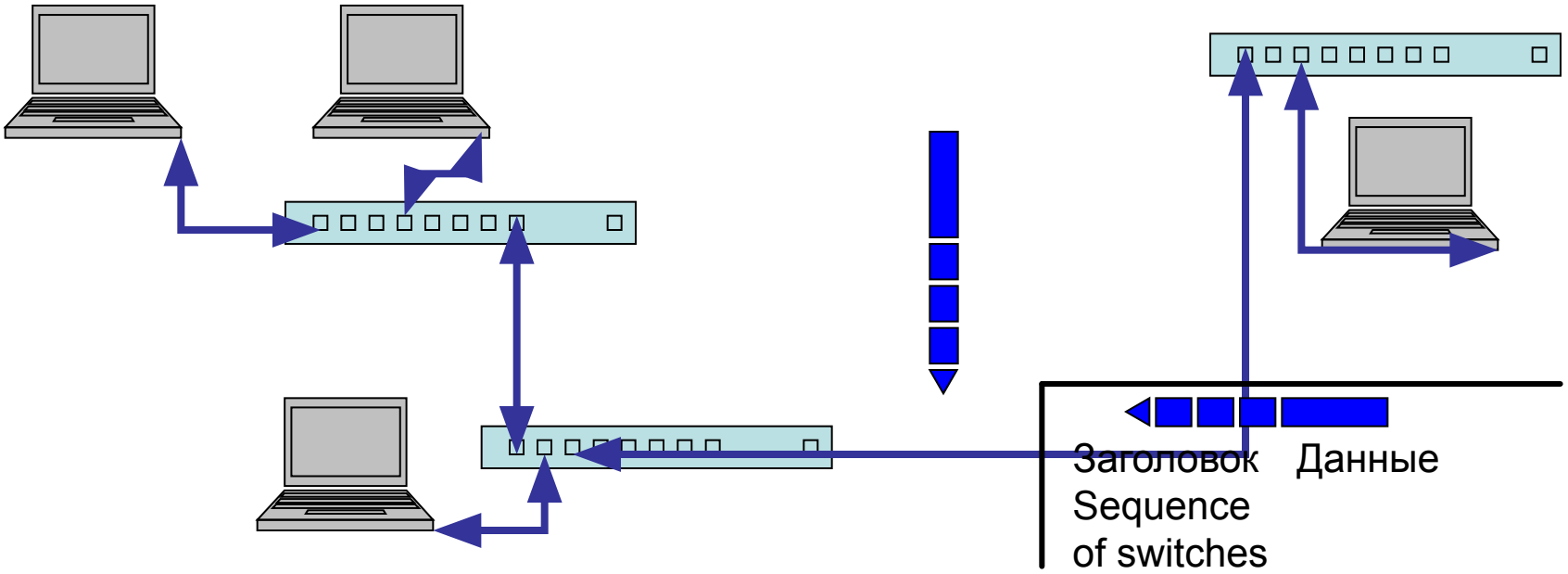
- Switch типа маршрутизатор:
- Пакет (датаграмма) несет в заголовке только адрес пункта назначения. Маршрутизатор, используя этот адрес как ключ, сверяется с lookup-таблицей (routing table или forwarding table, между двумя есть тонкое различие). Таблица дает выходной порт маршрутизатора.
- Маршрут "прописывается" в каждый маршрутизатор при настройке или вычисляется бэкграундным процессом с помощью протоколов маршрутизации, и запоминается. Отправитель знает маршруты только до соседних узлов.

Token Ring switch



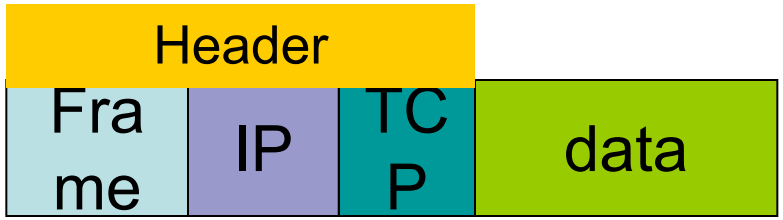
- Коммутация пакетов реализована с помощью source routing: каждый отправитель вычисляет маршрут и передает маршрут в заголовке пакета. Switch-и освобождаются от операции table lookup.
- Каждый отправитель/получатель пакетов должен участвовать в работе протоколов маршрутизации
- После прохождения switch-а, адрес перемещается в хвост последовательности адресов.

LAN Switches



- Адреса хостов, подключенных в порту switch-а, хранятся в content addressable memory. Номер порта для перенаправления пакета извлекается без lookup-а, одним обращением к памяти.
- Switch в локальной сети называется bridge (мост)
- Мосты прозрачны для пакетов – не нарушают их целостности
- Иногда мостом называют switch с двумя интерфейсами. В этой терминологии switch-ем называют Ethernet switch с несколькими (>2) интерфейсами.

Данные и switch-и



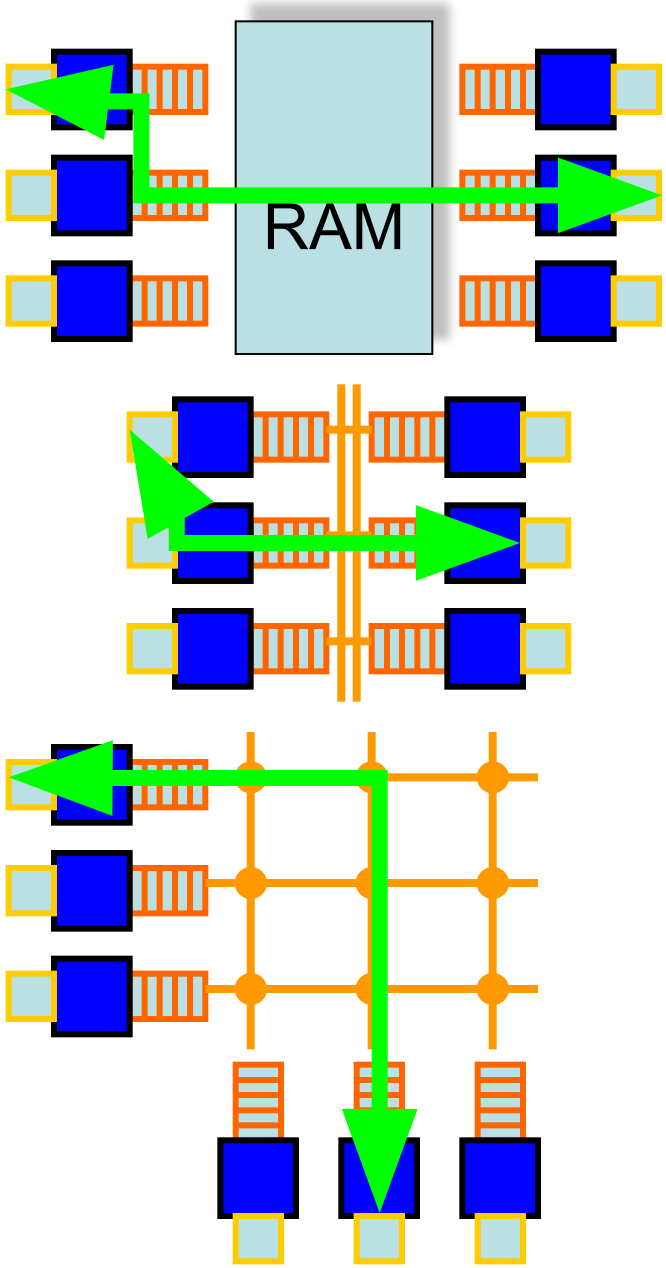
- Phy: repeater-ы и hub-ы. Общий доступ, расстояние для Ethernet не более 1500 м.
- DLC: switch-и и мосты
- Network layer: маршрутизаторы

- Cut-through switching: передача фрейма начинается немедленно по получении заголовка, не дожидаясь окончания приема данных и суффикса

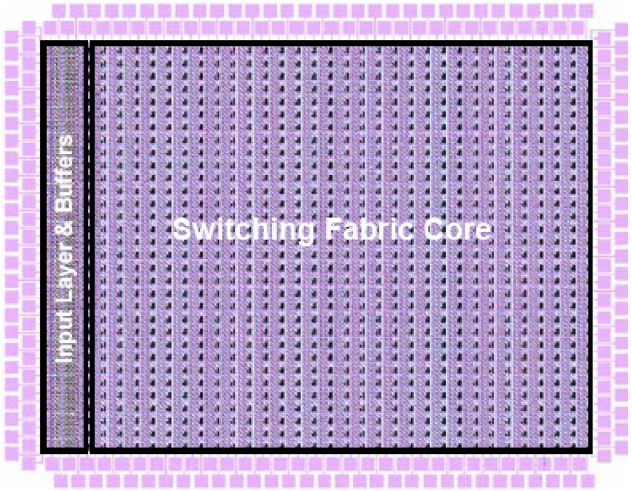
Электроника switch-a

- **Входные порты.**
  - Phy: Согласование электрических характеристик или преобразование света в электрические сигналы
  - Буферизуют входящие пакеты, выполняют lookup для определения выходных портов.
  - DLC: Анализ заголовка, данные – в switching fabric, контрольные пакеты (RIP, OSPF, IGMP) – в процессор.
  - Скорость линка OC48 2.5 Gbps. С 256-байтовыми пакетами, около миллиона операций lookup в секунду. Не линейный поиск, а поиск по дереву.
- **Switching fabric.** Соединяет входные порты с выходными портами. Сеть внутри сетевого устройства.
- **Выходные порты.** Буферизуют исходящие пакеты и выдают датаграммы на исходящий линк. В обратном порядке совершаются действия входных портов.
- **Процессор.** Исполняет протоколы маршрутизации, ведет таблицы маршрутизации и исполняет управляющие функции в маршрутизаторе/switch-е.

Switching fabric

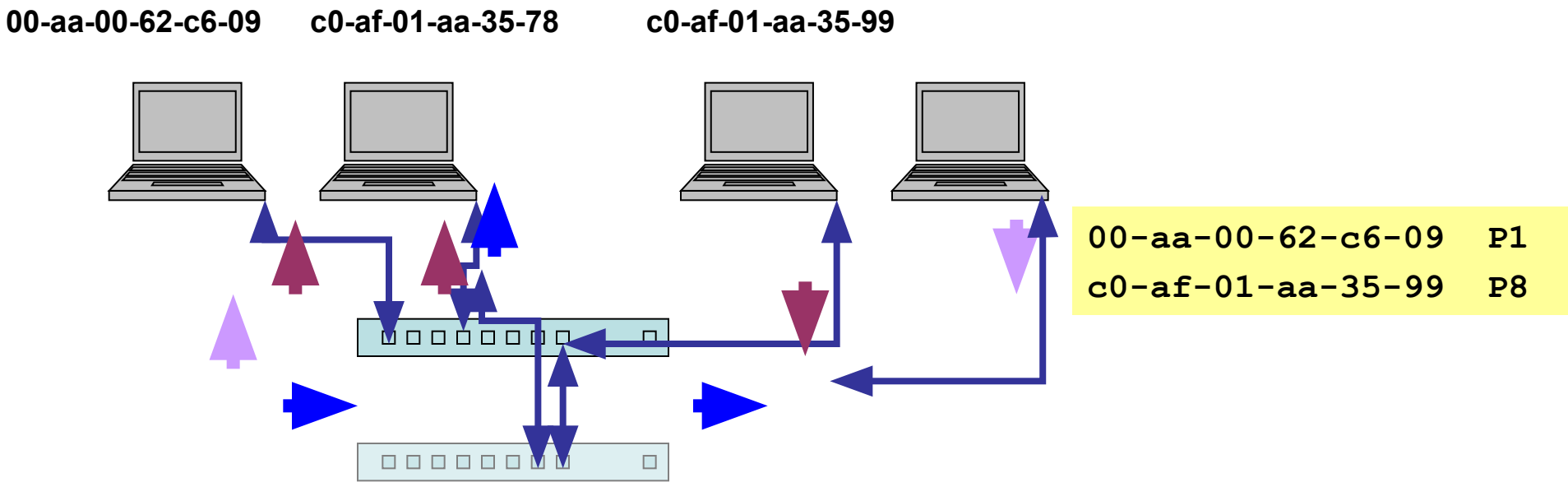


- Через память: процессор по прерыванию записывает из входного порта в память, затем считывает в выходной порт. В современных устройствах это процессор карты входных портов. CISCO Catalyst 8500, Bay Networks Accelar 1200
- Через шину: Cisco 1900, 1Gbps Packet Exchange Bus; 3Com CoreBuilder 5000, PacketChannel data bus 2 Gbps
- Через внутреннюю сеть: switch-и Cisco 12000, 60 Gbps



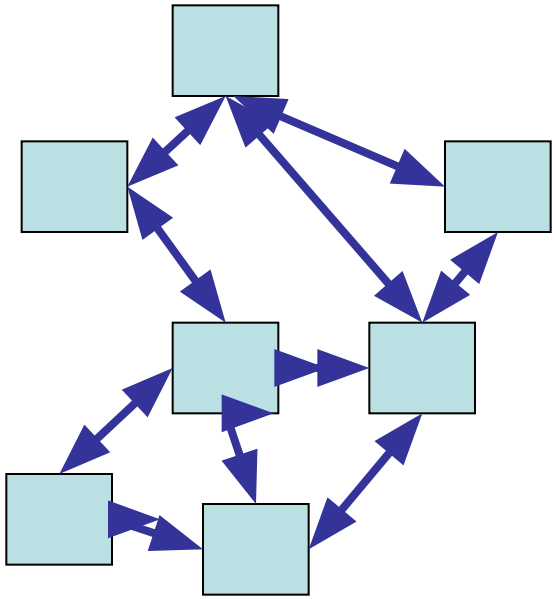
FLEXBAR, 32x32, 0.35m CMOS

Learning bridge



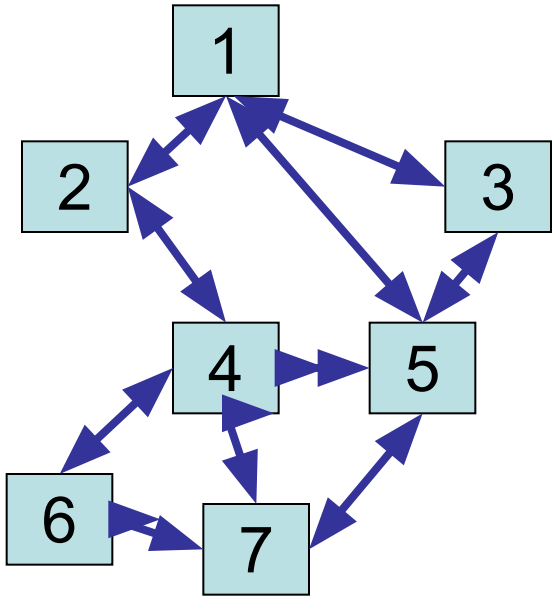
- Получив пакет, мост запоминает в кэше связку "адрес отправителя – порт".
- Если в кэше нет адреса получателя, пакет широковещается на все подключенные LAN-ы, кроме порта, с которого он получен.
- В кэше находятся адреса всех отправителей и портов, к которым подключены отправители
- Циклы приводят к дублированию и даже размножению пакетов.

Алгоритм spanning tree



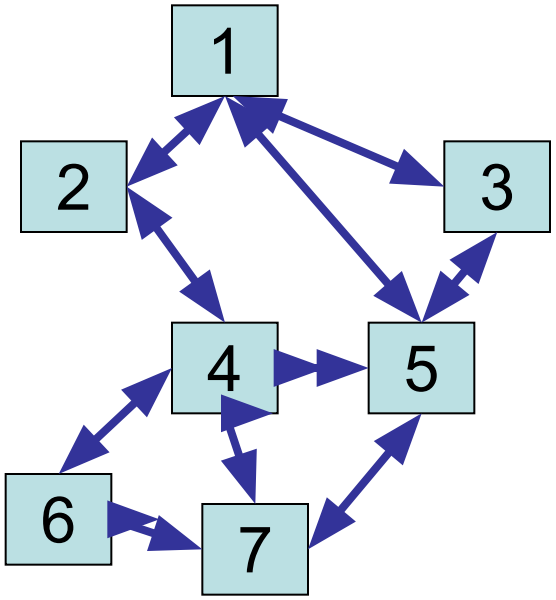
- Все узлы связаны, но без циклов
- По линкам, не принадлежащим дереву, могут передаваться только пакеты отправителя и получателя узлов этого линка (no forwarding across the link).
- Распределенный алгоритм:
  - Узлы выбирают root (если нет таблицы соединений, то себя, иначе – с минимальным ID)
  - Оставляем интерфейс на кратчайшем расстоянии от root-а
  - Каждый узел посылает сообщение M(Y,d,X)
    - Y – root (по мнению X на данный момент)
    - D – расстояние
    - X – сам узел

Алгоритм spanning tree



- Начало: каждый узел шлет сообщение M(X,0,X) соседям.
- Принимает сообщение. Если root ID полученного сообщения меньше того, который он считал root-ом, считает его новым root-ом.
- Добавляет 1 к расстоянию от root-а в полученном сообщении и считает это своим новым расстоянием от root-а. Находит интерфейсы не на кратчайшем расстоянии от root-а и исключает их из spanning tree.
- Узел 6: думает, что root.
- 6 шлет M(6,0,6)
- 6 получает (4,0,4), инкрементирует d=d+1
- 6 получает (4,1,7) от 7, исключает линк 6-7 из дерева
- 4 получает (1,1,2) от 2, назначает для себя 1 root-ом
- 4 шлет M(1,2,4) соседям
- 6 получает M(1,2,4) от 4, шлет M(1,3,6)
- 6 получает M(1,2,7) от 7, сохраняет интерфейс на 4 в дереве, шлет соседям M(1,3,6)

Алгоритм spanning tree



- Алгоритм должен быть устойчив к отказам root-а: в случае отказа, при перевыборах побеждает root с минимальным ID из оставшихся
- Алгоритм должен быть устойчив к отказам прочих switch-ей и линков: пересчет структуры spanning tree при необходимости
- Root switch периодически объявляет себя сообщениями M(1,0,1).
- Если нет сообщений ни от root-а, ни от других switch-ей, объявляет себя новым root-ом.

VLAN, Internetworking

- Виртуальные LAN: Пакеты из одного VLAN не широковещаются в другой VLAN
- VLAN-enabled switch-и изменяют заголовок: добавляют тэг VLAN-а в заголовок Ethernet-фрейма
- Ограничения LAN:
  - Switch-и хранят таблицы соответствий хост-порт. Снижает масштабируемость
  - Проблемы при соединении сетей с разными технологиями. Internetworking problem.

Протоколы маршрутизации

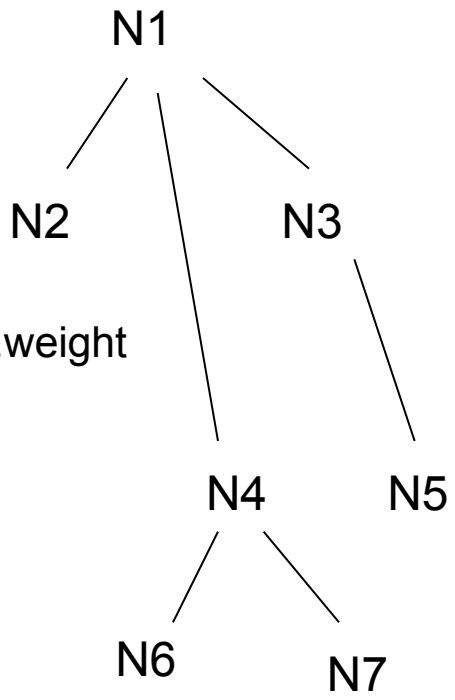
- Проактивные протоколы
  - Определяют маршруты вне зависимости от наличия трафика
  - Традиционные протоколы (link-state и distance-vector) – проактивные протоколы
    - Route Information Protocol
- Реактивные протоколы (on demand)
  - Поддерживают маршруты по мере возникновения надобности
- Гибридные протоколы
- RIP работает в Автономной системе, использует алгоритм DV:
  - Каждый узел вычисляет расстояния между собой и другими узлами в AS, заносит результат в таблицу.
  - Узел рассылает свою таблицу соседним узлам.
  - Получив таблицы от соседей, узел вычисляет кратчайшие маршруты к остальным узлам и обновляет свою таблицу.
- Управляющие пакеты могут быть уникастными и мультикастными.

Distance Vector Protocols

```
function BellmanFord(list vertices, list edges, vertex source)
// Берется граф (список вершин и сторон)
// модифицируем «веса» в вершинах так, чтобы атрибуты
// distance и predecessor содержали кратчайший путь.
// Шаг 1: Инициировать граф
for each vertex v in vertices:
if v is source then v.distance := 0
else v.distance := infinity v.predecessor := null

// Шаг 2: поэтапно обходя стороны, вычислять атрибуты
for i from 1 to size(vertices):
for each edge uv in edges:
u := uv.source v := uv.destination // uv is the edge from u to v
if v.distance > u.distance + uv.weight: v.distance := u.distance + uv.weight
v.predecessor := u

// Шаг 3: Проверка на циклы с отрицательным весом
for each edge uv in edges:
u := uv.source v := uv.destination
if v.distance > u.distance + uv.weight:
error "Graph contains a negative-weight cycle"
```



Distance Vector Protocols

- В ns, DV-протокол задается командой
  `$ns rtproto DV`
- Без этой команды маршрутизация вычисляется вначале симуляции и остается статической на протяжении всей симуляции, если нет команд явного указания маршрутизации
- С DV-протоколом определение маршрутов динамическое, при изменении коннективности с помощью сигнальных пакетов устанавливаются новые маршруты
- Сигнальные пакеты уникастные
- Изменение коннективности задается командами
  ```
  $ns rtmodel-at 1.0 down $n1 $n4
  $ns rtmodel-at 4.5 up $n1 $n4
  ```
- Можно также менять коннективность "на регулярной основе": модели экспоненциальная, детерминированная и с данными из файла
- Отказы узлов также можно симулировать

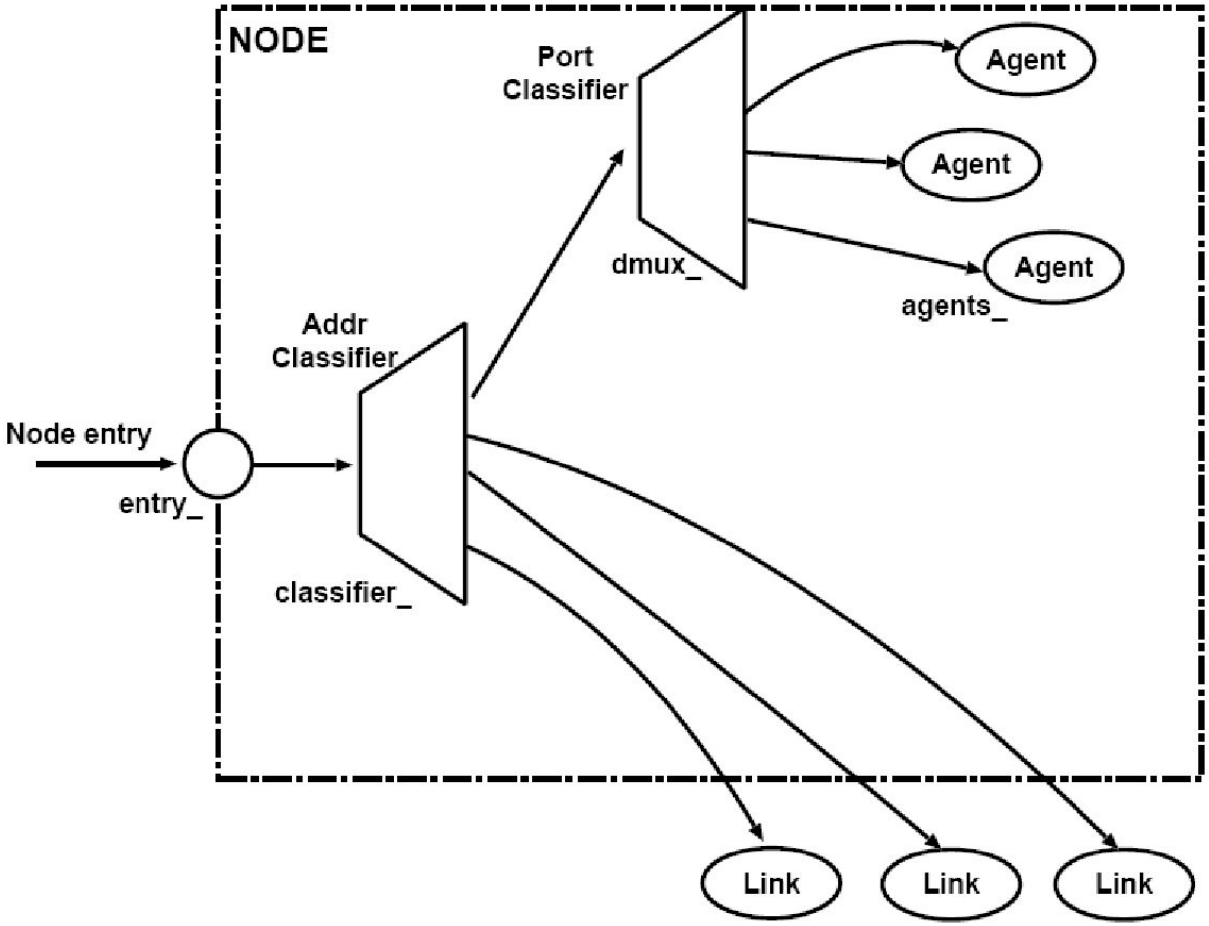
Мультикастные протоколы маршрутизации

- Участники группируются в мультикастные группы; один участник может входить в несколько групп.
- Принимают только члены групп, которым принимаемые сообщения адресованы; передавать можно без присоединения к группам.
- При симуляции в ns, нужно вначале задать мультикастную модель адресных классифайеров:
  $ns multicast
- Выделение мультикастного адреса
  set group1 [Node allocaddr]

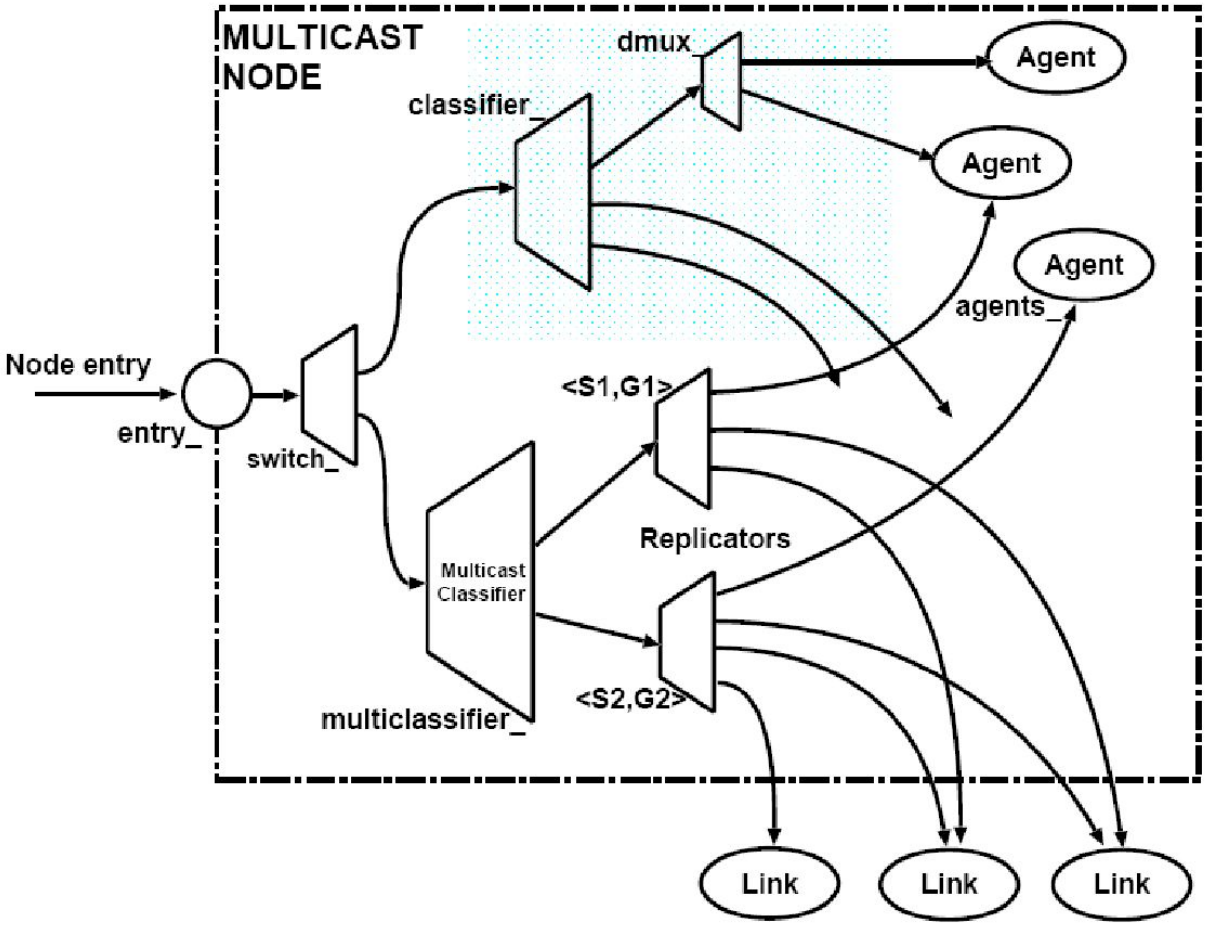
DM-протокол

- Dense mode, "плотный" режим
- Вначале сеть "наводняется" пакетами, а затем по уникастным таблицам вычисляется обратный маршрут

Уникастный узел в NS



Мультикастный узел в NS



Мультикастная группа

- Чтобы получать определенный поток, хост должен присоединиться к мультикастной группе , используя для этого протокол Internet Group Multicast Protocol, IGMP.
- Мультикастные адреса служат для спецификации некоторой группы IP-хостов, присоединившихся к группе и желающих получать поток.
- IANA выделила группу D: 224.0.0.0 по 239.255.255.255
  - 224.0.0.0 по 224.0.0.255 используются сетевыми протоколами на локальном сегменте. Маршрутизаторы никогда не транслируют пакеты для этих адресов за пределы группы
  - 239.0.0.0 по 239.255.255.255 может выделять организация для своих мультикастных потоков – локальный scope.
- 224.0.1.0 по 238.255.255.255 – глобальный scope. Некоторые адреса зарезервированы IANA. 224.0.1.1 используется в Network Time Protocol.
- Каждая AS получает группу мультикастных адресов из области 233.0.0.0/8 с номером AS во втором и третьем октетах.
- Шестнадцатеричный номер AS 62010 = F23A. F2=242; 3A=58. AS 62010 получает группу адресов 233.242.58.0 в свое распоряжение.

Мультикастные адреса на уровне DLC

- NIC на LAN-сегменте принимает пакеты только для своего MAC-адреса или для широковещательного адреса. Как принимать мультикастные пакеты?
- Бит 0 первого октета определяет бродкаст/мультикаст
- MAC-адреса с 0100.5e00.0000 по 0100.5eff.ffff принадлежат IANA. Адреса с 0100.5e00.0000 по 0100.5e7f.ffff выделены под мультикаст.
- Нижние 23 бита IP-мультикастного адреса транслируются в MAC-адрес. Полный IP -мультикастный адрес содержит 28 бит; 32 мультикастных группы транслируются на один MAC-адрес.

Internet Group Management Protocol

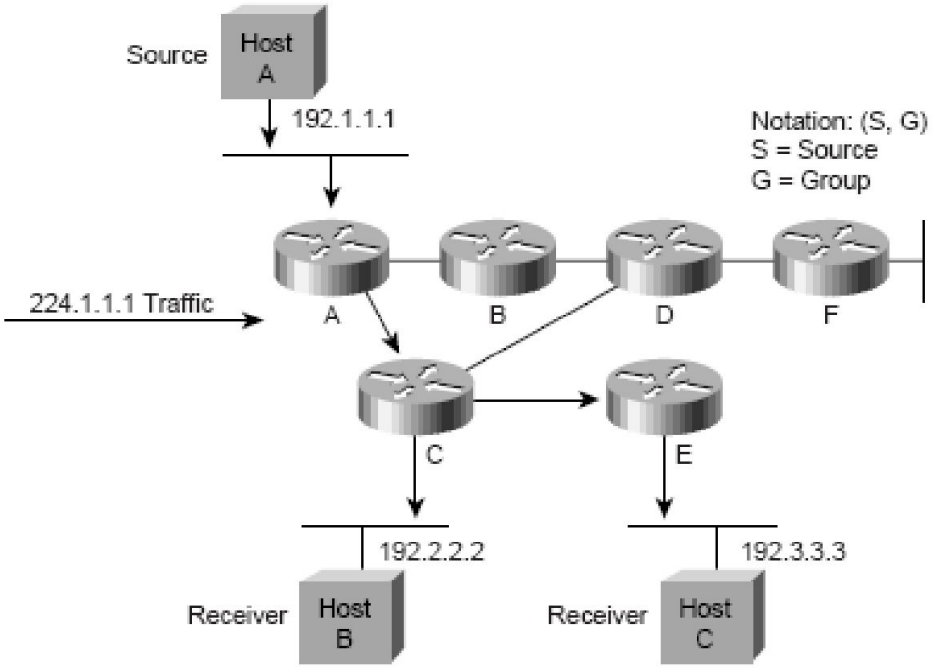
- Версия 1: два типа IGMP-сообщений
  - Запрос об участии в мультикастной группе
  - Отчет об участии в мультикастной группе
- Маршрутизатор периодически шлет запрос об участии в мультикастной группе. Если нет ответа на три последовательных запроса, прекращается трансляция мультикаст-потока в этот сегмент.
- Версия 2: четыре типа IGMP-сообщений
  - Запрос об участии в мультикастной группе
  - Отчет об участии в мультикастной группе по версии 1
  - Отчет об участии в мультикастной группе по версии 2
  - Сообщение об уходе из группы

Мультикаст в сети со switch-ами на уровне 2

- Switch просматривает информацию уровня 3 в пакетах IGMP (snooping). По этой информации switch добавляет/удаляет номер соответствующего порта в/из мультикастной таблицы.
- Сами IGMP-сообщения – это мультикастные сообщения, они неотличимы от мультикаста на уровне 2. Поэтому просматриваются ВСЕ мультикастные пакеты – нагрузка на switch. Для маломощных switch-ей CISCO разработало специальный протокол.

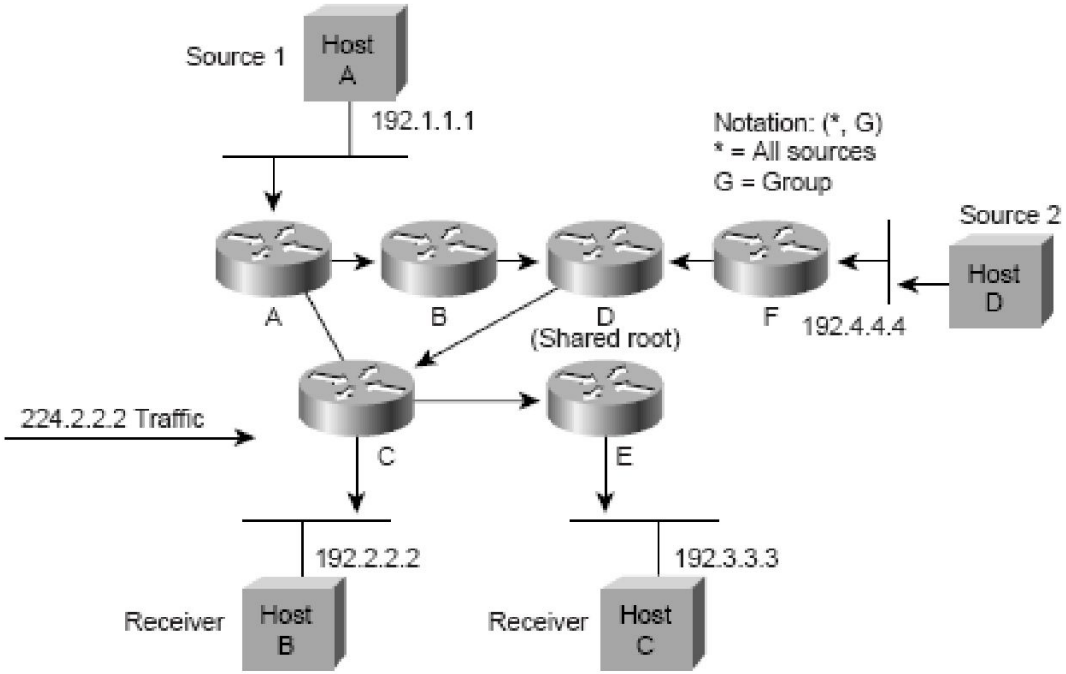
Деревья распределения мультикаста

- Source trees: источник мультикаста – корень дерева.
- Используется кратчайший путь, дерево называется shortest path tree (SPT)
- Нотация (S,G): в данном случае (192.1.1.1, 224.1.1.1)
- Если хост B источник: (192.2.2.2, 224.1.1.1)
- Оптимальные маршруты
- Маршрутизаторы должны хранить маршруты для каждого источника
- Хосты присоединяются к группам и покидают группы: деревья динамически обновляются



Деревья распределения мультикаста

- Shared trees используют общий корень дерева, который называется rendezvous point (RP)
- Нотация (*,G): в данном случае (*, 224.2.2.2)
- И SPT, и shared trees не образуют циклов. Сообщения реплицируются только в точках ветвления.
- В маршрутизаторах хранится маршрут только из RP.
- Хосты присоединяются к группам и покидают группы: дерево динамически обновляется

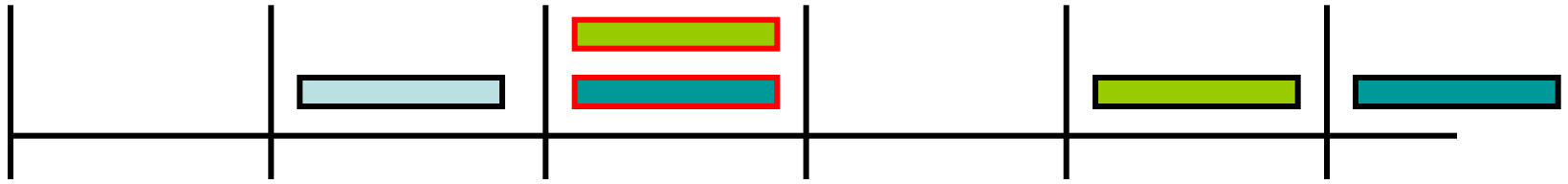


Multiple Access

- Примеры систем с общей средой передачи данных: ethernet с концентратором (hub), беспроводные сети, спутниковые каналы
- Как распределять ресурсы канала между пользователями: потребность в канале a priori неизвестна
- Switches в проводном ethernet позволяют заменить проблемы multiple access проблемами коммутации и маршрутизации.
- Time Division MA, Frequency Division MA, Code Division MA: каждый узел получает фиксированную часть канала (a fixed fraction of bandwidth). Эквивалентно технике circuit switching.
- Если позволить узлам доступ к каналу в любое время на любой частоте с любым кодом, возникает соперничество за ресурс канала (contention). Это соперничество проявляется в виде столкновений (collisions).
- Протоколы управления соперничеством и разрешением трудовых споров – столкновений:
  - polling;
  - резервирование канала и координация действий узлов;
  - использование результатов теории вероятностей, теории игр и теории очередей в планировании случайного доступа

Aloha

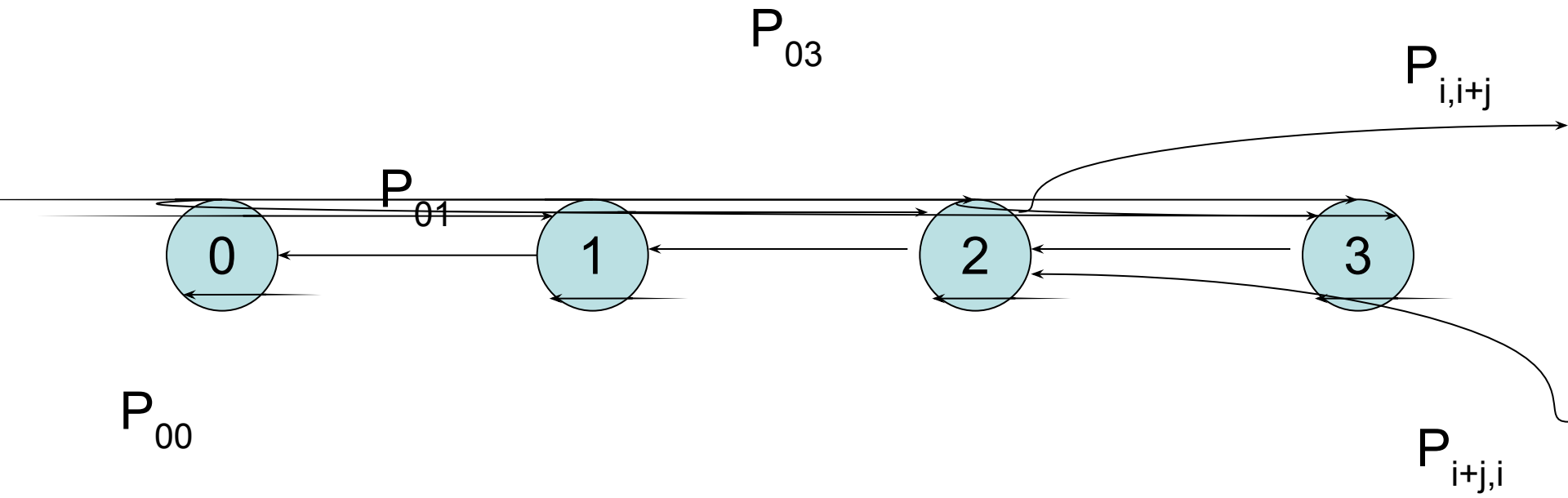
- Достижение максимальной эффективности передачи данных ОДНОМУ приемнику от НЕСКОЛЬКИХ передатчиков
- Алоха с временнЫми слотами:
  - пакеты фиксированного размера;
  - передача пакета начинается от начала слота => необходима синхронизация передатчиков с часами канала
  - передача в слоте более одного пакета => столкновение, необходимо повторить передачу для КАЖДОГО столкнувшегося пакета после случайной задержки



Алоха II

- Модель:
- Пуассоновское распределение времени появления пакетов
- Все пакеты, участвующие в столкновении, теряются, но! (capture model)
- Узел НЕМЕДЛЕННО узнает о состоянии текущего слота: idle (0), success(1), collision (e).
- Появившийся из более высокого слоя пакет передается в НЕПОСРЕДСТВЕННО СЛЕДУЮЩЕМ слоте
- Если передача от узла претерпела столкновение, узел переводится в состояние backlogged (завал), и начинается передача пакета с определенной вероятностью q в каждом следующем слоте, пока не удастся передать пакет.
- Пребывание в состоянии backlogged не увеличивает вероятности появления нового пакета для передачи в данном узле: очередь в узлах = 1.

Марковская цепь для Алохи со слотами

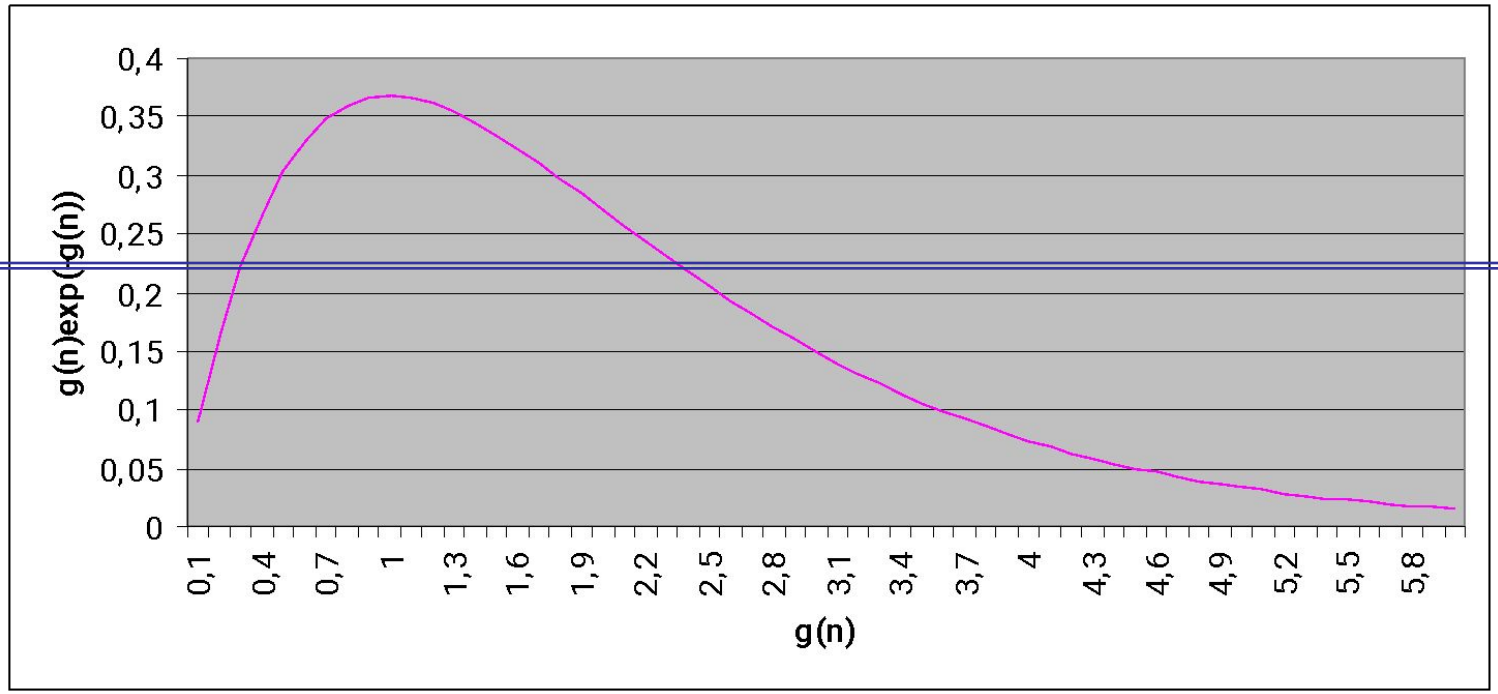


- N – кол-во узлов в состоянии backlogged
- Система нестабильна, N $\rightarrow\infty$

Алоха III

- g(n) – среднее кол-во пакетов, передаваемых в слоте, в состоянии n (n backlogged узлов)
  $g(n) = \lambda + nq$
- Распределение по кол-ву пакетов передаваемых в слоте также Пуассоновское со средним g(n)
- P(m попыток) = $g(n)^m \cdot \exp(-g(n))/m!$
- P(нет попыток) = $\exp(-g(n))$
- P(успешная передача) = $g(n) \cdot \exp(-g(n))$
- P(есть столкновение) = 1 – P(нет попыток) – P(успешная передача)

Производительность Алохи



- Производительность = доле слотов с успешной передачей; при стабильной работе, Производительность = скорости поступления данных = λ
- Максимум при g(n) = 1
- Максимальная производительность = 1/e ≈ 0,36 пакетов/слот
- Дрейф в состоянии n = ожидание изменения кол-ва слотов в состоянии backlog за один слот = λ – g(n)·exp(–g(n))

Стабилизация неустойчивости

- Если выбирать q на основании оценки кол-ва слотов в состоянии backlog для достижения максимальной производительности, наступает также стабилизация
- Если развивается нестабильность и передачи всех вновь прибывших пакетов претерпевают столкновения, $g(n) = nq$ и $P(\text{успех}) = g(n)\cdot\exp(-g(n)) = nq(1-q)^{n-1}$
- P(успех) максимальна при q=1/n (= 1, если n=0)
- В стабильном состоянии вероятность успеха = 1/e, вероятность холостого слота такая же, вероятность столкновения 1-2/e
- Узел оценивает кол-во узлов в backlog-е увеличением оценки при столкновении и уменьшением оценки при успешной передаче или в холостом слоте
- Алгоритм
  - новая оценка кол-ва узлов в бэклоге $n_{k+1} = n_k + \lambda + 1/(e-2)$ при столкновении и $\max\{\lambda, n_k + \lambda - 1\}$ в остальным случаях
  - $q = 1/n_{k+1}$
- гарантирует стабильность протокола при $\lambda < 1/e$

Алоха без слотов

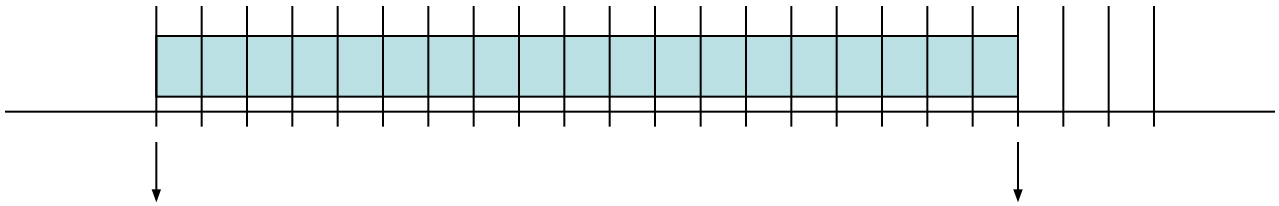
- Попытка передачи успешна, если с обеих сторон на протяжении длины пакета не было других попыток передачи
- $P(\text{успех}) = \exp(-g(n)) \cdot \exp(-g(n)) = \exp(-2g(n))$
- Пропускная способность $g(n) \cdot \exp(-2g(n))$
- Максимум при $g(n)=1/2 \approx 0.18$
- Не нужна синхронизация, пакеты не обязаны быть одинаковыми

CSMA

- "Вежливая" версия Алохи: передача начинается, если канал свободен
- Если канал занят, узел переходит в состояние backlogged
- При освобождении канала, backlogged узел начинает передачу с вероятностью q
- q=1 – настойчивый алгоритм
- q<1 – ненастойчивый алгоритм

CSMA

- τ – максимальная задержка распространения сигнала по каналу
- Введем дискретизацию по времени для лучшего уяснения принципов CSMA. Т.е., рассмотрим вариант CSMA со слотами. На практике таких реализаций не существует.
- Размер слота – τ, совпадает с максимальной задержкой распространения по каналу
- Временная продолжительность пакета T_p, продолжительность слота τ. Введем параметр $\beta = \tau / T_p$, и будем измерять время в единицах продолжительности пакета.



Алгоритм CSMA со слотами

- При появлении нового пакета с верхнего уровня
  - Если слот холостой, в следующем слоте начать передачу
  - Если слот занят, узел переходит в состояние backlogged
  - Если, в ходе передачи, возникает коллизия, узел также переходит в состояние backlogged
- Узлы в состоянии backlogged предпринимают попытку передачи только дождавшись холостого слота (после него). В ненастойчивом режиме, попытка предпринимается с вероятностью $q<1$.
  - За каждым периодом активности (передача или коллизия) следует по крайней мере один слот
- Рабочее время разбивается на эпохи:
  - Успешная передача пакета, за которой следует обязательный холостой слот, продолжительность эпохи = $1+\beta$
  - Коллизия, за которой следует обязательный холостой слот, продолжительность эпохи = $1+\beta$
  - Холостой слот сам по себе, продолжительность эпохи = β

Анализ CSMA

- Состояние системы – кол-во узлов в стадии backlogged
- Примем, что моменты перехода из одного состояния в другое совпадают с окончаниями холостых слотов
- Математическое ожидание продолжительности времени между переходами из состояния *n* в другое состояние:
- $T(n) = \beta + (1 - (1-q)^n \cdot \exp(-\lambda\beta))$

Анализ CSMA

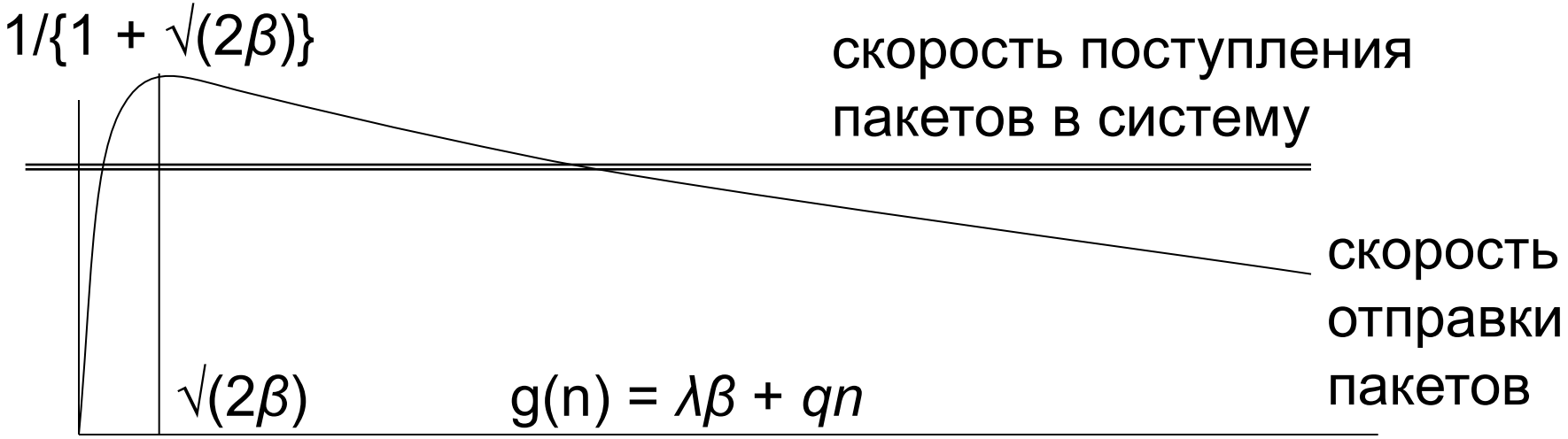
- Состояние системы – кол-во узлов в стадии backlogged
- Примем, что моменты перехода из одного состояния в другое совпадают с окончаниями холостых слотов
- Математическое ожидание продолжительности времени между переходами из состояния *n* в другое состояние:
  $T(n) = \beta + (1 - (1-q)^n \cdot \exp(-\lambda\beta))$
- Для очень "ненастойчивого" ($q \ll 1$) алгоритма, $(1-q)^n \approx \exp(-qn)$, следовательно:
  $T(n) = \beta + (1 - \exp(-\lambda\beta - qn))$
- При этом: в начале каждой эпохи каждый backlogged узел порывается, с вероятностью q, начать передачу; также с вероятностью единица предпринимается попытка передачи пакетов, появившихся из верхнего слоя в течение предыдущего холостого слота.
- В состоянии n количество пакетов, попытка передачи которых предпринимается, следует приблизительно распределению Пуассона с параметром $g(n) = \lambda\beta + qn$

Анализ CSMA

- Вероятность успешной передачи за эпоху составляет
  $P(\text{успех}) = g(n) \cdot \exp(-g(n))$
- Математическое ожидание продолжительности эпохи
  $T(n) = \beta + (1 - \exp(-g(n))$
- Вероятность успешной передачи в единицу времени
  $g(n) \cdot \exp(-g(n)) / \{\beta + (1 - \exp(-g(n))\}$
  и равна устоявшейся скорости отправки пакетов, которая должна по замыслу превышать λ.

Максимальная пропускная способность CSMA

- Оптимальное значение g(n) получается опять же нахождением экстремума и составляет
  $g(n) \approx \sqrt{(2\beta)}$; $\lambda < 1/\{1 + \sqrt{(2\beta)}\}$
- Производительность тем выше, чем меньше β : время распространения сигнала в канале сдерживает пропускную способность канала.
- Проблема стабильности аналогична стабильности Алохи, но менее критична из-за малости β .

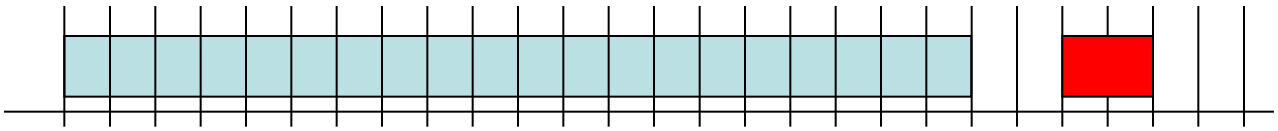


Реализация CSMA на практике

- Из-за отсутствия слотов, чуть ниже пропускная способность, выраженная через β (сравните с Алохой), но и значение β меньше, так как фактически имеем дело с усредненным временем распространения.

CSMA/CD

- При обнаружении коллизии, узел НЕМЕДЛЕННО прекращает передачу
- Протокол:
  - Все узлы слушают канал
  - Поступает пакет с верхнего уровня:
    - Канал свободен, узел начинает передачу
    - Канал занят – двоичный экспоненциальный бэкофф
  - Если во время передачи узел обнаруживает коллизию, передача прекращается немедленно
    - Узел входит в двоичный экспоненциальный бэкофф
- Задержка распространения τ. Чтобы коллизия была обнаружена всеми узлами, может потребоваться время распространения из конца в конец и обратно = $2 \cdot \tau$. "Воображаемые" слоты продолжительности $2 \cdot \tau$.



Анализ CSMA/CD

- N пользователей, каждый пытается начать передачу в свободном слоте с вероятностью p.
- Рассчитываемая вероятность включает и вновь прибывшие пакеты, и ретрансмиссии.
- P(i узлов пытаются) = $C_i^N P^i (1–P)^{N–i}$
- P(успешная передача) = $N \cdot P(1–P)$
- Максимум при P=1/N (ср. теорема Литтла) – как в Алохе
- P(успешная передача) (при $N \to \infty$) $\to 1/e$
- Среднее кол-во слотов, за которое произойдет успешная передача = *e*.
- Если слот захвачен, передача далее идет без помех

Анализ CSMA/CD

- Среднее значение интервала времени между успешными передачами:
  T_S = <холостые/коллизии> +
  <продолжительность пакета> +
  <среднее время до начала следующего слота>
  = (e–1)·2·τ + T_P + τ
- Эффективность = T_P / T_S
  = 1/{1+β·(2e–1)} = 1/(1+4.4β); λ < 1/(1+4.4β);
  CSMA без CD: λ < 1/{1 + $\sqrt{(2\beta)}$}
- 10 Mb/s, 1000 бит, T_P = 10^{-4}сек; 1500 м;
  t=5·10^{-6} сек; β = 5·10^{-2}; Эффективность = 80%