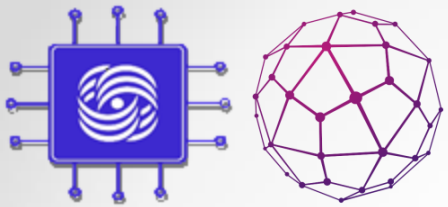
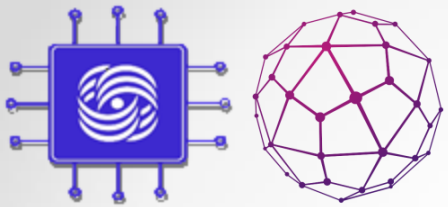


Внутреннее устройство коммутаторов -- механизмы обеспечения качества сервиса

Доп. главы Компьютерных сетей и телекоммуникации

к.ф.-м.н. Чемерицкий Е.В.



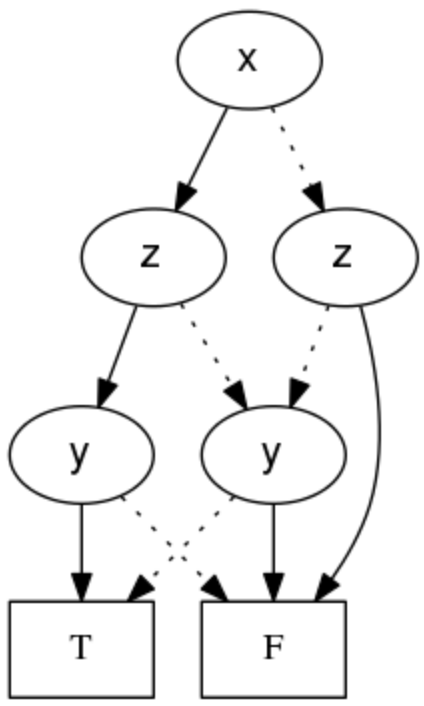
Рефераты!

- 49 человек зарегистрировано в HotCRP
- 23 реферата сохранено в системе
- 12 человек ни разу не выполнили вход

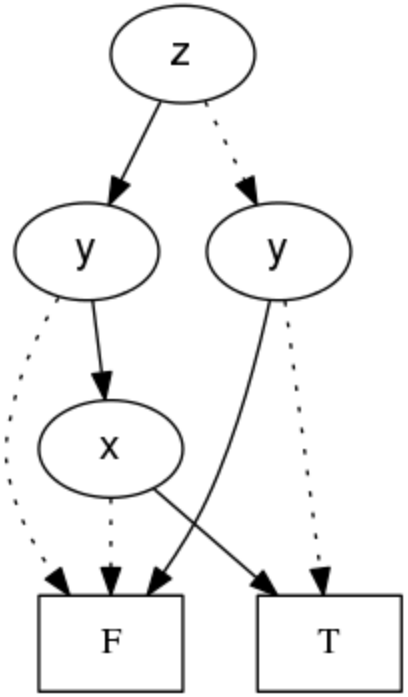
- Сегодня 20 ноября – если сдать реферат до полуночи, его максимальная оценка составит ¾ от изначально возможной

Результаты проверки

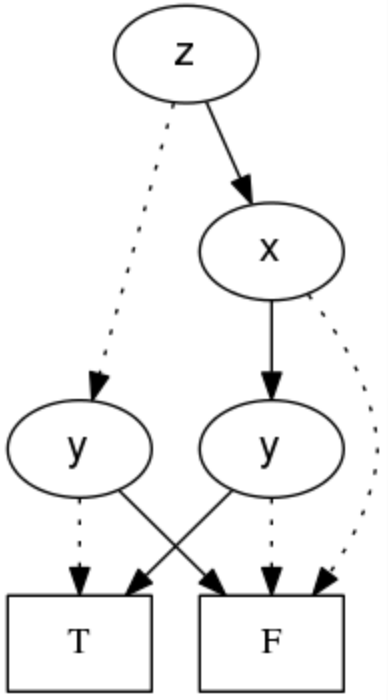
$$xyz \vee \bar{y}\bar{z}$$



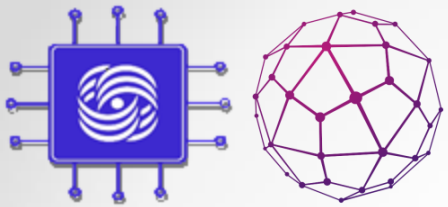
$x\ z\ y$



$z\ y\ x$

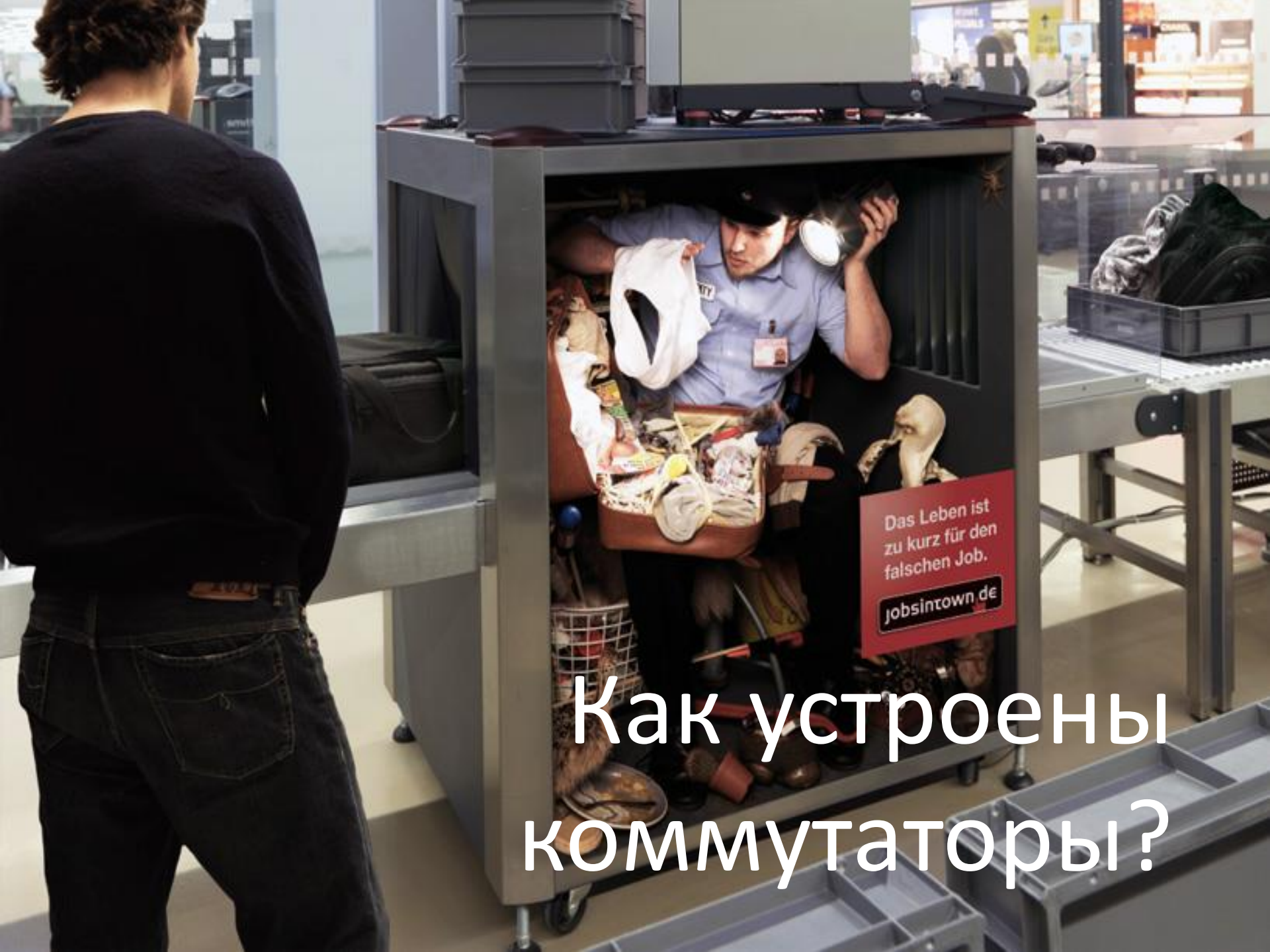


$z\ x\ y$

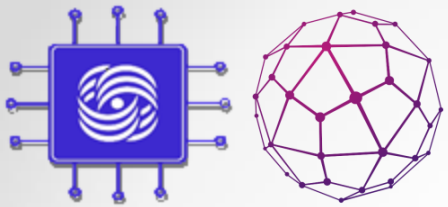


План лекции

- Варианты компановки коммутатора
- Работа коммутационной матрицы
- Внутреннее устройство буферов пакетов

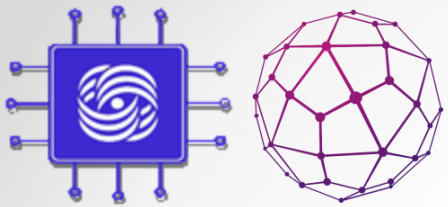


Как устроены коммутаторы?



Классификация коммутаторов по поколениям (1)

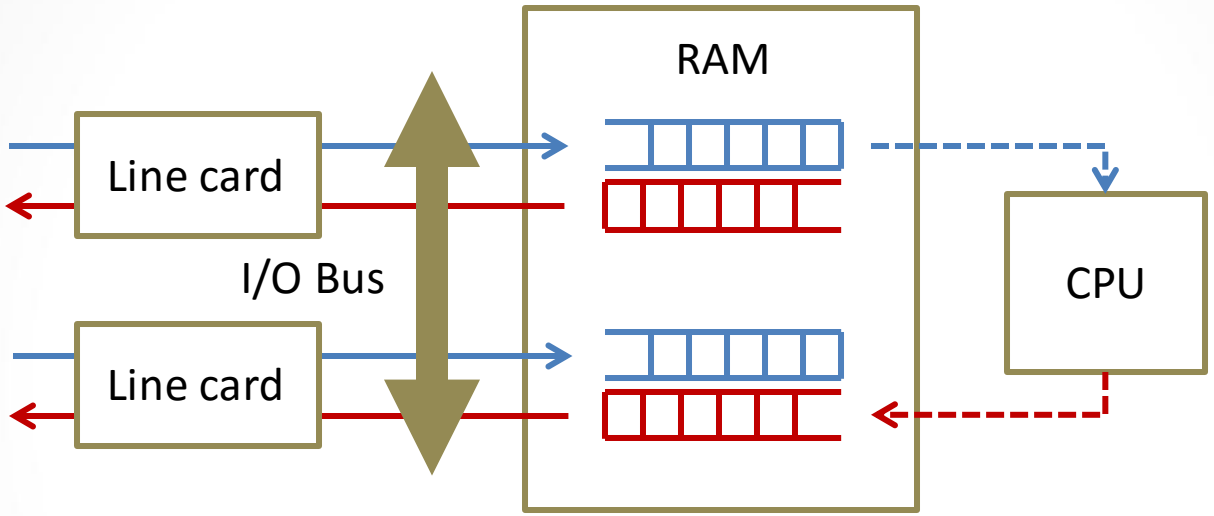
- Поколения отражают достигнутые характеристики производительности, а не коренные изменение технологии
- Эволюция достигается за счёт изменения баланса между стоимостью и сложностью коммутационного устройства
- Каждое из поколений заняло свою нишу и продолжает использоваться сегодня



Классификация коммутаторов по поколениям (2)

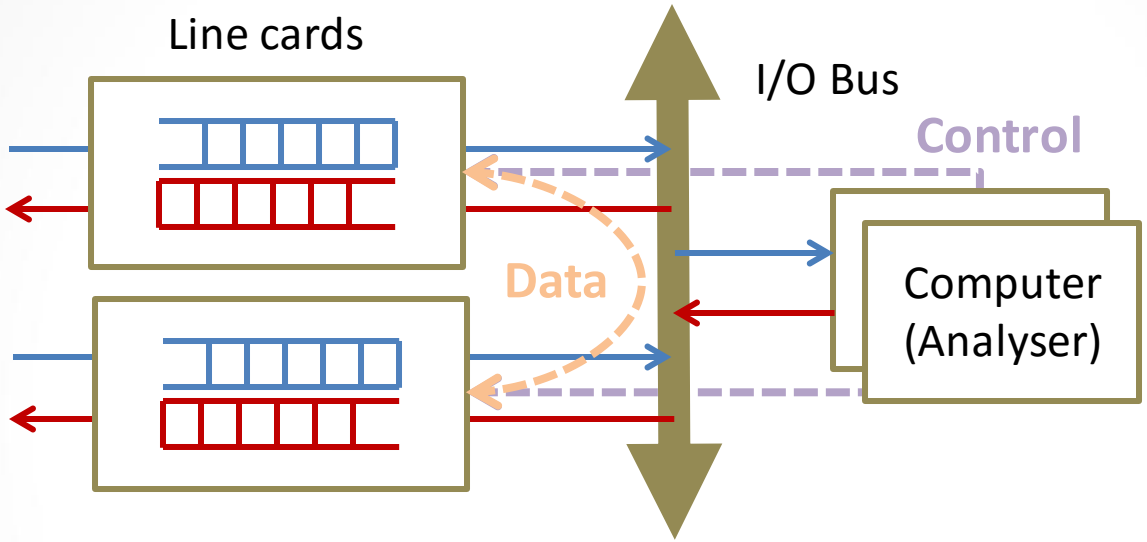
1. Интерфейсные Процессоры Сообщений – SISD компьютеры с несколькими сетевыми интерфейсами *[см. William Yeager]*
2. Распределённая MIMD архитектура с собственными контроллерами на интерфейсах
3. Переход от архитектуры с единой шиной передачи данных к коммутационным матрицам

Первое поколение коммутаторов

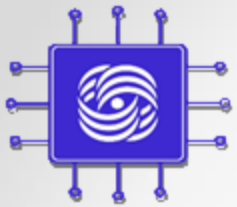


- Большинство домашних Ethernet коммутаторов и маршрутизаторов
- Bottleneck'ом может быть шина данных или процессор – в зависимости от типа трафика и производительности этих компонентов

Второе поколение коммутаторов



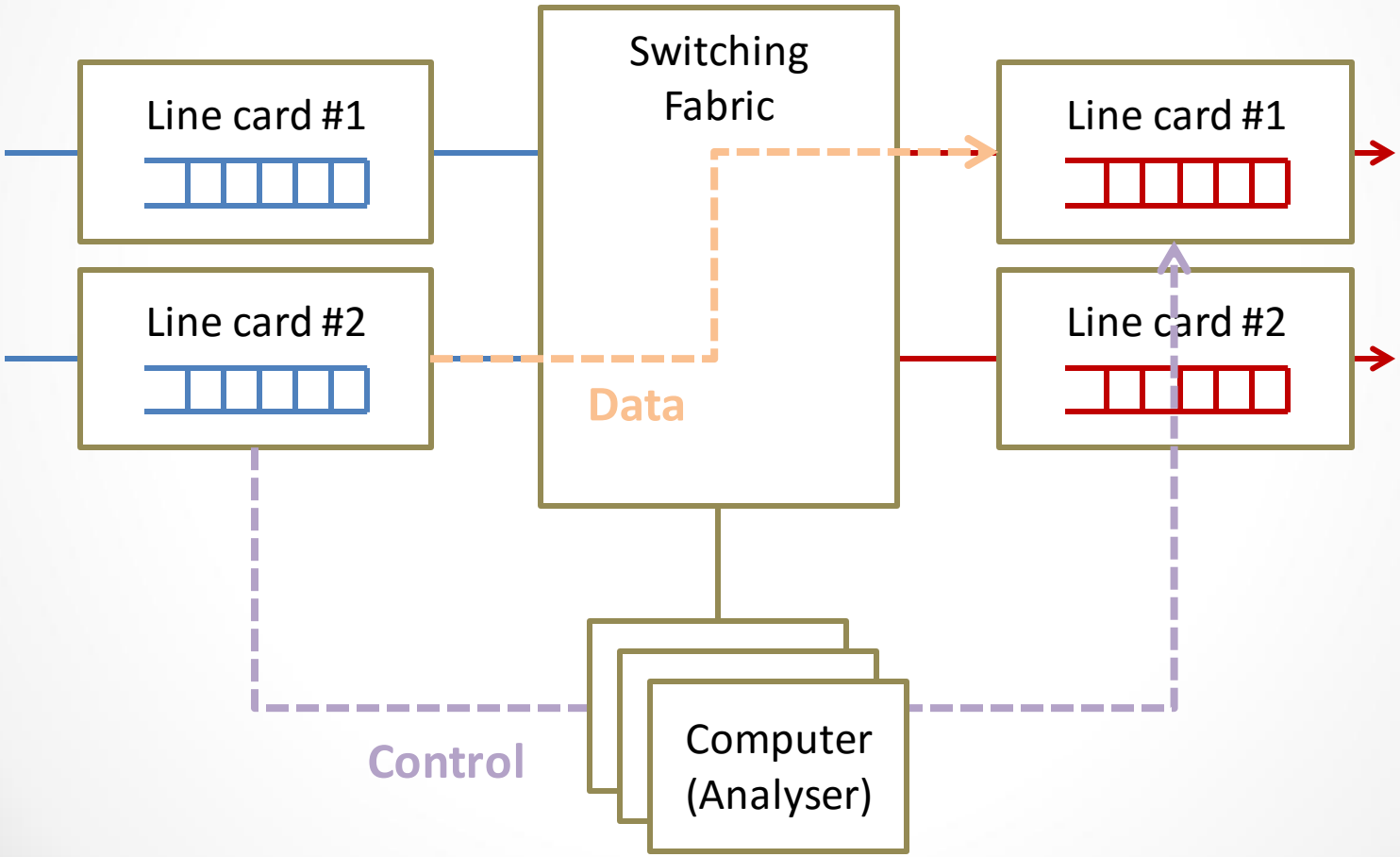
- Умные сетевые интерфейсы с собственным контроллером и встроенной памятью
- Данные пересылаются между картами напрямую, минуя оперативную память
- Слабое место – общая шина передачи данных

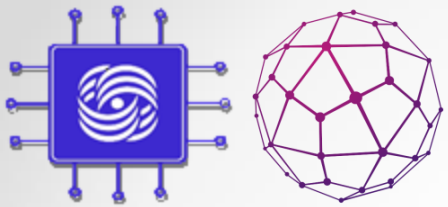



К какому поколению относятся решения в духе NFV?

- Самые современные реализации программных коммутаторов и некоторых ***виртуальных сетевых функций*** (VNF) – коммутаторы второго поколения
- Анализ пакетов занимает больше времени, чем их передача между интерфейсами
- Пример: NVWare – CGNAT с пропускной способностью до 120 Gbit/sec

Третье поколение коммутаторов





Третье поколение коммутаторов

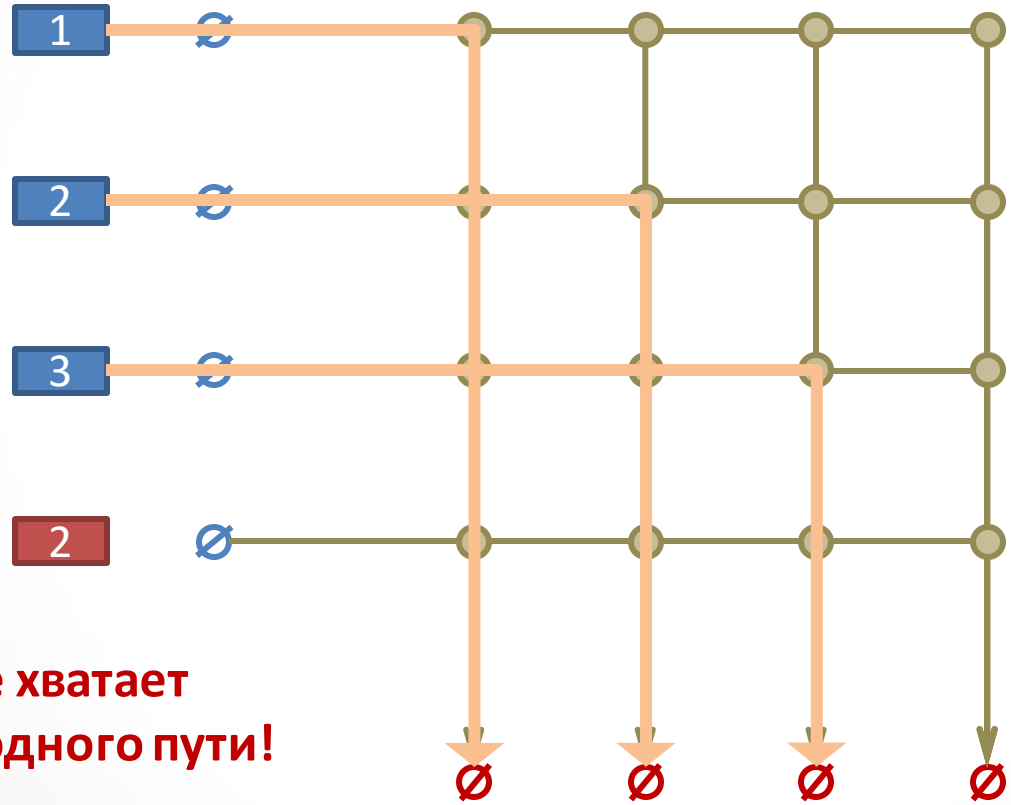
Если во втором поколении для передачи пакетов используется общая шина, то в третьем – ***коммутационная матрица***

Коммутационная матрица способна передавать между N интерфейсами сразу несколько пакетов одновременно (имеет K transmission planes):

- Для шины передачи данных $K = 1$
- Матрица переключателей $N \times N$: $1 \leq k \leq N$ (в зависимости от нагрузки)
- Матрица переключателей $N \times N^2$: $\mathrm{k} = N$ (вне зависимости от нагрузки)

Crossbar Switching Fabric

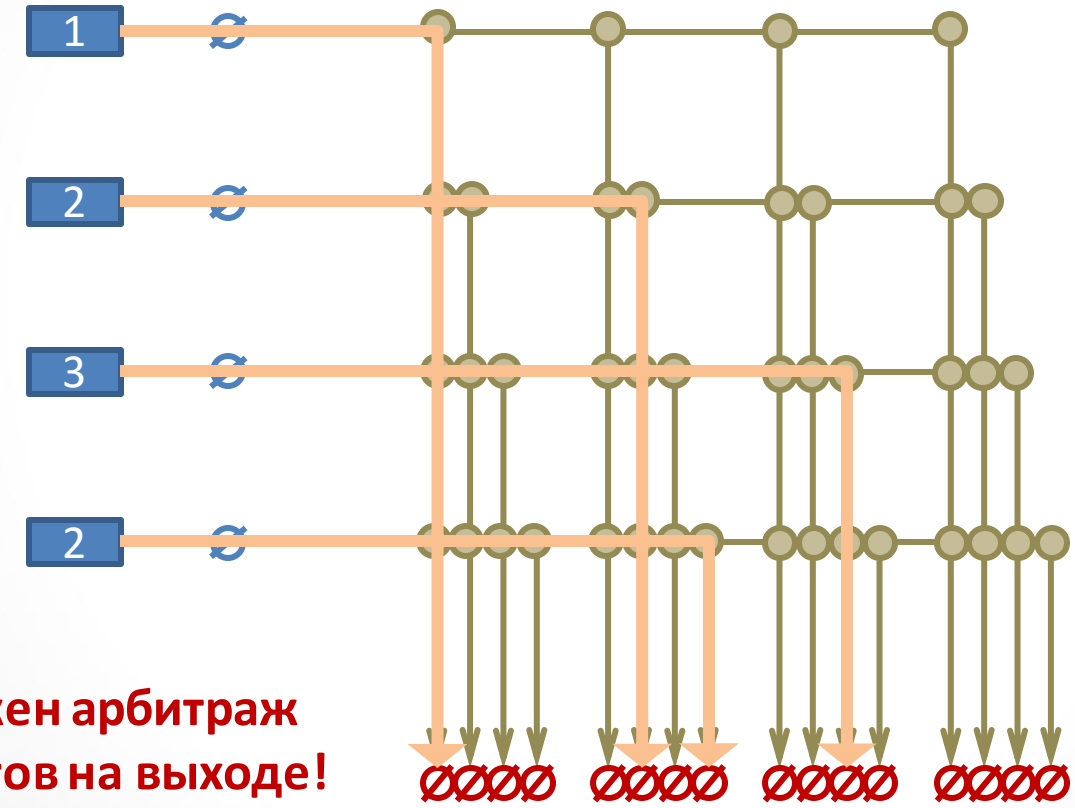
N^2 переключателей



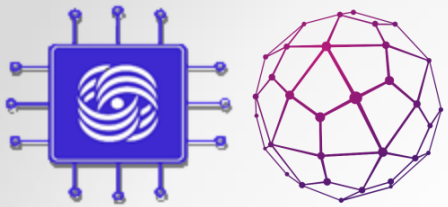
Не хватает свободного пути!

Crossbar Switching Fabric

N^3 переключателей



Нужен арбитраж пакетов на выходе!



Режимы коммутации

Δ_f -- задержка коммутации пакета (FIFO)

Δ_s -- задержка сериализации пакета

Δ_h -- задержка обработки пакета (FIFO)

Store-and-Forward: $\Delta_f = \Delta_s + \Delta_h$

Обработка начинается после получения всего пакета

Работает в терминах пакетов

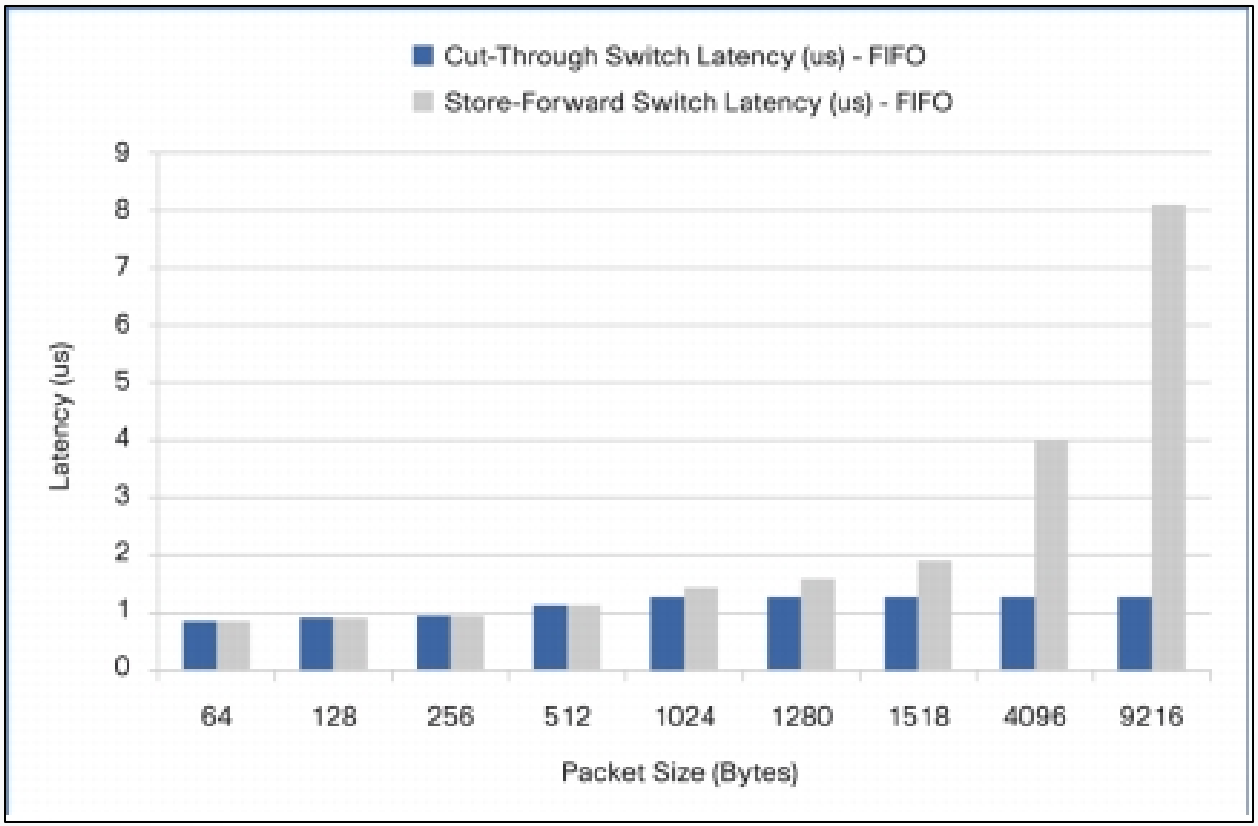
Cut-Trough [& Fragment free]: $\Delta_f = d + \Delta_h$

d – время сериализации битов заголовка

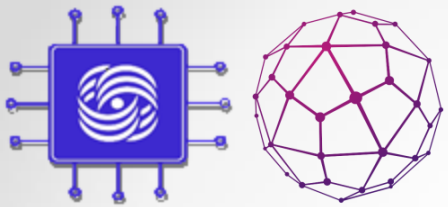
Обработка начинается сразу после получения *достаточного количества* битов заголовка пакета

Работает в терминах ***ячеек*** фиксированной длины

Режимы коммутации

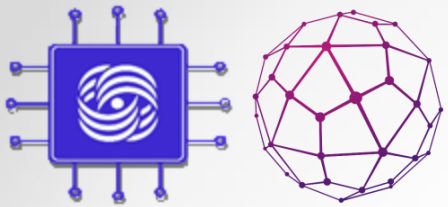


Store-and-forward vs. Cut-Trough на канале в 10 Gb



Зачем нужна буферизация?

- Линии связи не могут передавать сразу несколько пакетов одновременно
- Что делать если несколько пакетов нужно отправить через одну и ту же линию?
  - Сбросить один из пакетов
  - Сохранить пакет в буфер
- Классическая задача проектирования коммутатора – где расположить буферы, чтобы собрать наилучшее устройство:
  - Максимальная производительность
  - Хорошая масштабируемость
  - Минимальная стоимость

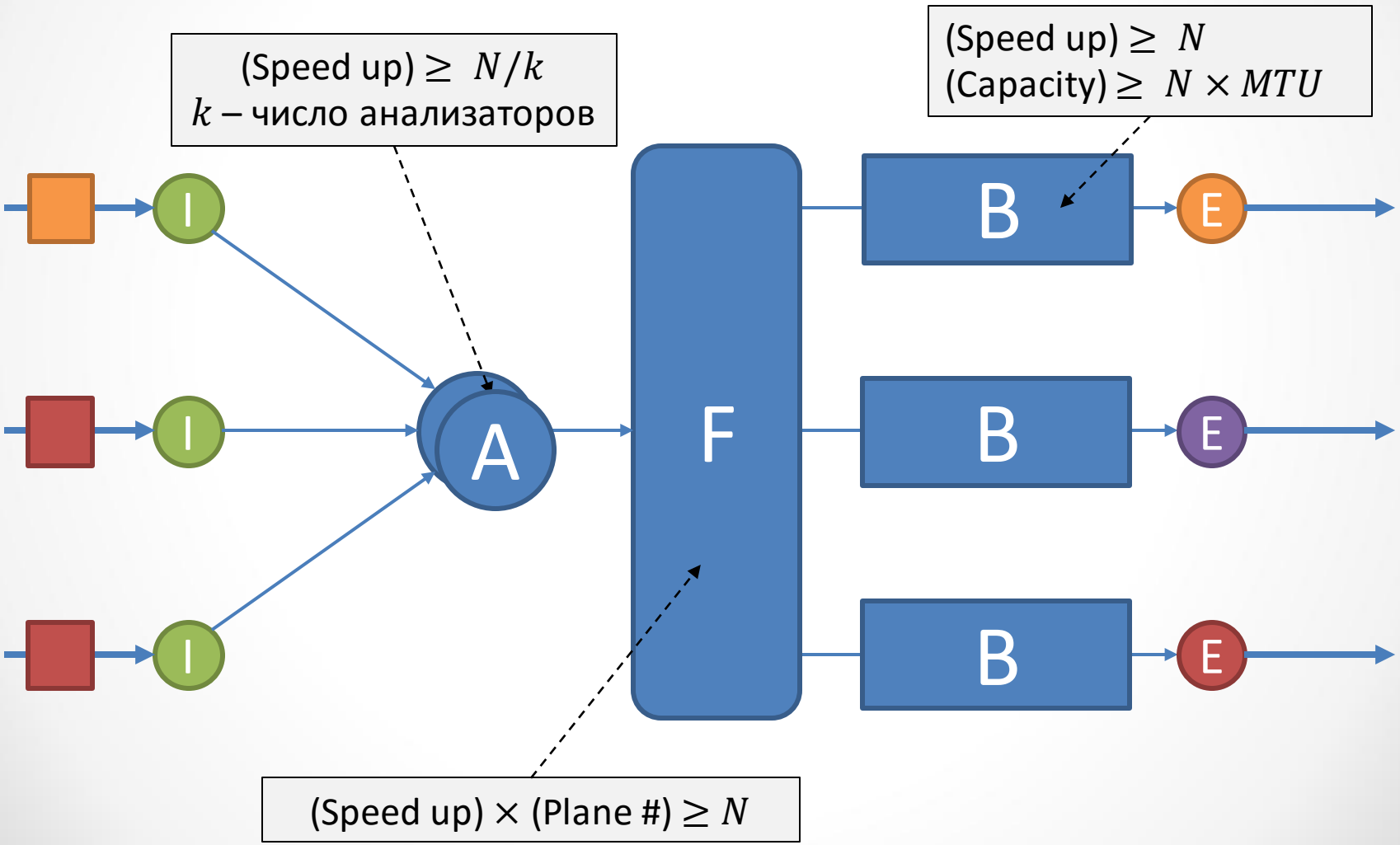


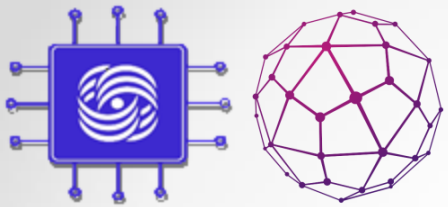
Измерение производительности коммутаторов

RFCs 2544 & 6815

- Если распределение трафика таково, что для каждого выходного порта суммарная скорость поступления данных, которые нужно через него передать, не превосходит скорости подключённой к нему линии связи, то распределение называется ***приемлемым для коммутатора (admissible)***
- Если коммутатор не сбрасывает пакеты при поступлении трафика с приемлемым для него распределением, то говорят, что этот коммутатор удовлетворяет требованиям ***full backplane***

Буферизация на выходе
Output Queuing

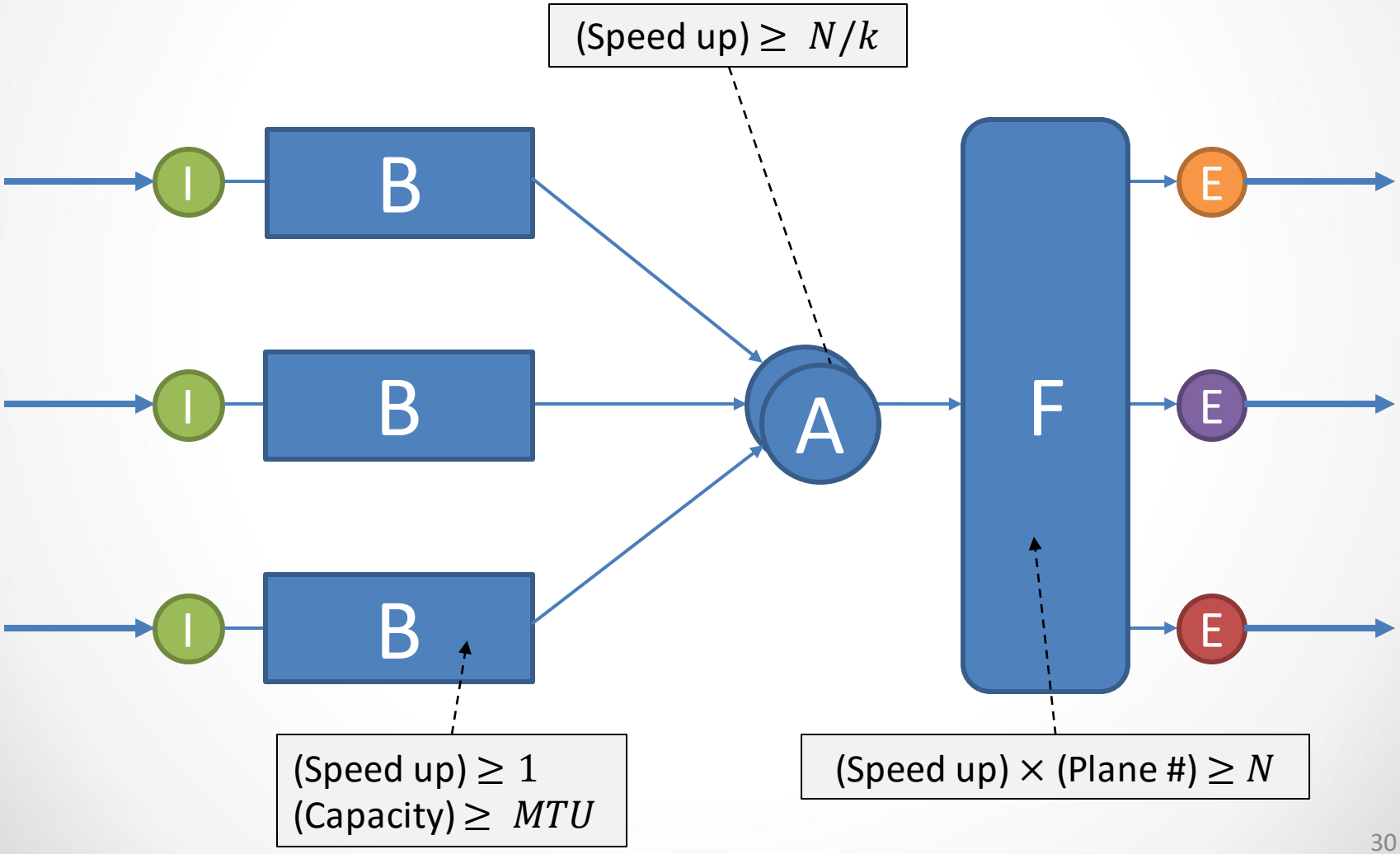




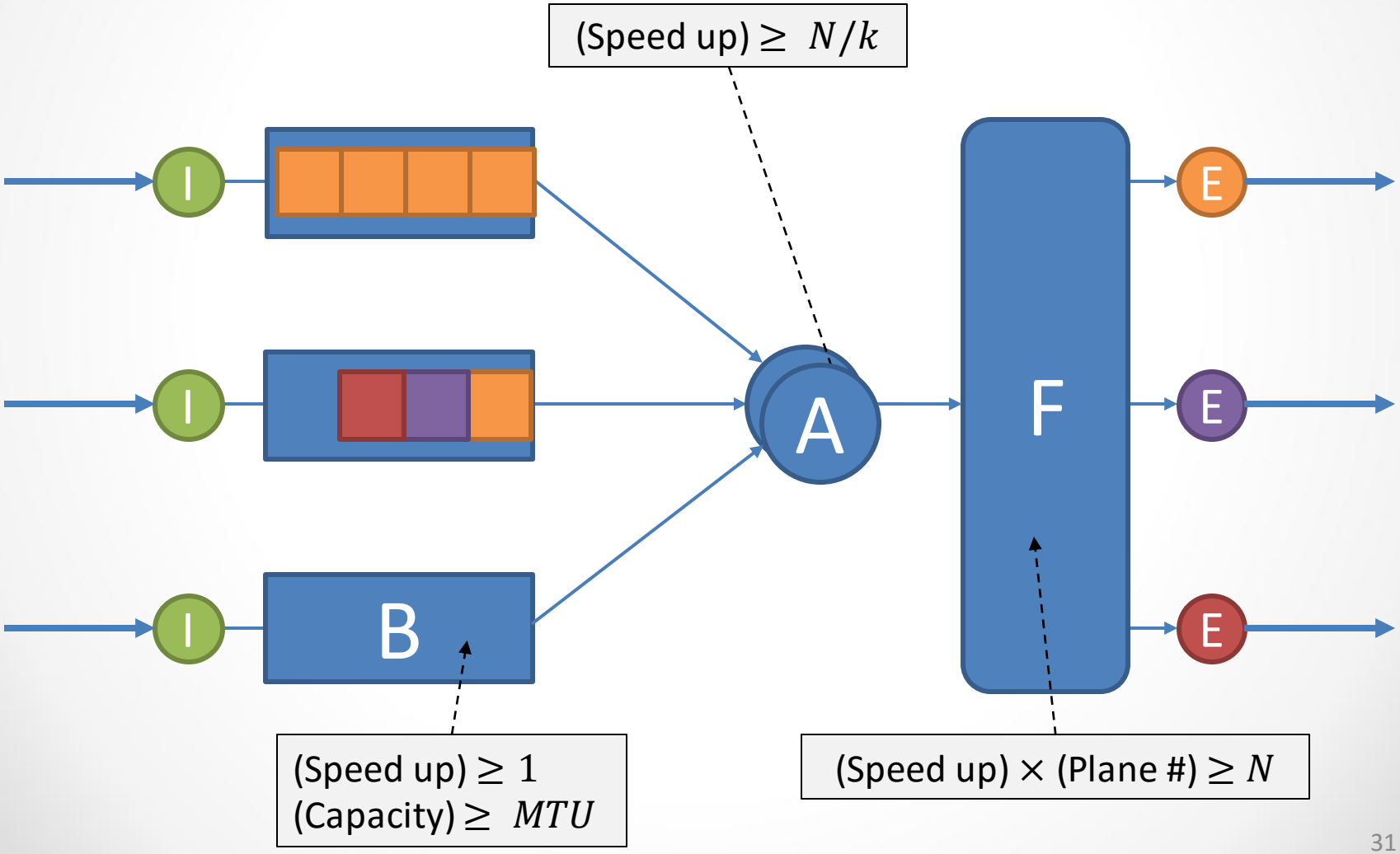
Производительность коммутатора

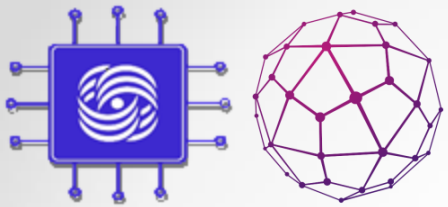
- Анализаторы должны работать с ускорением N/k, где k – количество анализаторов
- Скорость работы матрица должна превышать скорость линий связи в N раз
- Буферные блоки должны работать с ускорением (***speed up***) не менее N, их объём – составлять не менее N MTU
  - Каковы должна быть частота и ширина шины доступа к памяти, чтобы поддерживать работу современных коммутаторов?

Буферизация на входе
Input Queuing



Буферизация на входе
Input Queuing

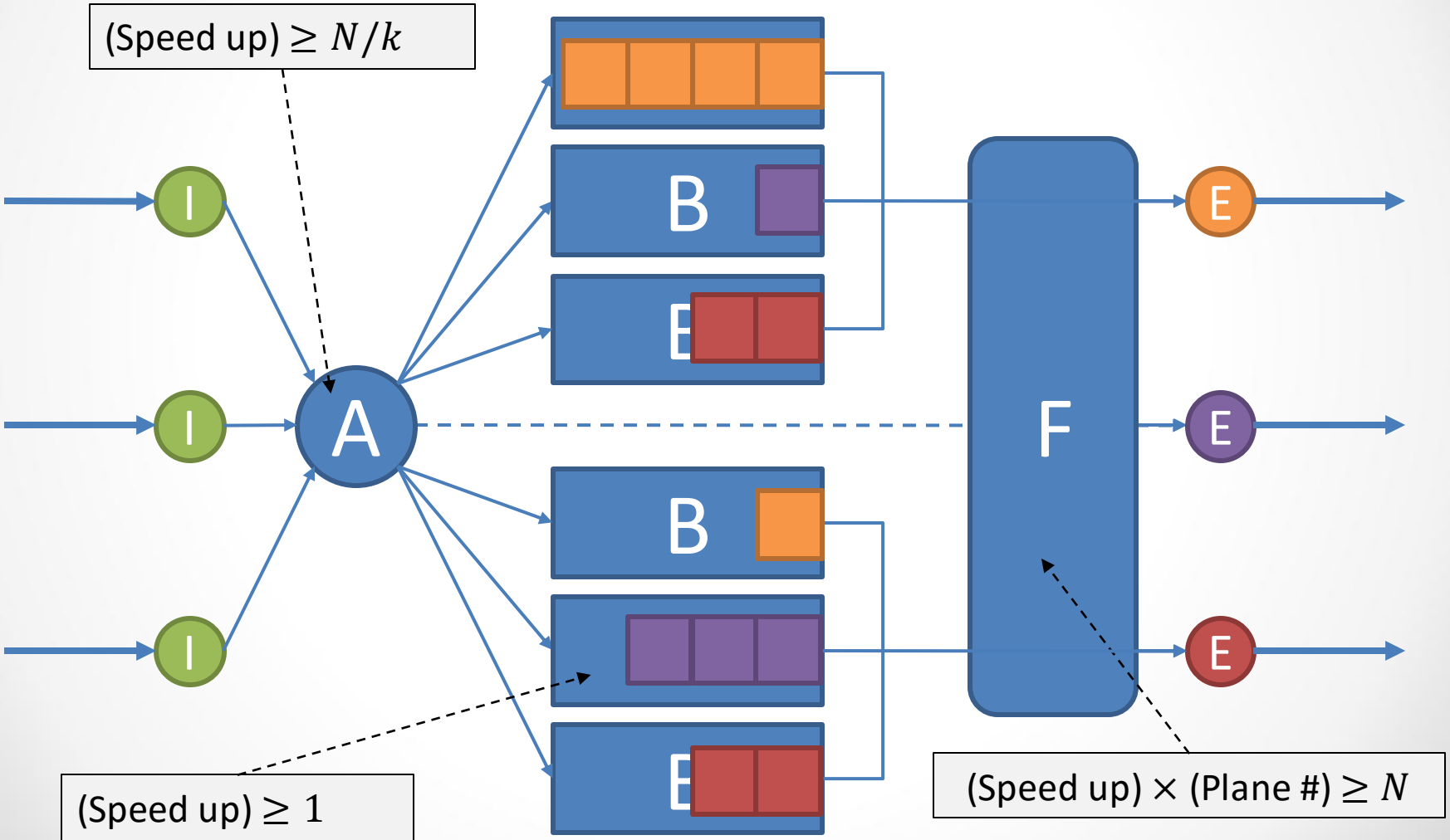


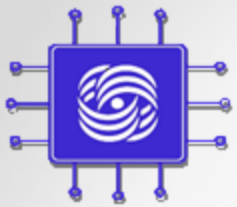


Буферизация на входе Input Queuing

- Нет необходимости в сверх-быстрой памяти
- Если пакеты из нескольких входных портов начинают конкурировать за один и тот же вход коммутационной фабрики, возникает блокировка пакетов, находящихся за ними – ***Head Of Line (HOL) Blocking***
- При равномерном распределении маршрутов передачи пакетов производительность IQ-коммутатора равна менее 59% показателя коммутатора с буферизацией на выходе
- M. Karo; M. Hluchyj; S. Morgan ***Input Versus Output Queuing on a Space-Division Packet Switch (1987)***

Виртуальная буферизация на выходе
Virtual Output Queuing



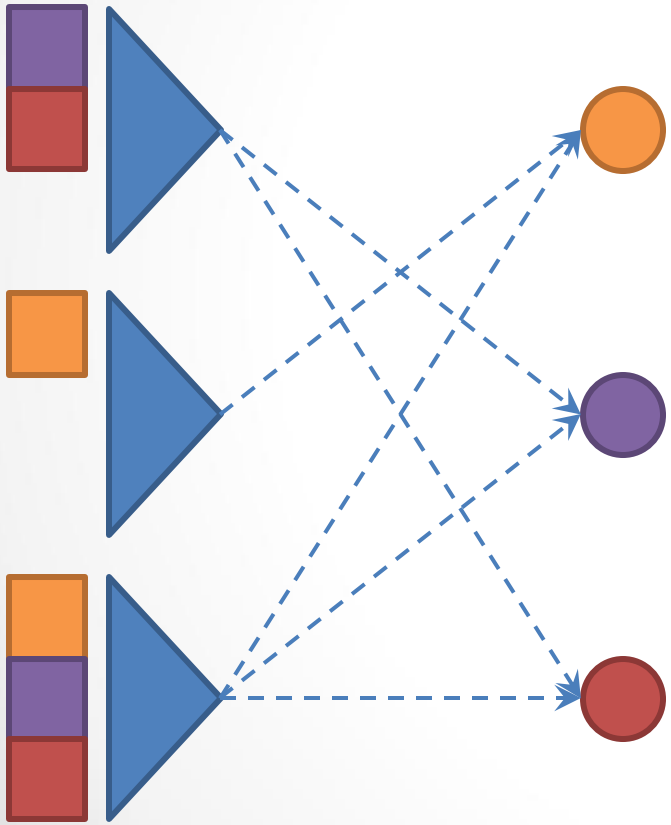



Виртуальная буферизация на выходе Virtual Output Queuing

N. McKeown A. Mekkittikul V. Anantharam J. Walrand
Achieving 100% Throughput in an Input-Queued Switch

- Проблема HOL blocking не возникает
- Появляется N^2 очередей пакетов
- Коммутационная матрица с N^2 входами не требуется, но необходим алгоритм быстрого арбитража для выбора нужной очереди на каждом из интерфейсов
- Сложные динамические алгоритмы арбитража затруднительно реализовать в аппаратуре

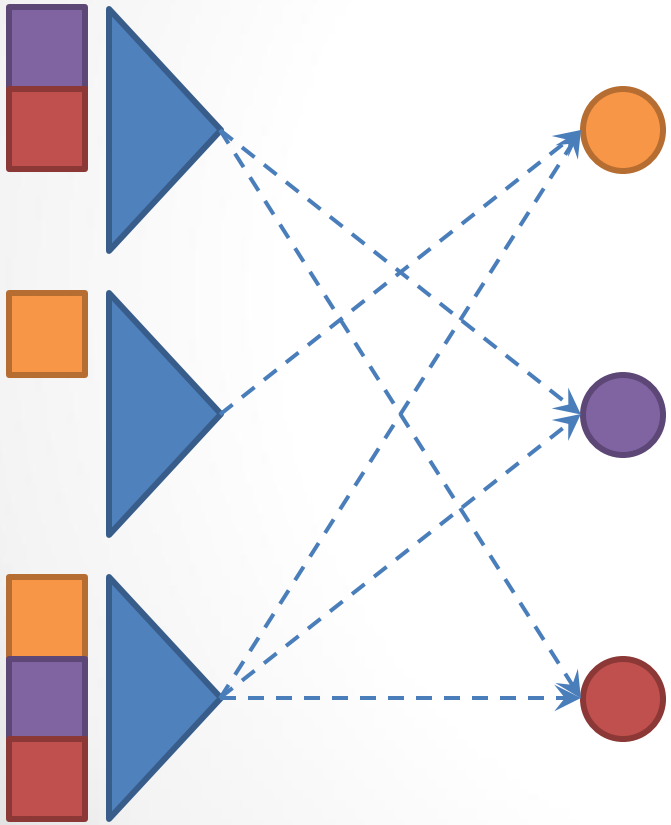
Алгоритмы арбитража коммутационной матрицы



Предположения модели:

- Коммутационная матрица имеет N ***плоскостей***
- Пакеты разбиваются на ***ячейки*** фиксированной длины (cells) при входе и восстанавливаются на выходе из матрицы
- Коммутационная матрица работает по ***тактам*** – на каждом такте она может по одной ячейке из каждого входа и поместить по одной ячейке на каждый свой выход

Алгоритмы арбитража коммутационной матрицы



Найти такой алгоритм арбитража коммутационной матрицы, при котором:

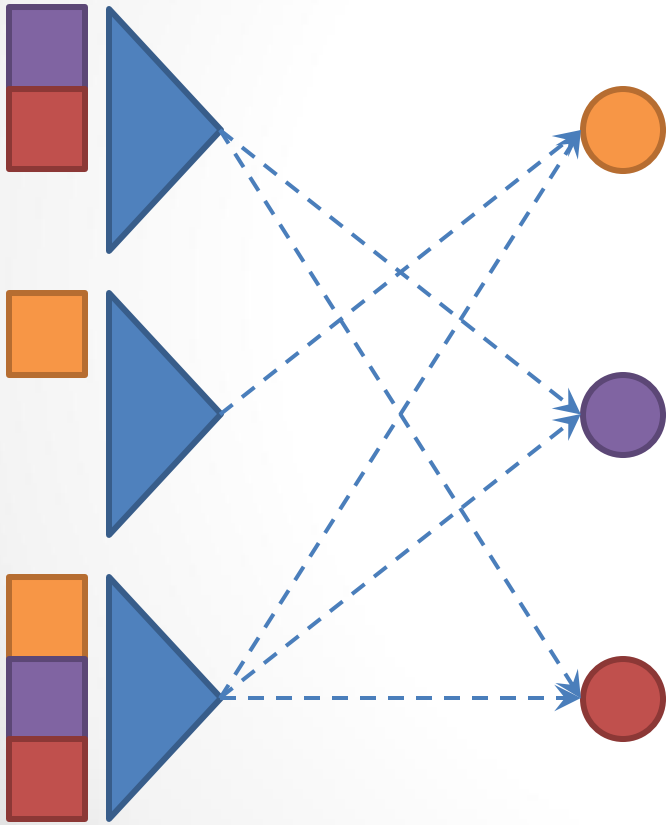
- ***(Производительность)***

Требования full backplane выполнялись бы без ускорения матрицы

- ***(Справедливость)***

Никогда не происходило бы удушение потоков трафика

Алгоритмы арбитража коммутационной матрицы

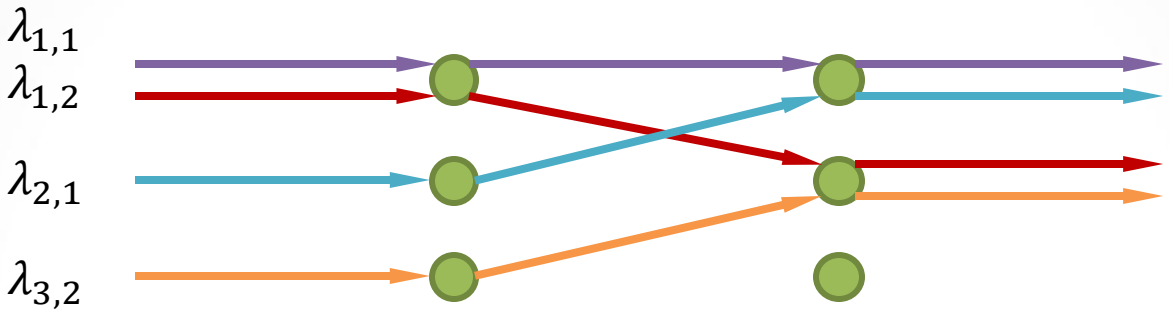


Гипотеза:

- Подходящий алгоритм должен передавать как можно большее количество ячеек на каждом такте работы матрицы – задача ***поиска наибольшего паросочетания***

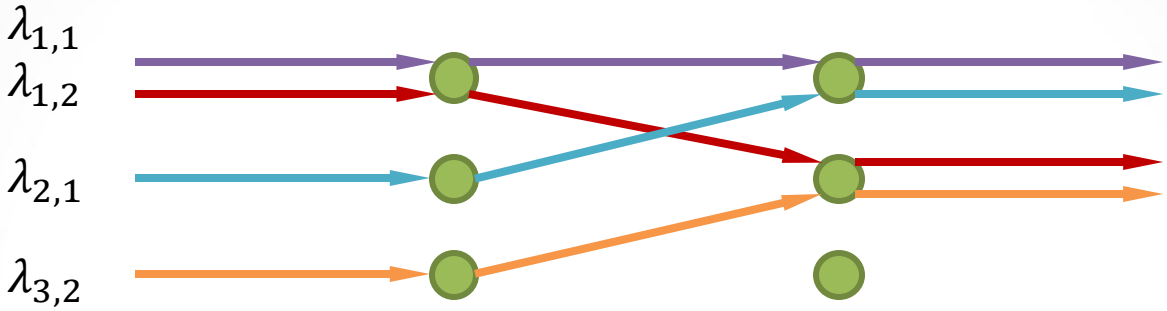
Гипотеза неверна, поскольку алгоритм не отличается ни производительностью, ни справедливостью планирования

Неприменимость алгоритма для наибольшего паросочетания (1)



- Пусть в ситуациях, когда существует несколько наибольших паросочетаний алгоритм выбирает один из них с равной вероятностью
- Пусть пропускная способность каждой из линий связи равна 1, скорость каждого из потоков $\lambda_{1,1} = \lambda_{1,2} = \lambda_{2,1} = \lambda_{2,2} = \frac{1}{2} - \delta$

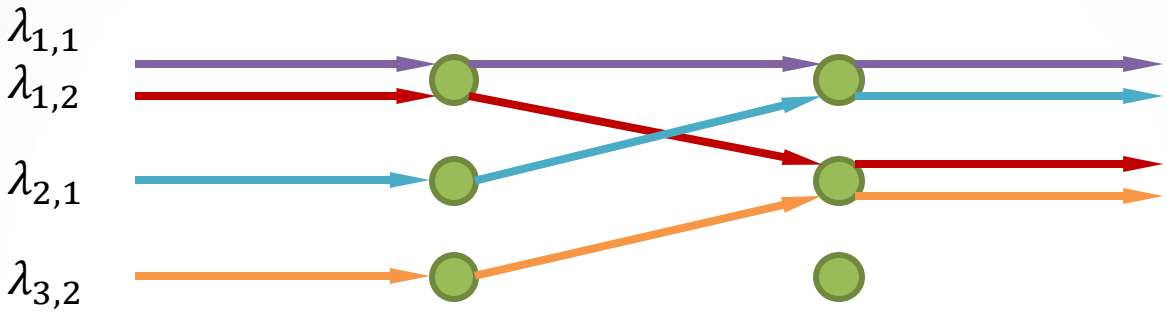
Неприменимость алгоритма для наибольшего паросочетания (1)



- Если и поток $\lambda_{2,1}$, и поток $\lambda_{3,2}$ готовы к передаче, то вход 1 может быть выбран с вероятностью не более 2/3, иначе – с вероятностью 1
- По формуле полной веростности наибольшая скорость поступления данных со входа 1 составляет:

$$\frac{2}{3}\left(\frac{1}{2}-\delta\right)^2+\left(1-\left(\frac{1}{2}-\delta\right)^2\right)=1-\frac{1}{3}\left(\frac{1}{2}-\delta\right)^2$$

Неприменимость алгоритма для наибольшего паросочетания (1)

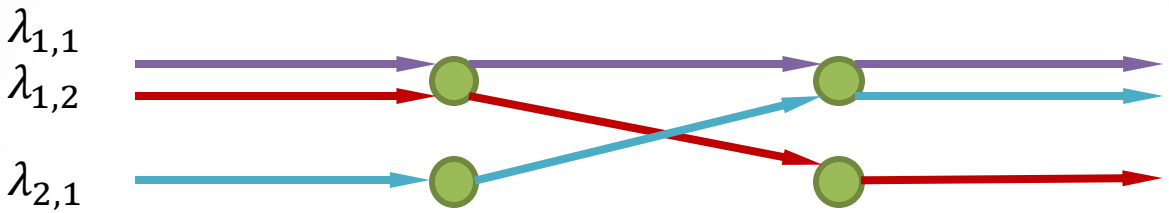


- Скорость поступления данных на вход 1 составляет $1 - 2\delta$
- Если скорость выборки данных будет меньше скорости их поступления, то требования full backplane нарушены:

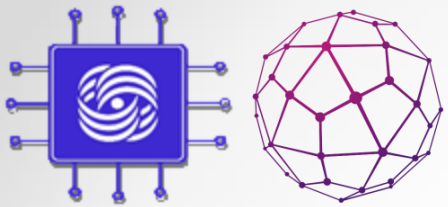
$$1 - 2\delta > 1 - \frac{1}{3}\left(\frac{1}{2} - \delta\right)^2$$

- Условие выполнено при $\delta < 0.0358$

Неприменимость алгоритма для наибольшего паросочетания (2)



- Пусть скорость поступления данных из потоков равна пропускной способности канала $\lambda_{1,1} = \lambda_{1,2} = \lambda_{2,1} = 1$
- Алгоритм для поиска наибольшего паросочетания всегда будет выбирать потоки $\lambda_{1,2}$ и $\lambda_{2,1}$
- Поток $\lambda_{1,1}$ будет испытывать ***удушение (starvation)***



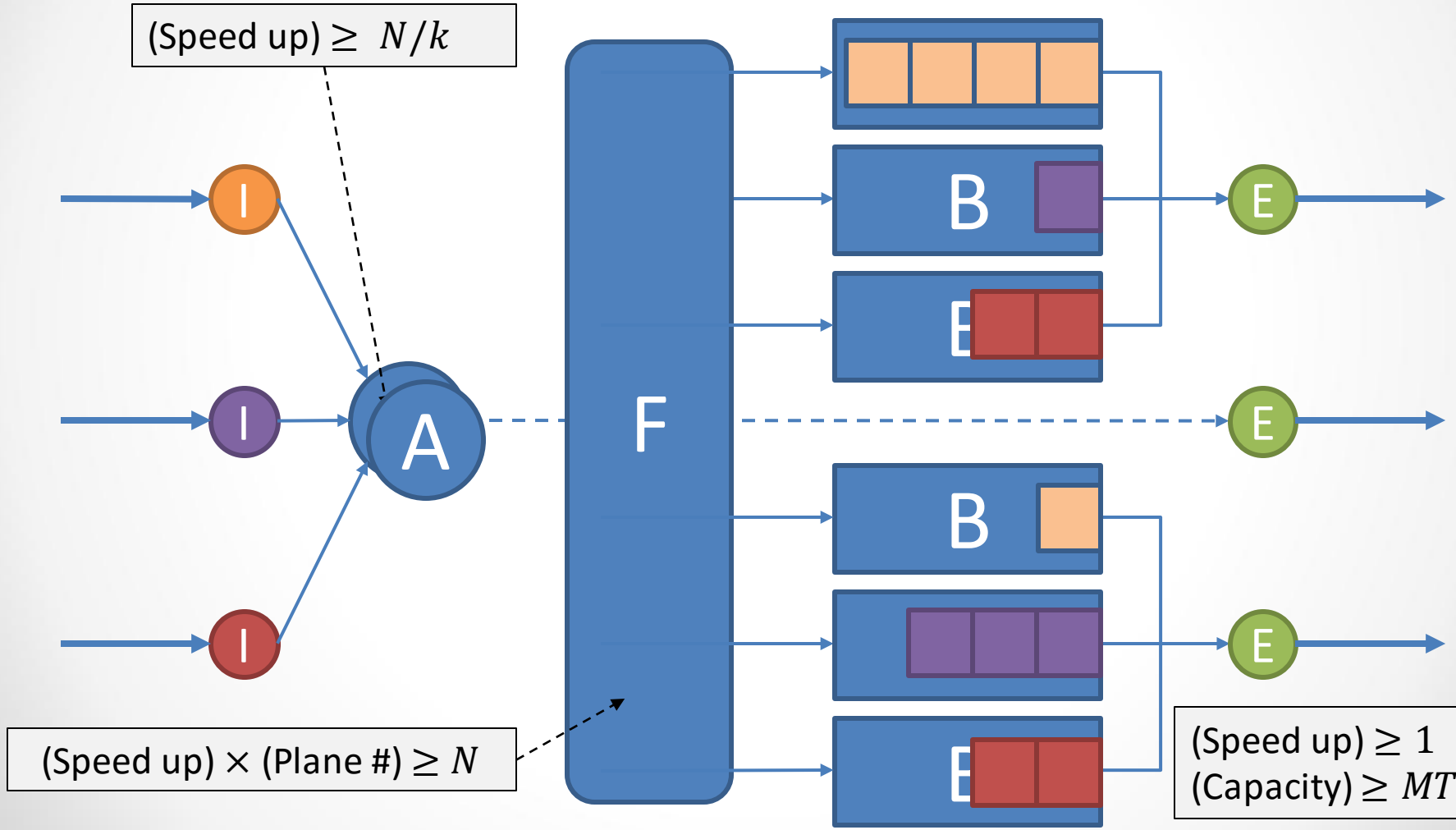
Алгоритм арбитража Oldest Cell First

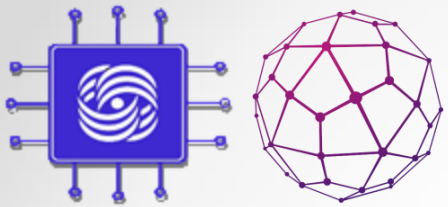
- Каждой из очередей присваивается собственный весовой коэффициент – количество тактов, когда находящаяся в ведущей позиции ячеяка не была выбрана
- Алгоритм арбитража выбирает паросочетания таким образом, чтобы максимизировать сумму весов выбранных им очередей

Недостатки:

- Высокая сложность реализации
- Алгоритм не учитывает задержку

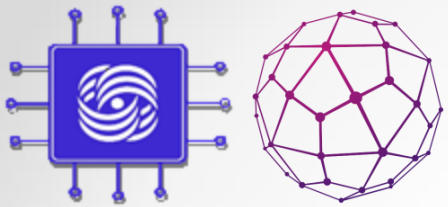
Множественная буферизация на выходе
Multiple Output Queuing





Множественная буферизация на выходе Multiple Output Queuing

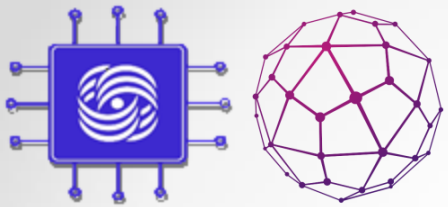
- Не нужен алгоритм арбитража коммутационной матрицы
- Необходимо усложнение логики для распределения пакетов по очередям на выходе из коммутационной матрицы и дополнительный планировщик пакетов для выборки данных из этих очередей



Практика построения коммутационных устройств

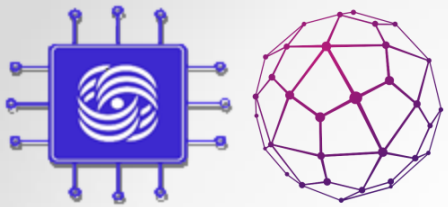
When the rubber meets the road…

- Модели редко используются в чистом виде
- Производители пытаются искать компромиссы между стоимостью и утилизацией каналов под различной нагрузкой
- На сегодняшний день наиболее распространены модели Combined Input-Output Queuing (CIOQ)
- Вместо полноценной коммутационной матрицы часто используются её упрощения – knockout switches или сети из переключателей



Распределение ресурсов между потоками

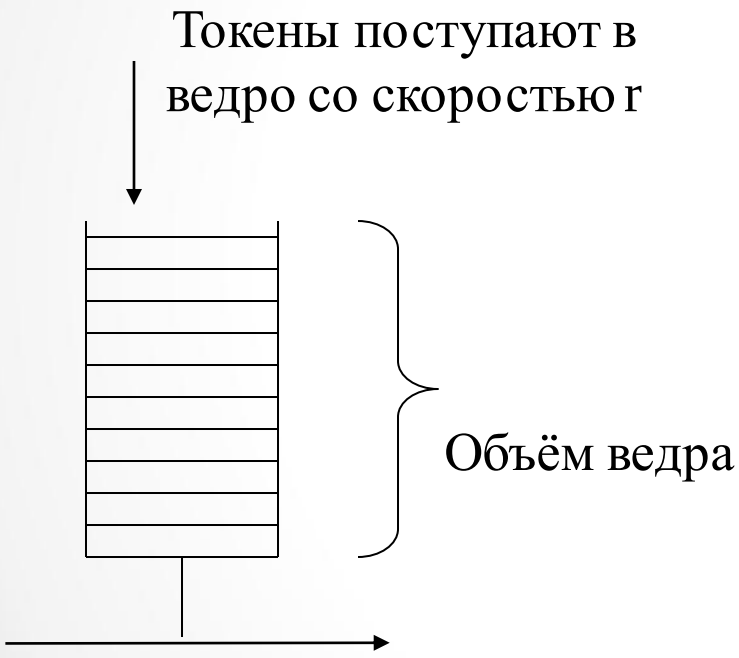
- ***Качество сервиса*** – качество обслуживания потоков на коммутационных устройствах: пропускная способность, задержка при передаче пакетов, уровень потерь, etc
- Качество обслуживания потоков напрямую связано с количеством выделенных на его обслуживание ***ресурсов***: доли пропускной способности канала, объёма буферной памяти и процессорного времени
- Управление качеством достигается с помощью:
  - Дисциплин очередизации потоков
  - Шейпинга трафика



Ограничение потоков

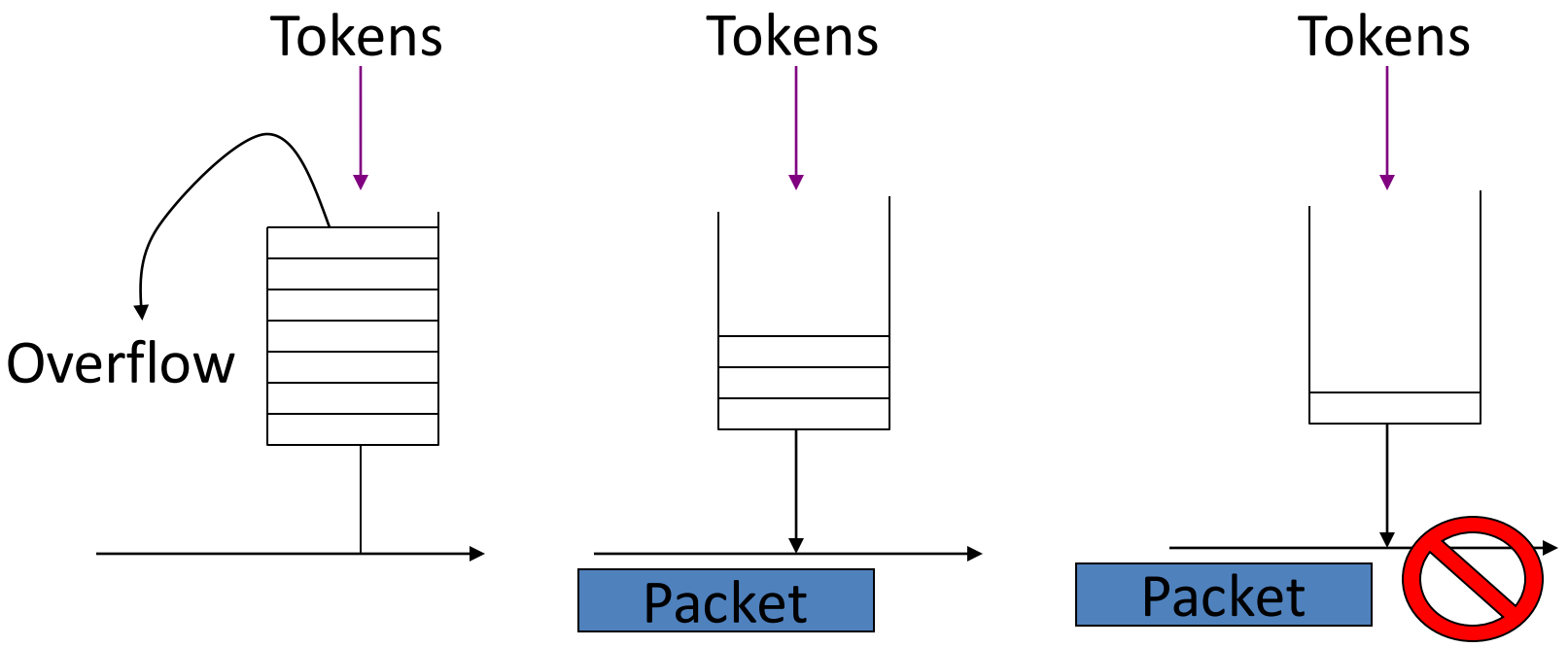
- ***Профиль потока*** определяется интенсивностью пересылки его данных (скорость, *всплески*, etc)
- ***Shaping & Policing*** трафика позволяет придавать профилям потоков необходимую форму с помощью встроенных средств коммутатора
- Если интенсивность потока превышает спецификации профиля:
  - shaping задерживает обработку пакетов
  - policing сбрасывает пакеты

Алгоритм Token Bucket



- Если ведро заполняется, то поступающие токены игнорируются
- При отправке пакета размером N байтов из ведра извлекается N токенов
- Если при в ведре недостаточно токенов, то отправка пакета откладывается
- OpenFlow Metering

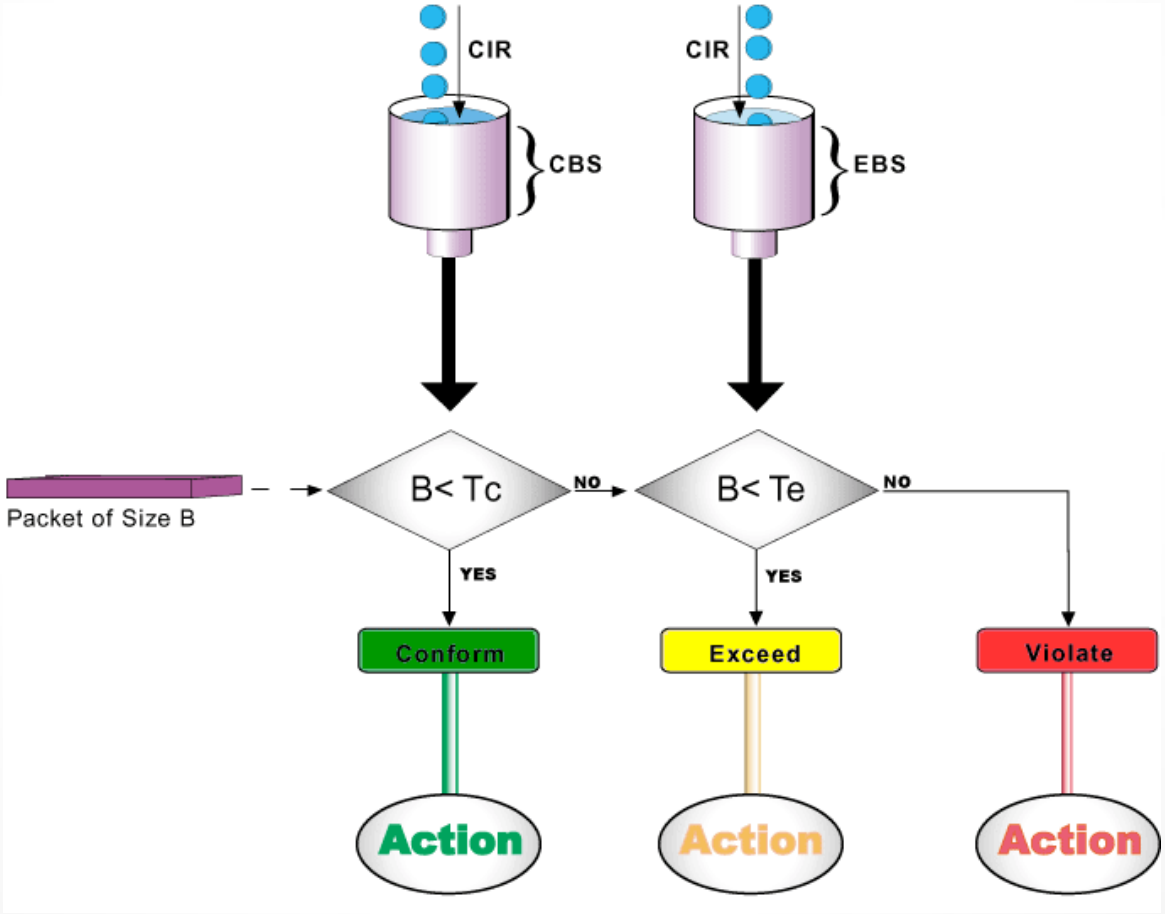
Token Bucket в картинках

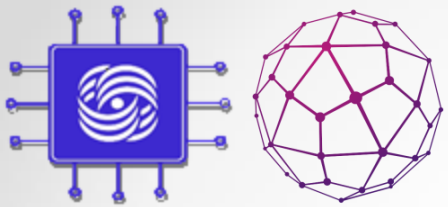


Если токенов достаточно, пакет отправляется, а количество токенов уменьшается

Иначе пакет ожидает, пока в ведре накопятся новые токены

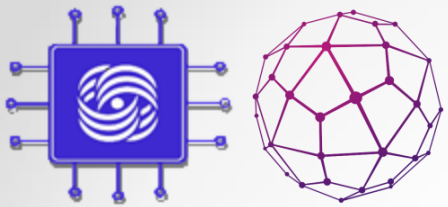
Использование нескольких алгоритмов текущего ведра





Дисциплины очередизации

- Дисциплина обслуживания очередей включает:
  - Планирование выборки пакетов из очереди
  - Политику сброса пакетов
- Выбор дисциплины очередизации определяет:
  - Распределение пропускной способности канала между потоками – какой пакет будет отправлен следующим?
  - Распределение буферной памяти – какой пакет будет сброшен, если памяти на всех не хватает
- Правила обслуживания очередей значительно влияют на задержку
- OpenFlow Enqueue

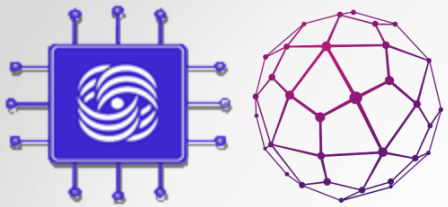


Типичная дисциплина очередизации

Самая простая из дисциплин

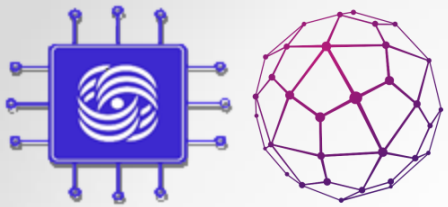
FIFO + drop-tail

- ***FIFO (first-in-first-out)***: пакеты выбираются из очереди в том же порядке, в котором они в эту очередь поступили
- ***Drop-tail***: если в очереди нет свободного места, то направленный в неё пакет сбрасывается, вне зависимости от его важности



Недостатки FIFO + Drop-tail

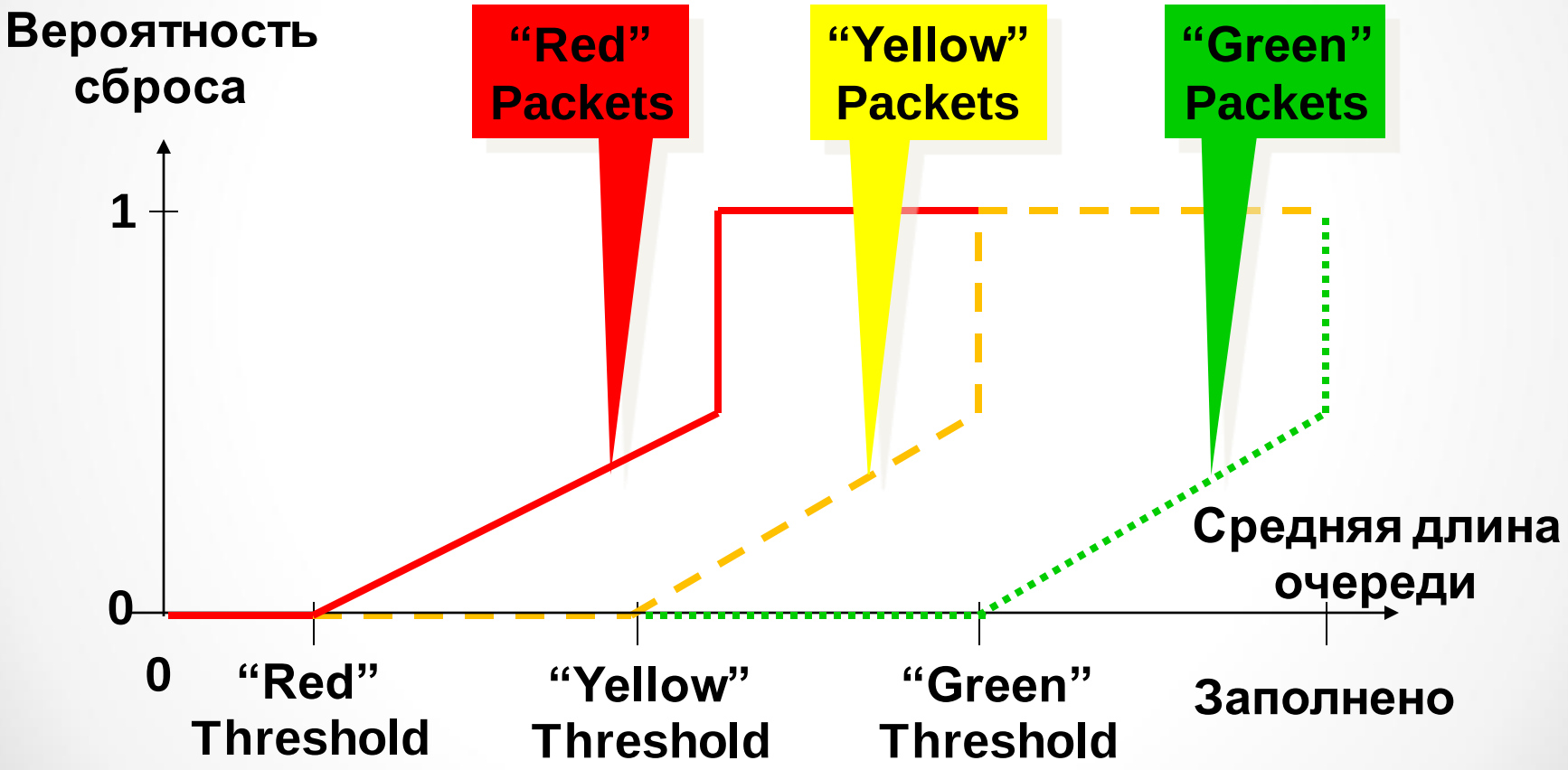
- Блокировка потоков (Lock-out)
  - Позволяет неограниченно захватывать ресурсы
  - Чем больше интенсивность потока, тем больше ресурсов он получает
- Потоки обрабатываются с одним качеством
- Проблема полных очередей
  - Приводит у увеличению сквозной задержки
  - Наступает эффект синхронизации TCP трафика -- очередь то переполняется, то простаивает
- Потоки с большими всплесками ущемляются сильнее других



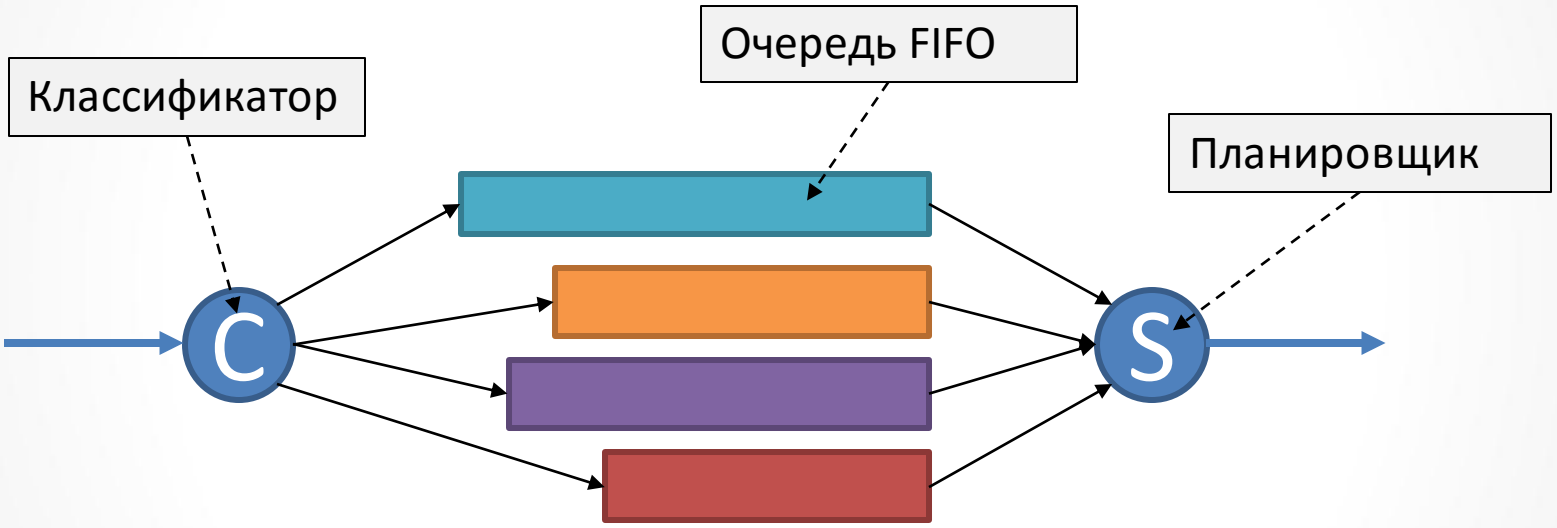
Обход проблемы блокировки потоков

- Случайный сборос пакетов (random drop)
  - Если пакет пребывает в очередь, и в ней нет места для его размещения, из неё удаляется случайный пакет
- Сброс пакетов заранее, до переполненеия очереди (random early detection)
  - Вычисляет среднюю загруженность очереди x
  - Если $x < x_{min}$ - пакеты не сбрасываются
  - Если $x > x_{max}$ - сбрасывается поступающий пакет
  - Иначе, пакет сбрасывается с веростностью, линейно зависящей от близости к пороговым значениям

RED с несколькими пороговыми значениями

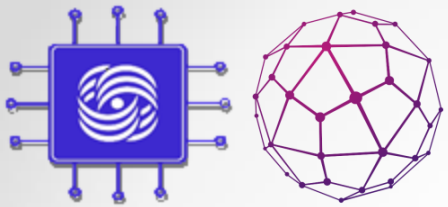


Устройство буферного блока



Классификатор распределяет пакеты по очередям

Планировщик выбирает пакеты из очередей

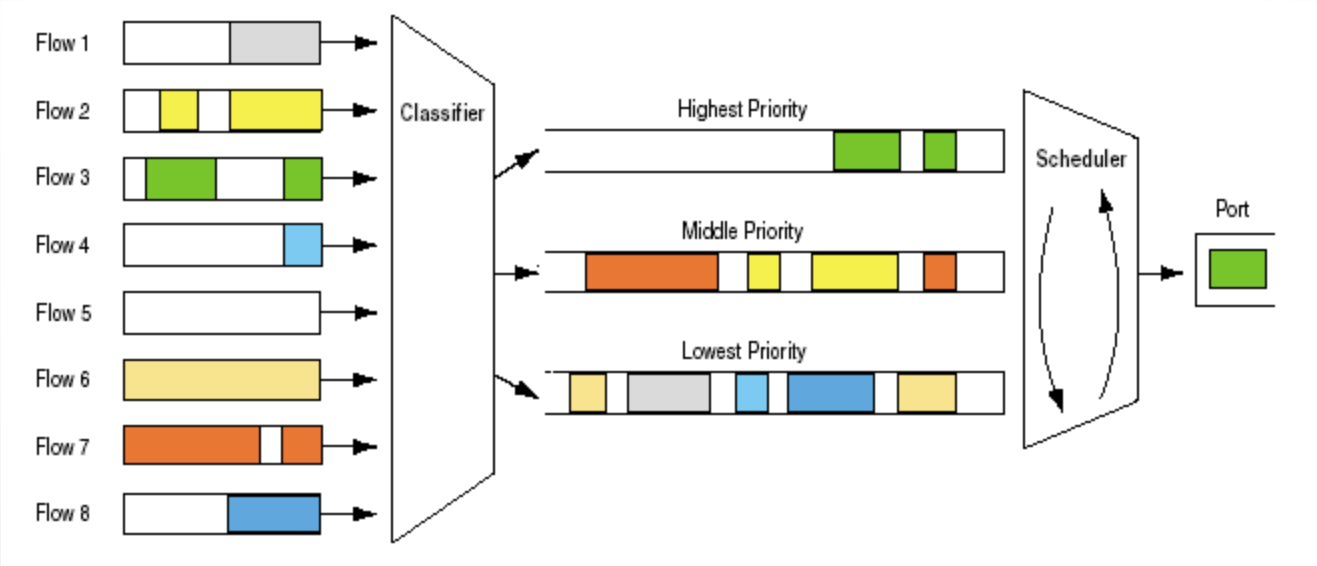


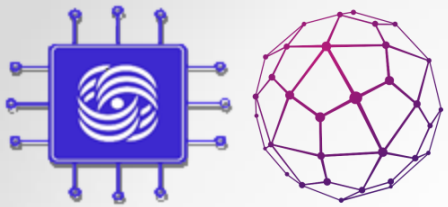
Распространённые дисциплины очередизации

- First-In-First-Out (FIFO)
- Priority Queuing (PQ)
- Fair Queuing (FQ)
- Weighted Fair Queuing (WFQ)
- Weighted Round Robin (WRR)
- Shared Round Robin (SRR)
- Deficit Weighted Round Robin (DWRR)

Статические приоритеты Priority Queuing (PQ)

- Пакеты распределяются по нескольким очередям
- Каждой очереди назначается собственный приоритет
- Планировщик извлекает пакет из очереди лишь в том случае, если все очереди с большим приоритетом пусты
- Каждая из очередей обслуживается по дисциплине FIFO





Статические приоритеты

Достоинства

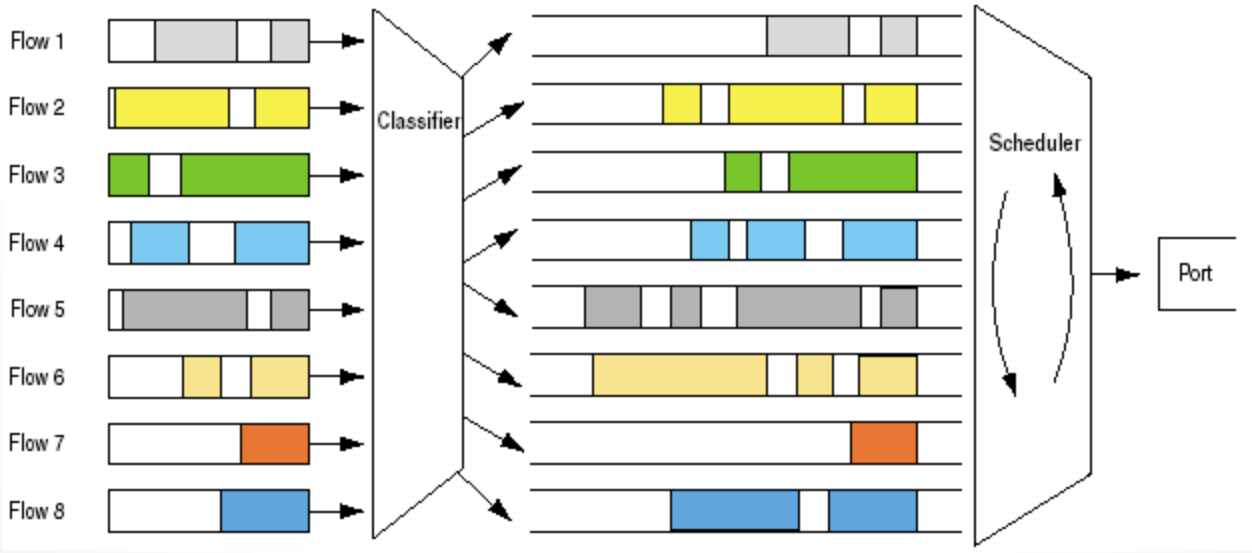
- Позволяет организовать дифференцирование трафика простым в реализации способом
- Возможность передачи данных с низкой задержкой

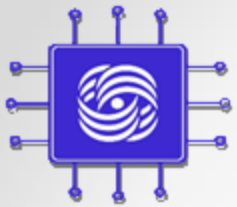
Недостатки

- Существует опасность удушения потоков – лучше использовать дополнительный rate-control
- Низкоприоритетный трафик может испытывать существенные задержки
- Борьба между потоками, направленными в одну и ту же очередь, сохраняется

Справедливая очередизация Fair Queuing (FQ)

- Для *каждого потока* выделяется собственная очередь временного хранения пакетов
- Пакеты выбираются из очередей циклически



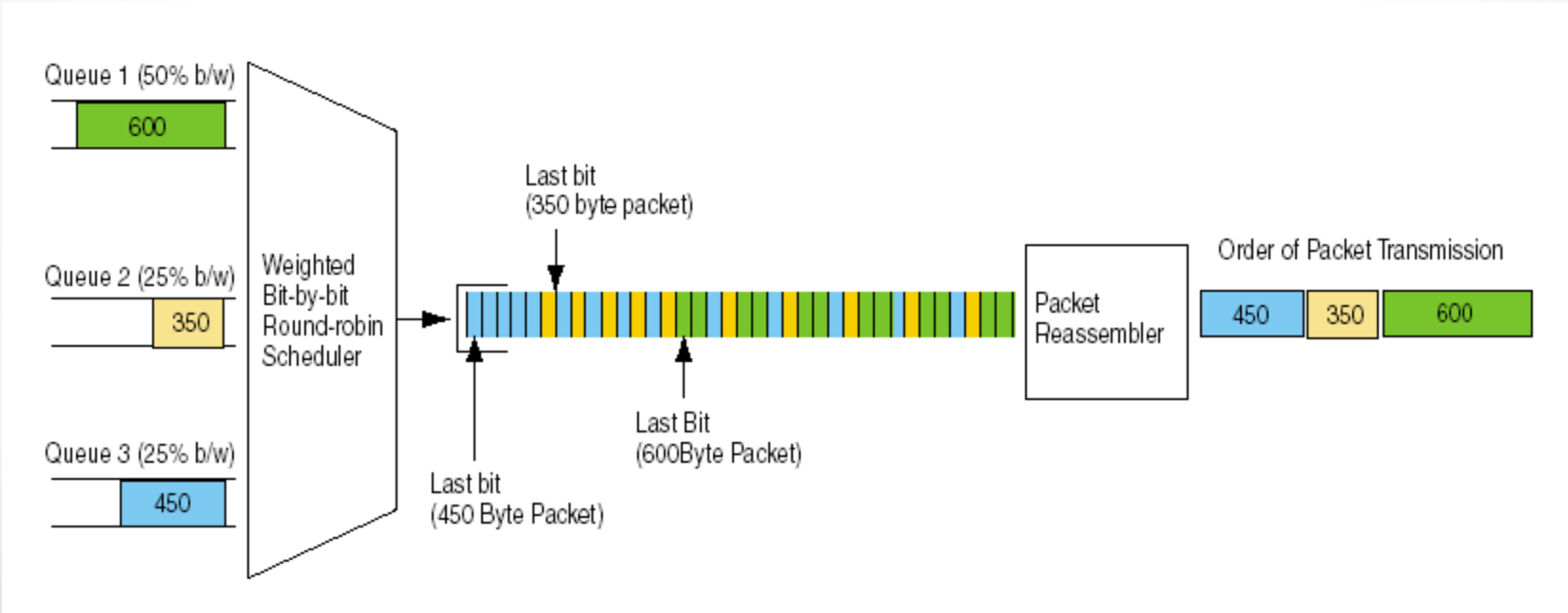



Справедливая очередизация Fair Queuing (FQ)

- Потоки изолированы друг от друга
- Реализация на базе аппаратных очередей затруднительна
- Может быть использован совместо с другими дисциплинами обслуживания
- Дифференциация между потоками передачи данных не производится, потоки с разными требованиями к пропускной способности не поддерживаются
- Нет механизмов для передачи real-time трафика
- Потоки с пакетами больших размеров получают преимущество

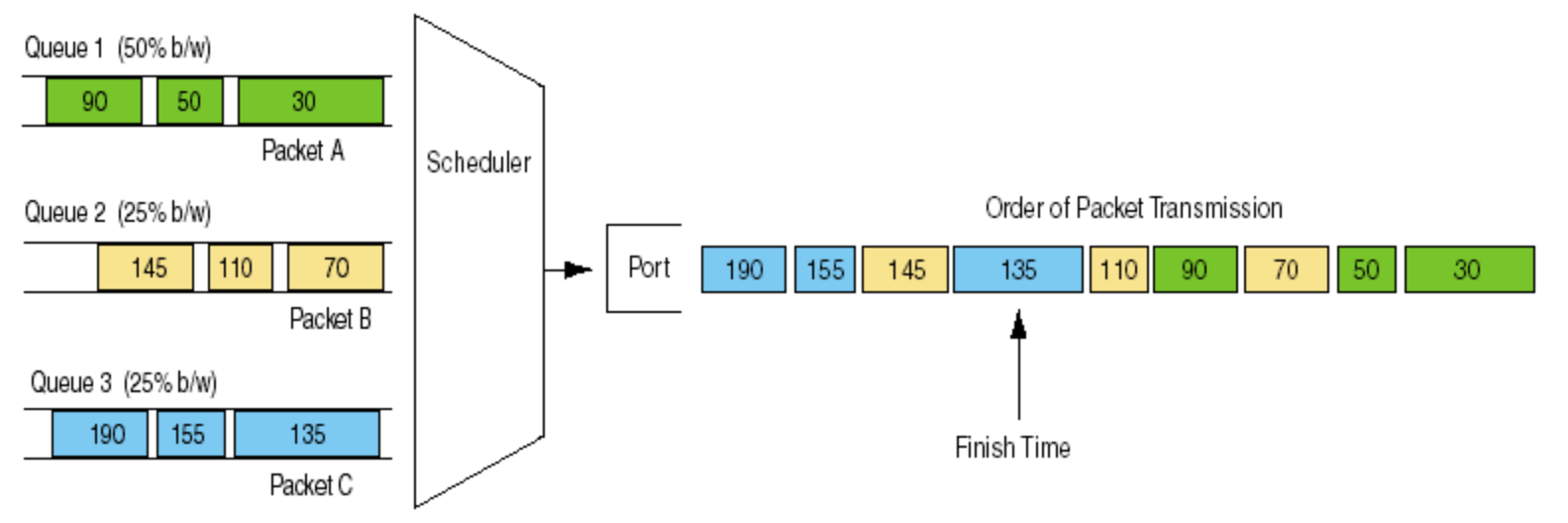
Взвешенная справедливая очередизация (WFQ)

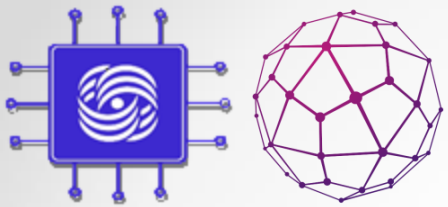
- Использует веса для поддержки потоков с разными требованиями к пропускной способности
- Учитывает размер пакетов при планировании выборки пакетов из очереди
-



Взвешенная справедливая очередизация (WFQ)

- Абстракция fluid модели приближается путём вычилсения времени завершения передачи пакета
- Планировщик выбирает пакеты с наименьшим расчётным временем завершения передачи





Взвешенная справедливая очередизация (WFQ)

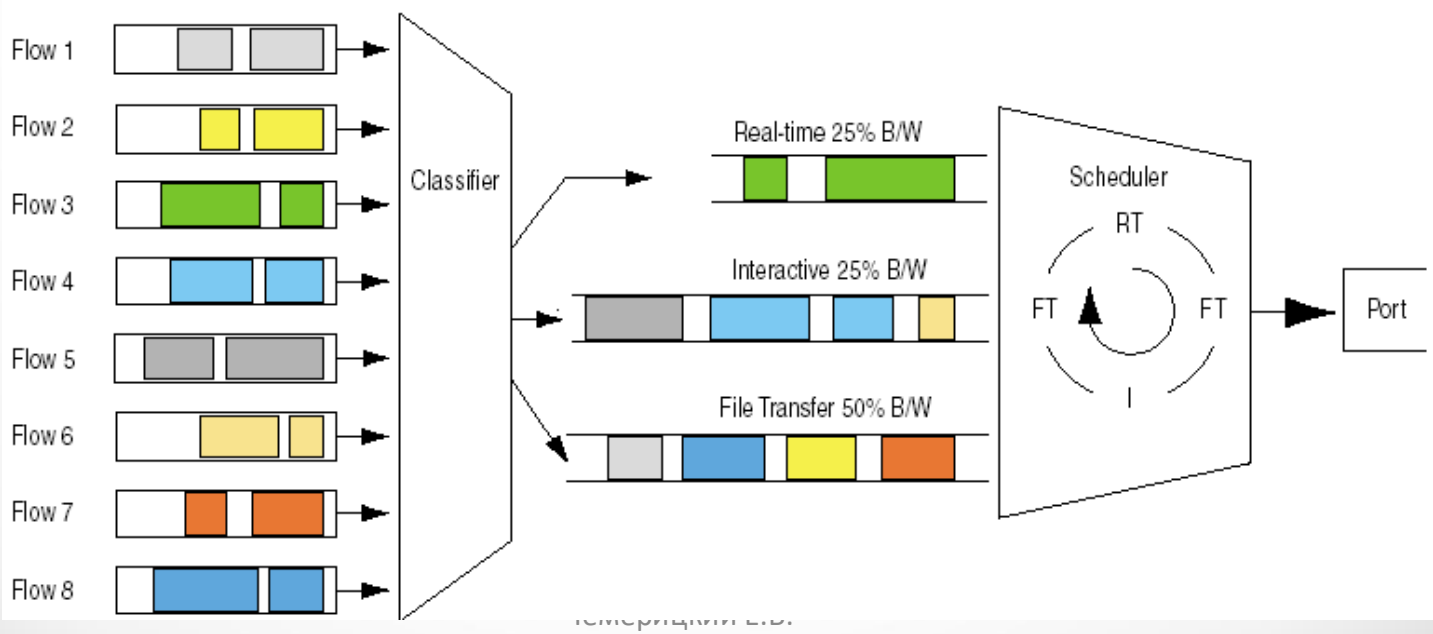
- Сложность аппаратной реализации
- Высокая сложность алгоритма – для каждой очереди необходимо поддерживать состояние – временную метку, и обновлять её значение при каждом прибытии или отправке пакета
- Плохая масштабируемость

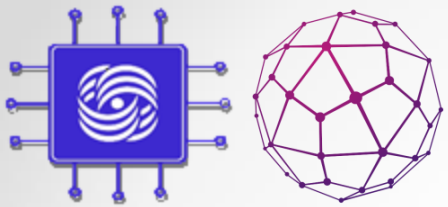
Существуют более продвинутые аналоги:

- Self-clocking Fair Queuing (SCFQ)
- Worst case Fair weighted Fair Queuing (WF^2Q)
- Worst case Fair weighted Fair Queuing+ (WF^2Q+)

Weighted Round-Robin

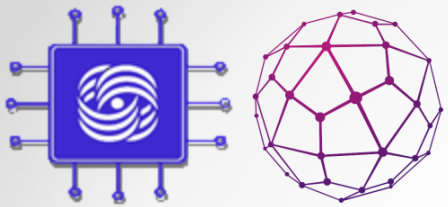
- Потоки распределяются на несколько классов, для каждого класса создаётся своя очередь
- Очереди обслуживаются циклически
- Количество пакетов, извлечённых из очереди соответствует её весовому коэффициенту





Weighted Round-Robin

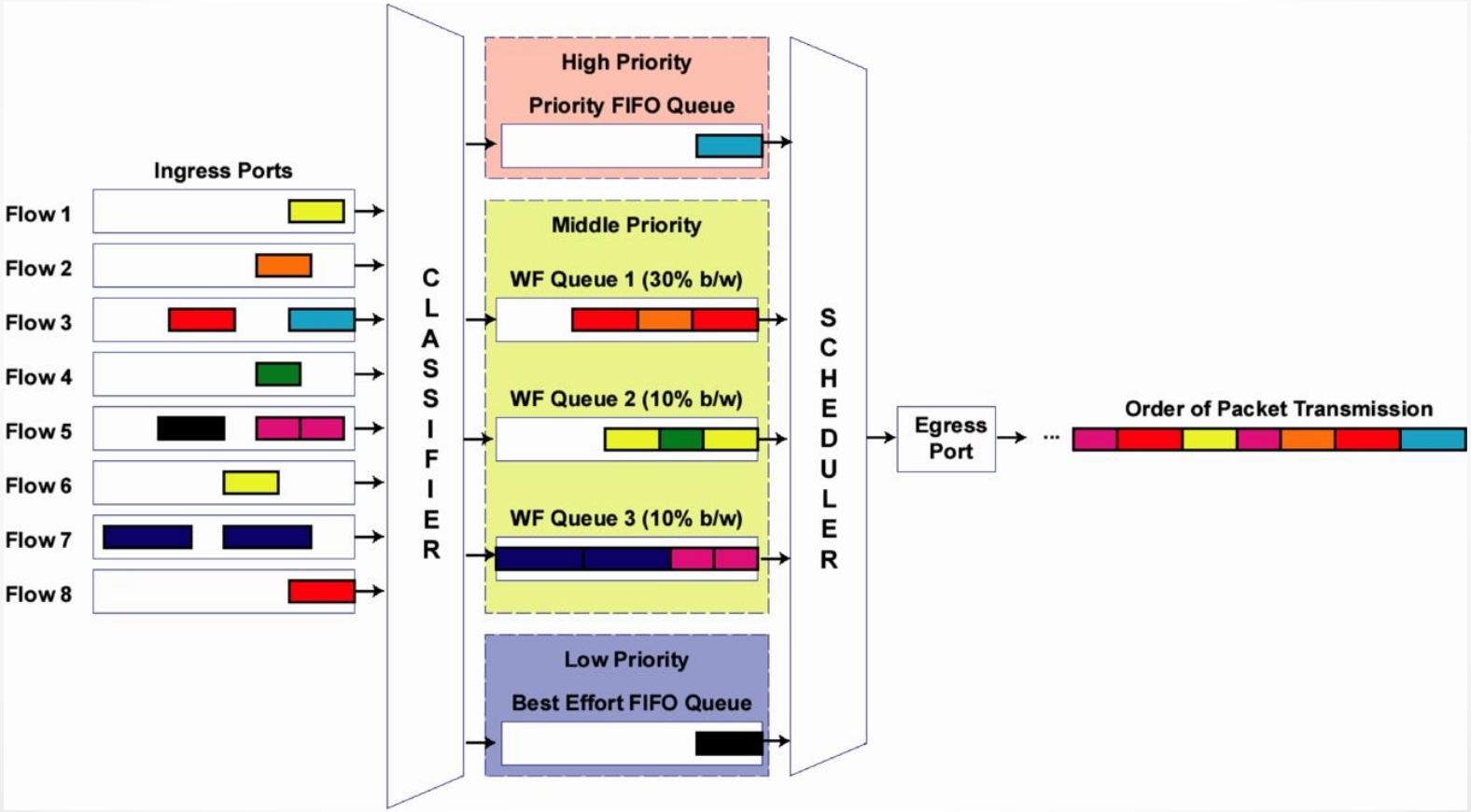
- Простая модель – алгоритм может быть реализован в аппаратуре
- Очереди для трафика средней и низкой важности не игнорируются – каждый класс потоков получает свою долю ресурсов
- Хорошо работает только при условии, что размеры пакетов равны
- Не очень хорошо перемешивает трафик, сохраняя последовательности подряд идущих пакетов из одного потока

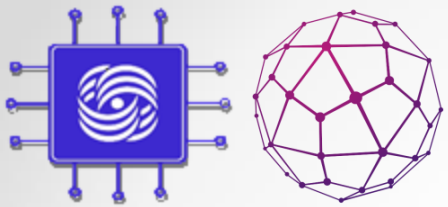


Доработки дисциплины WRR

- SRR – shaped round-robin
  - Выборка происходит по раундам
  - Очередь исключается, если её вес меньше номера раунда
  - За раунд выбирается по одному пакету от каждой из участвующих в этом раунде очередей
  - Если в раунде не осталось ни одной непустой очереди, начинаем с первого раунда
- DWRR – deplicit weighted-round robin
  - Выборка осуществляется не по пакетам, а по байтам – решается проблема пакетов разного размера

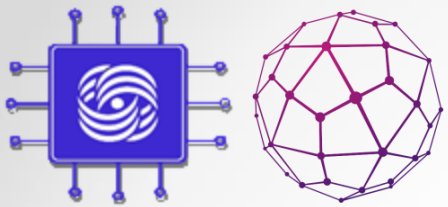
Комбинирование нескольких дисциплин





Заключение

- Этапы эволюции коммутаторов
- Классификация коммутаторов по методам буферизации
- Механизмы для обеспечения контроля качества – шейпинг, политики сброса пакетов, дисциплины планирования пакетов из очередей



Вопросы:

- Каковы преимущества и недостатки архитектуры коммутатора с буферизацией на выходе?
- Сколько очередей внутри коммутатора с N портами, виртуальной буферизацией на входе, WRR из 5 очередей и 3-х уровневым Random Early Detection необходимо для поддержания требований качества сервиса?