Ответы на Вопросы к экзамену по курсу «Дополнительные главы компьютерных сетей» (2015)

Подготовлено: Кирилл Шохин и avasite

Оглавление (*кликабельно*)

1. Оптические транспортные сети и сетевые хранилища данных ... 3

1.1. Тенденции и требования рынка телекоммуникации. Значение волоконно- оптических систем для современных компьютерных сетей. ... 4

1.2. Структура и типы услуг современной сети телеком оператора. Физические свойства носителей. ... 5

1.3. Теоретические основы передачи электромагнитных сигналов. Основные понятия и их взаимосвязь. Способы кодирования данных. ... 8

1.4. Методы ускорения передачи. Мультиплексирование. Примеры стандартов мультиплексирования. ... 9

1.5. Synchronous Digital Hierachy (SDH): основные понятия, структура и иерархия протокольных единиц данных (PDU). 11

1.6. WDM системы передачи данных: основные понятия, емкость, принципы работы, форматы модуляции сигналов. 12

1.7. Optical Transport Hierarchy (стандарт ITU G.709): назначение, базовые механизмы, достоинства. ... 14

1.8. Иерархия интерфейсов и PDU в стандарте ITU G.709. Примеры применения OTN. ... 15

1.9. Структура и мультиплексирование PDU в стандарте ITU G.709. ... 20

1.10. Механизмы мониторинга, обнаружения и коррекции ошибок вы стандарте ITU G.709. ... 24

1.11. Сравнение серверно-ориентированной архитектуры со Storage-ориентированной архитектурой. Внутренняя организация Дисковой ПодСистемы (ДПС). ... 25

1.12. RAID дисковые массивы и их уровни. Горячее резервирование, способы ускорения работы ДПС. ... 26

1.13. Интеллектуальные ДПС: удаленное зеркалирование, групповая консистентность, LUN маскирование. Методы повышения устойчивости работы ДПС. ... 29

1.14. Тракт от CPU до ДПС. SCSI интерфейс: структура, адресация устройств, организация СХД на SCSI. ... 30

1.15. Fibre Channel: основные характеристики, структура стека протоколов, топологии, типы портов. ... 31

1.16. FC-0, FC-1: характеристики физической среды, кодировка, упорядоченные наборы, управление линией. ... 32

1.17. FC-2: структура кадра, организация передачи данных, управление потоком, классы обслуживания. ... 33

1.18. FC-3, FC-4: сервисы, имена, адреса, сервисы среды коммутации. ... 34

1.19. Примеры топологий, интеграция FC_SAN с IP сетями. ... 36

1.20. Примеры синергии SDN и NFV технологий ... 38

1.21. Проблемы безопасности в SDN сетях. ... 38

2. Программно-Конфигурируемые Сети и Виртуализация Сетевых Функций ... 39

2.1. Проблемы традиционных сетей. Основные принципы SDN. Архитектура SDN. Преимущества SDN. Примеры применения. Абстракции в IT и в SDN. ... 39

2.2. Протокол OpenFlow. Структура OpenFlow коммутатора и контроллера. Таблица потоков. Основные сообщения протокола OpenFlow. Принципы установки правил. Суть вопроса “SDN == OpenFlow?“. ... 41

2.3. OpenFlow 1.3. Несколько таблиц потоков, групповые таблицы, Meter таблицы, механизм отказоустойчивости контроллеров. Пример приложения по маршрутизации в SDN/OpenFlow. ... 43

2.4. Варианты применения SDN/OpenFlow в корпоративном сегменте, телеком. операторы и сервис провайдеры, ЦОД и облачные вычисления. ... 45

2.5. OpenFlow контроллер. Архитектура и принцип работы. Требования к контроллеру OpenFlow. Экспериментальное исследование и методика. Достоинства и недостатки методики. ... 47

2.6. Производительность и программируемость OpenFlow контроллеров. Способы улучшения производительности. Проблематика Northbound API и варианты решения. ... 49

2.7. Распределенный уровень управления в SDN/OpenFlow. Основные угрозы. Стратегии резервирования. Основные задачи и варианты решения. ... 50

2.8. Виртуализация сетевых сервисов (NFV). Проблемы телеком. операторов. Уровни развития NFV. Архитектура и основные термины по ETSI. Варианты применения. 54

2.9. Проблематика производительности сетевых сервисов. Суть проблемы, узкие места и варианты решения. 56

2.10. Одновременное применение концепций NFV и SDN. Основные задачи. Примеры. 57

3. Методы моделирования и анализа компьютерных сетей 58

3.1. Методы моделирования компьютерных сетей. Понятия модели, точности моделирования. Плюсы и минусы каждого метода моделирования. 58

3.2. Методы имитационного моделирования компьютерных сетей. Системная динамика: основные понятия, примеры моделей. 59

3.3. Методы имитационного моделирования компьютерных сетей. Агентное моделирование: основные понятия, примеры моделей. 61

3.4. Методы имитационного моделирования компьютерных сетей. Дискретно-событийное моделирование: основные понятия, примеры моделей. 62

3.5. Архитектура системы NPS. LXC контейнеры. Особенности моделирования глобальных компьютерных сетей. 63

3.6. Основы контейнерной визуализации. Проект Docker: цели проекта, основные преимущества, базовые команды управления. 64

3.7. Мониторинг сетевого трафика. Утилиты tcpdump, wireshark. Привести примеры фильтров tcpdump на L2, L3, L4, L7 уровнях сетевого трафика. 66

4. Методы верификации функционирования компьютерных сетей и управления качеством сервиса 67

4.1. Основные понятия и определения из области формальных методов. Задача формальной верификации на примере алгоритма Петерсона. Формальная модель, спецификация поведения, алгоритм верификации. 67

4.2. Задача формальной верификации конфигурации сети на примере средства VERMONT. Формальная модель, спецификация поведения, алгоритм верификации. 70

4.3. Варианты постановки задачи синтеза консистентного обновления конфигурации сети. Алгоритм трёхфазного обновления конфигурации сети с помощью тегирования. 72

4.4. Классификация коммутационных устройств по поколениям. Варианты компоновки коммутаторов в зависимости от метода буферизации. Требования к производительности блоков коммутатора. 73

4.5. Устройство коммутационной матрицы. Принципы передачи пакетов через коммутационную матрицу при виртуальной буферизации на выходе. Неприменимость алгоритма поиска наибольшего паросочетания для выборки пакетов. 76

4.6. Механизмы управления качеством сервиса на уровне коммутатора. Ограничение интенсивности потоков по алгоритму token bucket. Дисциплины очередизации: сброс и выборка пакетов. 78

4.7. Принципы функционирования протокола резервирования ресурсов RSVP. Модели управления качеством сервиса в сети Интернет: IntServ и DiffServ. 83

4.8. Связь задачи управления качеством с задачей распределения сетевых ресурсов. Управление качеством с помощью планирования маршрутов и многопоточной маршрутизации. 85

4.9. Основные понятия сетевого исчисления. Функции поступления и отправки, задержка и отставание, кривые нагрузки и сервиса. Оценки отставания, задержки и интенсивности выходного потока. 87

4.10. Построение оценки для сквозной задержки передачи данных через сеть с помощью алгоритма SFA. Причины низкой точности алгоритма SFA при его применении в предположениях модели DiffServ. 89

Материал взят со слайдов 2015 года, лекторами глав являлись соответственно Смелянский, Шалимов, Антоненко, Чемерицкий.

Все билеты написаны разумно.

!!! – обозначение вопроса (по ним работает поиск)

1. Оптические транспортные сети и сетевые хранилища данных

Аббревиатуры этой главы:

бод === символов/с (символ – то, что передаётся одним перепадом в сигнале (за раз сигнал может передать несколько бит))	JBOD – Just Bunch of Disks
OTN – Optical Transport Network	SCSI - Small Computer Serial Interface
WDM - Wavelength Division Multiplexing (Спектральное уплотнение каналов)	HBA – Host Bus Adapter
DWDM – Dense Wavelength-division multiplexing (плотное частотное разделение каналов)	ДПС – Дисковая Подсистема
CWDM - Coarse Wavelength Division Multiplexing (Грубое спектральное мультиплексирование)	RAID – Redundant Array of Independent Disks
CDN – Content Delivery Network	BER – bit error rate
NAT – Network Address Translation	LUN – Local Unit Number
BRAS – Broadband remote access server	UPS - uninterruptible power supply
OSNR – Optical Signal-to-Noise Ration	HBA – Host bus adapter
ROADM – Reconfigurable Optical Add/Drop Multiplexer	SOF - Start of Frame
AWG – American Wire Guide	КсА – Кольцо с арбитражём
DWDM – самое первое и самое простое – просто оптоволокно для быстрой передачи сигнала.	WWN - World Wide Name - 64 бита
SDH - Synchronous Digital Hierarchy	WWPN - World Wide Portal Name
O-E-O – Optical-Electric-Optical	WWNN - World Wide Node Name
PDU – Protocol Data Unit	FCN – FiberChannel Name
BER – Bit Error Rate	FLOG – Fabric Login
OOK – On-Off Key	SAN – Storage Area Network
FEC – Forward Error Connection	
TCM – Tandem Connection Monitoring	Long-haul -> Metro core -> Metro access -> Premise
ROADM – Reconfigurable Optical Add/Drop Multiplexor	
OTH – Optical Transport Hierarchy	Длинна волны * частоту = скорость света
OTU – optical transport data unit	
ODU – optical channel data unit	Теорема Найквиста (1924г.) (или теорема Котельникова) - Взаимосвязь пропускной способности канала и ширины его полосы пропускания определяет $V_{max\ data\ rate} = 2H \log_2 L$, где H – ширина полосы пропускания, L – количество уровней сигнала. (эта теорема не учитывает шума в канале)
OCh – Optical Channel	
OSC – optical supervisory channel	$1dB = 10 \log_{10}(S/N)$, где S – сигнал, N – шум
OMS – Optical Multiplex Section	
FAS – Frame Alignment Signal	Теорема Шеннона - V_{max} (бит/с) = H (Гц) * $\log_2(1+S/N)$ bps, где H – полоса пропускания (Гц)
MFAS – Multi Frame Alignment Signal	
TTI – Trail Trace Identifier	

1.1.Тенденции и требования рынка телекоммуникации. Значение волоконно-оптических систем для современных компьютерных сетей.

<u>Эти данные Смелянский приводил на последней лекции, там будущее – это 2020 год, цифры вроде норм</u>:

Тенденции развития рынка информационных технологий

Мобильность

Виртуализация

Консолидация инфраструктуры

10.12.2015 проф. Смелянский Р.Л. Доп.главы комп. сетей: СХД 58

Тенденции развития рынка информационных технологий

Big Data

Всего с начала 2010 г. объем хранимых данных вырос в 50 раз

Центры обработки данных

- В 2014 году объем мирового рынка колокации в ЦОДах составил $22,8 млрд. Общая площадь размещения оборудования достигла 10,13 кв. км. (451 Research)
- Общее число дата-центров всех типов в 2017 г. вырастет до 8,6 млн (IDC)

Телеком

- Каждый из пользователей глобальной сети генерирует больше трафика, чем вся Всемирная паутина 30 лет назад
- В 2014 году интернет-трафик вырос, по сравнению с 1984 годом, в 2,7 миллиардов раз (Cisco)

10.12.2015 проф. Смелянский Р.Л. Доп.главы комп. сетей: СХД 59

<u>Следующие данные Смелянский приводил на первой лекции, и их он взял из старых 2-х, 3-х летних источников (там 2015 был будущим)</u>:

Рост пропускной способности в кабельных и беспроводных сетях: с 2000 года примерно в 10 раз каждые 5 лет.
LTE уже 10Мбит/с, FTTH – Fiber to the house – 100 Мбит/с и выше
2013 год – мобильные составляют 70% всех клиентов, почти 300 млн.

Количество голоса в мобиле почти не меняется последние 5 лет, а количество передаваемой информации растёт в квадрате.
В 2014 году количество передаваемых данных с проводных и беспроводных устройств сравнялось – 1700. Причём мобила растёт быстрее, чем десктопы.
На дно океанов кидают много оптики. Сейчас уже в атлантике засвечено волокон на 40-50 Тб/с.

OTN – 34% планируемых капитальных расходов мирового рынка.
76% респондентов планируют развертывание коммутации OTN.
OTN играет центральную роль в 77% узлов дальней связи и 46% узлов городских сетей;
94% респондентов признают коммутацию OTN ключевым средством повышения коэффициента использования длин волны 40G и 100G;
69% респондентов считают, что коммутация OTN позволит автоматизировать конфигурацию сети для реализации новых услуг.

Тенденции и требования рынка:
- Изменение модели вычислений (outsourcing & robosourcing);
- Быстрый рост траффика: к 2016 году объем трафика возрастет в 6 раз
- Изменение структуры траффика: к 2016 г. 90% - видеотрафик;
- Взрывной рост мобильности;
- Несоответствие темпов роста трафика и темпов роста доходов операторов

Финансовые роботы на биржах – борьба за миллисекунды в скорости доступа к биржевой информации.
Компании стараются избавляться от своих вычислительных центров, сокращая затраты на инфраструктуру и ее поддержку, за счет аутсорсинга услуг.

Закон Гилдера: рост пропускной способности сетей в целом, как минимум, в три раза превышает вычислительную мощность компьютеров, (т.е. компьютеры не генерируют столько трафика, сколько способна передать сеть.)
(*заметка: закон сомнителен был даже раньше*)
Закон Гилдера нас не спасает более: трафик нарастает быстрее роста пропускной способности сетей. Любые вычисления при передаче трафика снижают скорость передачи.

Стоимость часа простоя (финансовый сектор – млн $, медиа – млн $, торговля – тыс $):
Финансовый сектор - Брокерские операции - 6,5 млн.долл.
Финансовый сектор - Авторизация оплаты с помощью кредитной карты - 2,6 млн.долл.
Медиа - Платное кабельное телевидение - 1,1 млн.долл.
Розничная торговля - «Магазин на диване» (ТВ) 113 тыс.долл.
Розничная торговля - Торговля по каталогу 90 тыс.долл.
Перевозки - Бронирование авиабилетов 89,5 тыс.долл.

Сеть – это платформа для услуг, непрерывно пополняемых и изменяющихся.

Услуги на базе облака (высокая пиковая скорость, низкие задержки, устойчивость к потерям), высокопроизводительные услуги (низкая задержка без потерь, отсутствие джиттера, надёжность), специальные услуги (выделенная инфраструктура, низкие задержки без потерь, надёжность, быстрое восстановление) – это всё то, что может предоставить OTN – Optical Transport Network (и только он).

Оптика – коммутация каналов, а на прикладном уровне идёт коммутация пакетов, вопрос как сделать коммутацию пакетов на уровне оптики – не решённый вопрос на сегодня.

1.2.Структура и типы услуг современной сети телеком оператора. Физические свойства носителей.

Современная бизнес-модель:
Доступ к транспорту должен быть бесплатным, платным должен стать контент и услуги.

Информация о пользователе:
- Использование: посещаемые сайты, звонки и сообщения (включая тип сообщений и их частоту)
- География: где находится мобильное устройство в конкретный момент (уровень точности может разниться от района к району)
- Демография: доход домохозяйства, число и возраст проживающих детей
- Уровень дохода: тарифный план, история платежей, паттерн совершения покупок
- Мультиплатформенность: использование данных на разных устройствах и типах подключения к сети (3G, WiFi и т.п.).
- 2011 - AT&T – запуск подразделения AdWorkds: поддержка целевой рекламы в web, мобильной среде и ТВ.
- 2013 – AdWorks открывает доступ к анализу данных 70 млн.пользователей.
- 2012 – Verizon - запуск инициативы Precision Market Insights – доступ к мобильным данным пользователей для маркетинговых и рекламных компаний.

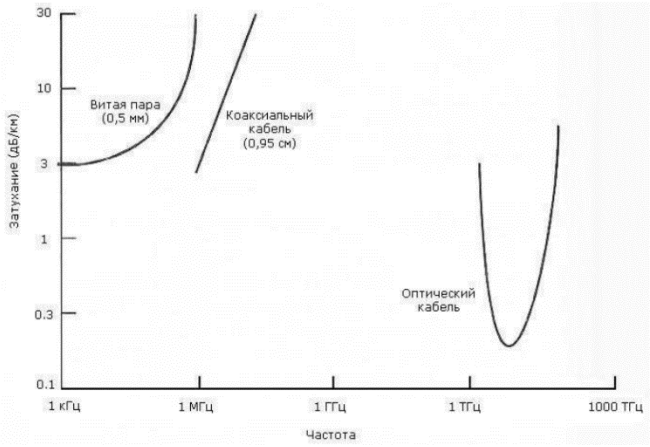
!!! типы услуг современной сети телеком оператора – что на это отвечать?

Устройство сети обычного телеком оператора:
Сеть насыщена региональными ЦОД, и сетевым оборудованием, функционирующим автономно.
Используется дорогое оборудование зарубежных вендоров.
Требуется дорогостоящая поддержка специалистов высокой квалификации.

Обычно есть некоторая центральная сеть – «core», от которой ответвляются aggregation контура (например, с CDN), от которых ответвляются региональные access хвосты, в которых могут стоять BRAS, NAT, …
Строже иерархия выглядит как: Long-haul -> Metro core -> Metro access -> Premise (ATM, Modems, Wireless)
OTN должно использоваться везде, кроме Premise.

Дополнительно (не часть билета):
***Развитие систем дальней связи**:*
TDM 1980-2000 - Скорость электронных компонентов - Максимальная скорость ~ 10 Гбит\с
WDM - Wavelength Division Multiplexing (Спектральное уплотнение каналов (спектральное мультиплексирование))
DWDM – Dense Wavelength-division multiplexing (плотное частотное разделение каналов)
CWDM - Coarse Wavelength Division Multiplexing (Грубое спектральное мультиплексирование)
увеличение скорости в 100 (DWDM) раз путем многоканальной передачи (плотность расположения каналов: 100 ГГц или 50 ГГц (0,8 или 0,4 нм)
CWDM – увеличение в 18 (макс) обычно в 8 раз (плотность расположения каналов: 20 нм)
Когерентные системы связи + многоуровневые форматы
один символ переносит 4 и более бит
Новые типы волокна: затухание в 1970 году 10дБ/км; в настоящее время 0,2 – 0,16 дБ/км; предел 0,14 дБ/км
Усилители: до 100 пролетов по 100 км
Коррекция ошибок: увеличение длины пролетов и/или их числа
Когерентные системы связи: больше чувствительность приемников (меньше требуемый OSNR), медленнее накопление нелинейных искажений
OSNR – Optical Signal-to-Noise Ration
ROADM – Reconfigurable Optical Add/Drop Multiplexer
AWG – American Wire Guide
Главное: laser module -> + time division multiplexing -> optical fiber amplifier -> +wave length devision multiplexing

Затухание в кабельных средах:



Физические свойства носителей:

Длинна волны * частоту = скорость света

	Frequency range	Typical Attenuation	Typical Delay	Repeater Spacing
Twisted pair (with loading)	0 to 3.5 kHz	0.2 dB/km @ 1kHz	50 мкс/км	2 km
Twisted pairs (multi-pair cables)	0 to 1 MHz	3 dB/km @ 1kHz	5 мкс/км	2 km
Coaxial cable	0 to 500 MHz	7 dB/km @ 10MHz (Sin - Sout =~5)	4 мкс/км	1 to 9 km
Optical fiber	180 – 370 THz	0.2 to 0.5 dB/km (Sin – Sout =~ 0.25)	5 мкс/км	40 km

Для оптики сильно устарело, т.к. это очень древняя таблица из его книжки.

Затухание в оптическом кабеле:

в мультимодовом кабеле (850нм) – 2.7 dB/км;

в мультимодовом кабеле (1310нм) – 0.75 dB/км;

в одномодовом кабеле (1310-1450нм) – 0.35 dB/км;

в одномодовом кабеле (1470-1610нм) – 0.25 dB/км;

Достоинства оптического спектра – при длине волны 1.4 мкм частота = (299 792 458) / (1 400) 10^-9 = 299792458*10^7/14 = 250 ТГц

В то время как wireless сети работают на частотах 1-10 GHz

Связь ширины спектра в нм с шириной спектра в ТГц:

Т.к. скорость света = частота * длинна волны, то неизменная ширина спектра при разных абсолютных частотах дают разную ширину волны.

Частота (ν) [ТГц]	Ширина спектра (Δν) [ТГц]	Ширина спектра (Δλ) [нм]	Коэфф.Кперевода [нм/ТГц]	Длина волны (λ) [нм]
200	0,1	0,75	7,5	1500
193,3	0,1	0,80	8,0	1550
187,3	0,1	0,85	8,5	2424

Спектр затухания сигнала в оптоволокне:

Диапазоны длин волн оптоволокна:

O - Original (основной) 1260…1360 нм

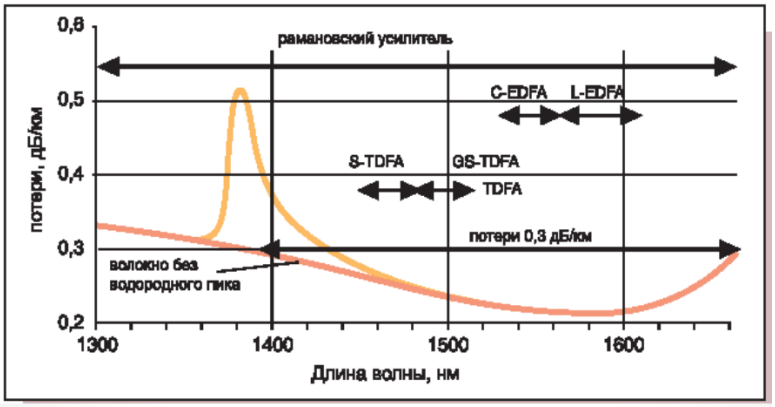
E - Extended (расширенный) 1360…1460 нм

S - Short wavelength (коротковолновый) 1460…1530 нм

C - Conventional (стандартный) основной 1530…1565 нм

L - Long wavelength (длинноволновый) 1565…1625 нм

U - Ultra-long wavelength (сверхдлинный) 1625…1675 нм

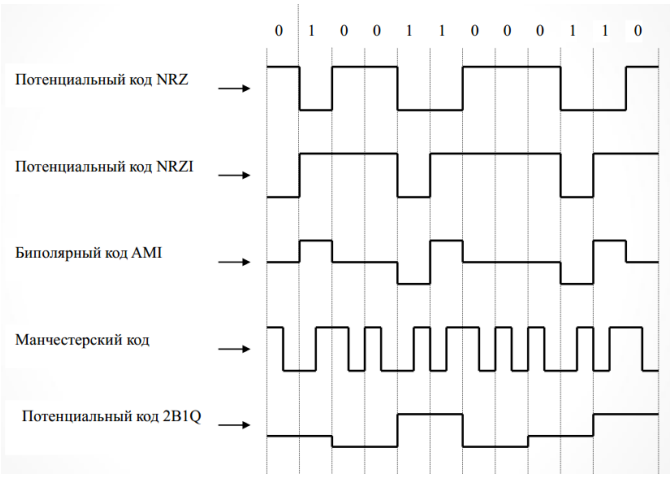


1.3.Теоретические основы передачи электромагнитных сигналов. Основные понятия и их взаимосвязь. Способы кодирования данных.

Скорость передачи, определения:
Число импульсов-символов, передаваемых системой связи за 1 секунду, называется **символьной скоростью** B_S, единица измерения – **бод**.
Количество информации, передаваемой системой связи за 1 секунду, называется **скоростью передачи информации (битовой скоростью)** $B_{INF} = B_S \cdot \log_2 M$
Если используется К не зависимых каналов передачи информации, то суммарная скорость передачи информации увеличивается в К раз: $B_{TOT} = B_{INF} \cdot K$

Взаимосвязь пропускной способности канала и его полосы пропускания
- шум в канале измеряется как соотношение мощности полезного сигнала к мощности шума: S/N (измеряется в децибелах 1dB = 10 $\log_{10}$ (S/N)).
- для случая канала с шумом есть Теорема Шеннона V_{max} (бит/с) = H (Гц) $\log_2$ (1+S/N) bps, где S/N - соотношение сигнал-шум в канале (обычно отношение мощностей); здесь уже неважно количество уровней в сигнале, это - теоретический предел, которой редко достигается на практике.

Способы кодирования данных:



Потенциальный код NRZ (0 – высокий потенциал, 1 – низкий потенциал)
Биполярный код NRZI (0 – нет перепада уровня сигнала в начале битного интервала, 1 – перепад уровня сигнала в начале интервала)
Биполярный код AMI (0 – отсутствие сигнала, 1 – положительный или отрицательный потенциал, обратный по отношению к потенциалу в предыдущий период)
Манчестерский код (0 – переход с высокого на низкий потенциал в середине интервала, 1 – переход с низкого на высокий потенциал в середине интервала)
Потенциальный код 2B1Q (Использует 4 уровня сигналов, значение уровня определяет значение пары битов данных)

NRZ – Non-Return to Zero – без возврата к нулю на битовом интервале
- Основным недостатком этого кода является отсутствие синхронизации.
- Модификацией NRZ кода и хорошим примером дифференциального кодирования является NRZ-I код

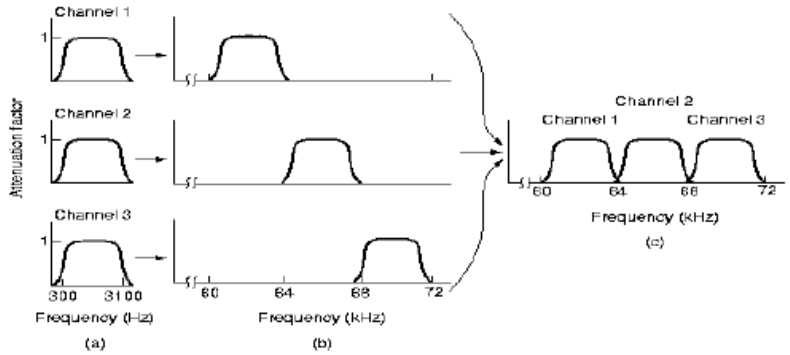
В линиях связи без усилителей источник шумов – приёмник.

Дифференциальные коды (коды, у которых всегда есть перепад, не зависимо от передаваемого бита) – хороши автосинхронизацией, но плохи тем, что для них нужна повышенная частота (или возможность АЦП распознавать более 2-х уровней)

1.4.Методы ускорения передачи. Мультиплексирование. Примеры стандартов мультиплексирования.

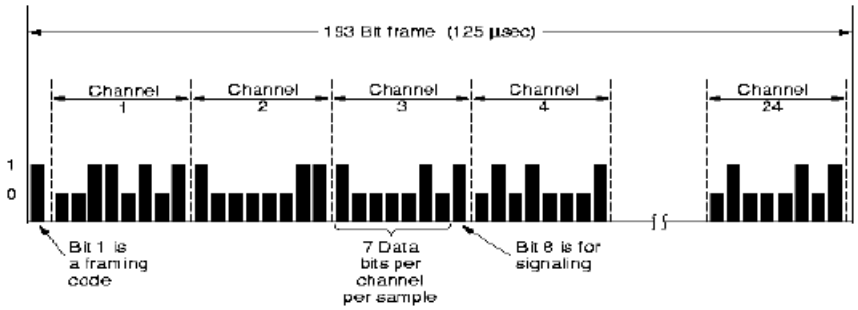
!!! Методы ускорения передачи – это он мультиплексирование имеет ввиду?

Мультиплексирование – уплотнение:
- Мультиплексирование с разделением частот.



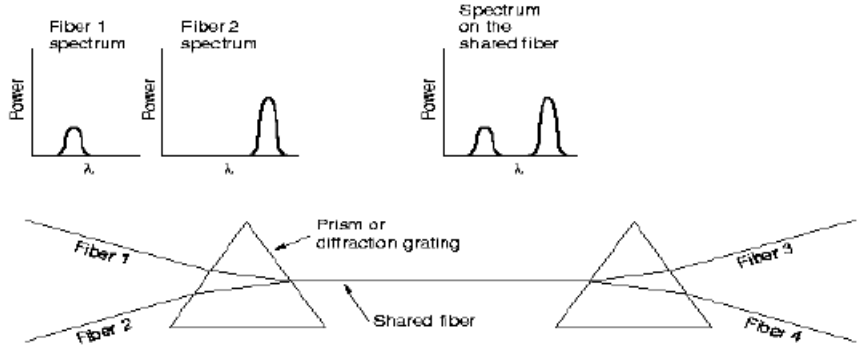
Частоты в кабеле не перемешиваются + их можно друг от друга отделить.

- Мультиплексирование с разделением по времени. (65 нс на бит)



Тут просто в один кадр по очереди укладываются биты из разных потоков.

- Спектральное мультиплексирование (по длине волны).

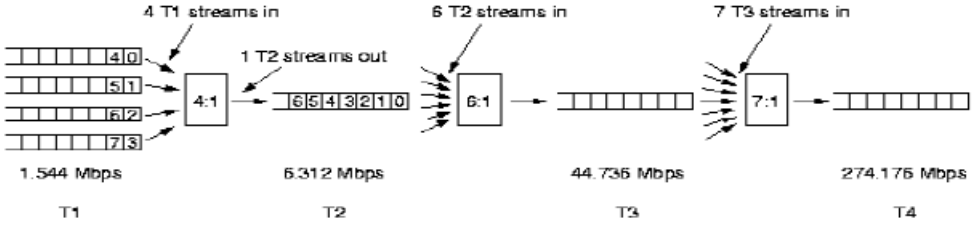


Здесь суть в линзе – разные длинны волн/частоты имеют разные коэффициенты преломления, поэтому пучёк можно сложить, передать по одному кабелю, а потом разложить.

Смелянский этого вроде не говорил:

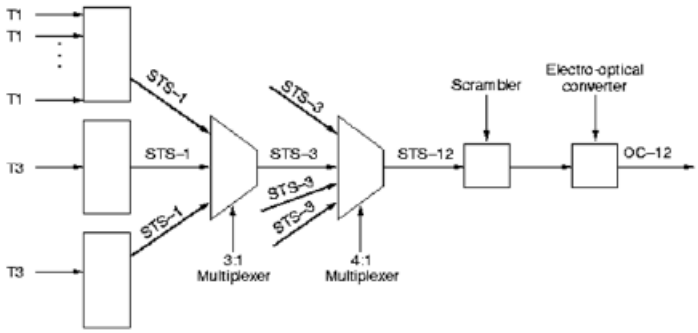
В чём разница между мультиплексированием спектральным и по частоте? – физически – никакой, т.к. частота умножить на длину волны = скорость распространения сигнала (т.е. скорость света (говорят, что это константа))
Просто исторически так сложилось, что в радиоэлектронике такое мультиплексирование называют «по частоте», а в оптике это называют «по длине волны» («спектральное»).
(в радиоэлектронике частоты отделяют друг от друга резонансными контурами из конденсатора и катушки, а в оптике – это делают при помощи линзы)

Мультиплексирование по стандарту T:



На уровне T3 уже 0.5 нс на бит. (4-6-7)
Приходящие потоки называются притоками.

Мультиплексирование в SONET и SDH:



Согласно стандарту T1, 4 канала T1 (4* 1.5Мб/с) ~ 6Мб/с могут быть объединены в один T2, затем 6 T2 ~ 36 Мб/с в один T3 и 7 T3 в один T4. Тем самым на уровне T4 максимальная скорость равна 274,176 Мбит/с. Согласно E1 (это цифровой поток передачи данных, пришедший из телефонной связи, там 30 потоков по 64 кбит/с и ещё 2 таких же для сигнализации), группировать можно только 4 канала, но зато есть 4 уровня вложенности, а не три, как в T1. Поэтому скорость передачи в этом случае E1 = 2,048; E2 = 8,848; E3 = 34,304; E4 = 139,264; E5 = 565,148 Мбит/с.
!!! Здесь картинки не соответствуют сказанному тексту + странные какие вылезли T1 и E1

Мультиплексирование происходит побайтно. Например, когда три STS-1 притока, каждый со скоростью 51,84 Мбит/с, объединяют в один STS-3 приток со скоростью 155,52 Мбит/с, мультиплексор сначала берет 1-й байт 1-го притока, затем 1-й байт 2-го притока, затем 1-й байт 3-го. Только после этого он переходит ко вторым байтам этих притоков. Кадр STS-3 состоит из 270x9=2430 байтов и занимает 125 мкс. Таким образом, на этом уровне битовая скорость равна 155,52 Мбит/с. На слайде приведены основные данные об иерархии мультиплексирования в SONET и SDH.

SONET		SDH	Data rate (Mbps)		
Electrical	Optical	Optical	Gross	SPE	User
STS-1	OC-1		51.84	50.112	49.536
STS-3	OC-3	STM-1	155.52	150.336	148.608
STS-9	OC-9	STM-3	466.56	451.008	445.824
STS-12	OC-12	STM-4	622.08	601.344	594.432
STS-18	OC-18	STM-6	933.12	902.016	891.648
STS-24	OC-24	STM-8	1244.16	1202.688	1188.864
STS-36	OC-36	STM-12	1866.24	1804.032	1783.296
STS-48	OC-48	STM-16	2488.32	2405.376	2377.728

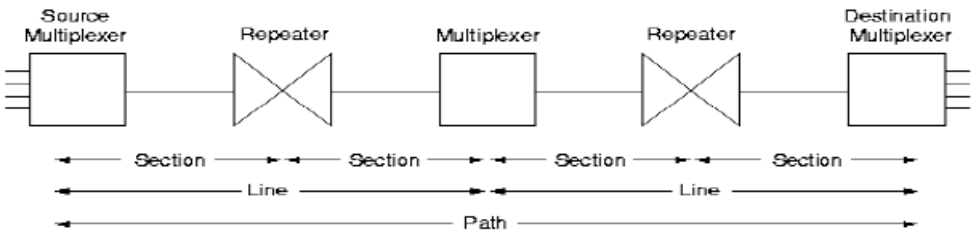
1.5.Synchronous Digital Hierachy (SDH): основные понятия, структура и иерархия протокольных единиц данных (PDU).

!!! Иерархия протокольных единиц PDU для SDH ???? Может быть вопросы спутались, и имелось ввиду не здесь, а в OTN спрашивать про PDU?

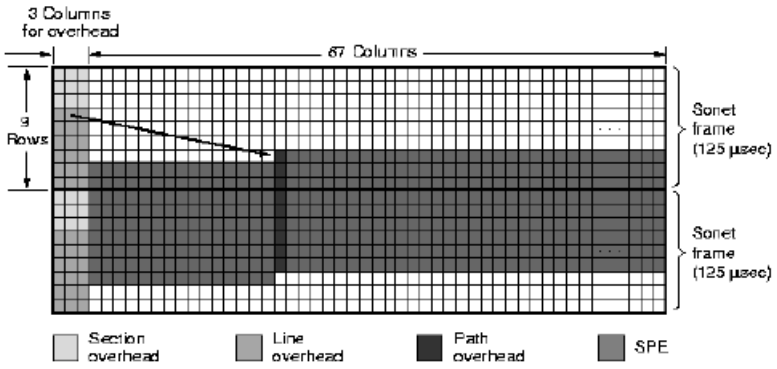
SONET/SDH (на сегодня используется до уровня T3, хотя стандарт позволяет и до T4) (SONET используется в Америке, SDH – в Европе (и у нас), но в целом это одно и то же):
Стандарт должен был позволить:
- использовать разные физические среды в сети. Это требовало проработки стандарта на кодировку на физическом уровне, выбор длины волны, частоты, временных характеристик, структуры фрейма.
- обеспечить иерархическое мультиплексирование нескольких цифровых каналов. Унифицировать Американские, Европейские и Японские цифровые системы.
- определить правила функционирования, администрирования и поддержки.

На уровне оптики происходит (без преобразования в электричество):
В системе единые часы (генератор импульсов)

Система передачи данных SONET состоит из коммутаторов, мультиплексоров и повторителей, соединенных оптическими линиями. На рисунке – определения «секция», «линия», «путь»



Кадр SONET содержит 810 байт и занимает 125 мкс. SONET допускает разные топологии каналов связи, но чаще это двунаправленное кольцо (почему это так, мы увидим, когда будем рассматривать системы FDDI). Так как система SONET синхронная, то кадры генерируются строго один за другим без перерывов вне зависимости от того, есть данные на передачу или нет. Скорость в 8000 кадров/с как раз соответствует каналам с ИКМ-модуляцией, используемым в цифровой телефонии. Исходя из этого, нетрудно подсчитать, что пропускная способность канала SONET равна 51,84 Мбит/с.



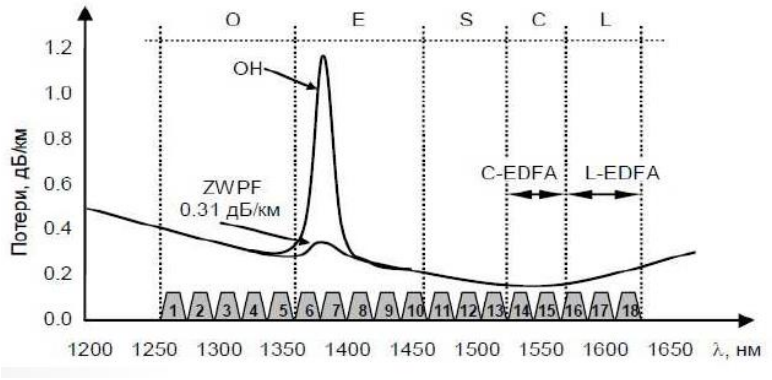
Для описания кадра SONET представим его 810 байт в виде матрицы 9 строк на 90 столбцов. Каждый элемент матрицы – один байт. Первые три элемента в каждой строке – это служебная информация, используемая для администрирования и управления передачей. Первые три элемента первых трех строк образуют заголовок секции, в следующих 6 строках – заголовок линии. Заголовки секции генерируются и проверяются в начале и в конце каждой секции. Аналогичным образом поступают на каждой линии с заголовком линии. 8000 кадров в секунду образуют основной канал, называемый Synchronous Transport Signal-1 (STS-1).
Оставшиеся в 87 столбцах и 9 строках 783 байта приходятся на данные пользователей, которые образуют так называемый SPE-конверт (Synchronous Payload Envelope). SPE-конверт содержит как данные пользователя, так и служебную информацию, которая занимает 13 байт. SPE-конверт может начинаться с любого байта в оставшихся 9x87 байтах. Начало SPE-конверта указано в первых двух байтах 3-й строки. Учитывая, что в SONET генерируется 8 000 кадров в секунду, получаем, что полезная пропускная способность составит 8000 x 783 x 8 = 50,112 Мбит/с. Нетрудно видеть, что такая организация работы канала предполагает плотную его загрузку со стороны абонентов (сравните с организацией работы каналов по IEEE 802).
Итак, есть следующие элементы в канале: section overhead, line overhead, path overhead, SPE.

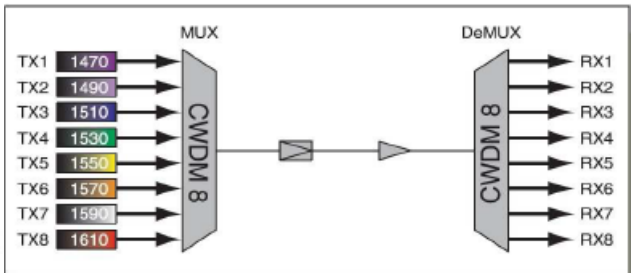
Иерархия section -> line -> path важна для реализации того, что в OTN называют TCM.
TCM на пальцах: аналогично тому, как это есть в SDH – в OTN присутствует многоуровневость, и есть аналогичные section, line и path. В случае если с каналом что-то случилось, и посланный сигнал не дошёл до получателя, то получатель должен как-то уведомить об этом отправителя. Для этого в обратном направлении (другом кабеле), шлётся соответствующий флаг в заголовке. Другое устройство получив сообщение об ошибке, перестраивается на использование альтернативного канала/маршрута.
Важность многоуровневости в том, что если на текущем уровне система смогла починить поломку, то она не рапортует об этом вышестоящему уровню, а если не смогла, то и в вышестоящем уровне шлются флаги ошибки, и тогда уже тот уровень пытается использовать альтернативный путь.
Важно то, что всё это происходит аппаратно, а не программно, в связи с чем задержка на восстановление канала занимает меньше 50 мс (при мелкой поломке вообще в 5 мс можно уложиться). В то время как программно это никак не меньше 300 мс.

1.6.WDM системы передачи данных: основные понятия, емкость, принципы работы, форматы модуляции сигналов.

Принцип работы:

Стандартом определены 18 спектральных каналов CWDM (Coarse Wave Devision Multiplexing), обычно используются 8.

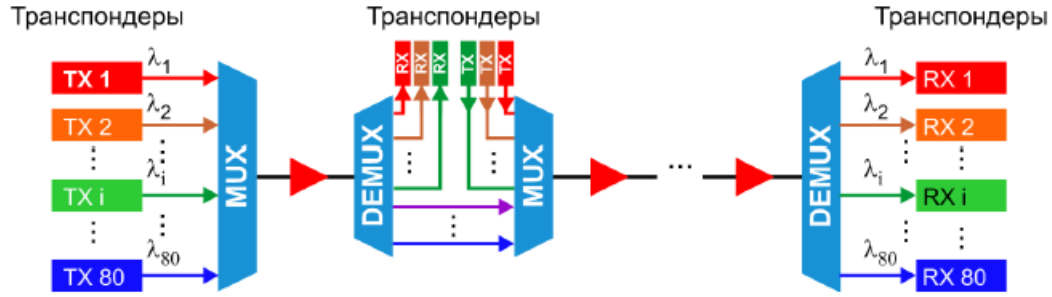




На 2-й картинке треугольничек – это мультиплексор.

Всё стало возможно с появлением соответствующих когерентных излучателей.

DWDM (Dense Wavelength-division multiplexing):



• Прозрачная передача протоколов: OTN OTU1/2/3/4, SDH STM-1/4/16/64/256, Ethernet FE/GE/10GE/100GE и др.

• Одновременное усиление всех спектральных каналов.

• Высокая емкость сети при одновременной передаче множества каналов.

• Быстрый апгрейд за счет ввода новых каналов без остановки старых. Мультисервисность.

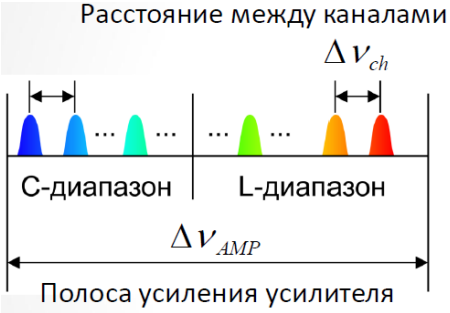
• Высокая скорость, до 160 каналов по 40 Гб/с.

• Длина регенерационного участка до 2000км. Одновременное усиление всех каналов одним оптическим усилителем.

• Объединение сетей различных производителей

• Не требует единой синхронизации

Ёмкость системы передачи:



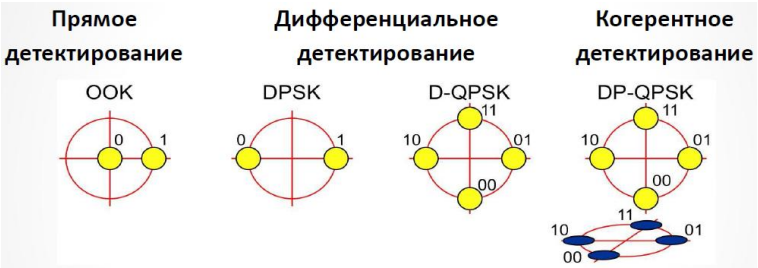
Число каналов: $N_{ch} = \Delta\nu_{AMP} / \Delta\nu_{ch}$

Скорость передачи: $B_\Sigma = B_{ch} \cdot N_{ch}$

Спектральная эффективность: $SE = B_\Sigma / \Delta\nu_{AMP} = B_{ch} / \Delta\nu_{ch}$

При $\Delta\nu_{ch}$ = 50 ГГц и B_{ch} = 100 Гбит/с, SE = 2 (бит/с) Гц

Форматы модуляции:



• OOK (On-Off Key) (Так же используется обозначение NRZ ASK) – при этой модуляции единицам “1” соответствует наличие оптического излучения, нулям “0” отсутствие излучения.

• DPSK – нулю и единице соответствуют сигналы, несущие которых смещены друг относительно друга на π, амплитуда излучения постоянна.

• D-QPSK – в одном символе содержится информация сразу о двух переданных битах, четырем значениям символа соответствуют четыре фазы: 0, π/2, π, 3π/2
• DP-QPSK – передаются два независимых потока QPSK в двух поляризациях
Похоже Смелянский в 2-х средних случаях подразумевал фазовую модуляцию. (вообще существуют понятия обо всех 3-х модуляциях – фазовой, частотной и амплитудной (хотя частотную модуляцию не пользуют на практике))

1.7.Optical Transport Hierarchy (стандарт ITU G.709): назначение, базовые механизмы, достоинства.

OTN/OTH ITU G.709 назначение:
"…to cater for the transmission needs of today's wide range of digital services, and to assist network evolution to higher bandwidths and improved network performance." - … чтобы удовлетворить потребности передачи данных современного широкого спектра цифровых услуг, и, чтобы помочь сети эволюционировать до более высокой пропускной способности и улучшенной производительности сети.

Базовые механизмы OTN:
- Усиленный механизм обнаружения ошибок – «FEC»
- Многоуровневый сквозной мониторинг соединений - Tandem Connection Monitoring (TCM)
- Прозрачная передача данных пользователя.
- Многоуровневая коммутация («switching scalability»).
- *наверно, я бы ещё добавил – возможность упаковывать и передавать совершенно разные (хоть проприетарные) протоколы выше L1, т.е. OTN – это такая труба, в которую можно бросить что угодно, хоть другой OTN или SDH или ip, … и на выходе из трубы получится абсолютно тоже самое с точностью до времён и даже частот.*

Достоинства иерархично: Управление, Мониторинг, FEC
Минусы: требует нового оборудования и смену системы управления.

FEC (Forward Error Connection):
- увеличить длину оптических линий
- число каналов в DWDM системе (можно плотнее ужать частоты)
- снизить мощность сигналов
- снизить требования к оптическим компонентам приема/передачи, что снижает их стоимость
- увеличить число ретрансляторов: передача аналоговая => происходит усиление шума при каждой ретрансляции сигнала, снижение порогового значения S/N позволяет увеличить число ретрансляций.

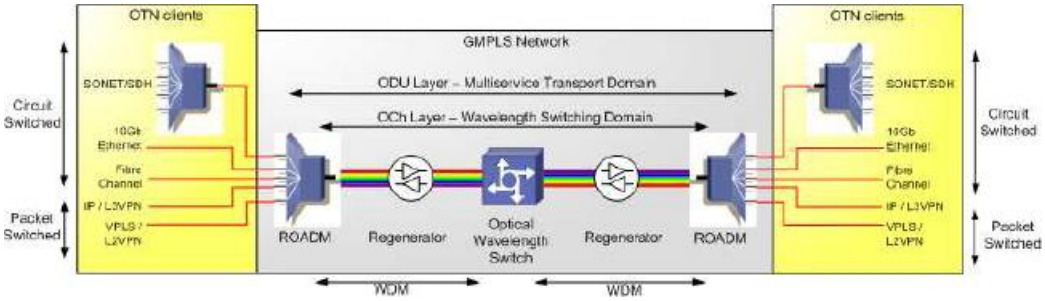
В качестве FEC стандарт рекомендует использовать Reed-Solomon Code RS (255, 239)

t = длина символа = 8 бит
n = число символов в кодослове = 255 байт
k = количество информационных символов = 239 байт
2t = n-k = 255 – 239 = 16, t = 8
RSS (255, 239) max length $n = 2^t - 1 = 255$
Сам Reed-Solomon построен на принципах кода Хемминга, т.е. добавление битов четности для группы битов. 16 контрольных байт позволяют выявлять и корректировать до 8 байт. RS код выявляет ошибки на уровне символа, т.е. байта. Не важно сколько ошибочных бит в байте: все 8 или только 1 бит.
Code word в ODU мультиплексируются, что позволяет эффективно бороться с групповыми ошибками.
Например, если 64 codewords с возможностью исправления до 8 символов у каждого замультиплексировать, то получим структуру, где можно обнаруживать и справлять до 512 ошибок! И не важно в байте будет одна ошибка или все 8 бит.

Сквозной мониторинг на всех уровнях (TCM – Tandem Connection Monitoring):
TCM на пальцах: аналогично тому, как это есть в SDH – в OTN присутствует многоуровневость, и есть аналогичные section, line и path. В случае если с каналом что-то случилось, и посланный сигнал не дошёл до получателя, то получатель должен как-то уведомить об этом отправителя. Для этого в обратном направлении (другом кабеле),

шлётся соответствующий флаг в заголовке. Другое устройство получив сообщение об ошибке, перестраивается на использование альтернативного канала/маршрута.
Важность многоуровневости в том, что если на текущем уровне система смогла починить поломку, то она не рапортует об этом вышестоящему уровню, а если не смогла, то и в вышестоящем уровне шлются флаги ошибки, и тогда уже тот уровень пытается использовать альтернативный путь.
Важно то, что всё это происходит аппаратно, а не программно, в связи с чем задержка на восстановление канала занимает меньше 50 мс. В то время как программно это никак не меньше 300 мс.

- В SDH был транзитный мониторинг, но только внутри домена одного оператора. Здесь эта концепция получила расширение и на междоменный мониторинг.
- Конечный пользователь может мониторить QoS своего соединения, есть два уровня мониторинга внутри домена (на уровне линии и секции), есть мониторинг на меж-доменном уровне.
- G.709 определяет OPU, который может содержать весь SONET/SDH сигнал, т.е. может передавать четыре STM-16/OC-48 сигнала в одном OTU2 и при этом не модифицировать SONET/SDH заголовки. Таким образом передача таких сигналов от клиента полностью прозрачна (целостность сигнала не нарушается).
Оно также и прозрачно по времени. (*как я понимаю,* имеется ввиду, что относительное время в передаче битового сигнала клиента – не изменится после транспортировки через OTN (причём, это будет работать даже в случае, когда тактовая частота сигнала, который передаётся, не равен тактовой частоте сети, через которую передаём упакованный, например, SDH))
Масштабируемая упаковка:



Когда SONET/SDH был создан в середине 80-х, его основным предназначением была передача голосовых данных. Поэтому было определено два уровня упаковки: на скорости 1.5/2 Мб/с (E1/E1) и на скоростях 50/150 Мб/с. Однако скорости линий росли и разрыв в скоростях рос. Через некоторое время стало не эффективно использовать для высокоскоростного Ethernet, IP имеющуюся упаковку в SONET/SDH.
OTN был призван решить эту проблему и предоставить несколько уровней упаковки для разных видов сервисов.
Так в OTN может поддерживать разные скорости сервисов 1GE, 10GE, 40GE, 100GE, а также скорости соответствующие иерархии SDH (Signal Digital Hierarchy).

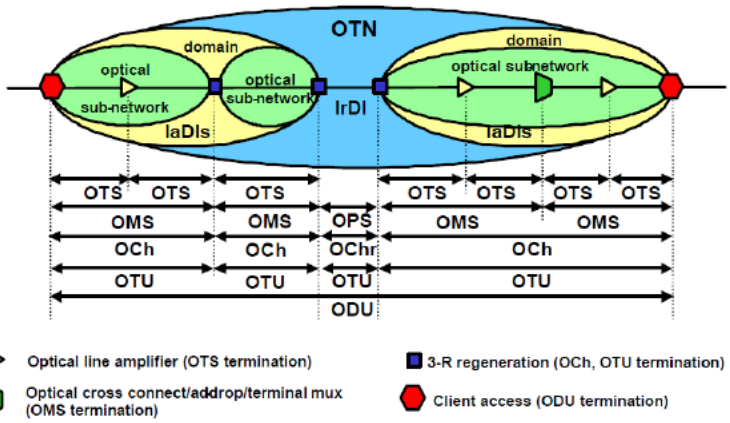
ROADM – Reconfigurable Optical Add/Drop Multiplexor.
Со временем, скорость линии выросла, в то время как скорость переключения (коммутации) не менялась, разница активно росла, появлялись новые сервисы, требовавшие большую пропускную способность.
Смежная и виртуальная конкатенация должны были решить проблему для некоторых сервисов, т.к. они позволяли работать выше скорости SONET/SDH битовой скорости коммутации. Но тем не менее разница до сих пор существует, даже не смотря на конкатенацию на уровнях STS-1/VC-4.
Для 4x10G до 40G SONET/SDH мультиплексора это означало 256 VC-4 в случае SDH или хуже, в случае SONET нужно обработать 768 STS-1-SPE. Это приводило к дополнительным нагрузкам не только на оборудование, но и на поддержку и настройку.
Идеальным решением была бы коммутация на уровне фотонов, но тогда она была бы привязана к битрейту конкретной длинны волны, а значит был бы привязан и сам сервис. Независимое решение – не возможно.
OTN предоставляет решение проблемы, по мене роста скорости передачи бит, дополнительная коммутация битового потока добавляется. В итоге оператор может предоставлять сервисы с разными скоростями (2.5G, 10G, …), не зависимо от длинны волны, за счёт мультиплексинга OTN.

1.8.Иерархия интерфейсов и PDU в стандарте ITU G.709. Примеры применения OTN.

Структура OTN на пальцах:

Нижним уровнем иерархии является Optical payload unit (OPU).
OPU передается из конца в конец всего тракта передачи сигнала, т.е. между терминальными мультиплексорами. Служебная информация OPU выполняет две функции:
– определение типа передаваемого сигнала (поле PSI). Специальное значение байта 0 PSI 20h показывает, что OPU содержит мультиплексированный сигнал (несколько ODU более низкого уровня), а байты 2-17 в этом случае определяют тип и номер каждого потока в мультиплексе.
- передача сигнала синхронизации в случае, если передаваемый сигнал – асинхронный (поле JC)
Таким образом, уровень OPU решает задачу инкапсулирования полезного сигнала в сигнал OTN и задачу мультиплексирования сигналов.
Следующим уровнем иерархии является Optical Data Unit (ODU).
ODU также передается из конца в конец тракта, однако ее функции связаны не с самим сигналом, как таковым, а с реализацией задач управления и мониторинга всего тракта передачи сигнала между конечными узлами. ODU выполняет следующие функции:
- передача в обратном направлении аварийных сообщений (PM)
- передача служебной информации при прохождении тракта по сетям различных операторов (поля TCMi, TCMACT)
- передача информации об обнаруженных ошибках и предполагаемом месте их расположения (поле FTFL)
- передача служебной информации из конца в конец тракта (поля GCC1/GCC2)
- передача информации о переключении основного и служебного каналов на резервный путь (вложенные поля APS/PCC)
Таким образом, уровень ODU решает задачи мониторинга и управления тракта передачи в целом, из конца в конец.
Верхним уровнем иерархии является Optical Transpot Unit (OTU).
В отличие от двух предыдущих уровней, информация на уровне OTU передается только в рамках секции мультиплексирования.
Уровень OTU выполняет следующие функции:
- framing, т.е. разбивка сигнала на кадры и мультикадры (поля FAS/MFAS)
- передачу обратного сигнала об обнаруженных в пределах секции мультиплексирования ошибках (SM)
- передача служебной информации в пределах секции мультиплексирования (поле GCC0)
- передача информации, необходимой для коррекции ошибок (FEC)

Структура OTN (специфицирован в G.872):



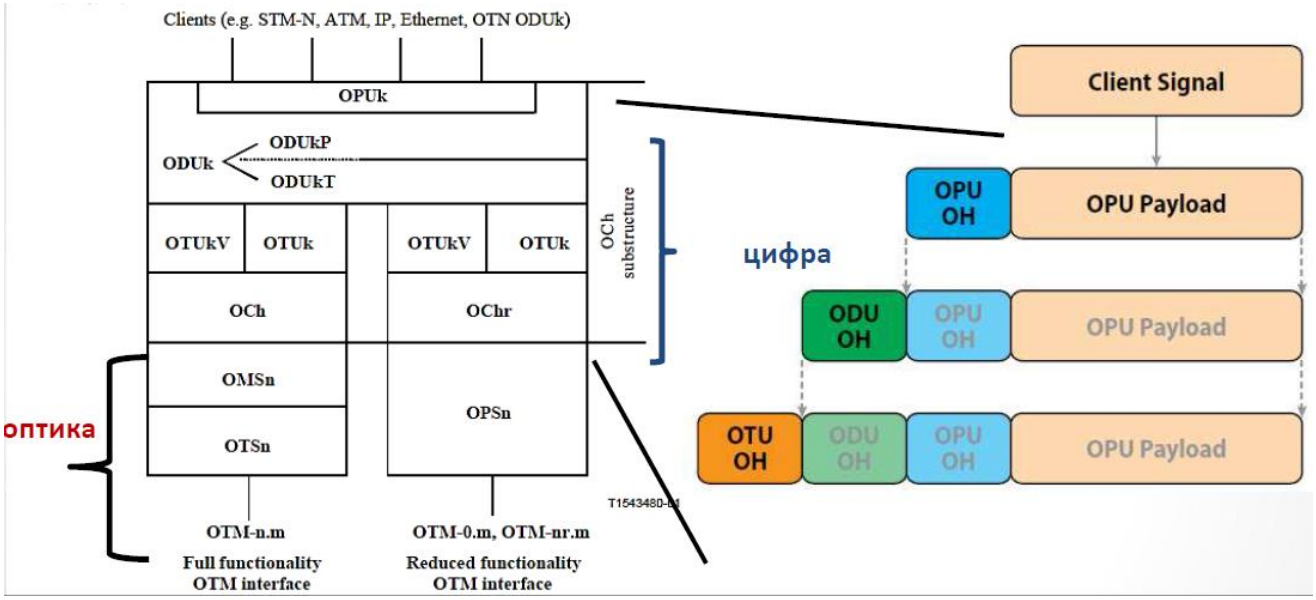
ODU -> OTU -> OCh -> OMS -> OTS

2 вида интерфейса:
- IrDI – (обычно на границе оператора) - inter-domain interface – содержит 3R обработку (Reamplification, Reshaping, Retiming) (усиление, восстановление (данных), восстановление синхронизации) на обоих концах интерфейса
- IaDI – (обычно внутри оператора) - intra-domain interface – без 3R обработки

OTM-n – Optical transport module-n – модуль, используемой для поддержки OTN интерфейсов. Их бывает 2 вида:
- OTM интерфейсы с полной функциональностью (OTM-n.m) (OCh – optical channel) (n – максимальное количество частот при мультиплексировании)

- OTM интерфейсы с урезанной функциональностью (OTM-0.m, OTM-nr.m, OTM-0.mvn). (OChr – optical channel reduced) (урезана не 3R обработка – бывает и так, и так, а урезаны показатели физической передачи (например, бесцветная передача))

OTN – иерархия PDU:



OTN поддерживает различные оптические архитектуры: кольцо, point-to-point.
С точки зрения эталонной модели мы находимся на уровне L1, но как видите тут возникает своя многоуровневая иерархия.
OTU – optical channel transport unit - контролирует условия передачи сигнала между конечными точками.
ODU – optical channel data unit, предоставляет:
- «tandem connection monitoring» (ODUkT) – многоуровневый сквозной мониторинг соединений
- end-to-end path supervision (ODUkP) – контроль пути точка-точка
- adaptation of client signals via the optical channel payload unit (OPUk) – адаптация сигналов клиентов посредством OPU
- adaptation of OTN ODUk signals via the optical channel payload unit (OPUk). – адаптация ODU сигналов посредством OPU

Полная структура OTM-n.m (n ≥ 1):
- optical transmission section (OTSn)
- optical multiplex section (OMSn)
- full functionality optical channel (OCh)
- completely or functionally standardized optical channel transport unit (OTUk/OTUkV)
- одна или более optical channel data unit (ODUk).

Типы ODU и их скоростные характеристики:

ODU Clients	ODU Server
1.25 Gbit/s bit rate area	ODU0
–	
2.5 Gbit/s bit rate area	ODU1
ODU0	
10 Gbit/s bit rate area	ODU2
ODU0, ODU1, ODUflex	
10.3125 Gbit/s bit rate area	ODU2e
–	
40 Gbit/s bit rate area	ODU3
ODU0, ODU1, ODU2, ODU2e, ODUflex	
100 Gbit/s bit rate area	ODU4
ODU0, ODU1, ODU2, ODU2e, ODU3, ODUflex,	

Как было отмечено OTN поддерживает несколько уровней упаковки, в зависимости от скоростных характеристик сервиса. Поэтому было введено несколько типов ODU для передачи OPU с данными разных сервисов.

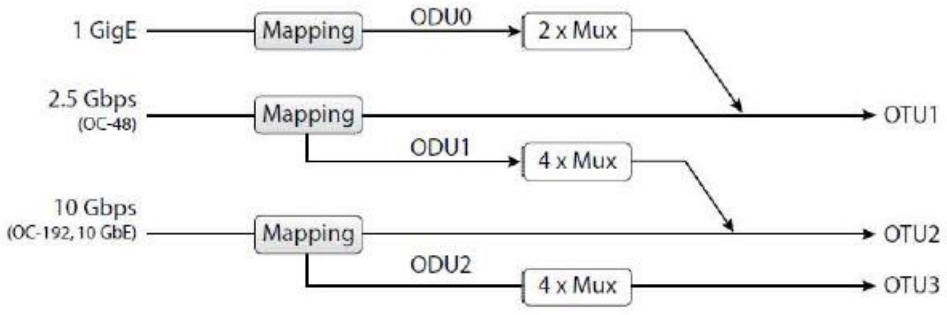
OTU скорости и интерфейсы:

G.709 Interface	Line Rate	Corresponding SONET/SDH Rate	Line Rate
OTU1	2.666 Gbit/s	OC-48/STM-16	2.488 Gbit/s
OTU2	10.709 Gbit/s	OC-192/STM-64	9.953 Gbit/s
OTU3	43.018 Gbit/s	OC-768/STM-256	39.813 Gbit/s

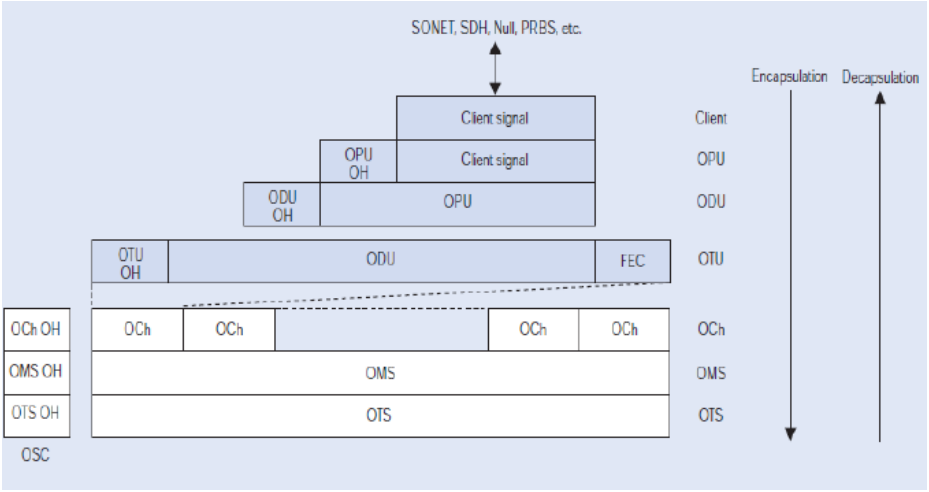
OTU и их соответствие структурам SDH.

По типу OTU однозначно определяется ее внутренняя структура и те вложенные структуры, которые там могут встретиться.

Соответствие между OTU и ODU:



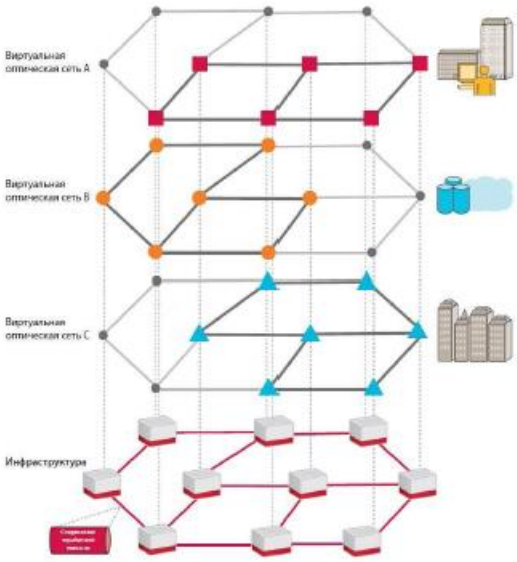
Базовые информационные структуры:



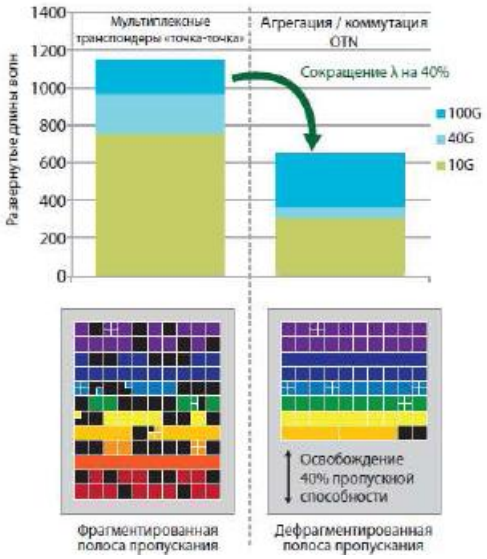
Важно понимать, какие структуры в оптике, а какие в электрике.

Каждый из OTUk (k = 1,2,3) передаётся, используя оптический канал OCh, на специализированной длине волны ITU.
Каждый из OCh, OMS и OTS уровней имеют свои собственные накладные расходы, которые передаются за пределами ITU структуры через OSC канал. Дополнительно OSC предоставляет поддержку сигналов и менеджмент данных на различных уровнях OTN.

Примеры применения OTN:



Новая модель виртуализации. По мере повышения скоростей по длине волны до 400 Гбит/с, 1 Тбит/с и более, OTN, в сочетании с уровнем управления, позволяет реализовать виртуализацию высокопроизводительной сети. На рисунке показана инфраструктура с одной базовой сетью терабитной пропускной способности, в которой созданы три виртуальные сети для поддержки различных моделей предоставления услуг. Например, Сеть A может обеспечить защищенные соединения многосвязной сети с высоким уровнем готовности для поддержки важнейших приложений выделенных линий для различных протоколов коммутации пакетов и хранения данных. Сеть B можно использовать для поддержки облачных услуг, позволяющих клиентам запланировать передачу крупных объемов данных для обеспечения их репликации или миграции виртуальных машин. Сеть C можно спроектировать для предоставления крупномасштабных частных оптических сетей VPN. OTN изначально позволяет поставщику услуг виртуализировать свою инфраструктуру с целью оптимизации предоставления сетевых услуг.



Коммутация OTN позволяет обеспечить оптимальный коэффициент использования пропускной способности, исключая фрагментацию линейной ёмкости, и поддерживает лучшую загрузку длин волны для разнородного клиентского трафика. Благодаря высокому коэффициенту использования длин волны, оптические линии DWDM ULH оптимизируются, устраняется необходимость преждевременного расширения ёмкости сети, как показано на рисунке 7. Оптимизация пропускной способности будет обязательным требованием, когда ёмкость одной длины волны выйдет за рамки 100G, достигнув 400G или 1 Тбит/с.

Недавняя модернизация магистральной сети одного из федеральных операторов в США со скоростями спектрального канала 40 и 100 Гбит/с для, примерно, 3 тыс. каналов 10G позволила сократить загрузку длин волн практически на 40% (т. е., данный поставщик освободил 40% текущей пропускной способности) при внедрении коммутации OTN наряду с 40G/100G. Таким образом, с точки зрения клиентских каналов, оператор сможет увеличить пропускную способность сети на 40% без установки дополнительного магистрального оборудования.

Динамическая инфраструктура. OTN и уровень управления позволяют сделать сеть динамической, обеспечивая реакцию на требования приложений более высокого уровня в режиме реального времени, закладывая основу для услуг следующего поколения, в числе которых облачные вычисления. Данная сеть может работать в режиме «клиент-сервер» для сетевых приложений, плавно и активно предоставляя и высвобождая пропускную способность в соответствии с требованиями уровня приложений, как показано на рисунке

OTN – достоинства:

1. OTN обеспечивает предсказуемое и простое предоставление услуг.
2. Эффективная интеграция коммутации OTN с существующей транспортной сетью OTN.
3. Упрощение операций.
4. Измерение задержки в режиме реального времени.
5. Виртуализация сети.
6. Базовая сеть без потерь.
7. Несколько классов обслуживания.
8. Усовершенствованные возможности сквозного мониторинга услуг.
9. Четкий путь развития до 100G и далее.
10. Динамическая инфраструктура.

Заключение по OTN:

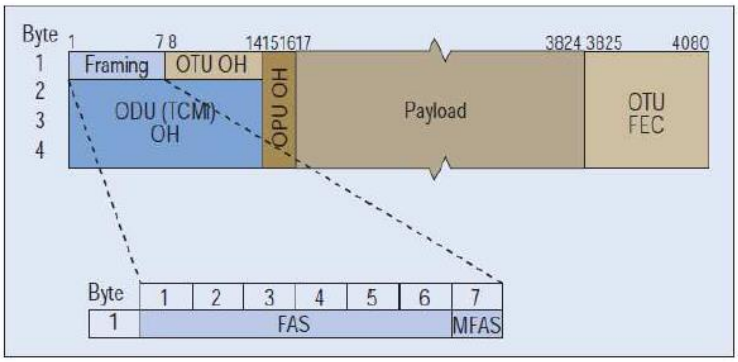
• Услуги для клиентов имеют множество различий: от скорости передачи данных до требований по качеству и уровню надежности.

• Сети с коммутацией пакетов не всегда могут удовлетворить строгие требования высокопроизводительных услуг, такие как минимальный уровень задержки, отсутствие потерь, высокая скорость передачи данных и предсказуемое время восстановления (не более 50 мс).

• OTN обеспечивает предсказуемую и простую модель предоставления услуг, дополняющую сети с коммутацией пакетов, благодаря уникальным возможностям и функциям, таким как прозрачность услуг, сквозной мониторинг, усиленную коррекцию ошибок, встроенные средства измерения задержки, которые необходимы для соответствия строгим требованиям услуг с высоким качеством канала и специальных сервисов.

1.9.Структура и мультиплексирование PDU в стандарте ITU G.709.

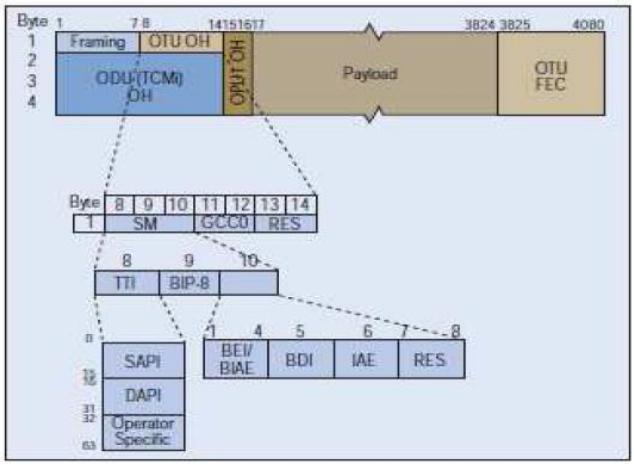
Структура OTU кадра:



FAS - первые 6 байт для определения границ кадра и выявления ситуации потери кадра.

MFAS – используется чтобы распространить функции управления (TCM, TTI – trail trace identifier) на несколько кадров сразу. Он увеличивается на 1 для каждого OTUk/ODUk кадра, позволяя структурировать по 256 фреймов.

OTU заголовок:

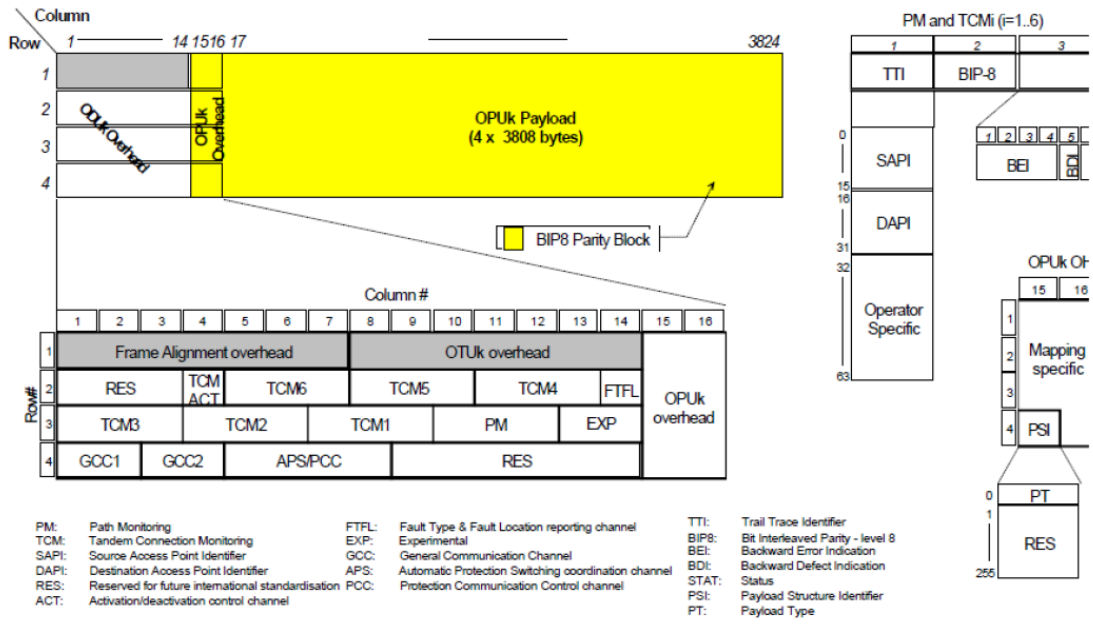


Section Monitoring (SM) состоит из байт: TTI, BIP-8, BEI, BIAE, BDI и IAE

- Trail Trace Identifier (TTI) – 64-битный мультикадровый TTI сигнал имеет ту же роль, что и J0 байт в SONET/SDH.
- Bit-Interleaved Parity (BIP-8) – покрывает OPU и полезную нагрузку клиента в G.709, и это значение «is inserted in the BIP-8 field of the second frame following calculation» (*вставляется начиная со второго кадра*).
- Backward Defect Indication (BDI) – когда AIS (Alarm Indication Signal) шлётся в обратном направлении в качестве ответа на индикатор об ошибке сигнала (например, FTFL), ответ в прямом направлении о продолжении соединения называется сигналом backward defect indication (BDI). Флаг BDI подымается, в качестве тревоги, когда его получают в 5-ти последовательных фреймах.
- Backward Error Indication (BEI) – используется для детектирования «скольжения» («slip») кадра, которое может произойти в OTU.
- Backward Incoming Alignment Error (BIAE) 3-х битное значение «010» в поле статуса (STAT) в прямом направлении. Соответствующий BIAE в обратном направлении вставляется в виде последовательности «1011» в поле BEI/BIAE SM.
- STAT – 3 бита, обозначающих сигналы (AIS, OCI, TCMi, IAE)
- General Communication Channel 0 (GCC0) – канал для передачи информации между конечными точками уровня OTU.

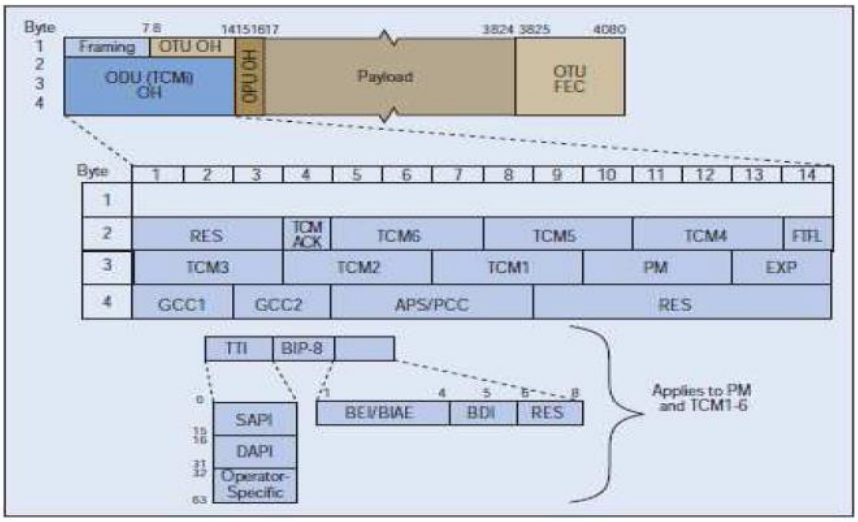
RES – зарезервированные байты.

Структура ODU кадра:



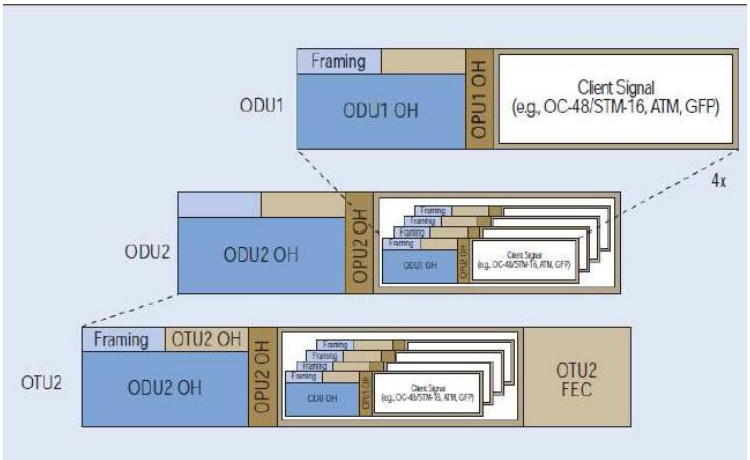
Эта структура обеспечивает следующий сервис:
Сквозной мониторинг прохождения ODU
Контроль пути следования
Унификацию представления данных пользователя не зависимо от вида сервиса в user space
Интеграцию с OTU для передачи по OCh

ODU заголовок:



- RES – зарезервированные и неиспользуемые байты.
- Path Monitoring (PM) - позволяет отслеживать отдельные части сети а так же локализировать место падений. PM содержит в себе похожие подполя, по сравнению с полями из SM, включая: TTI, BIP-8, BEI, BDI и Status (STAT).
- PM trail trace identifier (TTI) – однобайтовое поле, схожее с J0 байтом в SONET/SDH. Используется, чтобы идентифицировать сигнал от источника к получателю в рамках сети. TTI содержит поле - access point identifier (API), которое используется, чтобы специфицировать source access point identifier (SAPI) и destination access point identifier (DAPI). xAPI содержит информацию о стране появления трафика, сетевом операторе и административных деталях.
- PM bit-interleaved parity (BIP-8) – однобайтовое поле, используемое для детектирования ошибок. BIP-8 байт добавляет битовую избыточность – 8-битный код, подсчитанный для всего OPU, и добавленный в BIP-8 SM 2-мя фреймами позже.
- PM backward defect indication (BDI) - один бит, несущий информацию о функционировании передачи сигнала в обратном направлении.
- PM backward error indication (BEI) and backward incoming alignment error (BIAE) сигналы несут информацию «interleaved-bit blocks, detected in error in the upstream direction». Эти биты также используются для передачи IAE в обратном направлении.
- PM status (STAT) – 3-х битное поле, используемое для обозначения наличия maintenance signals.
- Tandem Connection Monitoring (TCMi) поля, являющиеся частью ODU заголовков, определяют шесть ODU TCM подуровней. Каждый TCM подуровень содержит TTI, BIP-8, BEI/BIAE, BDI и STAT, ассоциированные с TCMi уровнями (i =1,2,3,4,5,6).
- Tandem Connection Monitoring Activation/Deactivation (TCM ACT) – однобайтовое поле, на текущий момент неопределено в стандарте.
- Fault Type and Fault Location (FTFL) – однобайтовое поле в ODU заголовке и используется для транспортировки fault type and fault location (FTFL) сообщение, распространённое среди 256-байтового мультифрейма для посылки индикаторов ошибок в прямом и обратном направлении. Прямому направлению соответствуют байты с 0-го по 127, остальные – обратному направлению.

ODU мультиплексирование:

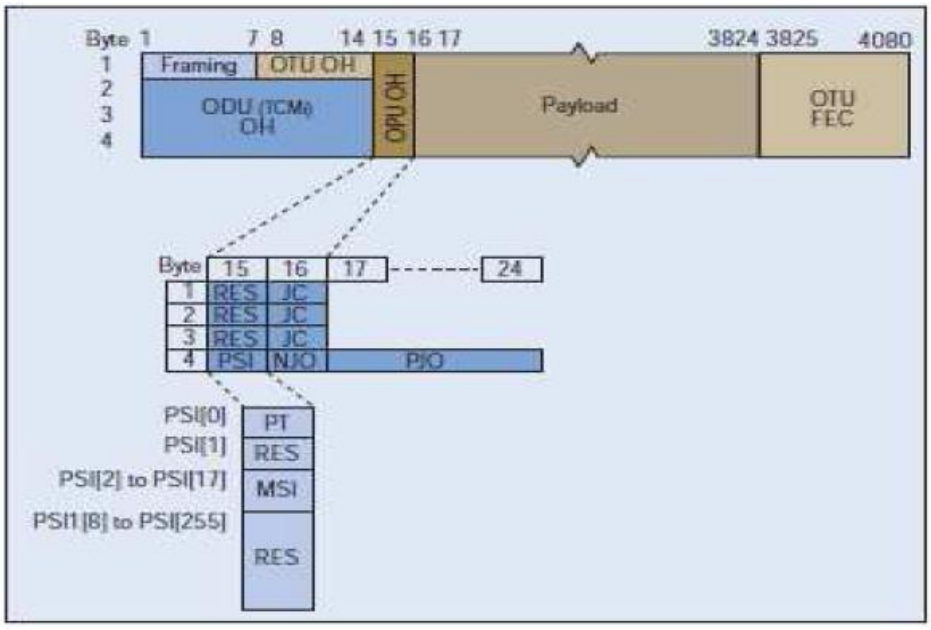


В OTN (G.709) определены функции мультиплексирования, позволяющие 4 ODU1 вложить в ODU2, 4 ODU2 вложить в ODU3, а также можно смешивать ODU1 и ODU2 в ODU3.
ODU мультиплексирование очень важно для оптимального использования сети. Обычно сигнал клиента – 2,5 Gbit/s, которые влазят в одну длину волны DWDM, что эффективно для передачи на малые дистанции, но для выделения целой длинны волны под клиента при дальней передаче – это слишком дорого.

G.709 рекомендует OPU, которая предоставляет необходимые накладные расходы для организации ODU мультиплексирования.
Чтобы вложить 4 ODU1 в ODU2, в OPU2 разделён на несколько слотов данных – tributary slot (TS). В процессе мультиплексирования байты ODU1 вкладываются в 4-ре TS.
Multiplex structure identifier (MSI) используется, чтобы определять тип мультиплексирования на трансмиттере. MSI состоит из PSI байтов от 2 до 17, но когда мультиплексируется ODU1 в ODU2, только байты от 2 до 5-ти имеют значение, остальные установлены в 0, т.к. они используются для мультиплексирования в ODU3.
Информация, передаваемая в MSI:
- Тип ODU, передаваемый в OPU слоте.
- Соответствие порта данных слоту данных.
- В случае ODU1 в ODU2, соответствие порта данных и слотов данных – фиксированно.
- Для ODU2 добавляются избыточные данные, после чего ODU2 оборачивается в OTU2.

Функционал мультиплексирования 4-х OC-48/STM-16 сигналов ODU в один OTU2 – являются бит-, время- и задержко- прозрачными. Т.е. целостность всего пользовательского сигнала сохраняется.

Структура OPU кадра:

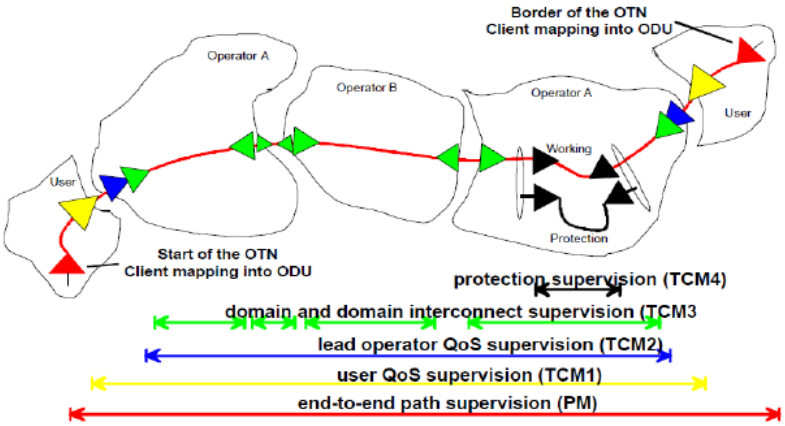


Payload Structure Identifier (PSI) – создан для транспортировки 256-байтного сообщения с MFAS.

Payload Type (PT) – содержит идентификатор, который устанавливает тип полезной нагрузки переносимой в OPU, а текущий момент не определён в стандарте.
Multiplex Structure Identifier (MSI) – located in the mapping-specific area of the PSI signal (PSI[2] to PSI[17], and it is used to encode the ODU multiplex structure in the OPU).
Justification Control (JC) - negative justification opportunity (JJO) and positive justification opportunity (PJO) сигналы используются в ODU мультиплексировании, чтобы принять решение о том, как маппировать сигнал клиента.

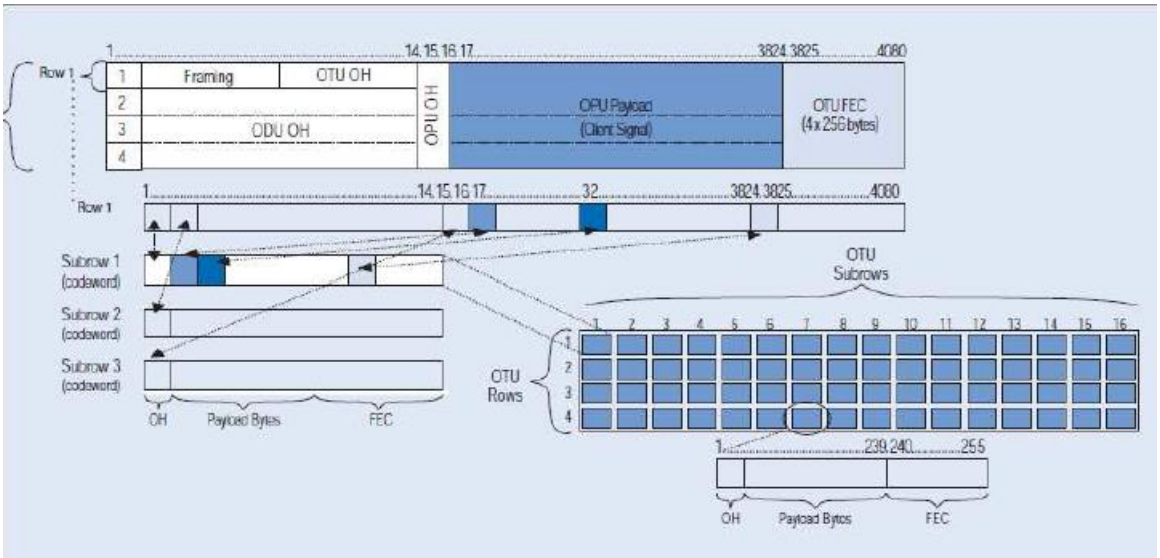
1.10. Механизмы мониторинга, обнаружения и коррекции ошибок вы стандарте ITU G.709.

Мониторинг в OTN:



- Path Monitoring (PM) состоит из следующих байт: TTI, BIP-8, BEI, BIAE, BDI и IAE.
- Trail Trace Identifier (TTI) – 64-х байтный мульти-кадровый сигнал – аналогичен J0 байту в SONET/SDH.
- Bit-Interleaved Parity (BIP-8) - ODU PM содержит BIP-8 поле, покрывающее OPU and полезную нагрузку клиента в G.709 кадре.
- Backward Defect Indication (BDI) – AIS-направленный сигнал в прямом направлении, шлётся в качестве ответа на индикатор ошибки, подобно FTFL или приходящему ODU-AIS. (In the upstream direction, the response to continuity, connectivity and maintenance signals is a backward defect indication (BDI) signal indicated by a bit found in the PM and TCMi. BDI is raised as an alarm when it has been received for five consecutive frames)
- STAT – 3 бита, обозначающих сигналы (AIS, OCI, TCMi, IAE)

Коррекция ошибок, FEC механизм:



Весь payload разбиваем на блоки по 238 байт + 1 байт из заголовка, добавляем 16 байт RS code. Как раз получаем (255, 239) параметры RS code. Все 16 блоков по 16 байт FEC каждой строки собирают в конце строки.
Перемешивание даёт устойчивость к групповым ошибкам.

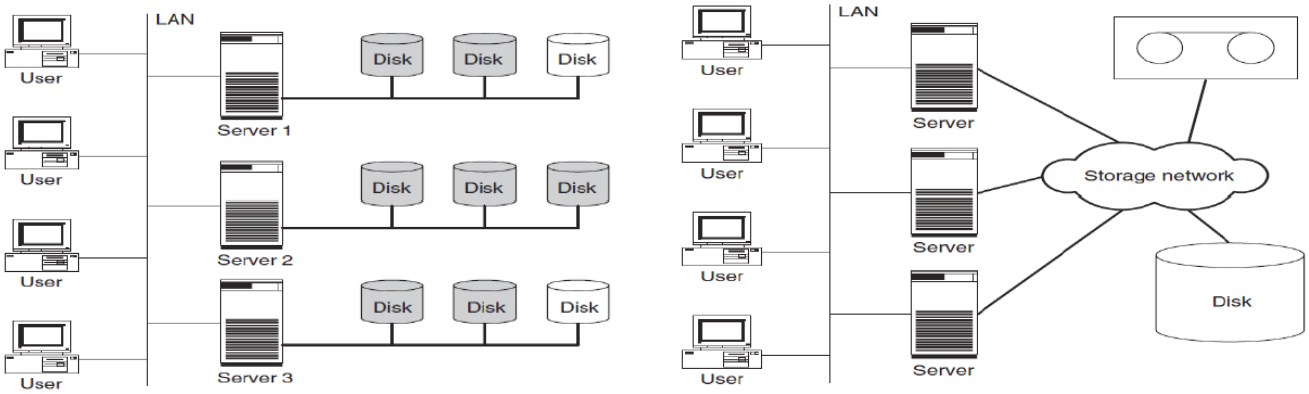
FEC (Forward Error Connection):
- увеличить длину оптических линий
- число каналов в DWDM системе (можно плотнее ужать частоты)
- снизить мощность сигналов
- снизить требования к оптическим компонентам приема/передачи, что снижает их стоимость
- увеличить число ретрансляторов: передача аналоговая => происходит усиление шума при каждой ретрансляции сигнала, снижение порогового значения S/N позволяет увеличить число ретрансляций.

В качестве FEC стандарт рекомендует использовать Reed-Solomon Code RS (255, 239)

t = длина символа = 8 бит
n = число символов в кодослове = 255 байт
k = количество информационных символов = 239 байт
2t = n-k = 255 – 239 = 16, t = 8
RSS (255, 239) max length n = 2^t -1 = 255
Сам Reed-Solomon построен на принципах кода Хемминга, т.е. добавление битов четности для группы битов. 16 контрольных байт позволяют выявлять и корректировать до 8 байт. RS код выявляет ошибки на уровне символа, т.е. байта. Не важно сколько ошибочных бит в байте: все 8 или только 1 бит.
Code word в ODU мультиплексируются, что позволяет эффективно бороться с групповыми ошибками.
Например, если 64 codewords с возможностью исправления до 8 символов у каждого замультиплексировать, то получим структуру, где можно обнаруживать и справлять до 512 ошибок! И не важно в байте будет одна ошибка или все 8 бит.

1.11. Сравнение серверно-ориентированной архитектуры со Storage-ориентированной архитектурой. Внутренняя организация Дисковой ПодСистемы (ДПС).

Серверно-ориентированая и Storage-ориентированная архитектура:

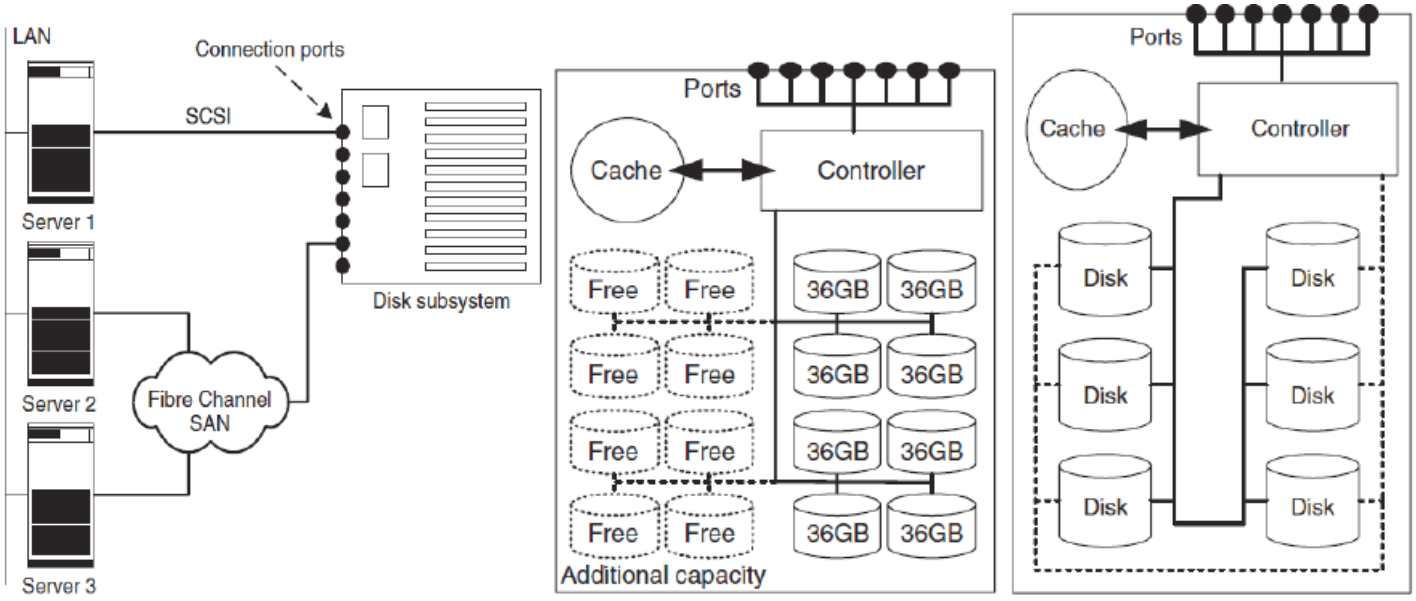


<u>В серверно-ориентированной</u> если сервер упадёт, то будет потерян доступ к дискам.
<u>В storage-ориентированной</u>: архитектура поддерживает fault tolerance, high availability.

Small Computer Serial Interface - SCSI - потокол, который используется для общения (максимум 25 метров)

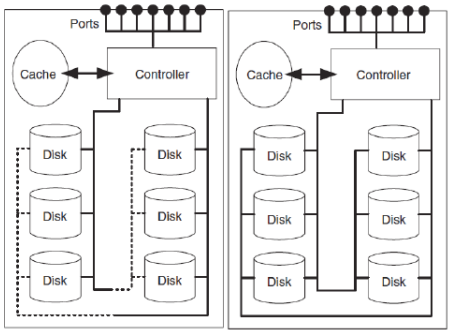
<u>Как происходит подключение серверов к этой системе</u>:
У дисковой системы есть контроллер, подсоединённый к HBA (host bus adapter) карте на сервере, которая через PCI подключена к центральному процессору.
Контроллер дискового массива - это специализированный компьютер для поддержки сервисов внешнего хранилища данных + нужно быть достаточно мощным в соответствии с пропускной способностью дисковой подсистемы.
Дисковые подсистемы отличаются набором портов и программным обеспечением.

Организация ДПС (Дисковой Подсистемы):



- могут использоваться все диски (и их может быть много маленьких), а могут и не все, + могут быть разных размеров

- могут быть соединены одной шиной, а могут быть соединены дублировано. При этом дублирование может быть с фиксированным распределением дисков, и с динамическим распределением:



Иногда сервер может сам хотеть использовать несколько дисков (в идеале такого случаться не должно и этим должен заниматься контроллер). А иногда это делает контроллер, распределяя данные между различным жёсткими дисками, снижая нагрузку.

1.12. RAID дисковые массивы и их уровни. Горячее резервирование, способы ускорения работы ДПС.

RAID – Redundant Array of Independent Disks

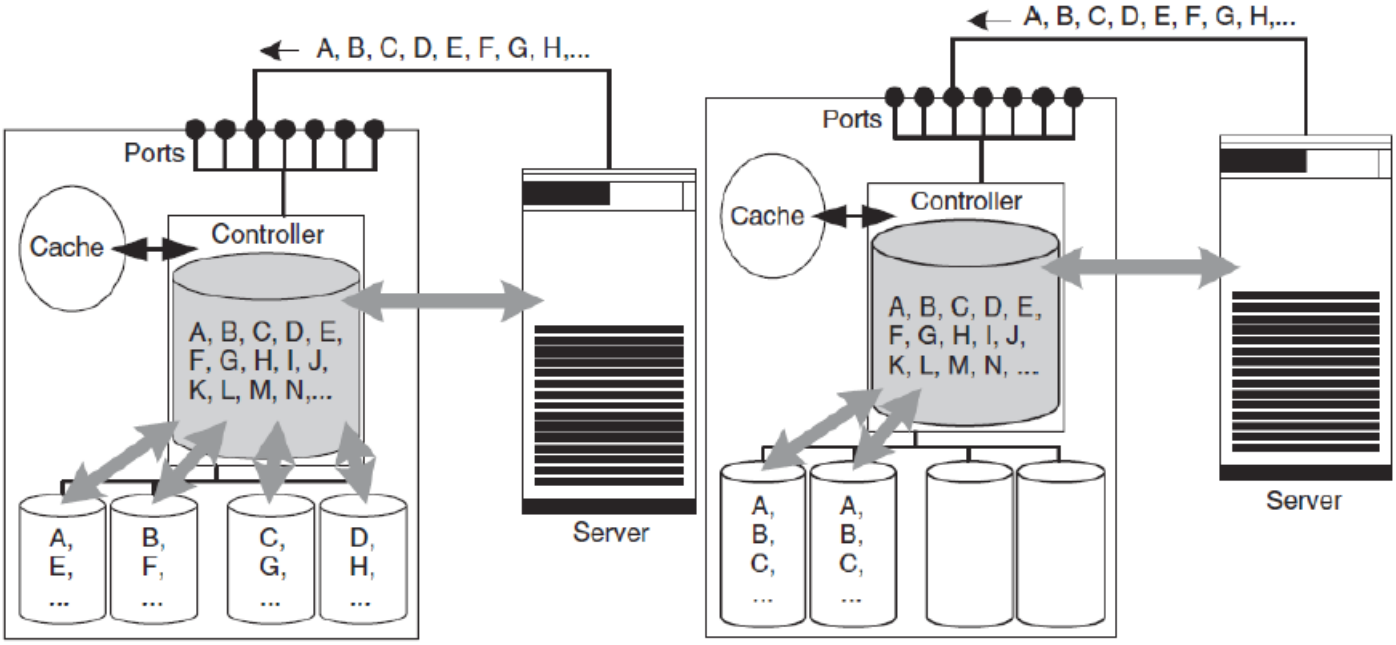
Контроллер объединяет жёсткие диски в виртуальные диски и даёт доступ к виртуальным дискам, а не физическим. При этом контроллер сам может заниматься распределением нагрузки между разными физическими дисками и заботиться об избыточности.

Благодаря кодам Хемминга чип (контроллер) может автоматически исправлять и детектировать ошибки, он так же может контролировать текущее состояние диска, чтобы понимать исправен он или нет.

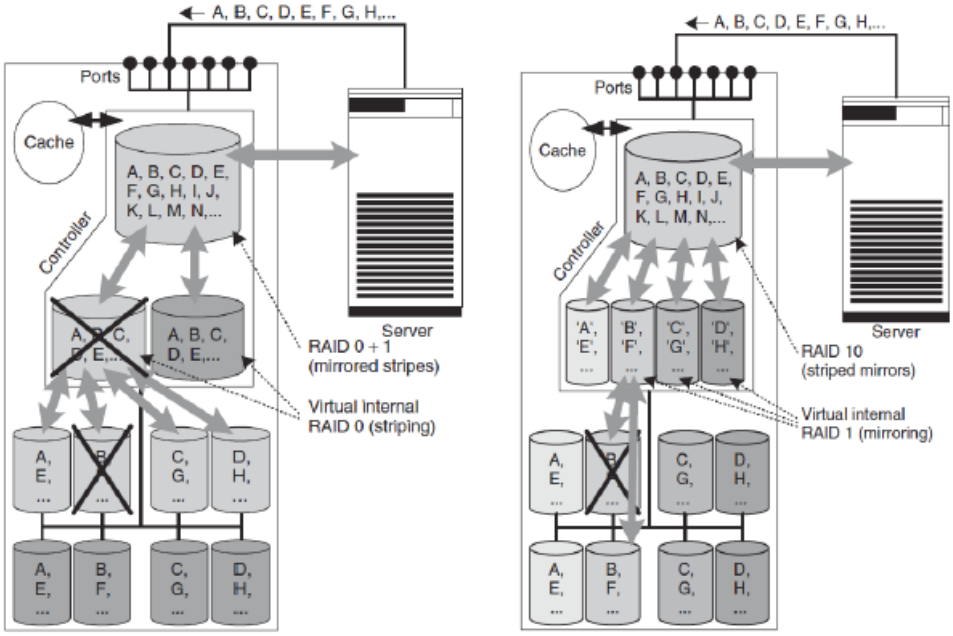
Горячий дисковый резерв – резервирование дискового хранилища, позволяющее сразу после детектирования поломки одного из дисков, автоматически перевести приложение на использование другого резервного диска.

Всего существует 10 уровней RAID (вообще-то их 0-7, + комбинации 1,5 и 6 с raid 0). 2,3,9,8 - практически не используются

RAID0 - просто поочерёдная запись данных в соответствии с Round Robin. Избыточности в классическом понимании - нет! Суть в том, что если отказывает диск, то умирает не подряд идущий блок данных, а понемногу отовсюду (расчёт идёт на то, что приложение заранее заложило избыточность в свои данные и сможет их восстановить).

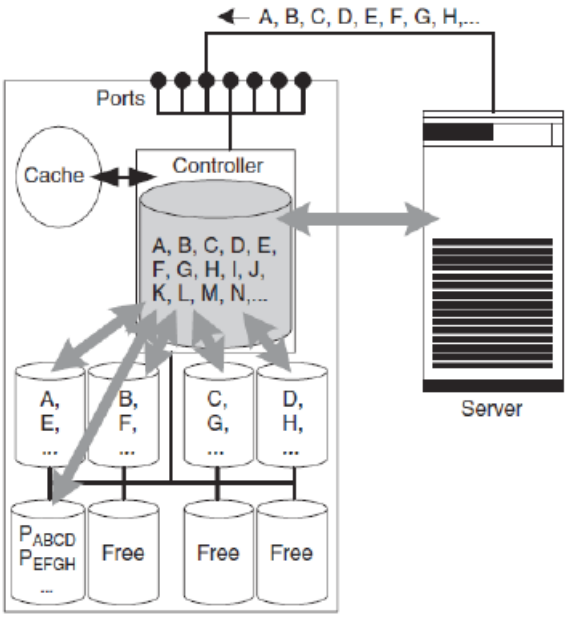


RAID 0 RAID 1



RAID 0+1 RAID 10

RAID4 - просто подсчитывает контрольную сумму некоторого блока данных (xor). Можно легко восстановить потерянный блок, если мы знаем контрольную сумму и все диски, кроме одного битого.
RAID5 - отличается от RAID4 только тем, что раскидывает контрольные суммы между дисками по Round Robin
RAID6 - просто имеет дополнительно к RAID5 дублирование контрольных сумм (как RAID1) на диск чётности.
В случае перезаписи какого-либо блока, то для пересчёта контрольной суммы нужно лишь заксорить её с предыдущим значением блока, а потом заксорить с новым значением. Т.е. запись требует предварительного чтения изменяемого блока + чтение и запись контрольных сумм.
RAID 6 использует дополнительный диск четности - увеличение затрат - увеличение времени операции записи и коррекции



RAID 4

- Современные диски 1TB с BER (bit error rate) 10^-15 => 100 TB одним сектором без ошибок не считать
- 10 дисковых массивов по 16 x 1TB будут терять один массив 1 раз в год
- Режим эксплуатации теперь 7X24

RAID level	Fault-tolerance	Read performance	Write performance	Space requirement
RAID 0	None	Good	Very good	Minimal
RAID 1	High	Poor	Poor	High
RAID 10	Very high	Very good	Good	High
RAID 4	High	Good	Very very poor	Low
RAID 5	High	Good	Very poor	Low
RAID 6	Very high	Good	Very very poor	Low

Немного википедии:
RAID 0 — дисковый массив повышенной производительности с чередованием, без отказоустойчивости;
RAID 1 — зеркальный дисковый массив;
RAID 2 — зарезервирован для массивов, которые применяют код Хемминга;
RAID 3 и 4 — дисковые массивы с чередованием и выделенным диском чётности;
RAID 5 — дисковый массив с чередованием и «невыделенным диском чётности»;
RAID 6 — дисковый массив с чередованием, использующий две контрольные суммы, вычисляемые двумя независимыми способами;
RAID 10 — массив RAID 0, построенный из массивов RAID 1;
RAID 01 — массив RAID 1, построенный из массивов RAID 0 (имеет низкую отказоустойчивость);
RAID 50 — массив RAID 0, построенный из массивов RAID 5;
RAID 05 — массив RAID 5, построенный из массивов RAID 0;
RAID 60 — массив RAID 0, построенный из массивов RAID 6;
RAID 06 — массив RAID 6, построенный из массивов RAID 0.

Caching (ускорение дисковых хранилищ):
- кеш на уровне HD
- кеш на уровне контроллера дисковой системы при записи (ГБ кеша; важно сохранить данные, даже при отключении питания; важно для блочных приложений)
- кеш для ускорения чтения контроллером дисковой системы

Переполение кеша - это очень плохо, т.к. это потеря данных, поэтому нужен сервис по управлению потоком («скользяще окно»), чтобы контролировать переполнение буфера у получателя.

Кеш не знает ни структуры данных, ни характеристик потоков.

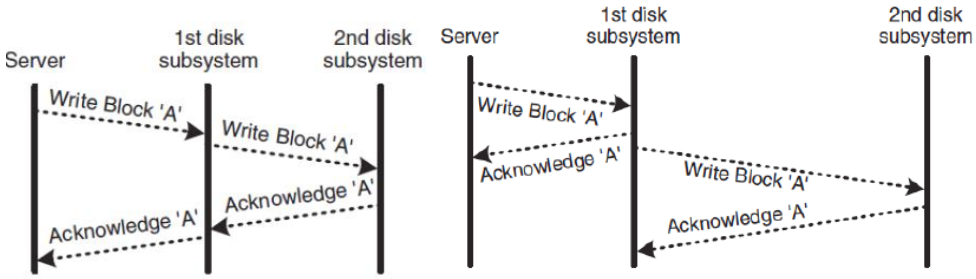
1.13. Интеллектуальные ДПС: удаленное зеркалирование, групповая консистентность, LUN маскирование. Методы повышения устойчивости работы ДПС.

Зеркалирование (локальное) происходит, начиная с тех данных, с которых начнёт чтение второй сервер.
При удалённом зеркалировании, контроллер автоматически отправляет данные в удалённое хранилище данных через интернет (другому контроллеру). В случае, если первое хранилище накроется, то система начнёт брать данные из резервного хранилища.
При зеркалировании очень актуален вопрос когерентности данных.

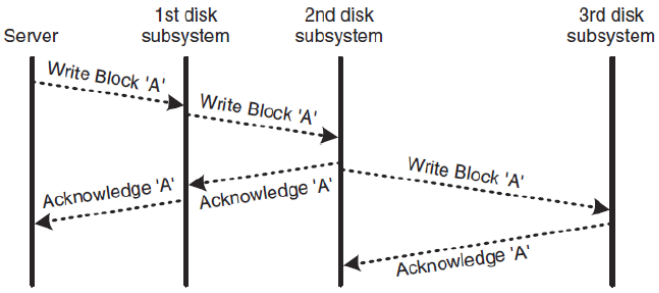
Мгновенное копирование использует либо RAID1, либо 10. На самом деле сначала производится операция сравнения, а потом копирования выявленных изменений.

Если какое-то приложение на сервере начало операцию записи, но по каким-то причинам упало и отвалилось и данные остаются в некотором неопределённом положении.
Против этого мы имеем 3-й уровень избыточности. Тут происходит запись в уже имеющиеся данные, но если что-то отвалилось, то из 3-й копии возвращаются исходные данные.
(именно 3-й копии, потому что со 2-й копией работает другой сервер)
Обычно все имеют 3 цода: в москве, в питере и где-то в глубине россии (3-й это как раз backup цод).

Синхронное и асинхронное зеркалирование:

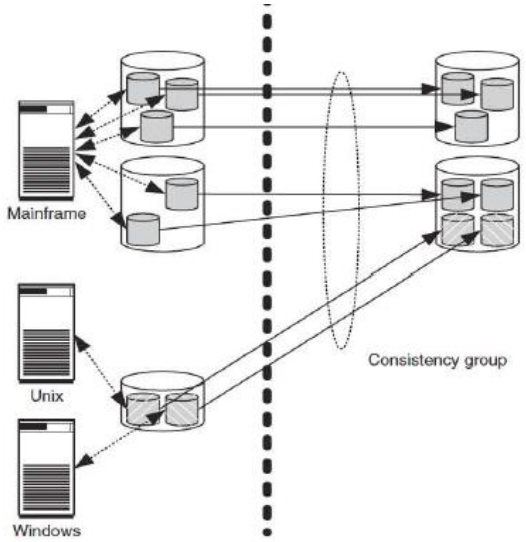


Комбинированная схема:



Полезно знать CAP-теорему: одновременно можно реализовать лишь 2 из 3-х: Consistency, Availability и Partition Tolerance (возможность расщепления распределённой системы на изолированные части, и при этом каждая часть сама по себе - корректна).

Групповая консистентность:
ЦОД Backup ЦОД (через него реализуется консистентность)



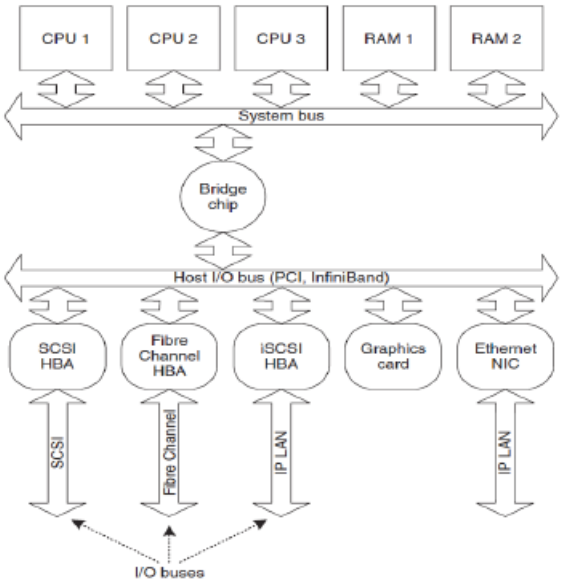
Консистентная группа – это некоторый контейнер, который может содержать множество блоков данных из разных источников, которые является копией исходных данных, находящихся в некотором консистентном состоянии (т.е. там не происходит такого, что, например, часть данных одна, а другая часть – это немного изменившиеся данные, которые были считаны в последующий момент времени)

LUN маскирование - Local Unit Number:
На шине SCSI у каждого порта должен быть уникальный номер, и всего может быть лишь 16 номеров (а раньше вообще только 8). однако может быть 16 "подадресов" - LUN.
LUN - позволяет задавать маски, чтобы не все сервера всё видели. (маска задаётся именно на LUN), тем самым изолируя чужие диски от сервера. Если не поддерживается маскирование, то каждый сервер видит все диски. Это плохо с точки зрения безопасности, т.к. сервер может сломаться сам или его могут хакнуть, и тогда он может испортить чужие данные.

Методы повышения устойчивости работоспособности ДПС:
• Данные распределяют по нескольким дискам с помощью механизмов RAID и снабжают избыточными данными (блоки четности).
• На каждом физическом диске данные закодированы кодом Хемминга. Кроме этого диск оснащен подсистемой самодиагностики, которая контролирует частоту ошибок, вибрацию шпинделя и т.д. Это позволяет проактивно прогнозировать отказы диска.
• Каждый диск подсоединен к контроллеру хотя бы через две внутренние шины.
• Контроллер дисковой подсистемы может быть продублирован. Выход одного экземпляра, автоматически будет активизировать следующий экземпляр.
• Дублируются UPS (uninterruptible power supply) (бесперебойник), системы охлаждения так же. ДС подключают к разным электрическим сетям.
• Сервер соединяют с ДС через несколько линий.
• Мгновенное копирование используют от логических ошибок.
Например, создание мгновенной копии данных через каждый час, тогда в случае сбоя и уничтожения какой-то таблицы, она может быть восстановлена.
• Удаленное зеркалирование используют от физического уничтожения или повреждения оборудования. В сочетании с мгновенным копирование эти сервисы гарантируют сохранение и консистентность данных даже для нескольких виртуальных дисков или дисковых подсистем.
• LUN маскирование защищает от несанкционированного доступа, упрощает работу системного администратора, защищает от случайных сбоев в работе приложений серверов и их оборудования.

1.14. Тракт от CPU до ДПС. SCSI интерфейс: структура, адресация устройств, организация СХД на SCSI.

Тракт от CPU до ДПС:



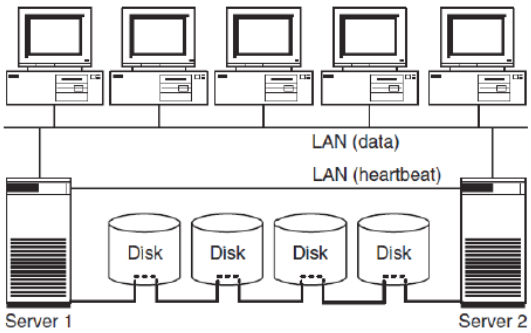
HBA – Host bus adapter

SCSI - Small Computer Serial Interface:

SCSI version	MByte/s MByte/s	Bus width	Max. no. of devices
SCSI-2	5	8	8
Wide Ultra SCSI	40	16	16
Wide Ultra SCSI	40	16	8
Wide Ultra SCSI	40	16	4
Ultra2 SCSI	40	8	8
Wide Ultra2 SCSI	80	16	16
Ultra3 SCSI	160	16	16
Ultra320 SCSI	320	16	16

SCSI поддерживает до примерно 15 млн адресов, но лишь 16 портов. Дальность – 25 метров.
У 16 адресов шины есть конкретный приоритет. И обслуживание происходит в соответствии с приоритетами. (чем старше адрес, тем приоритетнее устройство).

Организация СХД на SCSI:



SCSI хочется сохранить, т.к. под неё написано много приложений

1.15. Fibre Channel: основные характеристики, структура стека протоколов, топологии, типы портов.

В FiberChannel данные передаются последовательно. Оптоволокно позволяет большие расстояния и большую скорость, малая задержка, мало ошибок.
SCSI - это всё-таки шина. FiberChannel - это стек протоколов.

FiberChannel:
- I/O канал
- Сетевое взаимодействие

- Fiber Channel сопрягаем с IPI (Intelligent Peripheral Interface), SCSI, HIPPI (High Performance Parallel Interface), ATM, IP и 802.2.
- Fibre Channel - n x100МБ/с при длинах канала 10 км и более, где n – число каналов. Предельная скорость передачи 4,25 Гбод.
- В качестве физической среды может использоваться одномодовое или мультимодовое оптическое волокно. Допускается применение медного коаксиального кабеля и витых пар (при скоростях до 200 МБ/с).

- FC-0 определяет физические характеристики интерфейса и среды, включая кабели, разъемы, драйверы (ECL, LED, лазеры), передатчики и приемники. Вместе с FC-1 этот уровень образует физический слой.
- FC-1 определяет метод кодирования/декодирования (8B/10B) и протокол передачи, где объединяется пересылка данных и синхронизирующей информации.
- FC-2 определяет правила сигнального протокола, классы услуг, топологию, методику сегментации, задает формат кадра и описывает передачу информационных кадров.
- FC-3 определяет работу нескольких портов на одном узле и обеспечивает общие виды сервиса (шифрование, сжатие, RAID)
- FC-4 обеспечивает реализацию набора прикладных команд и протоколов вышележащего уровня (например, для SCSI, IPI, IEEE 802, SBCCS, HIPPI, IPFC (IP over FC), ATM и т.д.) (Драйвер SCSI будет опираться на драйвер HBA fiberchannel, когда нужно использовать приложение, работающее со скази поверх FiberChannel.)

Топологии FC:
Точка-точка, кольцо с арбитражем, коммутатор.
«Fabric» – это коммутационная среда.

Типы портов FC:
• N-Port – порт определяет характеристики для соединения через коммутатор или P2P.
• F-Port (Fabic port) – порт для подключения к коммутатору.
• L-Port – порт для КсА (кольца с арбитражём)
• NL-Port – работает как N Port или как L-Port. Можно подключать порт через коммутатор или в КсА. (например, чтобы поставить коммутатор, который будет связывать несколько колец)
• FL-Port – для подключения коммутатора в КсА
• E-Port – для соединения двух FC коммутаторов
• G-Port – может настраиваться как E или FL в зависимости от подключения.
• B-Port – для соединения двух FC коммутаторов через ATM, SDH, Ethernet или IP. Например, две FC SAN могут быть соединены через WAN.

1.16. FC-0, FC-1: характеристики физической среды, кодировка, упорядоченные наборы, управление линией.

FC-0 разъёмы, кабели, кодировка:
• Скорость от 100 МБ/с до 10ГБ/с ожидается до 20ГБ/с в одном направлении
• Есть два вида оборудования Base 2 (*кажется, это коммутаторы уровня brocase/hp silkworm – вроде бы с тех пор поменяли архитектуру коммутаторов*) и Base 10
• Передача последовательная
• BER = 10^12, т.е. для линии 100Мб/с ошибка не чаще чем раз в 16,6 мин. Протоколы верхнего уровня для обнаружения ошибок оптимизированы под эту величину BER (протоколы обнаружения ошибок находятся на уровне F3, F4)
• Преимущества оптического кабеля (расстояние)

FC-1 кодировка, упорядоченные наборы, управление линией:
• 8b/10b кодирование
Преобразование 8-битовых последовательностей в 10-битовые.
Если идут 5 подряд нулей, то вставляется единица, если 5 подряд нулей, то вставляется единицей.
Если 5-ти подряд нету, то 8 бит добиваются 2-мя разрядами.

При этом используется потенциальное кодирование, что экономит нам на частоте (ведь импульсное кодирование требует повышенной частоты)

• Transmission words

Data word: SOF (Start of Frame), 4 bytes, EOF

Ordered set: EOF, K28.5, SOF

• Управление линией

!!! что нужно сказать про управление линией?

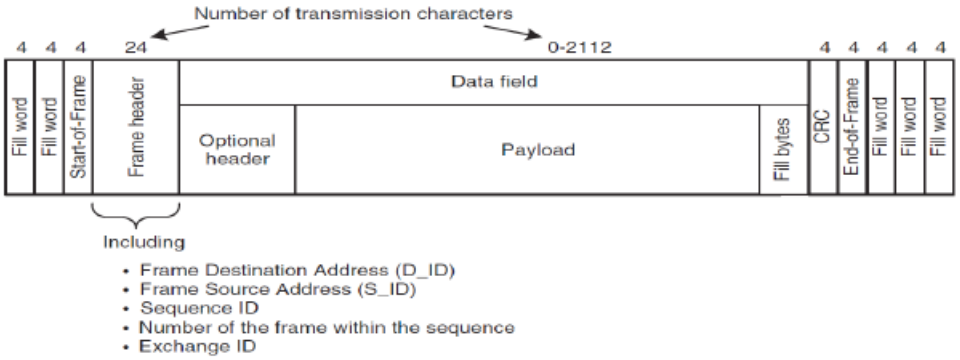
Приёмник всё время читает словами - 4 последовательности по 10 бит с префиксом SOF и концовкой EOF. (приёмник настроен на принятие именно 10 бит (поэтому нельзя не вставлять биты, даже тогда, когда они не нужны))

(*Потенциальное кодирование - кодирование за счёт поддержки потенциала (например, напряжения). Импульсное кодирование - кодирование, например, NRZ (чтобы единицы подряд не приводили к постоянному держанию одного постоянного потенциала). 0 от 1 отличается перепадом сигнала.*)

K28.5 - это специальная комбинация, которая никогда не может встретиться в data word - используется для проверки линии, приёмник с помощью этой последовательности может всегда проверить состояние линии и её работоспособность

1.17. FC-2: структура кадра, организация передачи данных, управление потоком, классы обслуживания.

FC-2 структура кадра:



• Определение размера передаваемых данных

- Exchange – сессия между логическими сущностями (процессы)
- Sequence – последовательность кадров
- Frame – управления и данных (2 112 Б)

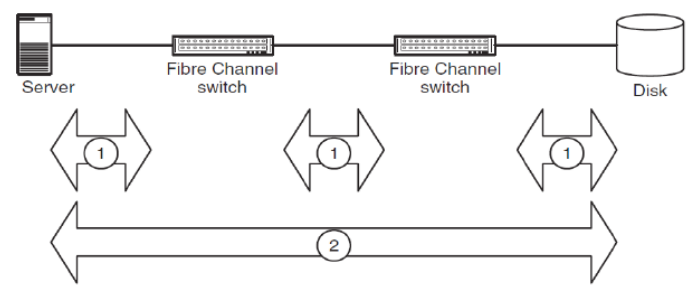
• Управление потоком

• Классы сервиса

управление только на соединении fabric (коммутация) и P2P

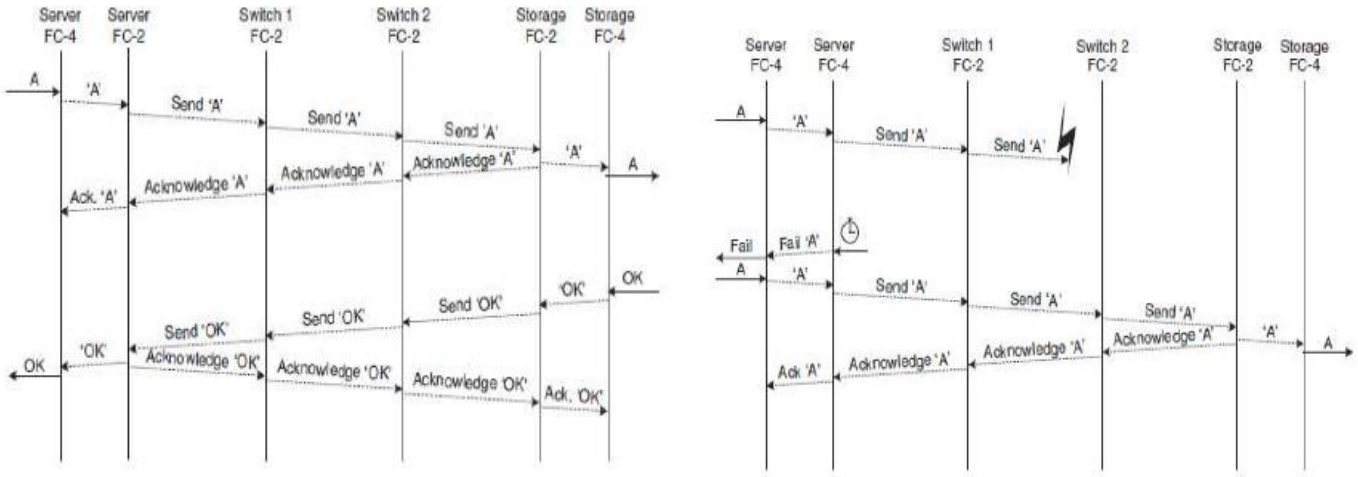
F-port позволяет сообщить и договориться о размере последовательности фреймов, чтобы у получателя не случилось переполнение.

Управление потоком:

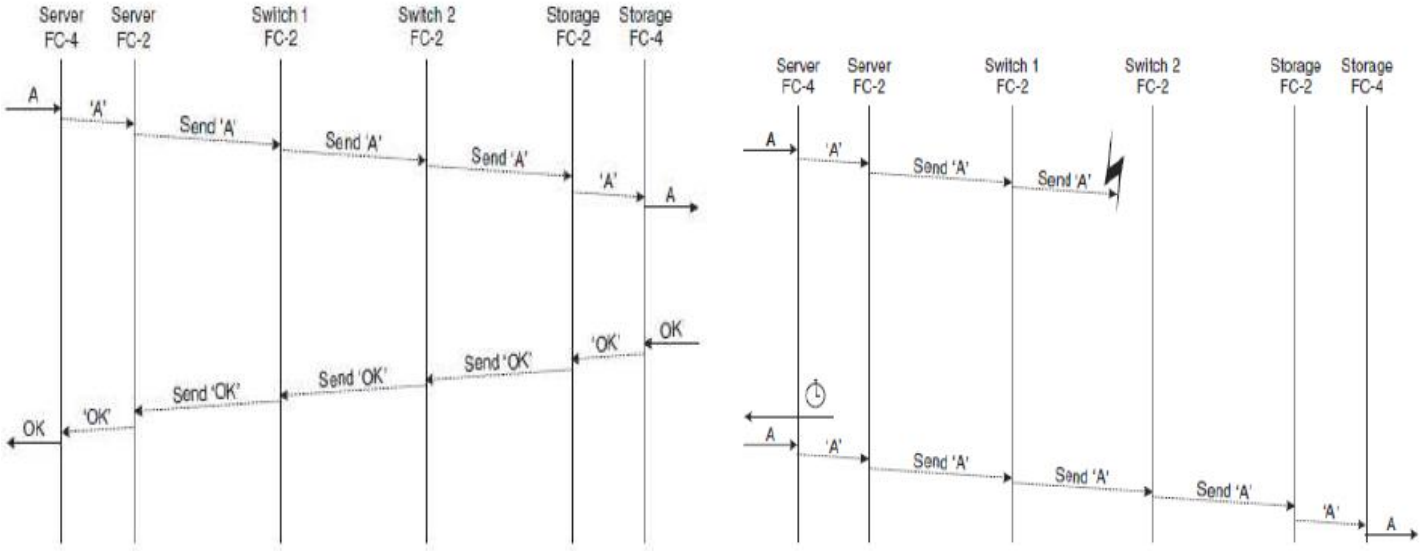


- Кредитная схема
- E2E flow control (2)
- Link flow control (1)

Классы обслуживания:
классы 4,5,6 на текущий момент не реализованы и на рынке не встречаются.
Класс 1 Соединение точка-точка (end-to-end) между портами типа n_port через коммутацию каналов.
Класс удобен для аудио и видео приложений, например, видеоконференций. После установления соединения используется вся доступная полоса пропускания канала. При этом гарантируется, что кадры будут получены в том же порядке, в каком они были посланы. Есть управление потоком.
Класс 2 Без установления соединения с коммутацией пакетов, гарантирующий доставку данных. Порт может взаимодействовать одновременно с любым числом портов типа n_port в режиме дуплекс. Не гарантируется порядок доставки кадров, кроме соединения P2P или KcA. Есть управление потоком. Этот класс характерен для локальных сетей, где время доставки данных не является критическим.



Класс 3 Обмен дейтограммами без установления соединения и без гарантии доставки. Есть управление потоком. Применяется для каналов SCSI.



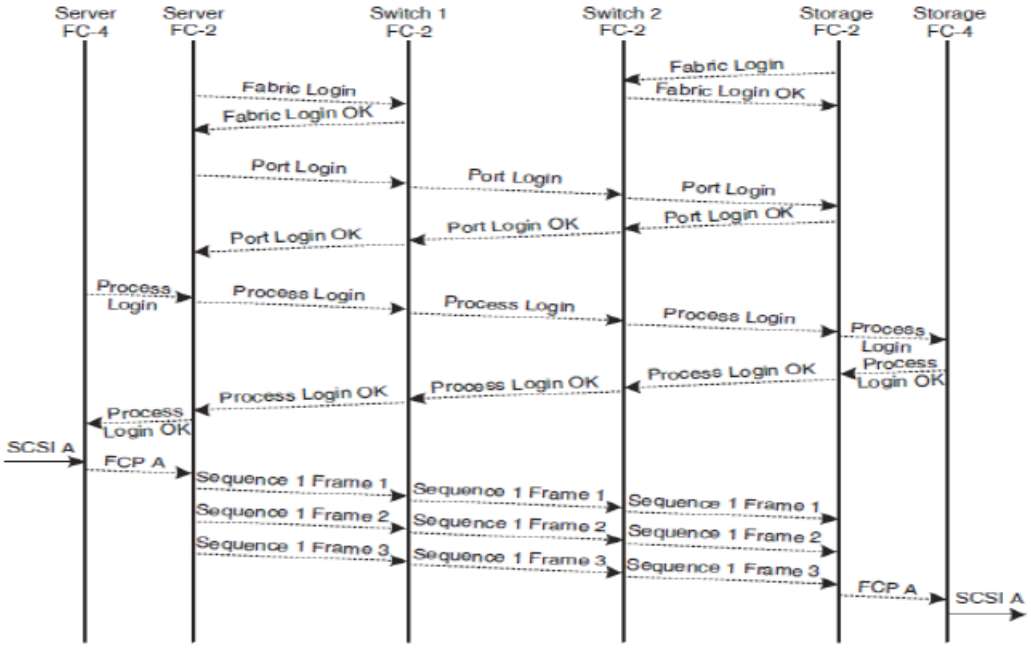
Класс 4 Обеспечивает выделение определенной доли пропускной способности канала с заданным качеством обслуживания (QoS). Только для топологии структура матрица с n_port. Гарантируется порядок доставки кадров.
Класс 5 Регламентирующие документы находятся в процессе подготовки.
Класс 6 Предусматривает групповое-обслуживание с коммутацией.

1.18. FC-3, FC-4: сервисы, имена, адреса, сервисы среды коммутации.

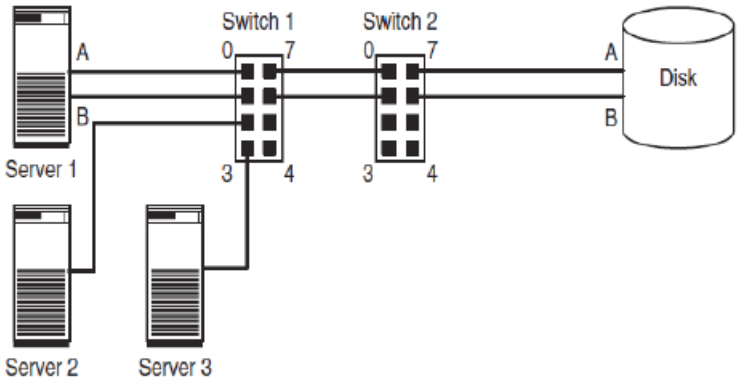
FC-3 сервисы:
• Распределение кадров по маршрутам между много портовыми устройствами для увеличения пропускной способности
• Детектирование перегрузок
• Формирование логических групп маршрутов для управления переполнения на маршруте или сбоя, чтобы не загружать верхние уровни стека.

• Компрессия передаваемых данных (на HBA)
• Шифрование данных (на HBA)
• Зеркалирование

Службы линии - идентификация и адресация:



Адресация:



Port_ID	WWPN	WWNN	Device
010000	20000003 EAFE2C31	2100000C EAFE2C31	Server 1, Port A
010100	20000003 C10E8CC2	2100000C EAFE2C31	Server 1, Port B
010200	10000007 FE667122	10000007 FE667122	Server 2
010300	20000003 3CCD4431	2100000A EA331231	Server 3
020600	20000003 EAFE4C31	50000003 214CC4EF	Disk, Port B
020700	20000003 EAFE8C31	50000003 214CC4EF	Disk, Port A

WWN - World Wide Name - 64 бита
WWPN - World Wide Portal Name
WWNN - World Wide Node Name
FCN – FiberChannel Name

• Имена и адреса в FC (устройствам присваиваются 64-битные имена, потом при подключении им выделяется адрес)
• У всех устройств FC сети есть 64 битные имена
• WWN vs FCN (строго говоря FCN – это уникальное имя в рамках сети, а WWN – это уникальное имя на весь мир, на практике т.к. всё работает в рамках сети, их не различают)
• WWN: WWPN. WWNN (имя сервера и имя пары порт:сервер – это разные вещи, и им присваиваются в соответствии с этим разные адреса)
• FLOG (Fabric Login) – 24 bit port address (требуется соответствующий логин для порта, прежде чем данные могут быть отправлены)
• S_ID vs D_ID
• KcA 8 bit AL_PA (Arbitrated Loop Physical Address)

Когда некоторый узел соединяется с некоторым портом, то он получает 64 разрядный адрес от коммутатора и происходит сопоставление имени и адреса.

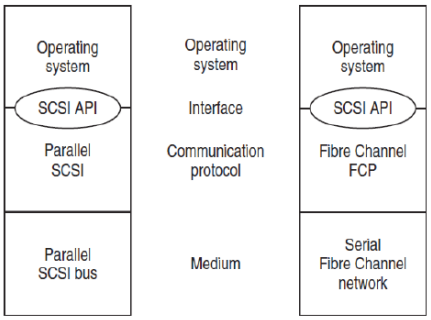
В случае кольца с арбитражём - там адрес коротенький, потому что на таком кольце ограничено количество участников.
Один node name может иметь несколько port name (мало ли с каким портом происходит соединение)

Сервисы коммутационной среды:

Address	Description
0×FF FF FF	Broadcast addresses
0×FF FF FE	Fabric Login Server
0×FF FF FD	Fabric Controller
0×FF FF FC	Name Server
0×FF FF FB	Time Server
0×FF FF FA	Management Server
0×FF FF F9	Quality of Service Facilitator
0×FF FF F8	Alias Server
0×FF FF F7	Security Key Distribution Server
0×FF FF F6	Clock Synchronisation Server
0×FF FF F5	Multicast Server
0×FF FF F4	Reserved
0×FF FF F3	Reserved
0×FF FF F2	Reserved
0×FF FF F1	Reserved
0×FF FF F0	Reserved

- КСС (контроль качества сервиса) нужен для управления инфраструктурой и потоками в FC
- Все сервисы реализуют определённые сервера, которые имеют строго определенные адреса.
- FLOG сервер отвечает за обработку всех входящих fabric login request
- За всеми изменениями в FC сети следит fabric controller
- Name server – отвечает за БД всех имен N_Port-ов

FC-4 – ULP:

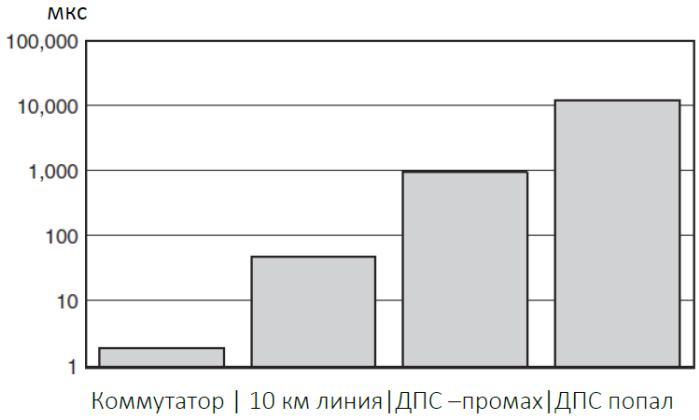


UPL - Upper Level Protocol - предназначен для обеспечения интерфейса для драйверной операционной системы, которая может работать по разным протоколам, в частности SCSI.

1.19. Примеры топологий, интеграция FC_SAN с IP сетями.

FiberChannel SAN – storage area network

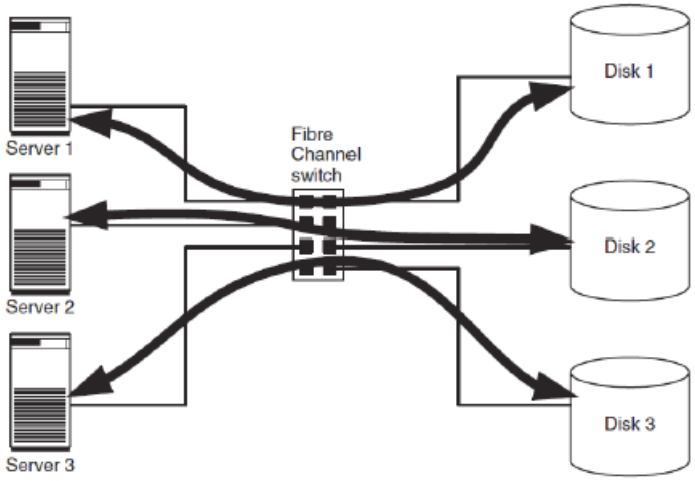
- P2P – FC до 10 км, SCSI не более 25 метров
- SCSI – медь, FC - оптика



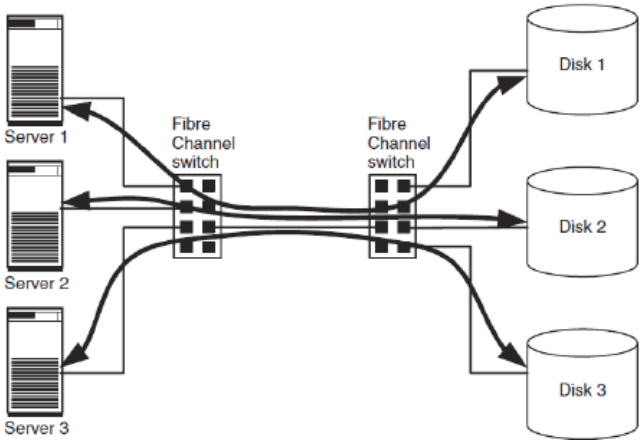
!!! Как получше интерпретировать эту картинку???

!!! Что нужно сказать о SAN и FC ?

Примеры топологий:

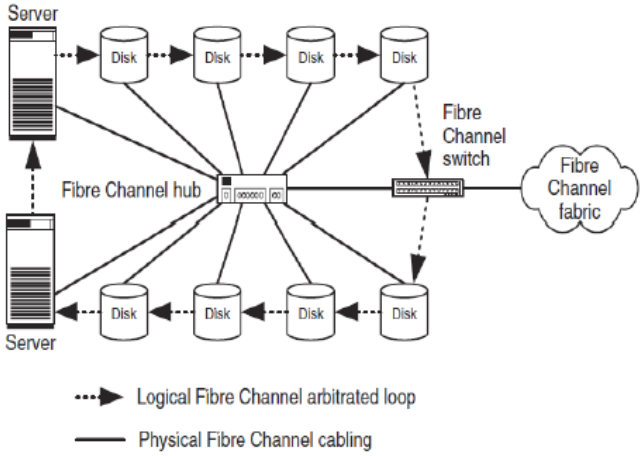


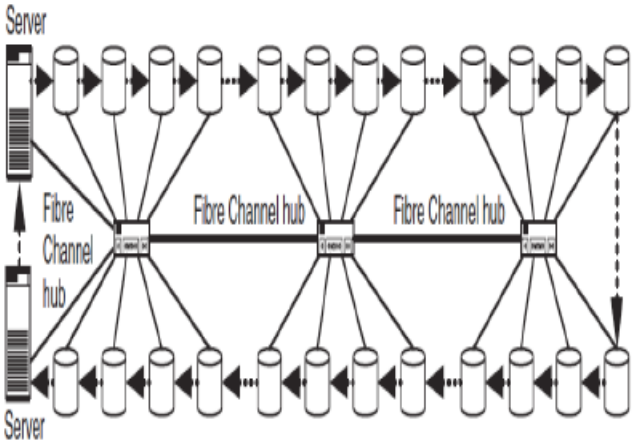
Тут всё ок



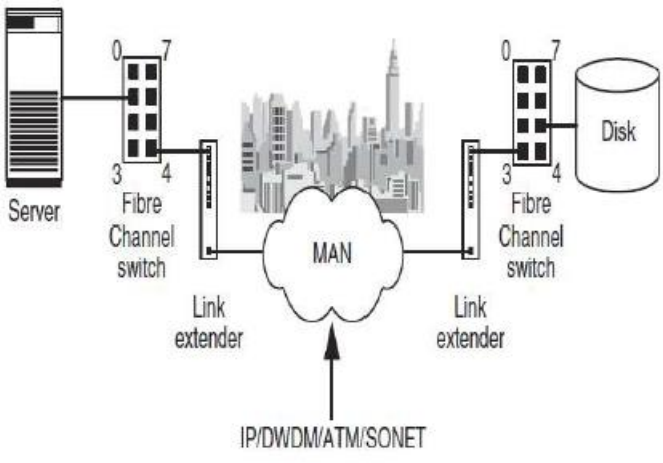
Тут плохо, т.к. придётся 3-х запихать в один кабель.

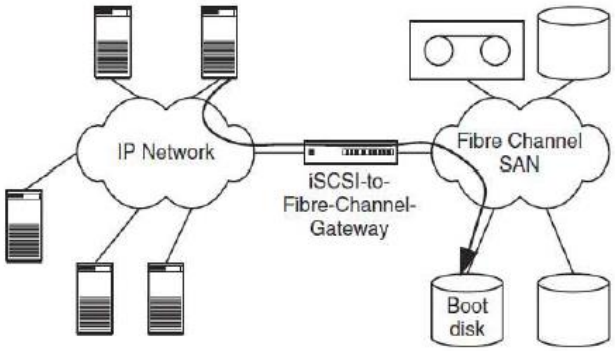
Топология кольца:



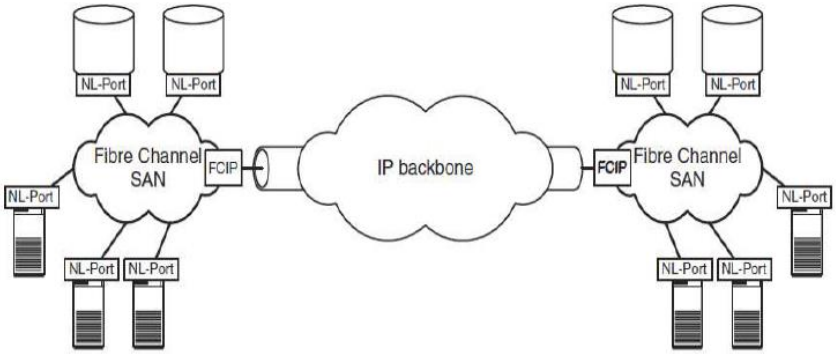


FC_SAN с IP сетями:





Сети IP и FC совершенно никак не стыкуются, их можно только лишь завернуть одну в другую (FC можно запихать в ip (SDH, например) или же ip запихать в FiberChannel)

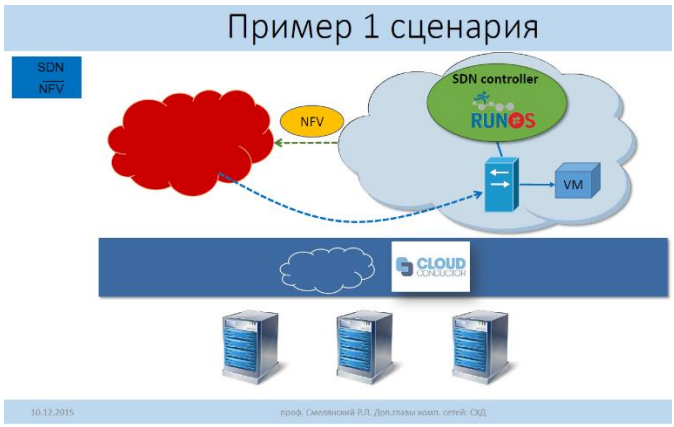


1.20. Примеры синергии SDN и NFV технологий

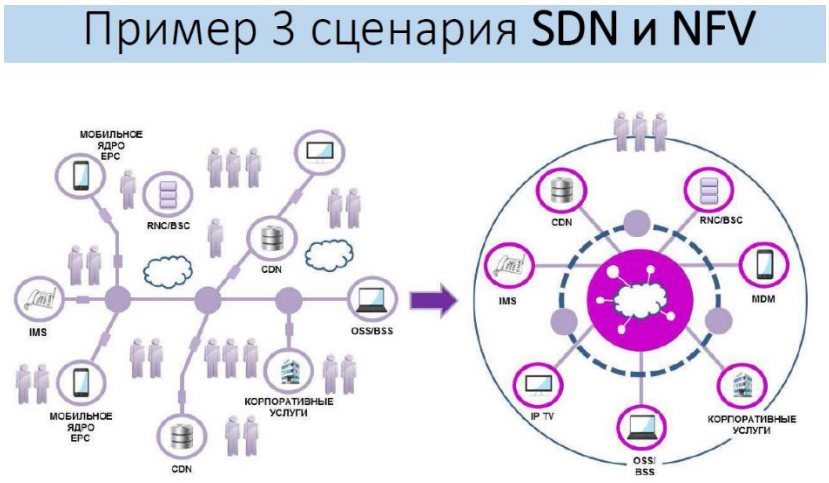
SDN – Software Defined Network – идея, заключающаяся в том, чтобы отделить физически управление потоками, от непосредственной коммутации/передачи данных (отделение dataplane от control plane), при этом объединяя управляющие блоки в один, называемый контроллером. А второе, что в sdn принципиально – это унификация интерфейсов управления.
NFV – Network Function Virtualization – идея о том, чтобы виртуализовать предоставление сетевых сервисов.

NFV и SDN могут хорошо подружиться, в случае если SDN контроллер реализовывать виртуально.

Смелянский выделил в симбиозе SDN и NFV 3 возможных их проявления:
SDN над NFV; NFV над SDN; SDN и NFV на равных



!!! Что символизирует красное облачко?



А ещё смотри билет 2.10

1.21. Проблемы безопасности в SDN сетях.

Безопасность в SDN – это хромая лошадь (даже жалко так обижать этих животных)

Контроллеры:
- Падают от некорректных данных
- Хакаются некорректными данными
- Падают от хитрых комбинаций корректных данных
- Просто падают, по разным причинам, типо memory leak, SEGSEGV, …, даже при штатной работе
- Не устойчивы к MITM (Man in the middle) (как сторона контроллера, так и сторона коммутатора), с аутентификацией – проблемы
- DoS – милое дело, можно даже без DDoS (на каналы, на контроллеры, на свитчи)
- В случае взлома контроллера – есть управление над всей сетью, в случае взлома коммутатора – либо возможен взлом контроллера, либо и без этого можно много всякого, типо сдеградировать сеть, прослушать трафик, подменить трафик, …

Устройства OF сети – DataPlane:
- Уязвимости программного обеспечения
  - неустойчивость кода к внешним воздействиям
  - код с уязвимостями
- Атаки с использованием вредоносного кода
- DDoS атаки
- Атака сетевых устройств изнутри сети
- Вредоносные устройства в OF-сети

Каналы связи:
- В каналах Open Flow используются SSL/TLS, но данные протоколы не являются обязательными
- Аутентификация между контроллером и OF устройствами
- DDoS атаки – поддержание насыщенности канала

Контроллер:
- Обеспечение безопасности контроллера
- Компрометация контроллера позволяет атакующим управлять всей сетью
- DDoS атаки на контроллер
- Поддельный контроллер может изменять топологию сети
- Строгий механизм аутентификации для доступа к SDN-контроллеру
- Целостность контроллера
- Внедрение нежелательной информации в контроллер

Control Plane:
- Требуется обеспечение безопасности control plane, управление авторизацией доступа для сетевых приложений
- Требуется аутентификация доступа приложений на control plane
- Сеть должна обслуживать требования бизнес приложений, и логика данный приложений определяет способы обеспечения безопасности

2. Программно-Конфигурируемые Сети и Виртуализация Сетевых Функций

2.1. Проблемы традиционных сетей. Основные принципы SDN. Архитектура SDN. Преимущества SDN. Примеры применения. Абстракции в IT и в SDN.

Проблемы традиционных сетей

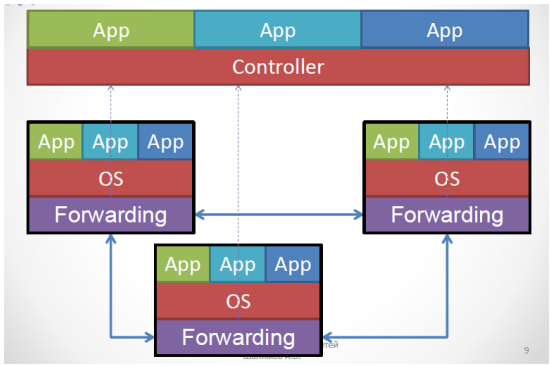
- Зависимость от производителя
- Ошибки в реализациях сетевых протоколов
- Миллионы строк закрытого проприетарного кода (6000+ RFC)

- Высокая стоимость оборудования
- Высокая стоимость эксплуатации
- Сложность управления большими сетями
- Сложность отладки
- “Закрытость” оборудования и программного обеспечения
- Сложность внедрения новых идей
- Неэффективность использования аппаратных ресурсов, энергоэффективность
- Растет сложность сетей => все больше проблем с поддержкой и т.п.

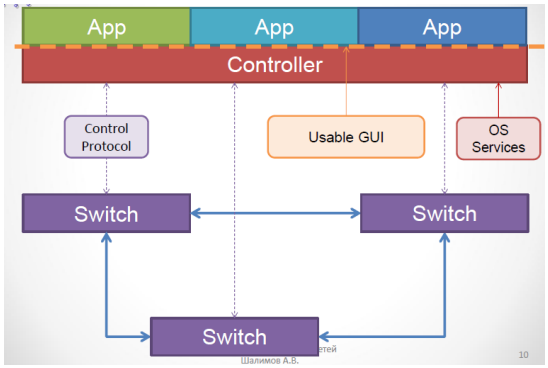
Программно-Конфигурируемые Сети (Software Defined Networking/SDN) – это разделение плоскости передачи и управления данными, позволяющее осуществлять программное управление плоскостью передачи, которое может быть физически или логически отделено от аппаратных коммутаторов и маршрутизаторов
1.Отделить управление сетевым оборудованием от управления передачей данных за счет создания специального программного обеспечения.
2.Перейти от управления отдельными экземплярами сетевого оборудования к управлению сетью в целом.
3.Создать интеллектуальный, программно-управляемый интерфейс между сетевыми приложениями и транспортной сетью.

На самом деле ПКС – как четвертое поколение сотовых телефонов, только в сфере сетевых технологий:
•Новые инструменты и функции;
•Простота администрирования;
•Открытость инновациям и экспериментам;
•Революция на ИТ-рынке

До появления SDN:



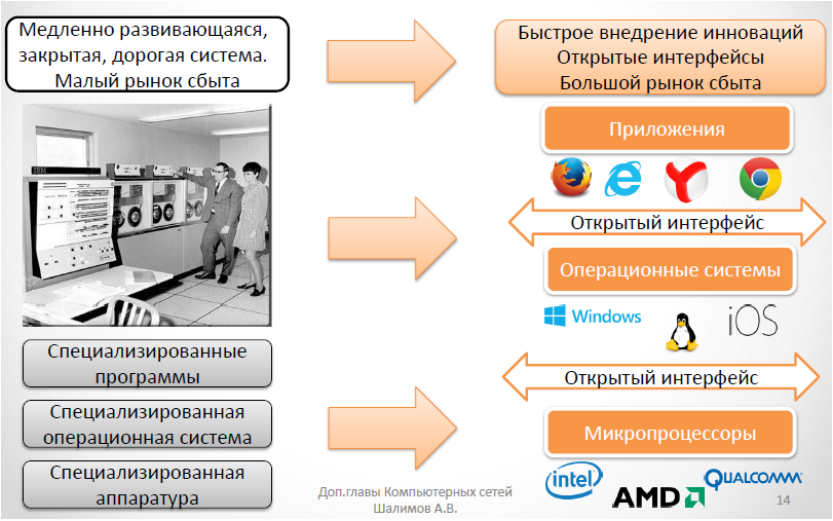
Что предлагается в SDN:



Т.е. Все управляющие функции мы выносим в контроллер, свитчи довольно глупые и умеют только пробрасывать пакеты, предварительно применив к ним некоторые действия, вся нагрузка по решению куда и что отправить лежит на контроллере.
Достоинства такого подхода:
•Удешевление оборудования (CAPEX)
•Облегчение управления сетью (OPEX)
•Программируемость, открытость, инновации

Примеры применения:
Уменьшение энергопотребления в ЦОД
–Отключение неиспользуемых коммутаторов и каналов на основе собранной информации о сети
–ElasticTree (Stanford): сокращение энергопотребления до 60%
–Применение в Google

Абстракция
В IT:



<u>В ЯП</u>:

•Машинные языки программирования - нет никакой абстракции

•Высоко-уровневые языки программирования, ОС + другие абстракции

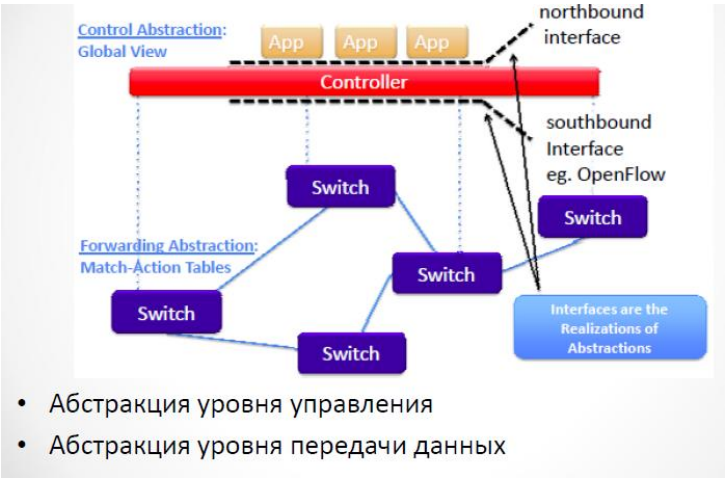
–Структуры данных, функции, переменные, файлы, виртуальная память, ...

•Современные языки программирования – еще больше абстракций

–Объекты, нити, семафоры, сборщик мусора

Абстракции упрощают программирование: проще писать, проще поддерживать, думать об алгоритме

<u>В SDN</u>:



В SDN контроллер имеет два интерфейса – для общения с приложениями (northbound interface) и для общения со свитчами (southbound interface) (строго говоря, концепция предполагает ещё присутствие east-west interface для взаимодействия нескольких экземпляров контроллера). Реализация этих интерфейсов есть абстракция на уровне управления и на уровне передачи данных соответственно.

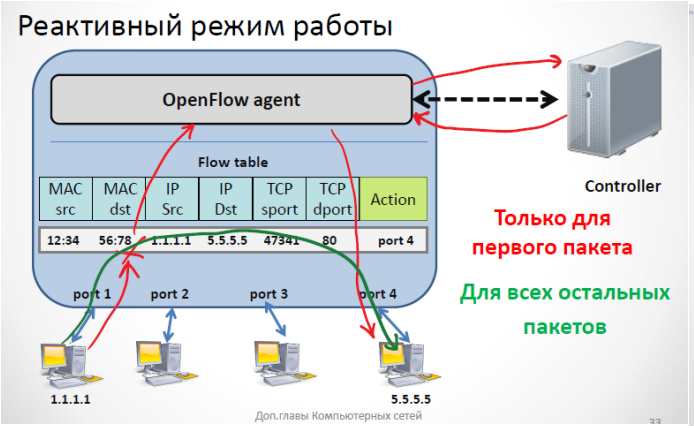
Для приложений это значит, что они имеют всю информацию о состоянии сети, которая им необходима, для свитчей абстракция заключается в match-action правилах.

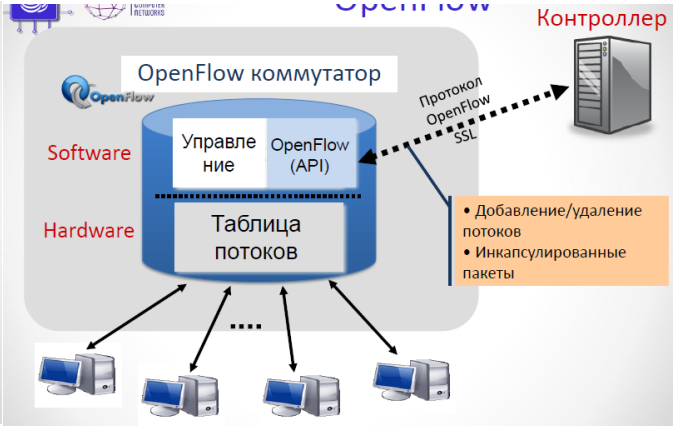
~~!!!Мб стоит еще что-нибудь добавить, хотя по существу вся абстракция как раз в двух интерфейсах~~

2.2.Протокол OpenFlow. Структура OpenFlow коммутатора и контроллера. Таблица потоков. Основные сообщения протокола OpenFlow. Принципы установки правил. Суть вопроса “SDN == OpenFlow?“.

Основная идея работы коммутатора и контроллера – на коммутаторе есть (судя по картинке аппаратная) таблица потоков, в которой хранятся правила, в соответствии с которыми коммутатор обрабатывает пакеты(можно заменять поля заголовка, отправить/сбросить пакет), если приходит пакет, который не подходит ни под одно правило, то на контроллер отправляется packet in, а контроллер уже решает, что делать с пакетом, и отправляет на коммутатор packet out, коммутатор разбирает, что ему ответил контроллер и добавляет (или не добавляет, если так сказал контроллер) новое правило.

Чтобы коммутатор и контроллер друг друга понимали (а также могли быть отчуждаемы) необходим стандарт – OpenFlow, который задаёт формат общения, и в частности, специфицирует, какие команды может отдавать контроллер коммутатору (а т.к. контроллер подразумевает, что коммутатор выполняет его команды, то это задаёт требования к коммутатору, к его возможностям)





OpenFlow контроллер

•Программа, TCP/IP сервер, ожидающий подключения коммутаторов.

•Отвечает за обеспечение взаимодействие приложения-коммутатор.

•Предоставляет важные сервисы (например, построение топологии, мониторинг хостов)

•API сетевой ОС или контроллер предоставляет возможность создавать приложения на основе централизованной модели программирования.

Поддерживается три типа сообщений:

•Сообщения контроллер-коммутатор

– Конфигурирование коммутатора

– Управление и контроль состояния

– Управление таблицами потоков

– Features, Configuration, Modify-State (flow-mod), Read-State (multipart request), Packet-out, Barrier, Role-Request

•Симметричные сообщения

–Отправка в обоих направлениях

–Обнаружение проблем соединения контроллера с коммутатором

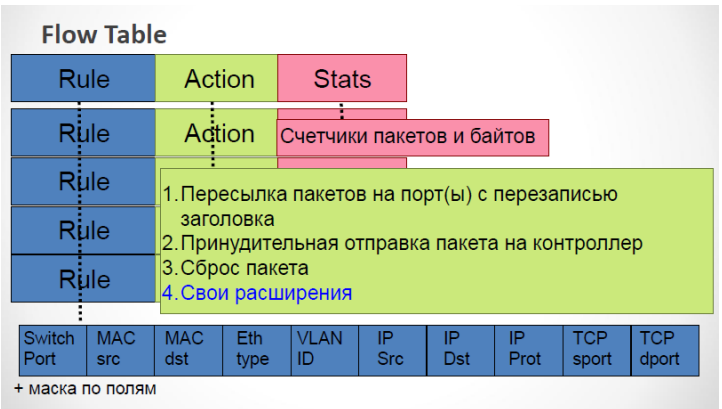
–Hello, Echo

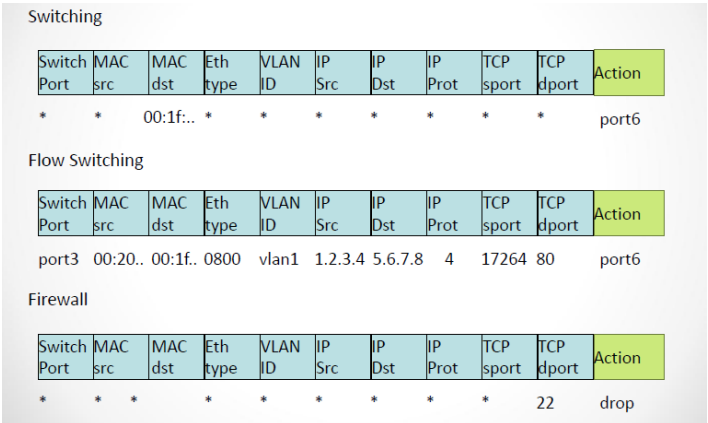
•Ассиметричные сообщения

–Отправка от коммутатора к контроллеру

–Объявляют об изменении состояния сети, состояния коммутаторов

–Packet-in, flow-removed, port-status, error





!!! Не очень понимаю, что значит «принципы установки правил»

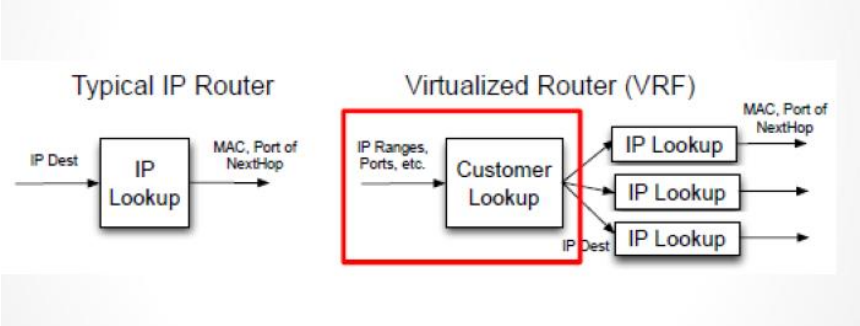
Как правила появляются на коммутаторе, я описал в первом абзаце, выбор подходящего правила происходит следующим образом – у каждого правила есть приоритет, выбирается наиболее приоритетное правило, под которое подходит пакет. Подходит – значит, что соответствующие (т.е. с учётом маски) поля заголовков пакетов совпадают, при этом правила могут быть инициированы приложениями на контроллере, и в зависимости от

порядка работы этих приложений, те или иные приложения могут принять решение остановить обработку пакета, не дав его другим приложениям.

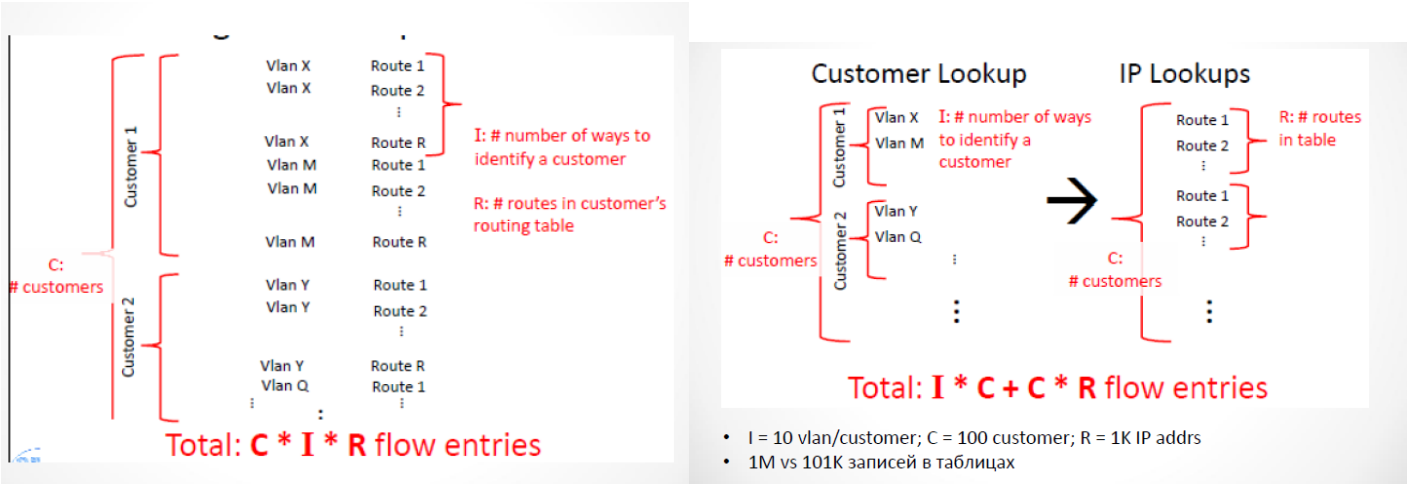
SDN != OpenFlow, т.к. OpenFlow – это протокол общения, описывающий интерфейс контроллера и коммутатора. А SDN – это концепция, подразумевающая существование некоторого контроллера, приложений под него, соответствующих коммутаторов, …
Де факто OpenFlow стал составляющей частью SDN, отвечающей за общение коммутатора с контроллером.

2.3.OpenFlow 1.3. Несколько таблиц потоков, групповые таблицы, Meter таблицы, механизм отказоустойчивости контроллеров. Пример приложения по маршрутизации в SDN/OpenFlow.

В OpenFlow 1.0 для хранения правил использовалась единственная таблица.
Но при таком подходе память, отведенная под хранения таблиц, может расходоваться неэффективно. Простой пример - Virtual Routing and Forwarding.

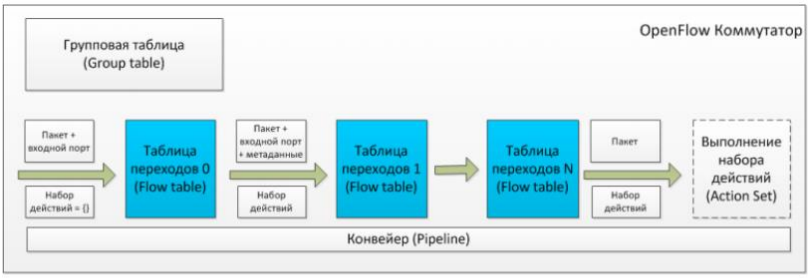


Т.е. сначала мы определяем пользователя, к нам обратившегося, затем для него уже смотрим, как маршрутизировать пакет.
Если мы храним все в одной таблице, то для каждого пользователя нам надо хранить все способы, которыми он может представиться и для каждого из способов хранить полный набор правил маршрутизации.



В случае 2-х таблиц, в одной таблице храним данные, по которым мы можем идентифицировать пользователя, а в другой – правила маршрутизации для каждого пользователя.

Таким образом, с пакетом происходит примерно следующее:

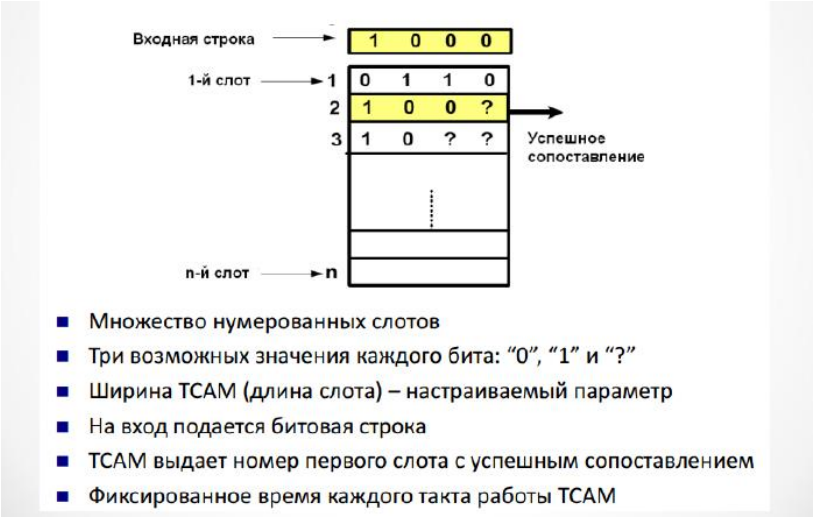


- Продвижение пакета только вперед
- Переход: модификация пакета, обновление набора действий, обновление метаданных

Также OpenFlow поддерживает групповые таблицы. Такая таблица позволяет объединять потоки группами и задавать действия для всей группы сразу.
Идентификатор группы – Тип группы – Счётчики – Контейнеры действий
Типы групп:
All – выполняются все контейнеры действий в группе
Select – выполняется только один контейнер действий в группе
Indirect – выполняется один определённый контейнер действий в группе
Fast failover – выполняется первый существующий (живой) контейнер действий
Польза:
•Экономия места для одинаковых действий
•Также для реализации сетевых механизмов:
–Multicast
–ECMP
–Active/Standby маршруты

Для реализации QoS и ограничения скорости используются meter таблицы.
–Для каждого потока или группы потоков
–Следит за превышение значений счетчиков
–Действия: drop или dscp remark
Meter Identifier – Meter Bands – Counters
Band Type – Rate – Counters – Type specific arguments

Для хранения таблиц используется TCAM (троичная память).

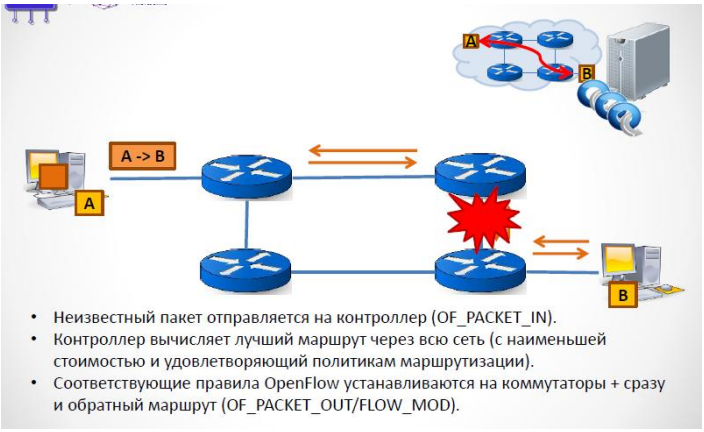


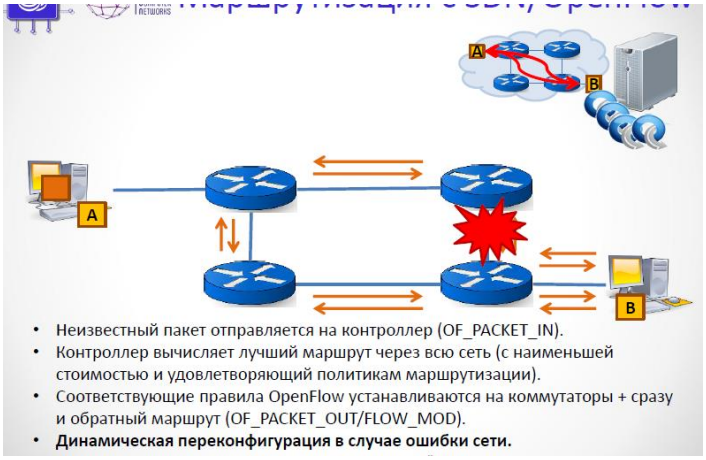
- Множество нумерованных слотов
- Три возможных значения каждого бита: “0”, “1” и “?”
- Ширина TCAM (длина слота) – настраиваемый параметр
- На вход подается битовая строка
- TCAM выдает номер первого слота с успешным сопоставлением
- Фиксированное время каждого такта работы TCAM

Т.к. контроллер централизованный, то естественно возникает вопрос о его отказоустойчивости. Один из возможных способов как-то ее обеспечить – использовать множество контроллеров в сети. OpenFlow обеспечивает механизм ролей. Поддерживается три роли – master, slave и equal (по умолчанию).

Распределением ролей занимаются сами контроллеры, смена производится через OFPT_ROLE_REQUEST. В случае equal – коммутатор одновременно взаимодействует с обоими контроллерами.
В итоге имеем, что каждый коммутатор подключен к нескольким контроллерам и, в случае отказа одного из них, может переключиться на работу с другим. В качестве дополнительного бонуса получаем возможность балансировать нагрузку на контроллеры.
Контроллер с ролью equal и master имеет одинаковые(полные) права доступа к свитчу. Различие в том, что свитч убеждается, что он подключен ровно к одному master'у. Slave-контроллер имеет read-only доступ к коммутатору.
Смена статуса происходит всегда в результате запроса одного из контроллеров. Свитч может быть одновременно подключен к нескольким equal контроллерам, нескольким slave-контроллерам и не более, чем одному master-контроллеру.

Пример маршрутизации.





2.4.Варианты применения SDN/OpenFlow в корпоративном сегменте, телеком. операторы и сервис провайдеры, ЦОД и облачные вычисления.

Области применения:
Компании
Телеком операторы и сервис провайдеры
ЦОД и облака

Корпоративная сеть
• Современные компания имеют сложную сетевую инфраструктуру:
– Большое количество сетевых элементов
– Разветвленная топология
– Набор различных политик маршрутизации и безопасности
• Сетевые администраторы отвечают за поддержания работы сетевой инфраструктуры:
– Сетевые инженеры руками переводят высокоуровневые политики в низкоуровневые команды
– Ручная настройка всех сетевых устройств
– Ограниченный инструментарий по управлению сетевыми устройствами
– Переучивание под каждого вендора
• Существующие системы управления (Примеры: Cisco Prime, HP OpenView, IBM Tivoli, OpenNMS)
- Предназначены для мониторинга состояния: топология, характеристики каналов, загрузка каналов и задержка.
- Основы на протоколе SNMP.
- Конфигурация оборудования по-прежнему происходит в ручном режиме.
Цель:
1) Сделать сеть управляемой без ручного доступа к оборудованию.
2) Повысить уровень абстракции управления сетью.

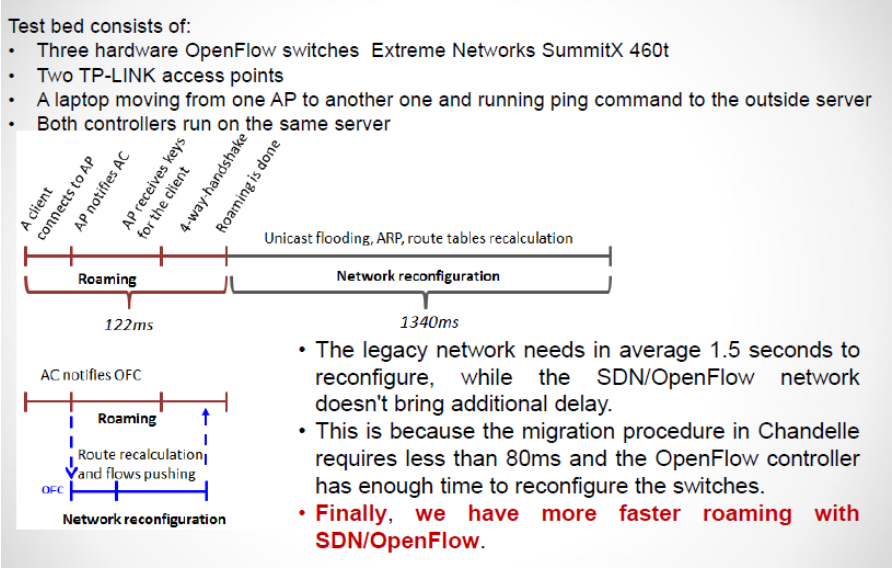
Семантическое управление сетью:
• Имена. Работа с именами хостов проще, чем запоминать связки IP и MAC адресов.

• Группы. Хосты объединяются в группы, удобно для задания политик. Вместо подсетей.
• Пути. Задавать маршруты через всю сеть сразу, а не для каждого устройства отдельно.

Бонус: в связи с тем, что теперь IP адреса скрыты, выбираем их сами, так как нам это удобно. Например, уменьшение правил, агрегирование потоков.

Основные функции:
• Автоматическое определение топологии.
• Мониторинг загрузки сети (пропускная способность, задержки).
• Именование.
• Группы.
• Выбор маршрутов.
• Поддежка QoS.
• Балансировка нагрузки.
• Firewall и ограничения доступа.
• NAT

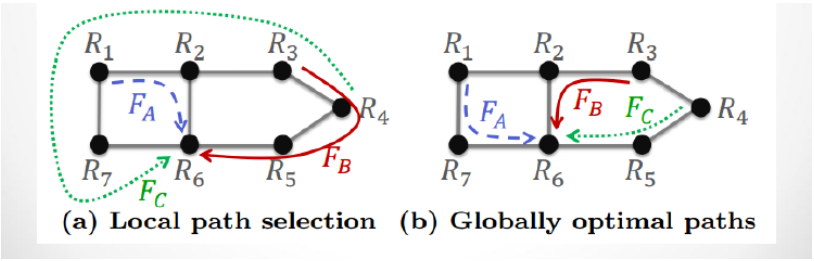
Пример использования SDN/OpenFlow для администрирования:



Есть две точки wi-fi доступа, между ними носят ноут и пингуют внешний сервер. За счет того, что при смене точки доступа не надо заново слать arp-ы, строить таблицы маршрутизации получаем большой выигрыш во времени.

Телеком

1. Интеллектуальный Traffic Engineering:
• Выбор оптимального пути
• Реакция на отказ канала
• Резервирование пропускной способности



Как применить все это на практике?
– Greenfield? !!! что это значит?
– Проблемы интеграции с традиционной сетью
• Нужно подыгрывать протоколам традиционной сети, т.е. правильно отвечать на запросы.
• Чем меньше стыков с традиционной сетью, тем лучше.
– Проблема интеграции с существующими системами управления
!!! что можно ещё здесь дописать?

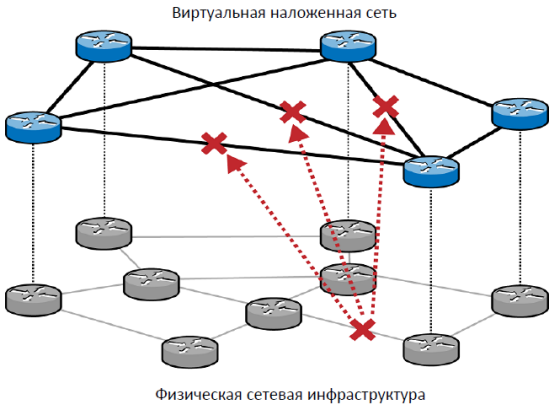
ЦОД/Облака

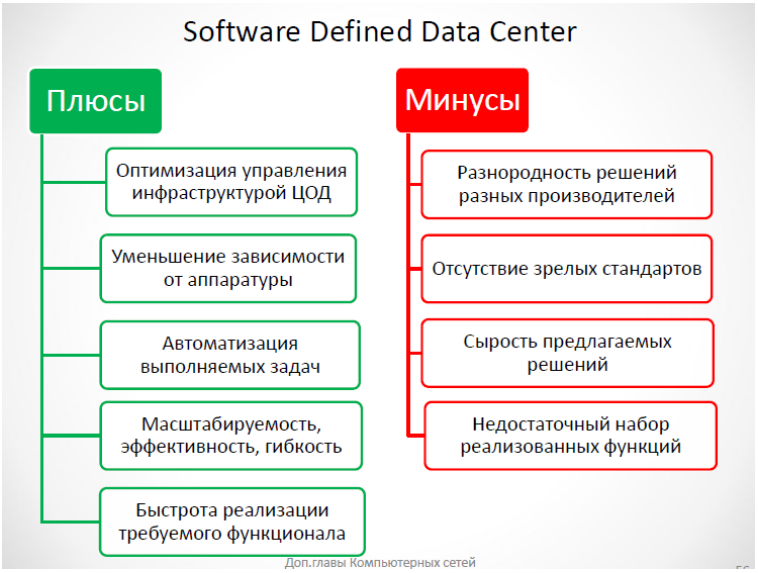
• Повышение утилизации оборудования и каналов
• Мониторинг и оптимизация потоков
• Виртуализация сети пользователей
• Балансировка нагрузки
• Обеспечение качества доступа

Как правило есть два SDN:

– Без OpenFlow (так в OpenStack)
ТОЛЬКО виртуальные каналы
Туннели, таблицы, новые VM, политики
– С OpenFlow для управления физическими устройствами

• Качество канала, определение узких мест



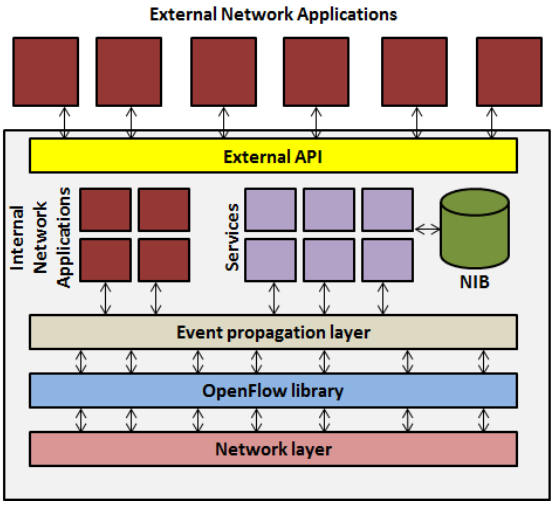


2.5.OpenFlow контроллер. Архитектура и принцип работы. Требования к контроллеру OpenFlow. Экспериментальное исследование и методика. Достоинства и недостатки методики.

OpenFlow контроллер – это:

• программа, TCP/IP сервер, ожидающий подключения коммутаторов
• Отвечает за обеспечение взаимодействие приложения-коммутатор.
• Предоставляет важные сервисы (например, построение топологии, мониторинг хостов)
• API сетевой ОС или контроллер предоставляет возможность создавать приложения на основе централизованной модели программирования.

Архитектура контроллера:



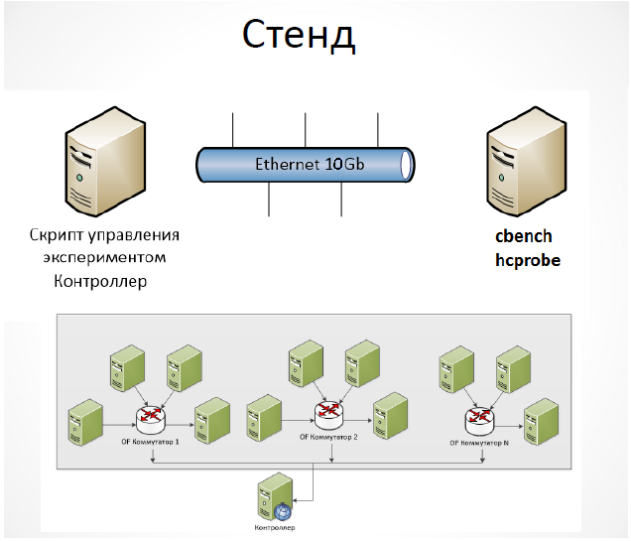
Контроллер имеет несколько уровней:
- Сетевой уровень, на нем обрабатываются пакеты. Сразу возникает требование быстрой обработки пакетов.
- Уровень библиотеки OpenFlow и уровень распространения событий.
- Далее идут внутренние и внешние приложения, которые принимают решения, что делать с пришедшим пакетом.
- Так же есть несколько сервисов (например, определение топологии) и база данных

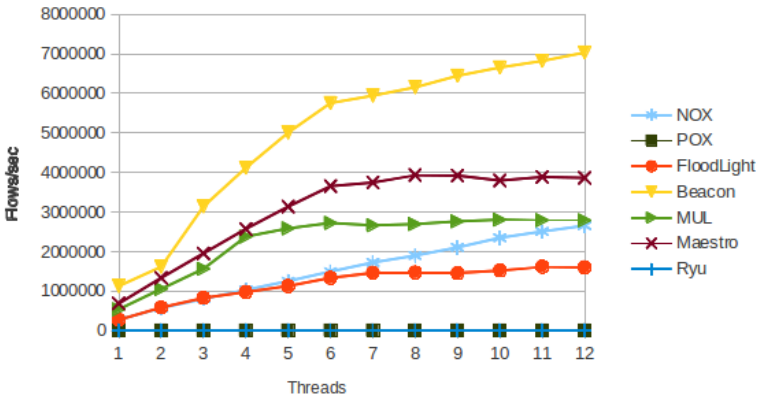
OpenFlow контроллер должен содержать в себе:
- Маршрутизация, виртуализация, мониторинг, балансировка, фильтрация, аутентификация, NAT, ARP, DNS, HDCP, BGP.
- Алгоритмы на графах, распределённые системы и базы данных, web.
- Многопоточные алгоритмы.
- Быстрая обработка пакетов.

Требования к контроллеру ПКС:
• Производительность
– Пропускная способность (events per second) (ЦОД требует обработку >10M событий в секунду)
– Задержка (us)
(Реактивные контроллеры более “чувствительные”)
• Надежность и безопасность (24/7)
• Программируемость
– Функциональность: приложения и сервисы
– Интерфейс программирования

Экспериментальное исследование:
• Производительность
– максимальное количество запросов на обработку
– время обработки запроса при заданной нагрузке
• Масштабируемость
– изменение показателей производительности при увеличении числа соединений с коммутаторами и при увеличении числа ядер процессора
• Надежность
– количество отказов за время тестирования при заданном профиле нагрузок
• Безопасность
– устойчивость к некорректно сформированным сообщениям протокола OpenFlow

Результаты сравнения (2013):





• Максимальная производительность 7 000 000 потоков в секунду.
• Минимальное время задержки от 50 до 75 мкс.
• Недостатки:
– Надежность контроллеров вызывала вопросы
– Производительность была не достаточна (DC >10M fps)

Недостатки методики:

Теоретически контроллеры будут вести себя по-разному в разных ситуациях, и на эту тему можно развести такой же разговор, как и на тему производительности суперкомпьютеров, Т.к. производительность зависит от характера (типа) нагрузки.

2.6. Производительность и программируемость OpenFlow контроллеров. Способы улучшения производительности. Проблематика Northbound API и варианты решения.

Повышение производительности:

Самые ресурсоемкие задачи:
• Взаимодействие с OpenFlow коммутаторами:
– использование многопоточности,
– учет загрузки нитей и перебалансировка
• Получение OpenFlow пакетов из канала:
– чтение пакетов из памяти сетевой карты, минуя сетевой стек OS Linux,
– переключение контекста,
– виртуальные адреса

Пример (In-kernel контроллер) - контроллер был реализован в ядре ОС Linux:
• Супер-производительный
– нет переключений контекста при сетевом взаимодействии
– меньше времени на работу с виртуальной памятью
• Но очень сложно разрабатывать свои приложения
– Низкоуровневый язык программирования
– Ограниченное число библиотек и средств отладки
– Высокий риск “положить” всю систему
• Производительность равна 30M fps
• Задержка 45us

В итоге имеем, что если мы делаем контроллер полностью в userspace, то имеем большую функциональность, но работает все медленно. Если полностью перенести контроллер в kernel-space, то писать приложения для него тяжело, риск все сломать, но зато все быстро работает.

Компромиссное решение – разместить наиболее используемые сервисы (например, определение топологии) и работу с пакетами в kernelspace, а приложения перевести в userspace, предоставив интерфейс взаимодействия kernel и user частей.

Программируемость

• На языке контроллера [быстро]
• На любом языке через REST интерфейс [медленно]
• Специальные языки программирования с другой абстракцией (например, Pyretic, Maple)

Проблематика NorthBound API

• NorthBound API – интерфейс между контроллером и приложениями
• Программирование с OpenFlow не простая задача!
– Сложно выполнять независимый задачи (routing, access control)
– Низкоуровневая абстракция
– Нужно помнить о правилах на коммутаторах
– Порядок установки правил на коммутаторах неизвестен
• Переносимость приложений между контроллерами

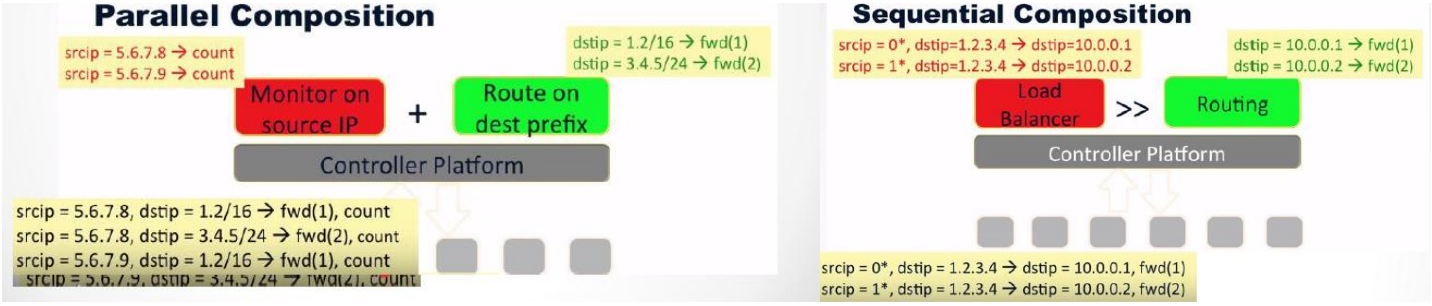
<u>В качестве возможной проблемы можно привести такой пример</u>:

Есть два приложения, работающие независимо. Одно создает правило, что какой-то поток надо направить на порт 1, а другое – что этот же поток надо направить на порт 5. Заранее неизвестно, в каком порядке расположатся правила в коммутаторе, а пакеты будут идти только по первому правилу, поэтому работать будет только одно приложения. Хотя можно было просто сделать отправку потока на два порта.

В качестве возможного решения подобных проблем придумали композицию приложений.

Два типа композиции (на примере, Pyretic)

– Параллельная – выполняет оба действия одновременно (forwarding и couting)
– Последовательная композиция – выполняет одно действие за другим (firewall, затем switch)



2.7.Распределенный уровень управления в SDN/OpenFlow. Основные угрозы. Стратегии резервирования. Основные задачи и варианты решения.

<u>Требования высокой доступности</u>:

• Сеть работает в режиме 365/24/7.
• Платформа управления ПКС должна работать непрерывно.
• Цель обеспечения высокой готовности – поддержание непрерывной работоспособности платформы управления, сетевых приложений.
• Причины простоя: обслуживание, программные и аппаратные ошибки, отказы оборудования, атаки, отключение электроэнергии, аварии.

При коэффициенте готовности 99% за год система простаивает 3,7 дней, при 99,999% – 5 минут.

<u>Возможные угрозы</u>:

- Потеря соединения с контроллером
- Отказ контроллера
- Перегрузка контроллера

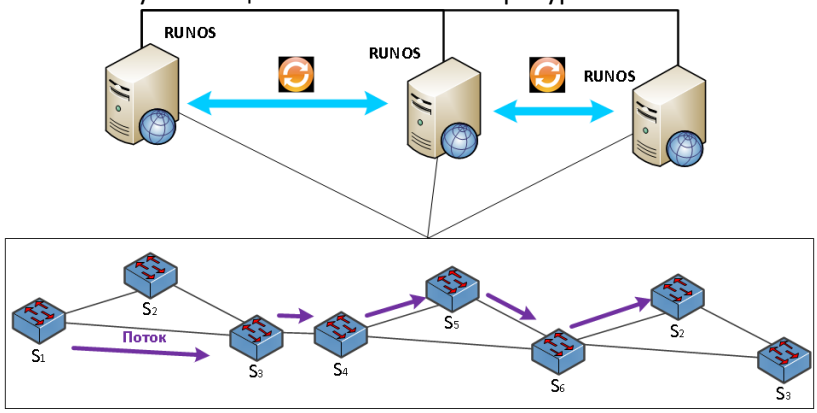
<u>Active/Standby (Passive) стратегии резервирования</u>:

• Холодное - [без синхронизации]

• Теплое - [периодическая синхронизация]
• Горячее - [постоянная синхронизация]

При отказе основного контроллера происходит переключение на работу с запасным. Такое решение решает проблему одиночного отказа контроллера, но две других остаются нерешенными. Особенности решения:
• OpenFlow ≥ 1.0
• Корпоративные сети.
• Не масштабируется
• Не полная утилизация вычислительных ресурсов

Другая стратегия резервирования – Active/Active.
– Ассиметричная (!!! *видимо* master-slave, т.е. контроллеры имеют неодинаковый доступ к свитчам)
– Симметричная (контроллеры работают совместно (не копируя действия друг друга, а именно разделяя задачи, и обмениваясь друг с другом необходимой информацией)
• Высокая сложность (Требуется координация контроллеров, поддержка глобального состояния)
• Высокая доступность (минимальное время простоя)
• Высокая утилизация вычислительных ресурсов

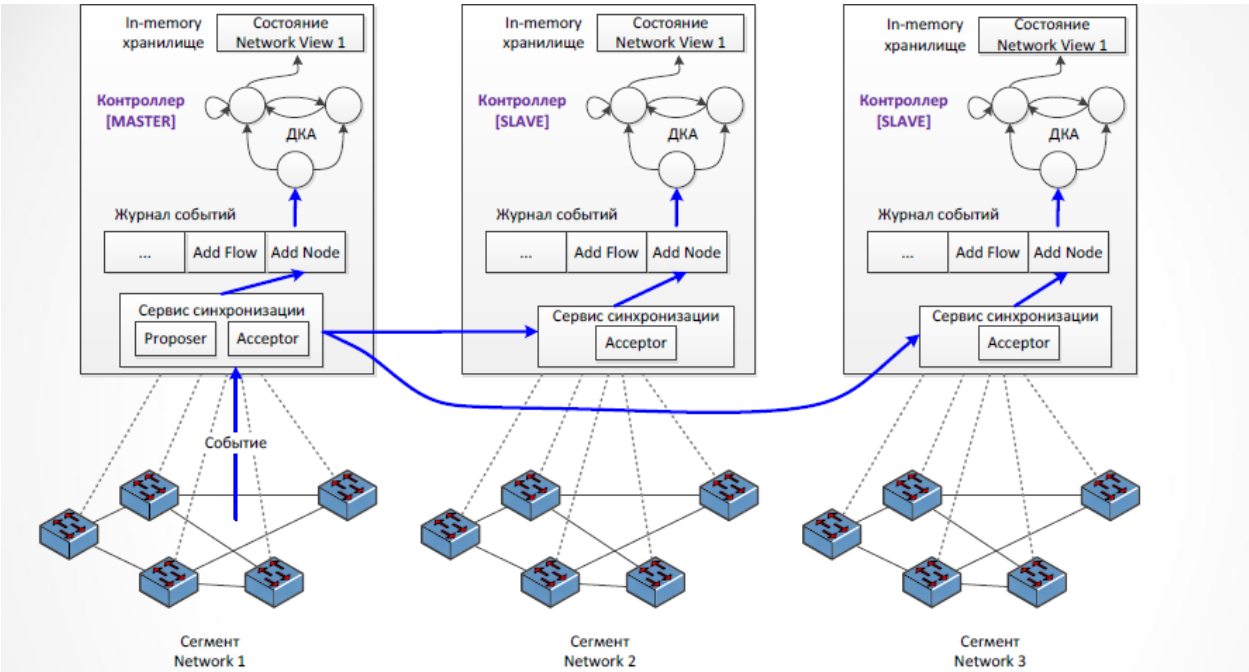


При этом контроллер является master для некоторой своей группы коммутаторов, но в то же время соединён со всеми остальными и является slave для них (т.е. он в любом случае слушает все коммутаторы).

Основные задачи:
1. Как синхронизовать контроллеры?
Решение основано на алгоритме «multi-paxos». Алгоритм заключается примерно в следующем:
Сначала лидер (в нашем случае – это master-контроллер) формирует предложение с уникальным идентификатором N. Затем каждый acceptor шлет лидеру сообщение, что он не будет принимать предложения с идентификатором меньше N (promise step). Если лидеру поступает достаточное количество обещаний, то он формирует запрос на изменение значения. Принимающий в свою очередь смотрит на идентификатор запроса и, если он пообещал принять его, то изменяет соответствующее значение.

```
Client   Proposer       Acceptor     Learner
   |         |          |  |  |       |  |  --- Following Requests ---
   X-------->|          |  |  |       |  |  Request
   |         X--------->|->|->|       |  |  Accept!(N,I+1,W)
   |         |<---------X--X--X------>|->|  Accepted(N,I+1,W)
   |<-----------------------------------X--X  Response
   |         |          |  |  |       |  |
```

Обеспечивает одновременное одинаковое выполнение команд контроллеров
Команды: изменение Network View, установление новых потоков (чтение NV), изменение ролей контроллеров



2. Как определить начальное распределение коммутаторов по контроллерам?

Критерий определения Master-контроллера:

–Задержка от коммутатора до контроллера, как мера близости.

• Ограничение: на количество коммутаторов в сегменте контроллера.

• Решение: Жадный алгоритм по значению задержки от коммутатора до контроллеров.

3. Как перераспределить управление коммутаторами при одиночном отказе контроллера?

Проактивная разработка сценариев восстановления

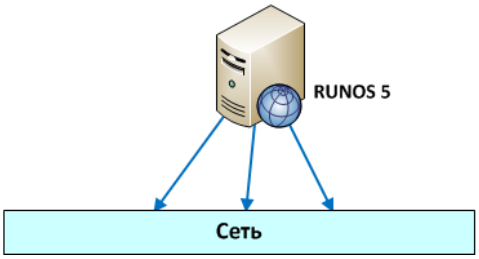
• Критерий выбора нового распределения: задержка от коммутатора до контроллера

• Алгоритм Балаша.

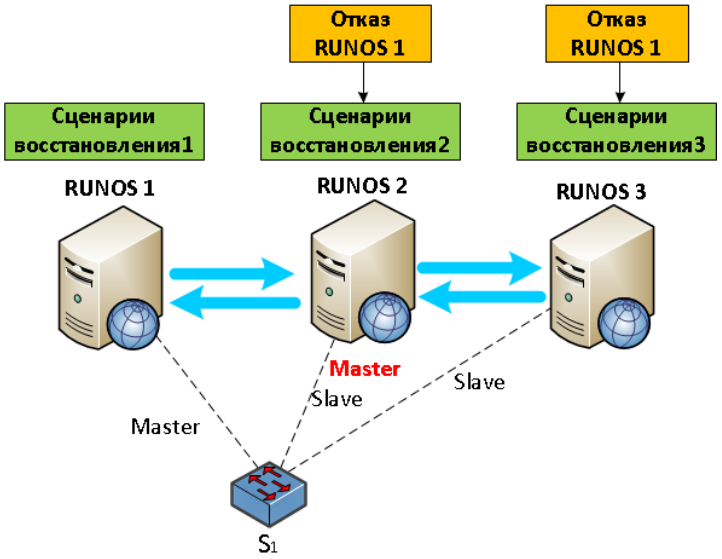
• Результат: перечень сценариев для каждого контроллера платформы управления.

Сценарии восстановления

Отказ контроллера	Перечень коммутаторов, для которых контроллер должен стать Master-ом
RUNOS 01	S3,S4
RUNOS 02	S1, S5
...	...
RUNOS N	S k



1. Оставшимися контроллерами фиксируется факт отказа контроллера и CID.
2. Каждый контроллер выбирает сценарий восстановления, соответствующий отказу контроллера CID.
3. В соответствии со сценарием каждый контроллер изменяет свою роль на коммутаторах.
4. Каждый контроллера информирует остальные контроллеры о завершении выполнения сценария восстановления.



4.Как восстановить управление коммутатором при потере соединения с контроллером?

1. Фиксируется отключение коммутатора Master-контроллером.
2. Инициируется проверка присутствия коммутатора в сети.
3. Осуществляется измерении задержки контроллерами, поддерживающими связь с коммутатором.
4. Master-контроллером становится контроллер с минимальной задержкой до коммутатора.
5. Новый Master-контроллер устанавливает свою роль на коммутаторе.

Здесь возникает проблема – как определить, отказал коммутатор или линия связи?

5.Как предотвратить перегрузку контроллера в платформе управления?

Для обнаружения перегрузки:

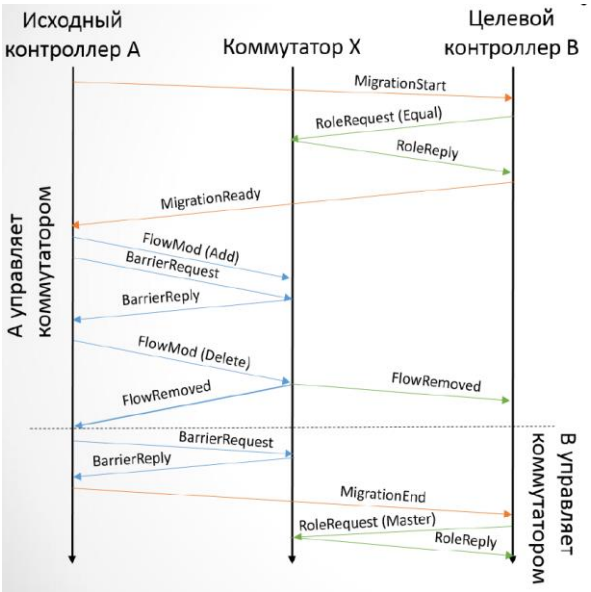
• Устанавливаются пороговые значения параметров загрузки контроллеров.

• Осуществляется постоянный мониторинг загрузки контроллеров - [количество коммутаторов, количество packet-in запросов в секунду, CPU, память]

• Факт перегрузки фиксируется при обнаружении превышения порогового значения.

Задача перераспределения коммутаторов по контроллерам: Необходимо перераспределить коммутаторы так, чтобы:

1. Нагрузка на каждый из контроллеров не превышала его производительности
2. Использовалось минимальное количество контроллеров.
3. При перестроении управления сетью было проведено минимальное количество передач управления коммутаторами.



Передача управления коммутатором должна обладать следующими свойствами:

• Свойство живучести: в любой момент времени для коммутатора существует контроллер, работающий в режиме Master. Для любой асинхронной команды, контроллер, который ее отправил, остается активным до тех пор, пока коммутатор не закончит ее обработку.

• Свойство безопасности: Ровно один контроллер обрабатывает каждое асинхронное сообщение от коммутатора.

Все решение задачи предлагается разить на 2 этапа:

1) Разобьем все коммутаторы на группы так, чтобы нагрузка каждой группы не превышала некоего порогового значения P.

2) Сформируем список необходимых операций передачи управления отдельными коммутаторами.

Более подробно:

Распределение коммутаторов по группам

• Входные данные:

– Топология сети

– Количество новых потоков с каждого коммутатора

– Максимально допустимая нагрузка на контроллер P

• Алгоритм разбиения графа на компоненты связности, чтобы суммарная нагрузка компоненты связности не превышала P.

• Выходные данные:

– Матрица, определяющая Master-контроллеры для коммутаторов и сегменты управления для каждого контроллера.

Распределение групп коммутаторов по контроллерам

• Входные данные:

– Текущая матрица распределения управления коммутаторами по контроллерам.

– Желаемая матрица распределения управления коммутаторами.

• Алгоритм: жадный алгоритм, минимизирующий количество передач управления коммутаторами между контроллерами платформы управления.

• Выходные данные:

– Список операций по передаче управления отдельными коммутаторами.

2.8. Виртуализация сетевых сервисов (NFV). Проблемы телеком. операторов. Уровни развития NFV. Архитектура и основные термины по ETSI. Варианты применения.

Проблемы телеком операторов:

– Количество трафика растет, требуется больше сетевого оборудования, НО доход не растет.

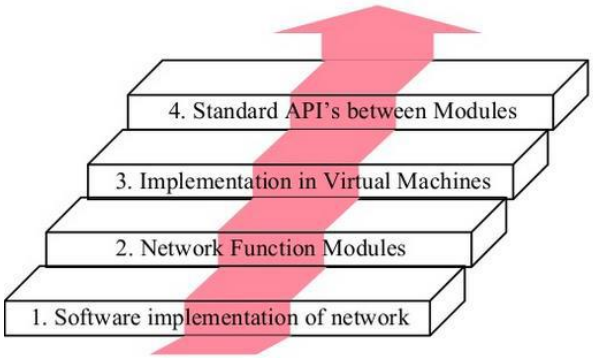
– Инфраструктура состоит из проприетарного дорого оборудования.

– Статическое распределение ресурсов.

– Вывод новых сервисов занимается до 18 месяцев.

Если раньше под каждую задачу создавалась специальная железка, то сейчас предлагается использовать стандартные сервера, с установленным на них софтом, выполняющим сетевые функции.

Уровни развития NFV:



1) Переход от аппаратных решений к программным. Роутеры, Firewalls, маршрутизаторы широкополосного удаленного доступа.
2) Функциональные модули и на уровне данных, и на уровне управления – NAT, DHCP, ограничение пропускной способности.
3) Переход к виртуальным машинам, со всеми вытекающими бонусами – масштабируемость, быстрое резервирование и т.д.
4) Создание стандартных интерфейсов взаимодействия, чтобы изолировать друг от друга компоненты. Для этого создали ISG (Industry Specification Group) в ETSI (European Telecom Standards Institute).

Преимущества

NFV - перенос сетевых функций на виртуальные машины:

• Упрощение развертывания и обновления как софта, так и железа

• Уменьшение стоимости за счет использования стандартных серверов
• Объединение сервисов в группы

Архитектура по ETSI

NFV имеет три основных компонента:

1) Виртуализованные сетевые функции (virtualized network functions, VNF) – программные реализации сетевых функций, которые могут быть размещены в рамках некоторой инфраструктуры.
2) Инфраструктупа виртуализации сетевых функций (Network function virtualization infrastructure, NFVI) – совокупность железа и софта, образующая среду, в которой работают VNF.
3) Фреймворк управления (Network functions virtualization management and orchestration architectural framework, NFV-MANO Architectural Framework) – множество функциональных блоков, данных, используемых этими блоками и интерфейсов, через которые эти блоки общаются с целью управления работой NFVI и VNF. По большому счету занимается созданием экземпляров VNF, наблюдением за их работой, их починкой. Важно отметить, что данная компонента должна уметь работать с VNF независимо от того что у этой VNF «под капотом».

Основные термины:

NF – Network Function – функциональный блок (абстрактная элементарная единица) с известными интерфейсами и функциональным поведением.

VNF – Virtualized Network Function – программная реализация NF, которая может быть развёрнута в виртуальной инфраструктуре.

VNF set – множество VNF (соединения между ними не специфицированы), специфицирован только абонентский шлюз

VNF Forwarding Graph – цепочка сервисов, в случае, когда порядок сетевых соединений важен (например, firewall, NAT, балансировщик)

NFV Infrastructure – NFVI – необходимый софт и железо, для развёртывания, управления и выполнения сетевых функций (включая потребности вычислительные, сетевые и storage).

NFVI Point of Presence (PoP) – местоположение NFVI (*наверно, местоположение в сети, относительно других*)

NFVI-PoP Network – внутренняя сеть

Transport Network – сеть, соединяющая PoP с другими PoP или внешними сетями

VNF Manager – контролирует жизненный цикл сетевых функций, например, создание, обновление, масштабирование, мониторинг, диагностика падений, …

Virtualized Infrastructure Manager – менеджмент вычислительных, сетевых, storage и программных ресурсов

Network Service – совокупность сетевых функций и спецификации их поведения

NFV Service – network service использующий хотя бы одну сетевую функцию (NF) и как минимум одну VNF (программная реализация сетевой функции)

User Service – сервисы, предоставляемые конечным пользователям/покупателям/подписчикам.

Deployment Behavior – NFVI ресурсы, которые требует VNF, например, количество виртуалок, памяти, диска, задержки, полоса подключения.

Operational Behavior – топология экземпляров VNF и жизненный цикл операций, таких как, старт, стоп, пауза, миграция, …

VNF Descriptor – Deployment Behavior + Operational Behavior

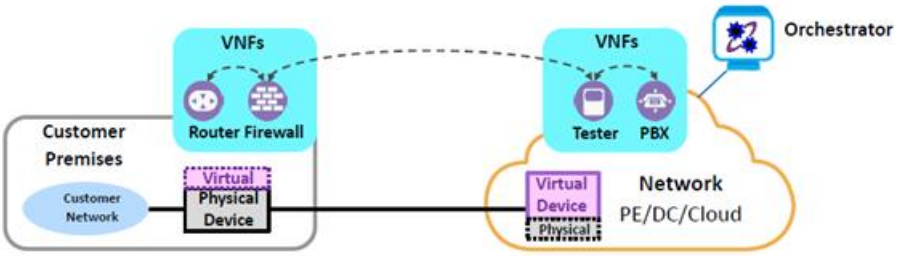
NFV Orchestrator – автоматизирует развёртку, функционирование, менеджмент и координацию VNF и NFVI.

Варианты применения:

- Облако:
  - NFVIaaS – NFV infrastructure as a service
  - VNFaaS – Virtual network functions as a service
  - VNF forwarding graphs (цепи сервисов)
  - VNPaaS – Virtual Network Platform as a service
- Мобила:
  - Виртуализация мобильной коры и IMS
  - Виртуализация мобильных базовых станций

- ЦОД:
    Виртуализация CDN (Content Delivery Network)
- жилой доступ (домохозяйства):
    Виртуализация домашнего окружения (Virtualization of the Home environment)
    Фиксированный доступ через NFV

Частные примеры:
vCPE – virtual Customer Premise Equipment:
- У клиента – маленькая коробочка
- Часть сервисов у клиента, часть в облаке



BRAS – для существующих решений одно подключение домохозяйства стоит 1$, с NFV – 0.1$
CG-NAT – 16$ -> 4$ -> 2$
NTT DoCoMo – динамическое перераспределение ресурсов

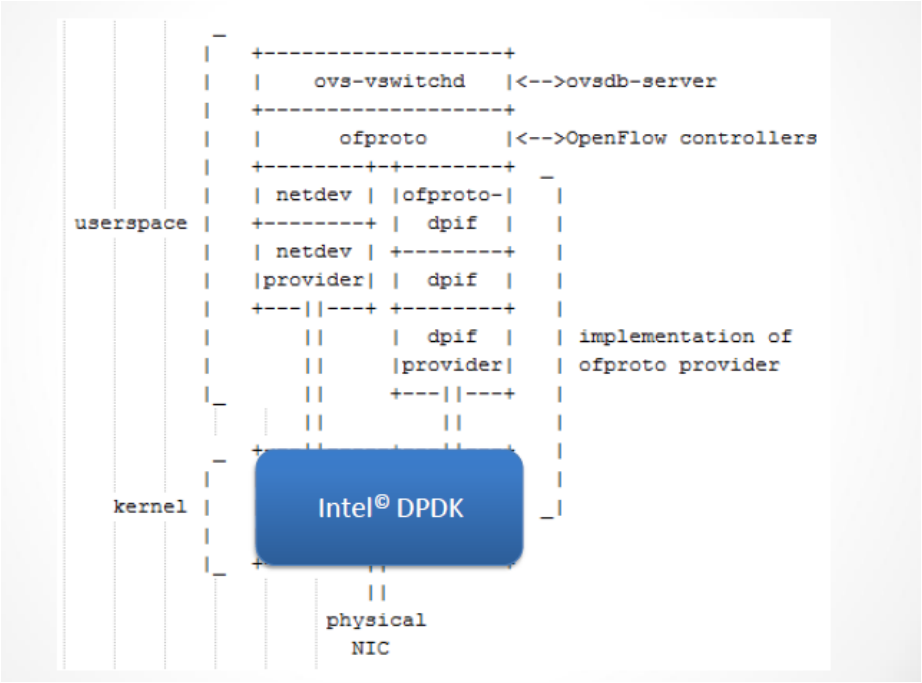
2.9.Проблематика производительности сетевых сервисов. Суть проблемы, узкие места и варианты решения.

Любой сервис у нас по большому счету состоит из трех компонентов – железной части, некоторой виртуальной машины, отвечающей за логику работы и некоторой сущности, которая этим всем добром управляет.
И если не пользоваться best-practices, то вся эта система работает раз в 10 медленнее, чем нужно, и чем могла бы работать.
Рассмотрим узкие места, имеющиеся в приложениях.
В первую очередь мы сталкиваемся с сетевым стеком линукса, который, несомненно, хорош, но зачастую избыточен и в итоге работает слишком медленно. Возможное решение – intel DPDK – набор библиотек, которые позволяют быстро обрабатывать пакеты. Основные особенности этого набора:

- Использование больших страниц (2мб/1гб)
- Объекты размещаются непрерывно по всем каналам оперативной памяти
- Адресное пространство сетевой карты доступно напрямую из userspace
- Неблокирующие очереди для передачи пакетов.
- Нет прерываний в драйверах DPDK – активный цикл.
- Активное использование SSE инструкций для обработки пакетов.
- Выделение целых ядер процессоров под задачи.

Два других узких места – open vSwitch и виртуальная машина.
Open vSwitch - это виртуальный программный коммутатор, который обеспечивает соединение между виртуальными машинами и физическими интерфейсами.
Поддерживает обычную Ethernet коммутацию с VLAN, SPAN, RSPAN, GRE, sFlow, Netflow. Частичная поддержка OpenFlow 1.2.

Архитектура Open vSwitch:



Видно, что состоит он из двух частей, одна работает в userspace и отвечает за работу с opeflow, другая, отвечающая за работу с пакетами, расположена в kernelspace и использует упомянутый Intel DPDK.

Работа с виртуальными машинами:
Virtio – стратегия виртуализации, при которой гостевая ОС знает, что она виртуальна и взаимодействует с гипервизором посредством специальных драйверов.
«+»: Прозрачно для приложений на виртуальных машинах
«-»: Медленно
IVSHMEM – технология, которая позволяет организовать data sharing между виртуальными машинами без копирования. Работает за счет того, что несколько больших страниц отображаются на одно IVSHMEM-устройство. Плюс добавляется файл с метаданными. Каждая машина работает с таким устройством, получая доступ к страницам памяти.
«+»: Самая высокая скорость
«-»: Требует “затачивания” сервиса под Intel vSwitch

VHOST (Это модуль ядра linux - vhost-net-module для интеграции с dpdk в случае работы под гипервизором KVM):
– Средняя скорость
– Прозрачно для приложений, написанных на DPDK

Требования к сервису:
- Возможность привязки VM к ядрам процессора (не совсем требование к сервису)
- Масштабирования сервиса на VM по имеющимся ядрам процессора
- Возможность запуска сервиса без VM

2.10. Одновременное применение концепций NFV и SDN. Основные задачи. Примеры.

SDN vs NFV:
- Концепция NFV была порождения в SDN.
- NFV и SDN являются различными концепциями, которые ладят вместе, могут быть использованы независимо.
- У обоих одна и та же цель (удешевление), но принципиально разные подходы.
- SDN нуждается в новых интерфейсах, приложениях.
NFV нуждается в перемещении сетевых приложений из железа в виртуальные контейнеры в COTS (Commercial-off-the-shelf) железо
- NFV уже есть, об NFV разговоров гораздо меньше, SDN – ещё впереди
- Виртуализация сама по себе предоставляет много необходимых возможностей

Основные задачи здесь – оркестрация и service chaining с помощью SDN.

А также смотри билет 1.20.

3. Методы моделирования и анализа компьютерных сетей

3.1. Методы моделирования компьютерных сетей. Понятия модели, точности моделирования. Плюсы и минусы каждого метода моделирования.

Модель – сущность/объект, который отображает процессы, протекающие в реальных системах с помощью математических или натурных средств. Отражение процессов осуществляется на основе оценки характеристик (зависимостей) или параметров процессов моделируемых систем
Основные условия выбора метода:
• Постановка задачи
• Составом, характером и объемом исходных данных
• Временем на решение исследовательской задачи

Методы моделирования:
– Натурного или физического
– Аналитического моделирования
– Имитационного моделирования
– Комбинированные методы моделирования

Натурное (физическое) моделирование
Измерение характеристик осуществляется на исследуемых системах в реальном времени (проведение экспериментов). Данные исследователь получает, ведя наблюдение за процессами в реальной системе
Достоинства
• Высокая адекватность модели реальной системе
• Высокая точность результатов

Недостатки
• Высокая стоимость создания модели
• Большие временные затраты
• Необходимость доработки отдельных узлов реально системы для проведения натурных экспериментов
Пример: исследование надёжностных характеристик сетей оператора связи

Аналитическое моделирование:
Модель представляется совокупностью аналитических выражений, которые отражают функциональные зависимости между параметрами реальной системы в процессе ее работы
Аналитические модели применяются для относительно простых систем, для исследования характеристик которых не требуется высока точность
Достоинства
– Простота и низкая стоимость модели
– Возможность быстро получить численные результаты
Недостатки
– Большое число допущений и ограничений
– Не высокая точность результатов
– Соответствие результатов определенным условиям
– Большая сложность аналитического описания функциональных зависимостей

Имитационное моделирование:

Имитационное моделирование — это метод исследования, при котором изучаемая система заменяется моделью, с достаточной точностью, описывающей реальную систему и с ней проводятся эксперименты с целью получения информации об этой системе. Экспериментирование с моделью называют имитацией (имитация — это постижение сути явления, не прибегая к экспериментам на реальном объекте).
Имитационное моделирование — это метод математического моделирования.
Существует класс объектов, для которых по различным причинам, не разработаны аналитические модели либо не разработаны аналитические методы решения полученной модели. В таких случаях математическая модель заменяется имитатором или имитационной моделью.

Достоинства имитационного моделирования:
– Высокая адекватность между физической сущностью описываемого процесса и его моделью
– Возможность описать сложную систему на достаточно высоком уровне детализации
– Значительно большие охват исследования, чем аналитическое моделирование
– Отсутствие ограничений на зависимости между параметрами модели
– Возможность оценки функционирования системы не только в стационарных состояниях, но и в переходных процессах (режимах)
– Получение большого числа данных об исследуемом объекте (закон распределения случайных величин, числовые значения абсолютные и относительные, и многое другое)
– Наиболее рациональное отношение «результат – затраты» по отношению к аналитическому и физическому моделированию

Недостатки имитационного моделирования

- Относительно большая сложность создания модели
- Необходимость высокой квалификации исследователя для написания модели
- Необходимость проведения верификации и валидации данных моделирования
  Верификация (от лат. verus — «истинный» и facere — «делать») — это подтверждение соответствия конечного продукта предопределённым эталонным требованиям.
  Валидация (англ. Validation) - подтверждение на основе представления объективных свидетельств того, что требования, предназначенные для конкретного использования или применения, точно и в полном объёме предопределены, а цель достигнута.
- Индивидуальность реализации. Для широкого применения моделью можно воспользоваться лишь при детальном описании ее построения

Комбинированные методы моделирования
– Модель представляется в комбинации методов моделирования
– Наиболее широко применяются имитационно-аналитические модели
– Степень применения методов моделирования определяет исследователь, исходя из поставленных задач, имеющихся ресурсов (знаний, ноутбука, Hadoop-кластера) и времени на проведение исследовательской работы (курсовик, диссертация…)

!!! Не было дано строгого определения точности моделирования

3.2.Методы имитационного моделирования компьютерных сетей. Системная динамика: основные понятия, примеры моделей.

Имитационное моделирование — это метод исследования, при котором изучаемая система заменяется моделью, с достаточной точностью описывающей реальную систему, и с ней проводятся эксперименты с целью получения информации об этой системе. Экспериментирование с моделью называют имитацией (имитация — это постижение сути явления, не прибегая к экспериментам на реальном объекте).
Имитационное моделирование — это метод математического моделирования. Существует класс объектов, для которых по различным причинам, не разработаны аналитические модели либо не разработаны аналитические

методы решения полученной модели. В таких случаях математическая модель заменяется имитатором или имитационной моделью.

Виды имитационного моделирования:
- Системная динамика
- Агентное моделирование
- Дискретно-событийное моделирование

Системная динамика – парадигма моделирования, где для исследуемой системы строятся графические диаграммы причинных связей и глобальных влияний одних параметров на другие во времени, а затем созданная на основе этих диаграмм модель имитируется на компьютере.
По сути, такой вид моделирования более всех других парадигм помогает понять суть происходящего выявления причинно-следственных связей между объектами и явлениями.
Пример: с помощью системной динамики строят модели бизнес-процессов, развития города, модели производства, динамики популяции, экологии и развития эпидемии. Метод основан Форрестером в 1950 годах.

Системно-динамическая модель состоит из набора абстрактных элементов, представляющих некие свойства моделируемой системы. Выделяются следующие типы элементов:
Уровни — характеризуют накопленные значения величин внутри системы.
Это могут быть товары на складе, товары в пути, банковская наличность, производственные площади, численность работающих. Уровни применимы не только к физическим величинам. Например, уровень осведомленности существенен при принятии решения. Уровни удовлетворения, оптимизма и негативных ожиданий влияют на экономическое поведение. Уровни представляют собой значения переменных, накопленные в результате разности между входящими и исходящими потоками. На диаграммах изображаются прямоугольниками.
Потоки — скорости изменения уровней. Например, потоки материалов, заказов, денежных средств, рабочей силы, оборудования, информации. Изображаются сплошными стрелками.
Функции решений (вентили) — функции зависимости потоков от уровней. Функция решения может иметь форму простого уравнения, определяющего реакцию потока на состояние одного или двух уровней.
Например, производительность транспортной системы может быть выражена количеством товаров в пути (уровень) и константой (запаздывание на время транспортировки). Более сложный пример: решение о найме рабочих может быть связано с уровнями имеющейся рабочей силы, среднего темпа поступления заказов, числа работников, проходящих курс обучения, числа вновь принятых работников, задолженности по невыполненным заказам, уровня запасов, наличия оборудования и материалов. Изображаются двумя треугольниками в виде бабочки.
Каналы информации, соединяющие вентили с уровнями. Изображаются штриховыми стрелками.
Линии задержки (запаздывания) — служат для имитации задержки потоков. Характеризуются параметрами среднего запаздывания и типом неустановившейся реакции. Второй параметр характеризует отклик элемента на изменение входного сигнала. Разные типы линий задержки имеют различный динамический отклик.
Вспомогательные переменные — располагаются в каналах информации между уровнями и функциями решений и определяют некоторую функцию. Изображаются кружком.

~~!!! Далее написана моя интерпретация того, что было сказано на лекции, было бы неплохо, подкорректировать, чтобы было действительно то, что на лекции говорили~~
Пример:
Применительно к сетям на лекции был пример про распространение сетевых червей. Задав связи между машинами в сети, скорость распространения червей и изначально зараженные машины можно моделировать процесс распространения червей в сети.
По сравнению с агентным моделированием основное отличие в том, что идем не от свойств частей системы, а от свойств системы как единого целого.

(это за рамками билета)
Системы построения имитационных моделей*:*
Это специальные системы моделирования, в которых есть

- специальные языки программирования (как общего применения, так и проблемно-ориентированные): GPSS/PC, GPSS/H, GPSS World, Object JPSS, Arena, SimProcess, Enterprise Dynamics, Auto-Mod, SIMPASS, …
- универсальные языки программирования: ~~pascal~~, си, …

Требования	SIMULA	GSPP	NS-2	NS-3	OPNET	OMnet++	Anylogic
различные уровни детализации	+	+	-	-	-	-	+
графический режим	-	-	+	+	+	+	+
представления работы ПО сети	-/+	-/+	+/-	+/-	-/+	-/+	-
отображение процесса моделирования	-	-	+	+	+	-	+
накопление данных по требованию	+	+	-	-	+	-	-
распределенное моделирование	-/+	-/+	-	-	-	-	-
описание процесса обработки PDU	+/-	+/-	+	+	+	+	-
избегание процессов идентификации и калибровки	-	-	-/+	-/+	-/+	-/+	-/+
моделирование сетей размера ГКС	-/+	-/+	-	-	-	-	-

3.3. Методы имитационного моделирования компьютерных сетей. Агентное моделирование: основные понятия, примеры моделей.

Имитационное моделирование —это метод исследования, при котором изучаемая система заменяется моделью, с достаточной точностью описывающей реальную систему, и с ней проводятся эксперименты с целью получения информации об этой системе. Экспериментирование с моделью называют имитацией (имитация — это постижение сути явления, не прибегая к экспериментам на реальном объекте).
Имитационное моделирование — это метод математического моделирования. Существует класс объектов, для которых по различным причинам, не разработаны аналитические модели либо не разработаны аналитические методы решения полученной модели. В таких случаях математическая модель заменяется имитатором или имитационной моделью.

Виды имитационного моделирования:
- Системная динамика
- Агентное моделирование
- Дискретно-событийное моделирование

Агентное моделирование – относительно новое (1990е-2000е гг.) направление в имитационном моделировании, которое используется для исследования децентрализованных систем, динамика функционирования которых определяется не глобальными правилами и законами (как в других парадигмах моделирования), а наоборот. Когда эти глобальные правила и законы являются результатом индивидуальной активности членов группы.
Цель агентных моделей – получить представление об этих глобальных правилах, общем поведении системы, исходя из предположений об индивидуальном поведении ее отдельных активных объектов и взаимодействии этих объектов в системе.
Агент – некая сущность, обладающая активностью, автономным поведением, может принимать решения в соответствии с некоторым набором правил, взаимодействовать с окружением, а также самостоятельно изменяться.

В качестве примера можно привести моделирование стаи птиц, зная поведение одной птицы, мы можем его запрогать и, обсчитывая поведение каждой особи с течением времени, моделировать поведение стаи.
Аналогично с распространением червей, задав поведение одного червя можно смоделировать распространение.
Из недостатков стоит отметить вычислительную сложность – моделирование систем из большого числа объектов требует много ресурсов.
По сравнению с системной динамикой основное отличие – что идем не от свойств системы как целого, а от частей системы.

3.4.Методы имитационного моделирования компьютерных сетей. Дискретно-событийное моделирование: основные понятия, примеры моделей.

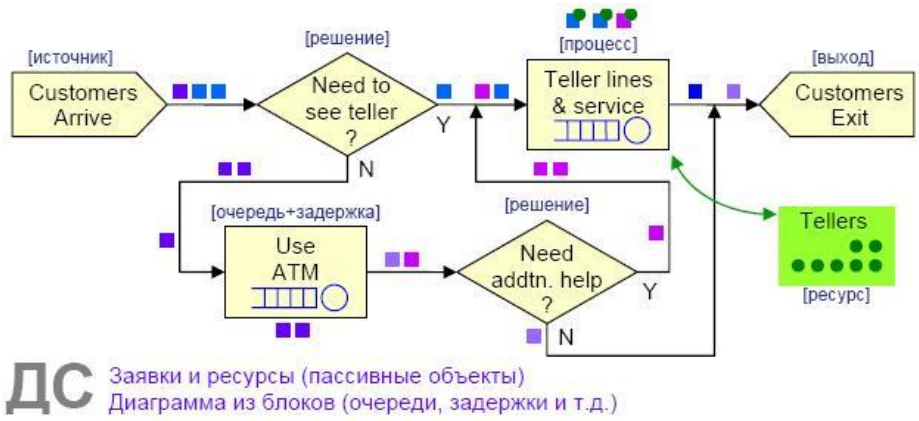
Имитационное моделирование —это метод исследования, при котором изучаемая система заменяется моделью, с достаточной точностью описывающей реальную систему, и с ней проводятся эксперименты с целью получения информации об этой системе. Экспериментирование с моделью называют имитацией (имитация — это постижение сути явления, не прибегая к экспериментам на реальном объекте).
Имитационное моделирование — это метод математического моделирования. Существует класс объектов, для которых по различным причинам, не разработаны аналитические модели либо не разработаны аналитические методы решения полученной модели. В таких случаях математическая модель заменяется имитатором или имитационной моделью.

Виды имитационного моделирования:
- Системная динамика
- Агентное моделирование
- Дискретно-событийное моделирование

Дискретно-событийное моделирование – подход к моделированию, предлагающий абстрагироваться от непрерывной природы событий и рассматривать только основные события моделируемой системы, такие как: «ожидание», «обработка заказа», «движение с грузом», «разгрузка» и другие.
Этот вид моделирования наиболее подходит для моделирования процессов в телекоммуникационных сетях.

Дискретно-событийное моделирование наиболее развито и имеет огромную сферу приложений – от логистики и систем массового обслуживания до транспортных и производственных систем.
Основан Джеффри Гордоном в 1960х годах.
Одной из особенностей такого типа моделирования является то, что система, по сути, не существует, когда нет событий. При этом можно ввести сущность часов, которые будут генерировать события-тики времени с заданной периодичностью, таким образом получив систему, работающую в «реальном» (в кавычках, потому что, вообще говоря, можно же любой промежуток между тиками задать) времени.
!!! Вообще, на википедии сказано, что часы – компонент любой модели, и нужны они для синхронизации событий. Но Антоненко на лекциях говорил примерно то, что написано выше, в принципе, я думаю, что зависит все от целей моделирования и часы действительно не всегда нужны.

Пример:



Пример на картинке – моделирование клиентов банка или какого-то подобного заведения. Есть источник событий – клиент, далее событие анализируется и либо помещается в очередь для использования АТМ, либо направляется к консультанту.

3.5.Архитектура системы NPS. LXC контейнеры. Особенности моделирования глобальных компьютерных сетей.

LXC (англ. Linux Containers) — система виртуализации на уровне операционной системы, позволяющая изолировать ресурсы друг от друга, за счёт чего приложениям будет казаться, что они находятся в отдельной виртуальной машине. LXC базируется на технологиях namespaces и cgroups. LXC не использует виртуальные машины, а создает виртуальное окружение с собственным пространством процессов, сетевым стеком, … Все экземпляры LXC используют один экземпляр ядра операционной системы.

Используемые операции с сетевым пространством имен:

1. Создание сетевого пространства имен
2. Ассоциирование интерфейса с сетевым пространством имен
3. Конфигурирование интерфейса в сетевом пространстве имен

Вообще говоря, LXC – просто набор утилит, который позволяет создавать namespaces (пространства имен) и конфигурировать для них cgroups (контрольные группы) (задают ограничения/лимиты).

Рассмотрим подробнее, что есть namespaces и cgroups.

О namespaces:

На данный момент, в linux есть 6 типов namespace: IPC, Network, Mount, PID, User и UTS (hostname).

Каждый namespace предназначен для того, чтобы создать некоторую абстракцию над каким-либо системным ресурсом и создать для процессов, работающих в этом пространстве имен, иллюзию того, что они работают с выделенным только для них ресурсом.

Mount namespaces – изоляция файловой системы.

UTS – изолирование двух идентификаторов nodename и domainname, что позволяет каждому контейнеру иметь свои имя хоста и домена.

IPC – изоляция ресурсов для взаимодействия между процессами.

PID – изоляция pid-ов процессов. Т.е. в разных контейнерах могут быть процессы с одинаковыми pid-ами. Это, в свою очередь, позволяет контейнерам мигрировать между хостами, при этом сохраняя pid-ы запущенных процессов.

Network – изоляция сетевых ресурсов. У каждого контейнера появляется свое (виртуальное) сетевое устройство. Например, можно запустить кучу веб-серверов на одном сервере, каждый будет слушать 80-й порт.

User – изоляция user и group ID. Это пространство имен позволяет играться с привилегиями пользователя в системе, например, внутри пространства имен пользователь м.б. рутом, и обычным пользователем вне такого пространства имен.

О cgroups:

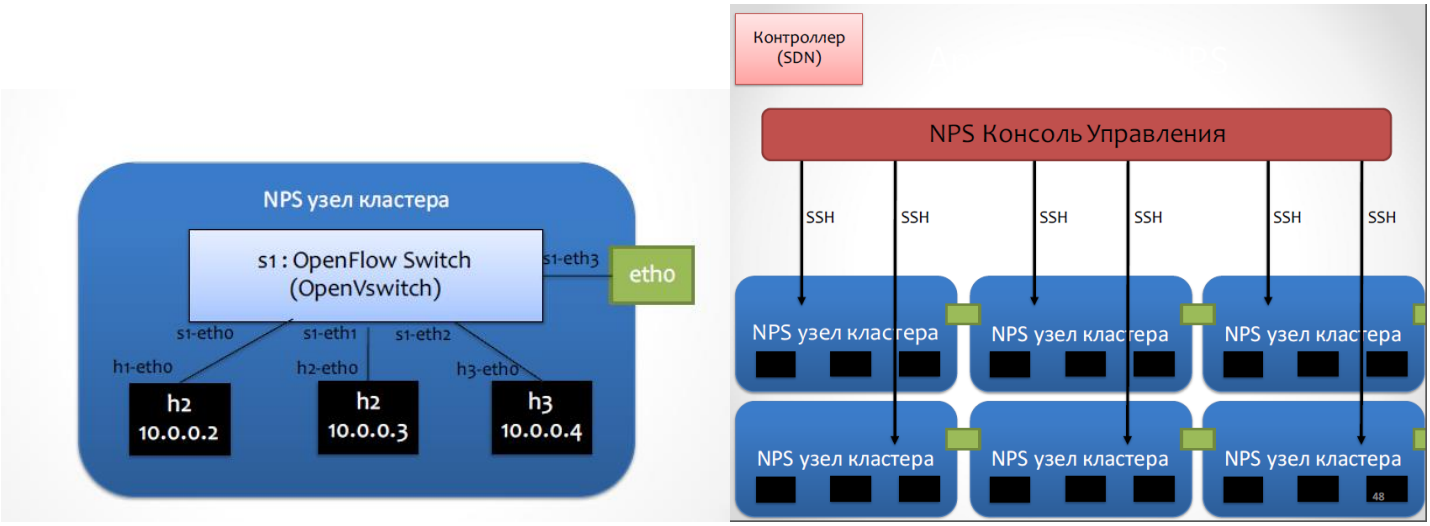
cgroups — механизм ядра Linux, который ограничивает и изолирует вычислительные ресурсы (процессорные, сетевые, ресурсы памяти, ресурсы ввода-вывода) для групп процессов. Механизм позволяет образовывать иерархические группы процессов с заданными ресурсными свойствами и обеспечивает программное управление ими.

Одна из целей механизма cgroups — предоставить единый программный интерфейс к целому спектру средств управления процессами, начиная с контроля единичного процесса (таких как, например, утилита nice) вплоть до полной виртуализации на уровне системы (как у OpenVZ, Linux-VServer, LXC). Механизм предоставляет следующие возможности:

- ограничение ресурсов (англ. resource limiting): использование памяти, в том числе виртуальной;
- приоритезацию: разным группам можно выделить разное количество процессорного ресурса и пропускной способности подсистемы ввода-вывода;
- учёт: подсчёт затрат тех либо иных ресурсов группой;
- управление: приостановку (freezing) групп, создание контрольных точек (checkpointing) и их перезагрузку.

NPS представляет из себя набор Docker контейнеров (NPS узлы кластера*, ещё их вроде называют доменами*), в каждом контейнере запускается свой Docker, который в рамках контейнера создаёт свои Docker контейнеры (h2, h3, …), являющиеся симулируемым узлами.

Таким образом, мы имеем docker внутри docker, однако следует понимать, что в рамках ос Linux вложенные контейнеры оформляются просто набор (не иерархический) контейнеров, где одни контейнеры могли унаследовать какие-либо свойства от других.
Благодаря такому дизайну NPS является хорошо масштабируемой системой.



NPS применяется для моделирования глобальных сетей.
Глобальная компьютерная сеть (ГКС) – сеть, состоящая не менее чем из 105 узлов.
Узел сети – точка в сети, имеющая уникальный сетевой адрес (может быть хост или сетевое устройство)
Хост – потребитель сетевого сервиса ГКС.
Свойства ГКС:
- сеть разделена на домены
- домены связаны между собой каналами «точка- точка»
Сетевое приложение – распределенная система, разные части которой обмениваются между собой данными.
Трафик сети – потоки данных между сетевыми приложениями.
Функционирование ГКС – процесс обработки и передачи сетевого трафика между узлами сети.
Имитационная модель (ИМ) ГКС – это математическая модель ГКС и ее реализация в вычислительной среде.
Точность ИМ ГКС (неформально) – совпадение моделируемых процессов обработки и передачи сетевого трафика между узлами в заданной топологии с процессами, происходящими в реальной сети.

Возможные проблемы:
Проблемы рассинхронизации между частями модели сети, расположенными на разных вычислителях.
!!! А поподробнее?

!!! Не отвечено на часть билета «особенности моделирования глобальных компьютерных сетей».

3.6.Основы контейнерной визуализации. Проект Docker: цели проекта, основные преимущества, базовые команды управления.

!!! В билете скорее всего подразумевается не «визуализации», а «виртуализации».

LXC (англ. Linux Containers) — система виртуализации на уровне операционной системы для запуска нескольких изолированных экземпляров операционной системы Linux на одном узле. LXC не использует виртуальные машины, а создает виртуальное окружение с собственным пространством процессов, сетевым стеком, …. Все экземпляры LXC используют один экземпляр ядра операционной системы.
Используемые операции с сетевым пространством имен:
1. Создание сетевого пространства имен
2. Ассоциирование интерфейса с сетевым пространством имен
3. Конфигурирование интерфейса в сетевом пространстве имен

Вообще говоря, LXC – просто набор утилит, который позволяет создавать namespaces (пространства имен) и конфигурировать для них cgroups (контрольные группы) (задают ограничения/лимиты).

Рассмотрим подробнее, что есть namespaces и cgroups.

О namespaces:

На данный момент, в linux есть 6 типов namespace: IPC, Network, Mount, PID, User и UTS (hostname).

Каждый namespace предназначен для того, чтобы создать некоторую абстракцию над каким-либо системным ресурсом и создать для процессов, работающих в этом пространстве имен, иллюзию того, что они работают с выделенным только для них ресурсом.

Mount namespaces – изоляция файловой системы.

UTS – изолирование двух идентификаторов nodename и domainname, что позволяет каждому контейнеру иметь свои имя хоста и домена.

IPC – изоляция ресурсов для взаимодействия между процессами.

PID – изоляция pid-ов процессов. Т.е. в разных контейнерах могут быть процессы с одинаковыми pid-ами. Это, в свою очередь, позволяет контейнерам мигрировать между хостами, при этом сохраняя pid-ы запущенных процессов.

Network – изоляция сетевых ресурсов. У каждого контейнера появляется свое (виртуальное) сетевое устройство. Например, можно запустить кучу веб-серверов на одном сервере, каждый будет слушать 80-й порт.

User – изоляция user и group ID. Это пространство имен позволяет играться с привилегиями пользователя в системе, например, внутри пространства имен пользователь м.б. рутом, и обычным пользователем вне такого пространства имен.

О cgroups:

cgroups — механизм ядра Linux, который ограничивает и изолирует вычислительные ресурсы (процессорные, сетевые, ресурсы памяти, ресурсы ввода-вывода) для групп процессов. Механизм позволяет образовывать иерархические группы процессов с заданными ресурсными свойствами и обеспечивает программное управление ими.

Одна из целей механизма cgroups — предоставить единый программный интерфейс к целому спектру средств управления процессами, начиная с контроля единичного процесса (таких как, например, утилита nice) вплоть до полной виртуализации на уровне системы (как у OpenVZ, Linux-VServer, LXC). Механизм предоставляет следующие возможности:

- ограничение ресурсов (англ. resource limiting): использование памяти, в том числе виртуальной;
- приоритезацию: разным группам можно выделить разное количество процессорного ресурса и пропускной способности подсистемы ввода-вывода;
- учёт: подсчёт затрат тех либо иных ресурсов группой;
- управление: приостановку (freezing) групп, создание контрольных точек (checkpointing) и их перезагрузку.

Docker — программное обеспечение для автоматизации развёртывания и управления приложениями в среде виртуализации на уровне операционной системы (раньше базировалось на LXC, теперь на libcontainer (cdjz разработка)). Позволяет «упаковать» приложение со всем его окружением и зависимостями в контейнер, который может быть перенесён на любой Linux-системе с поддержкой cgroups и namespaces в ядре, а также предоставляет среду по управлению контейнерами.

Программное обеспечение функционирует в среде Linux с ядром, поддерживающим cgroups и изоляцию пространств имён (namespaces); существуют сборки только для платформы x64.

Для экономии дискового пространства проект использует файловую систему Aufs с поддержкой технологии каскадно-объединённого монтирования: контейнеры используют образ базовой операционной системы, а изменения записываются в отдельную область (это позволяет мгновенно запустить сотни контейнеров). Также поддерживается размещение контейнеров в файловой системе Btrfs с включённым режимом копирования при записи.

Docker образ является dockerfile + некоторая сопутствующая информация (возможно зависимости, версия, …)

В состав программных средств входит демон — сервер контейнеров (запускается командой docker -d), клиентские средства, позволяющие из интерфейса командной строки управлять образами и контейнерами, а также API, позволяющий в стиле REST управлять контейнерами программно.

Демон обеспечивает полную изоляцию запускаемых на узле контейнеров на уровне файловой системы (у каждого контейнера собственная корневая файловая система), на уровне процессов (процессы имеют доступ только к собственной файловой системе контейнера, а ресурсы разделены средствами cgroups), на уровне сети (каждый контейнер имеет доступ только к привязанному к нему сетевому пространству имён и соответствующим виртуальным сетевым интерфейсам).

Набор клиентских средств позволяет запускать процессы в новых контейнерах (docker run), останавливать и запускать контейнеры (docker stop и docker start), приостанавливать и возобновлять процессы в контейнерах (docker pause и docker unpause). Серия команд позволяет осуществлять мониторинг запущенных процессов (docker ps по аналогии с ps в Unix-системах, docker top по аналогии с top и другие). Новые образы возможно создавать из специального сценарного файла (docker build, файл сценария носит название dockerfile), возможно записать все изменения, сделанные в контейнере в новый образ (docker commit). Все команды могут работать как с docker-демоном локальной системы, так и с любым сервером docker, доступным по сети. Кроме того, в интерфейсе командной строки встроены возможности по взаимодействию с публичным репозиторием Docker Hub, в котором размещены предварительно собранные образы контейнеров, например, команда docker search позволяет осуществить поиск образов среди размещённых в нём, образы можно скачивать в локальную систему (docker pull), возможно также отправить локально собранные образы в Docker Hub (docker push).
(build, pull, search, start, commit, run, kill, rm, pause, ps, top,)

Основные особенности:
Light-Weight
- Minimal overhead (cpu/io/network)
- Based on Linux containers
- Uses layered filesystem to save space (AUFS/LVM)
- Uses a copy-on-write filesystem to track changes
Portable
- Can run on any Linux system that supports LXC (today).
- 0.7 release includes support for RedHat/Fedora family.
- Raspberry pi support.
- Future plans to support other container tools (lmctfy, etc.)
- Possible future support for other operating systems (Solaris, OSX, Windows?)
Self-sufficient
- A Docker container contains everything it needs to run (их сравнивают с морскими контейнерами, когда груз (приложения) самых разнообразных характеристик и размеров (приложения с разными требованиями и зависимостями), упаковывают так, чтобы их можно было запускать универсальным образом)
- Minimal Base OS
- Libraries and frameworks
- Application code
- A docker container should be able to run anywhere that Docker can run.

3.7.Мониторинг сетевого трафика. Утилиты tcpdump, wireshark. Привести примеры фильтров tcpdump на L2, L3, L4, L7 уровнях сетевого трафика.

Мониторинг сетевого трафика может быть полезен, если вы столкнулись с новым типом сетевой атаки, пытаетесь понять, что не так с сетью, пишите приложение, использующее сеть и т.д., отлаживаете какой-нибудь blackbox, или ищете аномалии в сети, …
TcpDump и wireshark – утилиты, позволяющие сниффить трафик.
tcpdump – исключительно консольная, по дефолту не показывает содержимое пакетов, но зато есть почти везде, имеет очень широкий спектр фильтров.
wireshark – tcpdump с gui. Умеет парсить пакеты, адекватно выводя содержимое. Можно писать свои плагины, для поддержки новых протоколов. Также позволяет анализировать трафик (выделение т.н. conversations, статистика пакетов, рисует flow graph, есть функция follow tcp stream – показывает содержимое пакетов выбранного потока и т.д.). Поддерживает как фильтры захватываемого трафика, так и фильтрацию уже захваченного.
tcpdump может сохранить дамп трафика в файл, который потом можно лекто отобразить wireshark-ом.
При захвате пакетов бывает удобно использовать фильтры, чтобы отображались только пакеты, удовлетворяющие заданным условиям.
Пример фильтра L2: tcpdump -i eth0 arp, захватывает только arp-пакеты.
Пример фильтра L3: tcpdump -i eth0 ip[9] = 1, захватывает только icmp-пакеты.
(можно по аналогии захватывать только tcp пакеты, или только udp пакеты)

Пример фильтра L4: tcpdump -i eth0 tcp[13] & 0x08 != 0, захватывает только tcp-пакеты, у которых выставлен флаг psh.
tcpdump -i eth0 port http, захватывает только трафик, связанный с http портом (т.е. 80-й).
Пример фильтра L7: (это очень сомнительный пункт)
"GET " in hex=0x47455420 # tcpdump -i eth0 “tcp[32:4] = 0x47455420” (поиск по данным в пакете)
Для wireshark можно пользовать фильтр: ssl.handshake.cert_status_type == OCSP (хотя это скорее L6)

Условия в фильтрах можно соединять логическими связками and, or.
Можно ставить отрицание – not.

4. Методы верификации функционирования компьютерных сетей и управления качеством сервиса

4.1.Основные понятия и определения из области формальных методов. Задача формальной верификации на примере алгоритма Петерсона. Формальная модель, спецификация поведения, алгоритм верификации.

Формальные методы – такие методы спецификации, проектирования и анализа свойств систем, которые имеют строгое математическое обоснование
Верификация – проверка соответствия свойств системы заданной спецификации
Синтез – построения системы, свойства которой удовлетворяют заданной спецификации

Составные части метода формальной верификации:
Модель системы (язык описания модели) – математическое представление анализируемой системы
Язык спецификации – способ формального описания свойств системы
Метод верификации – алгоритм для проверки соответствия модели системы заданной спецификации

Типы исследуемых систем:
• Трансформирующие системы
- Преобразовывают входные данные в выходные
- Можно представить таблицей или формулой
• Реагирующие системы
- Поведение зависит от истории воздействий
- Можно представить конечным автоматом

Основные свойства математических моделей:
• Абстрактность - модель должна быть упрощением системы
• Адекватность - модель должна отражать свойства системы
• Компактность
- Размер и структурная сложность модели влияют на сложность решения задачи верификации
- Символьные методы – модель не перечисляет явным образом все состояния системы

Некоторые примеры формальных моделей:
• Таблицы
• Графовые представления
• Булевы формулы
• Двоичные решающие диаграммы
• Размеченные системы переходов
• Временные автоматы
• Сети Петри

Основные свойства языков спецификации:

• Выразительная мощность - язык должен охватывать исследуемые свойства
• Сложность решения задач
- Проверка эквивалентности формул
- Проверка непротиворечивости формул
- Формальная верификация
• Совместимость с моделью - сложность верификации напрямую зависит от согласованности модели с языком спецификации.

Некоторые примеры языков спецификации:
• Пропозициональная логика
• Логика предикатов первого порядка
• Логика высших порядков
• Логика Хоара
• Модальная логика
• Темпоральная логика
• Mu-исчисления

Методы формальной верификации:
• Неавтоматические - доказательство вручную
• Полуавтоматические - Theorem proving - полуразрешимость
• Автоматические - Model checking - комбинаторный взрыв

В качестве примера формальной верификации алгоритма рассмотрим формальную верификацию алгоритма Петерсона:
Алгоритм Петерсона – алгоритм для обработки критических секций без таких специальных средств, как семафоров.
*Исходные данные***:**
Глобальные переменные:

```
want1 = false;
want2 = false;
turn = 1;
```

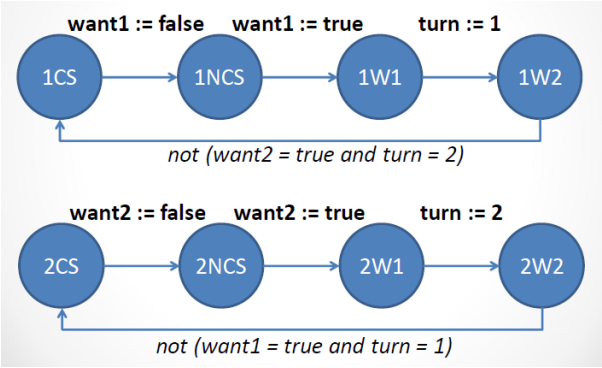
Процесс 1:

```
while (true)
{
    <noncritical section>
    want1 := true;
    turn := 1;
    while want2 and turn = 2 do { };
    <critical section>
    want1 := false;
}
```

Процесс 2:

```
while (true) {
    <noncritical section>
    want2 := true;
    turn := 2;
    while want1 and turn = 1 do { };
    <critical section>
    want2 := false;
}
```

В качестве формальной модели алгоритма будем использовать ДКА:



Спецификация свойств:
Данный алгоритм должен удовлетворять двум свойствам:
Свойство безопасности(safety)

Reachability analysis

Никогда не произойдёт ситуации, при которой оба процесса одновременно окажутся внутри критической секции.

$\mathbf{G}\neg(1CS \wedge 2CS)$

Свойство живучести(liveness)

Loop detection

Процесс, пожелавший попасть в критическую секцию, рано или поздно туда попадёт.

$\mathbf{G}(1NCS \rightarrow \mathbf{F}1CS) \wedge \mathbf{G}(2NCS \rightarrow \mathbf{F}2CS)$

Для проверки того, что система удовлетворяет указанным свойствам, будем использовать полный перебор диаграммы состояний системы из двух процессов. (!!! Далее не уверен, что правильно) Такая система будет состоять из 16 теоретически возможных состояний, часть из них будет недостижима, а остальные нас будут устраивать в соответствии со спецификацией свойств, что и будет являться доказательством того, что алгоритм обладает указанными свойствами.

(дальше материал за пределами билета)
***Формальная верификация vs тестирование**:*

Эффективность методов	*Формальная верификация*	*Тестирование*
Поиск ошибок	*средняя*	*высокая*
Доказательство корректности	*высокая*	*низкая*
Сложность использования	*высокая*	*низкая*

Типы ошибок	*Частая*	*Редкая*
Безобидная	*тестирование*	---
Критическая	*тестирование, формальная верификация*	*Формальная верификация*

Интернет очень резко развивался, и поэтому формальные методы стали менее популярны:
- разработка стала вестись больше методом проб и ошибок
- инженерная практика ценится выше теоретических изысканий

***Принцип построения компьютерных сетей**:*
- отказоустойчивость
- разнообразие сервисов передачи данных
- поддержка широкого диапазона сетей
- распределённое управление ресурсами
- рентабельность
- расширяемость
- учёт использования сетевых ресурсов

***Исследуемые свойства для сетевых протоколов**:*

- deadlocks – ожидание условий, которые никогда не выполнятся
- livelocks – выполнение инструкций протокола не приближает его к цели
- improper termination – протокол завершается, не достигнув цели
Число состояний бесконечно, поэтому метод их полного перебора не применим

***Исследуемые свойства Reachibility Analysis**:*
- Black Holes – тихое сбрасывание пакетов
- Forwarding Loops – зацикливание пакетов
- Достигнет ли пакет заданной точки?
- Ограничения на длину маршрутов
- Изоляция потоков
Нужен выразительный язык спецификации!
Как восстановить поведение сети?

***Верификация и синтез конфигураций сетевых экранов**:*
- проверка эквивалентности
- проверка избыточности
- синтез конфигурации, состоящей из минимального числа правил

***Формальные методы и ПКС**:*
• Новые протоколы управления сетевым оборудованием дали доступ к актуальным правилам обработки пакетов
• Централизация сети позволила собрать информацию о конфигурации сети в единой точке и отслеживать её изменение
• Появилась возможность адаптации наработок из области разработки ПО

***Data Plane Verification**:*
- Меньше предположений об управляющих приложениях на контроллере
- Проверяет выполнение политик на уровне проверки правил – метод нечувствителен к наведённым ошибкам внутри контроллера
- Естественная интерпретация понятия политики маршрутизации

<u>Примеры политик маршрутизации (требования к поведению компьютерной сети)</u>:
- Внешние потоки не достигают почтового сервера
- Исходящие потоки проходят через средство DPI
- Между каждой парой хостов в офисе есть маршрут
- Сети разных департаментов изолированы
- Все маршруты внутри сети короче шести хопов
- Пакеты не образуют циклов маршрутизации
- Хост A не может подлючиться к хосту B до тех пор, пока хост B не попытался подключиться к хосту A
- Пропускная способность соединения не меньше R, а задержка передачи данных – не превышает T

4.2.Задача формальной верификации конфигурации сети на примере средства VERMONT. Формальная модель, спецификация поведения, алгоритм верификации.

VERMÓNT - VERifying MONiTor
VERMONT проверяет правила обработки пакетов в таблицах коммутаторов на соответствие формальным спецификациям политик маршрутизации.
Для того, чтобы воспользоваться вермонтом нужно сделать две вещи:

1) Выразить требования к поведению сети с помощью специального языка спецификаций. Это одноразовая работа.
2) Предоставить топологию и файлы конфигурации коммутационных устройств. Это можно оптимизировать.

Как результат в течение всего функционирования сети имеем следующие плюсы:

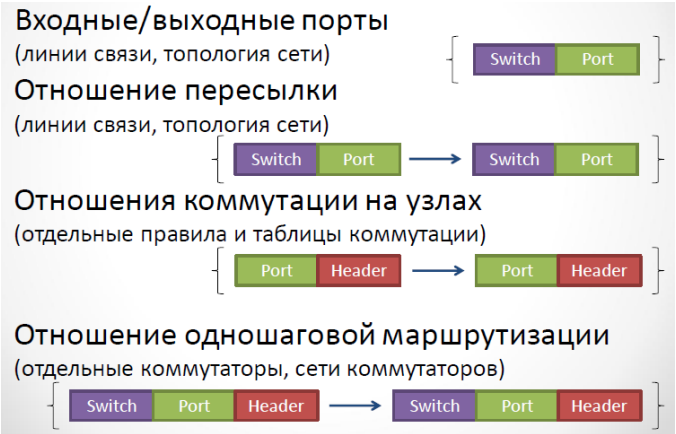
1) Гарантии работы сети в соответствии с ожиданиями
2) Выявление ошибочных конфигураций сети
3) Информация о причинах нарушения политики.

Описание модели.

Состояние пакета можно описать тремя наборами из {0, 1} (каждый набор имеет свою длину):

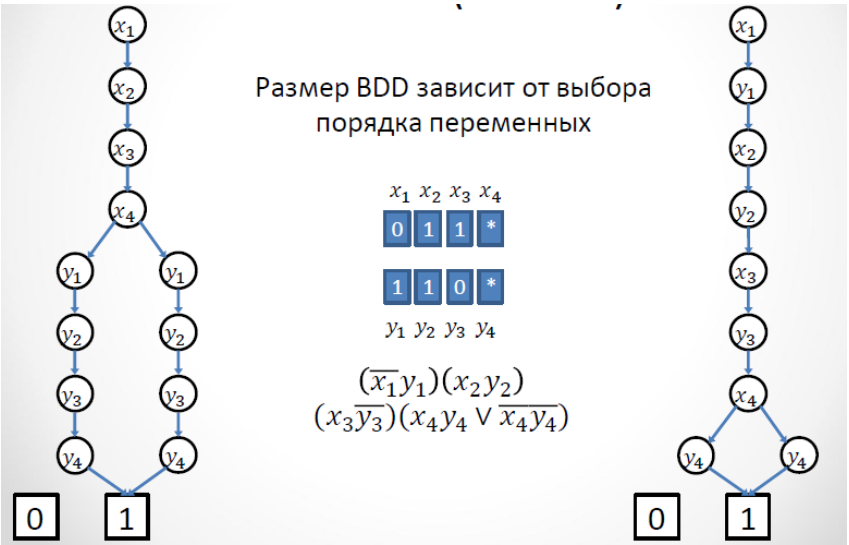
1) Имя коммутатора, на котором находится пакет
2) Номер порта
3) Заголовок пакета

Тогда сеть будет определяться набором отношений:



Затем по парам pattern-rewrite строим бинарную диаграмму решений:

(ROBDD – Reduced ordered binary decision diagram)



Для того, чтобы построить BDD по состоянию сети воспользуемся следующим алгоритмом:

Описание языка спецификаций:

Равенства	x[field] = const	x[field] = y[field]
Связки	$\varphi \wedge \varphi$	$\varphi \vee \varphi$ $\neg\varphi$
Кванторы	$\forall x.\, \varphi$	$\exists x.\, \varphi$
Замыкания	ψ^{+}	$\psi^{[i,j]}$

Правила вычисления замыканий

$$\psi^1(x,y) = \psi(x,y)$$
$$\psi^n(x,y) = \exists z:\ \psi^{n-1}(x,z) \wedge \psi(z,y)$$
$$\psi^{[i,j]}(x,y) = \psi^i(x,y) \vee \cdots \vee \psi^j(x,y)$$
$$\psi^{+}(x,y) = \psi^{[1,\infty]}(x,y)$$

Отношения, определяющие поведение сети

Входы	$In(x)$
Выходы	$Out(x)$
Шаг коммутации	$R(x,y)$
Отношение достижимости	$R^{+}(x,y)$

В итоге схема работы получается примерно следующая:

1) На основе описания топологии сети и свойств коммутаторов строятся модельные отношения в форме бинарных диаграмм решений
2) На языке спецификаций формулируются необходимые свойства
3) Верификатор на основе результатов первых двух пунктов определяет какие политики выполняются, а какие – нет, при каждой новой попытке загрузить новое правило в конфигурацию сети.

4.3.Варианты постановки задачи синтеза консистентного обновления конфигурации сети. Алгоритм трёхфазного обновления конфигурации сети с помощью тегирования.

Проблема консистентного обновления конфигурации:

• Конфигурация сети C задаёт привязку правил к коммутаторам и топологию сети

• Отношения In_C и R_C для конфигурации C определяет пути $path$ передачи пакетов

$s_0 \in In_C$; $(s_i, s_{i+1}) \in R_C$

$path = s_0, s_1, \ldots, s_i, s_{i+1}, \ldots$

• $Path(C)$–множество всех путей передачи пакетов для конфигурации C

• com–команда реконфигурации сети - добавление, удаление или модификация правила маршрутизации

• $com(C)$ - конфигурация, полученная в результате применения команды com к конфигурации C

• Если $\alpha = com_1, \ldots, com_k$, тогда $\alpha(C) = com_k(\ldots, com_1(C), \ldots)$

На множестве команд реконфигурации Com введено отношение частичного порядка $\prec$: если $com' \prec com''$, то com'' выполняется только после выполнения com'

• Пакет реконфигурации – множество команд реконфигурации, дополненное отношением частичного порядка $(Com, \prec)$.

Задача синтеза сетевых конфигураций:

Входные данные:

• Начальная конфигурация сети C_0

• Требования корректности и безопасности: постусловие Ψ, инвариант Φ.

Требования к решению:

• Такой пакет реконфигурации $(U, \prec)$, для любой линеаризации α_U которого выполнено:

1. $\alpha_U(C_0) \vDash \Psi$ (“$\vDash$” – означает, что в случаях, когда первое верно, то и второе тоже верно)
2. $\forall \alpha'\ (\alpha_U = \alpha'\alpha'' \Rightarrow \alpha'(C_0) \vDash \Phi)$

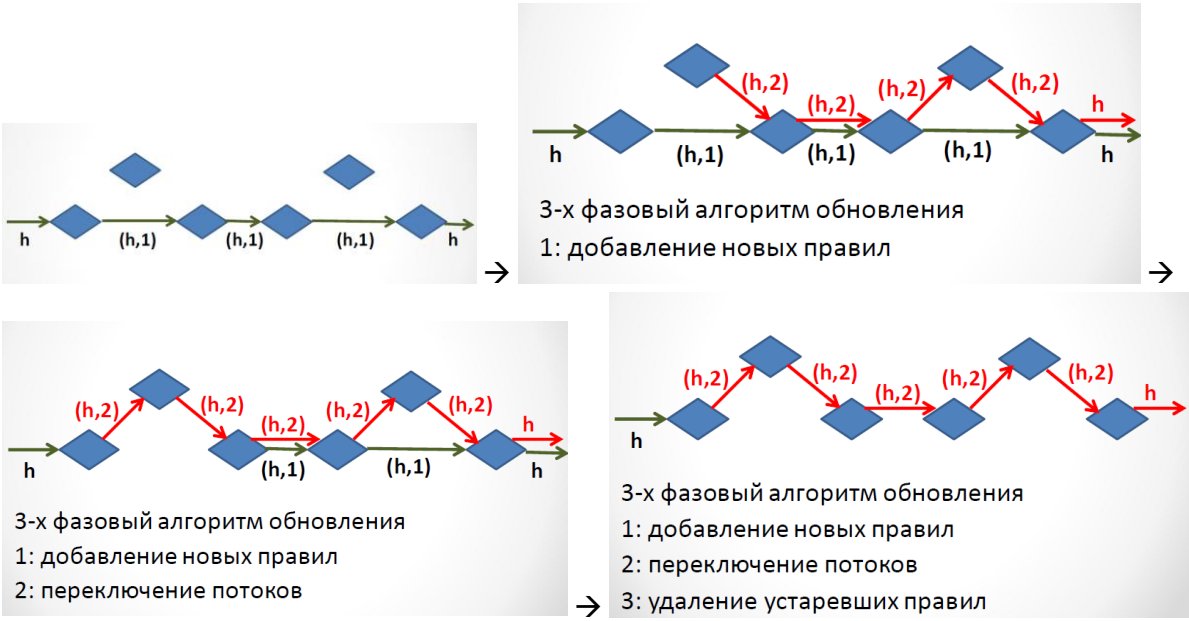
(т.е. при ∀ линеаризации конечный результат удовлетворяет постусловию, и на любой промежуточной стадии состояние удовлетворяет инварианту)

Задача синтеза сетевых конфигураций - построить такую конфигурацию C, которая удовлетворяла бы заданному постусловию Ф.

Возможные постановки:

1) Глобально-консистентное обновление сети
   Post-condition Ψ(X): X=C
   Invariant Φ(X): $\forall(In(s) \rightarrow ((Path(X,s) \subseteq Path(C))$ or $(Path(X,s) \subseteq Path(C_0))))$
   (для каждого свитча, s на котором может появиться пакет, либо множество путей в текущей топологии сети X из этого коммутатора вложено в множество всех путей в топологии сети C, либо в множество всех путей C_0) (~~!!! Возможен косяк с интерпретацией, но вроде норм~~)
2) Локально-консистентное обновление сети
   Post-condition Ψ(X): $Path(X) = Path(C_0) \backslash \{path_0\} \cup \{path_1\}$ (т.е. мы выкинули пути $path_0$ и добавили пути $path_1$)
   Invariant Φ(X): $(Path(C_0) \setminus path_0) \subseteq Path(X) \subseteq Path(C_0) \cup path_1$
3) Восстановление сети
   Постусловие Ψ(X): X=C
   Инвариант Φ(X): $Path(C_0) \cap Path(C) \subseteq Path(X) \subseteq Path(C_0) \cup Path(C)$
   Конфигурация C перешла в C_0 в результате удаления части правил
   Цель – восстановить C_0 из конфигурации C
4) Оптимизация таблицы потоков
   Постусловие Ψ(X): $Path(X) = Path(C)$ and $\forall Y$ $(Path(Y) = Path(C) \rightarrow f(X) \leq f(Y))$.
   Инвариант Φ(X): $Path(X) = Path(C)$

Обновление сети с помощью тегов:
Суть алгоритма в том, что у правил потока есть некоторый аналог временной метки. Далее – 3 этапа: добавить новые правила с инкрементированной временной меткой, переключить поток на работу с новыми правилами, удалить старые.

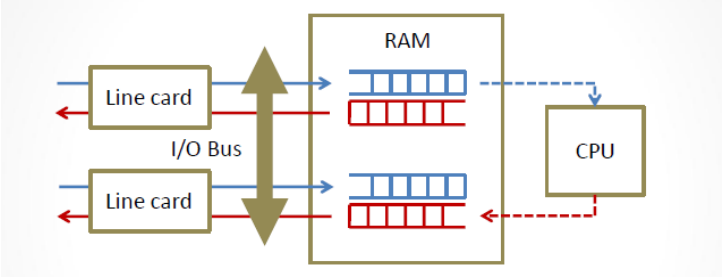


4.4.Классификация коммутационных устройств по поколениям. Варианты компоновки коммутаторов в зависимости от метода буферизации. Требования к производительности блоков коммутатора.

Классификация коммутаторов по поколениям:

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.

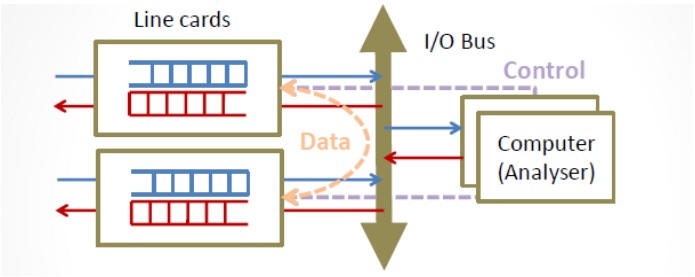
1. Интерфейсные Процессоры Сообщений – SISD компьютеры с несколькими сетевыми интерфейсами.



Большинство домашних Ethernet коммутаторов и маршрутизаторов

Bottleneck-ом может быть шина данных или процессор – в зависимости от типа трафика и производительности этих компонентов.

2. Распределённая MIMD архитектура с собственными контроллерами на интерфейсах

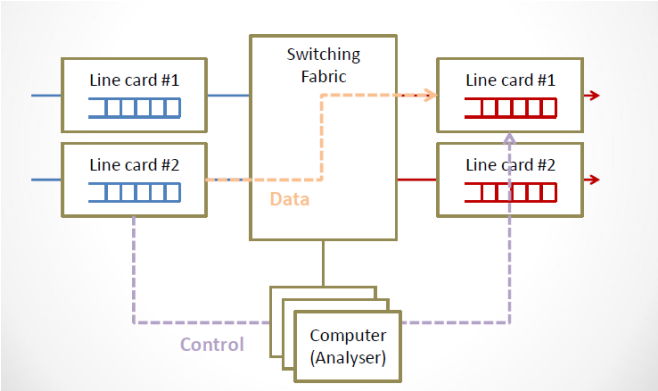


Умные сетевые интерфейсы с собственным контроллером и встроенной памятью. Данные пересылаются между картами напрямую, минуя оперативную память.

Слабое место – общая шина передачи данных.

Сюда относятся решения типо NFV, т.к. анализ пакетов занимает больше времени, чем их передача между интерфейсами.

3. Переход от архитектуры с единой шиной передачи данных к коммутационным матрицам



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) (K - количество одновременно передаваемых пакетов):

- Для шины передачи данных K=1 (количество одновременно передаваемых пакетов)
- Матрица переключателей $N \times N$: $1 \le k \le N$ (в зависимости от нагрузки)
- Матрица переключателей $N \times N^2$: k = N (вне зависимости от нагрузки)

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**).

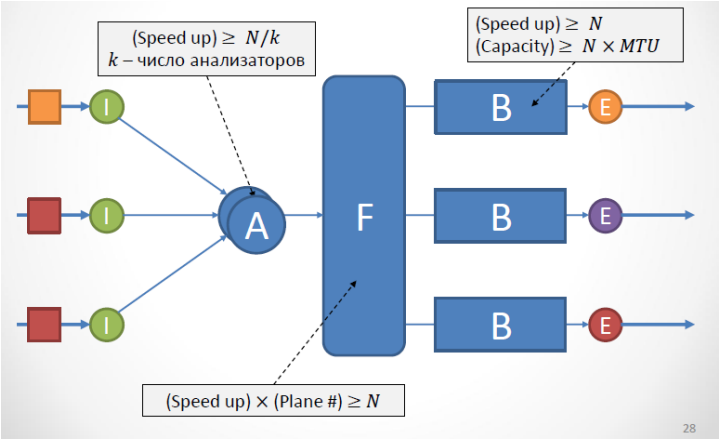
Если коммутатор не сбрасывает пакеты при поступлении трафика с приемлемым для него распределением, то говорят, что этот коммутатор удовлетворяет требованиям **full backplane**.

Буферизация на выходе

N – количество интерфейсов.

F – матрица коммутации

MTU – Maximum transmission unit



Производительность коммутатора:

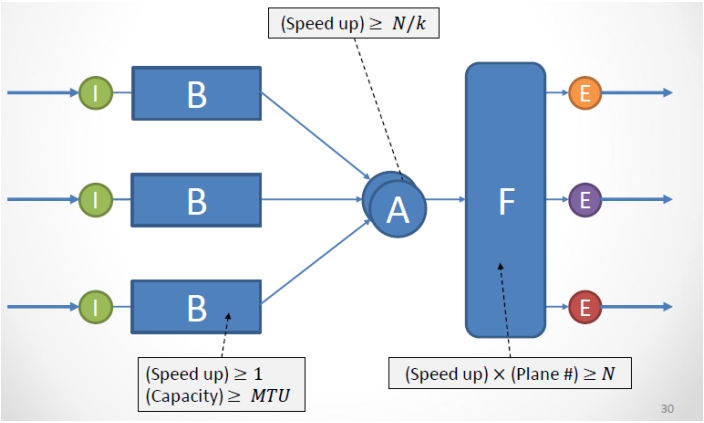
• Анализаторы должны работать с ускорением N/k, где –количество анализаторов

• Скорость работы матрицы 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

Недостаток такого подхода – нужны космические скорости доступа к памяти, чтобы справляться с нагрузкой современных коммутаторов.

Буферизация на входе

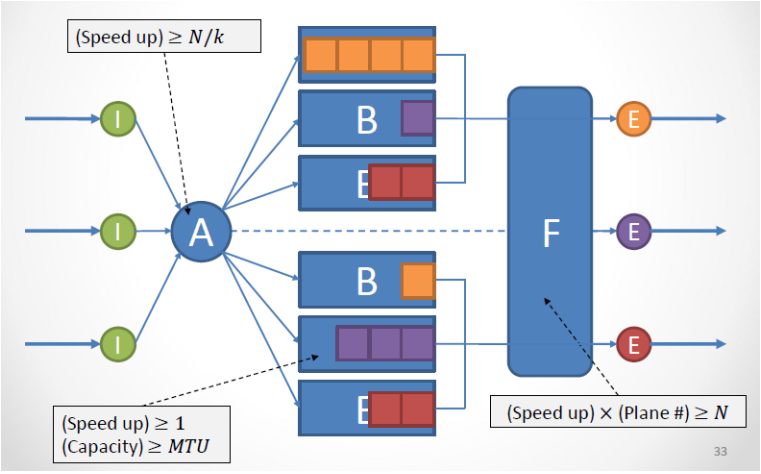


Нет необходимости в сверхбыстрой памяти.

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.

Виртуальная буферизация на выходе



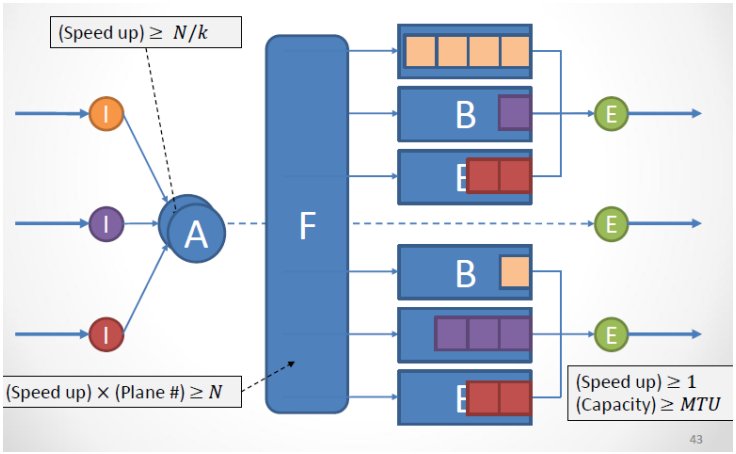
Проблема HOL blocking не возникает

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

Множественная буферизация на выходе



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.

(за пределами билета)

***Практика построения коммутационных устройств**:*

- Модели редко используются в чистом виде

- Производители пытаются искать компромиссы между стоимостью и утилизацией каналов под различной нагрузкой

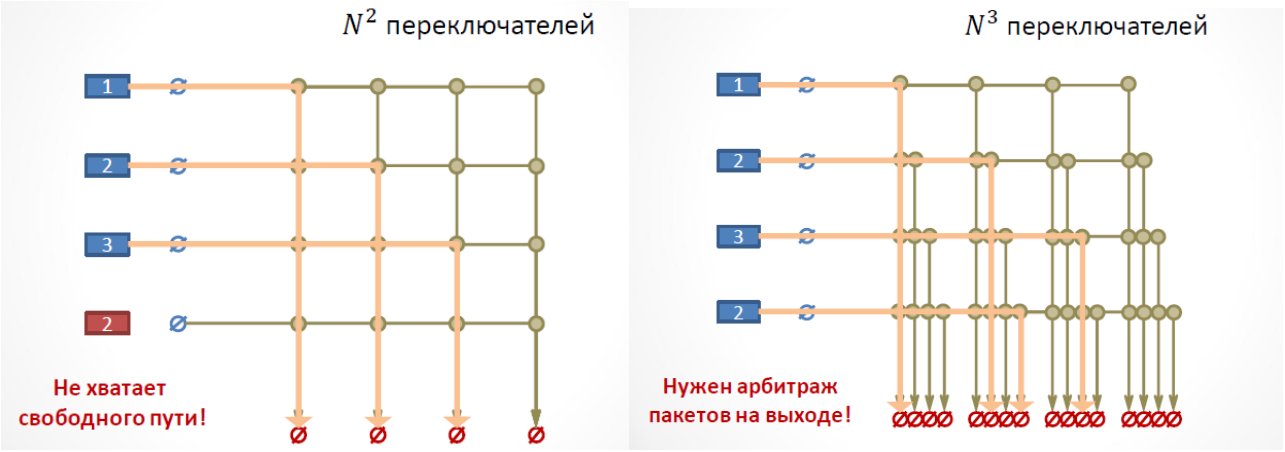
- На сегодняшний день наиболее распространены модели Combined Input-Output Queuing (CIOQ)

- Вместо полноценной коммутационной матрицы часто используются её упрощения – knockout switches или сети из переключателей

4.5.Устройство коммутационной матрицы. Принципы передачи пакетов через коммутационную матрицу при виртуальной буферизации на выходе. Неприменимость алгоритма поиска наибольшего паросочетания для выборки пакетов.

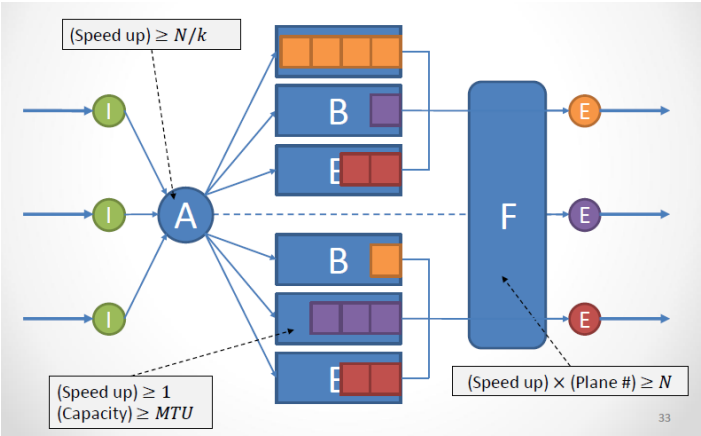
Если во втором поколении для передачи пакетов используется общая шина, то в третьем – коммутационная матрица. Коммутационная матрица способна передавать между N интерфейсами сразу несколько пакетов одновременно (имеет K transmission planes):

• Для шины передачи данных K=1
• Матрица переключателей $N \times N$: $1 \le k \le N$ (в зависимости от нагрузки)
• Матрица переключателей $N \times N^2$: k = N (вне зависимости от нагрузки)



Режимы коммутации
Δf - задержка коммутации пакета(FIFO)
Δs - задержка сериализации пакета
Δh - задержка обработки пакета(FIFO)
- Store-and-Forward (обработка начинается после получения всего пакета, работает в терминах пакетов): $\Delta f = \Delta s + \Delta h$
- Cut-Through [& Fragment free] (обработка начинается сразу после получения достаточного количества битов заголовка пакета, работает в терминах ячеек фиксированной длины): $\Delta f = d + \Delta h$ (d–время сериализации битов заголовка)

Виртуальная буферизация на выходе

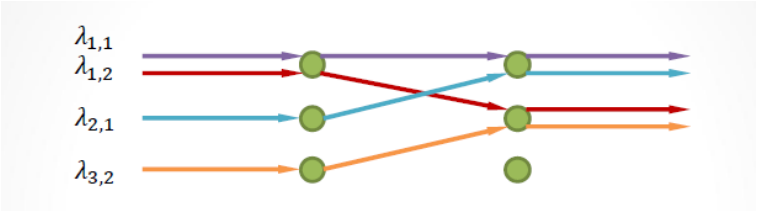


Проблема HOL blocking не возникает
Появляется N^2 очередей пакетов
Коммутационная матрица с N^2входами не требуется, но необходим алгоритм быстрого арбитража для выбора нужной очереди на каждом из интерфейсов
Сложные динамические алгоритмы арбитража затруднительно реализовать в аппаратуре

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 одну ячейку из каждого входа и поместить в одну ячейку на каждый свой выход
Надо найти такой алгоритм арбитража коммутационной матрицы, при котором:
- производительность (требования full backplane выполнялись бы без ускорения матрицы)
- справедливость (никогда не происходило бы ухудшение потоков трафика)

Гипотеза:

- подходящий алгоритм должен передавать как можно большее количество ячеек на каждом такте работы матрицы – задача поиска наибольшего паросочетания. Гипотеза **не верна** поскольку алгоритм не отличается ни производительностью, ни справедливостью.



Пусть в ситуациях, когда существует несколько наибольших паросочетаний алгоритм выбирает одно из них с равной вероятностью.

Пусть пропускная способность каждой из линий связи равна 1, скорость каждого из потоков $\lambda_{1,1}=\lambda_{1,2}=\lambda_{2,1}=\lambda_{2,2}=\frac{1}{2}-\delta$.

Если 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.

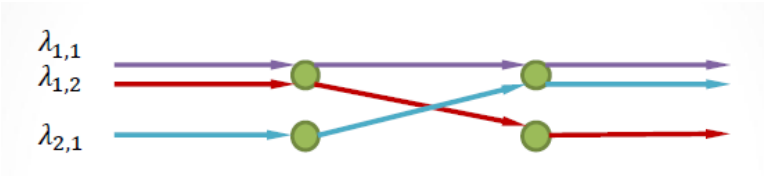
По формуле полной вероятности наибольшая скорость поступления данных со входа 1 составляет: $\frac{2}{3}(\frac{1}{2}-\delta)^2+(1-(\frac{1}{2}-\delta)^2)=1-\frac{1}{3}(\frac{1}{2}-\delta)^2$

Скорость поступления данных на вход 1 составляет $1-2\delta$

Если скорость выборки данных будет меньше скорости их поступления, то требования full backplane нарушены:

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

Условие выполнено при $\delta < 0.0358$.



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 ячейка не была выбрана

Алгоритм арбитража выбирает паросочетания таким образом, чтобы максимизировать сумму весов выбранных им очередей

Недостатки:

- Высокая сложность реализации
- Алгоритм не учитывает задержку

4.6.Механизмы управления качеством сервиса на уровне коммутатора. Ограничение интенсивности потоков по алгоритму token bucket. Дисциплины очередизации: сброс и выборка пакетов.

Качество сервиса – качество обслуживания потоков на коммутационных устройствах: пропускная способность, задержка при передаче пакетов, уровень потерь, …

Качество обслуживания потоков напрямую связано с количеством выделенных на его обслуживание ресурсов: доли пропускной способности канала, объёма буферной памяти и процессорного времени.

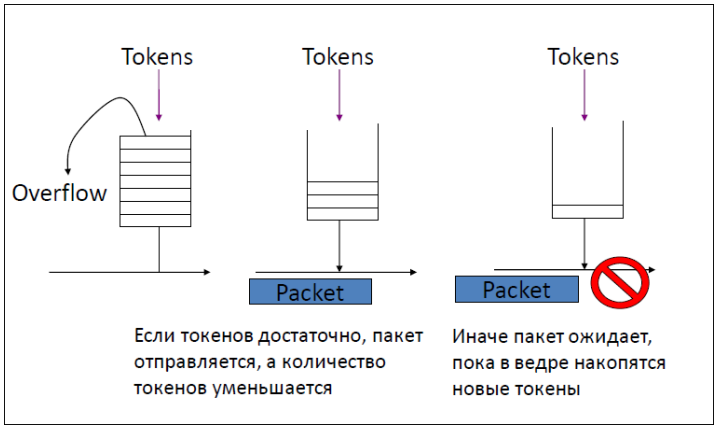
Управление качеством достигается с помощью дисциплин очередизации потоков и шейпинга трафика.

Ограничение потоков:

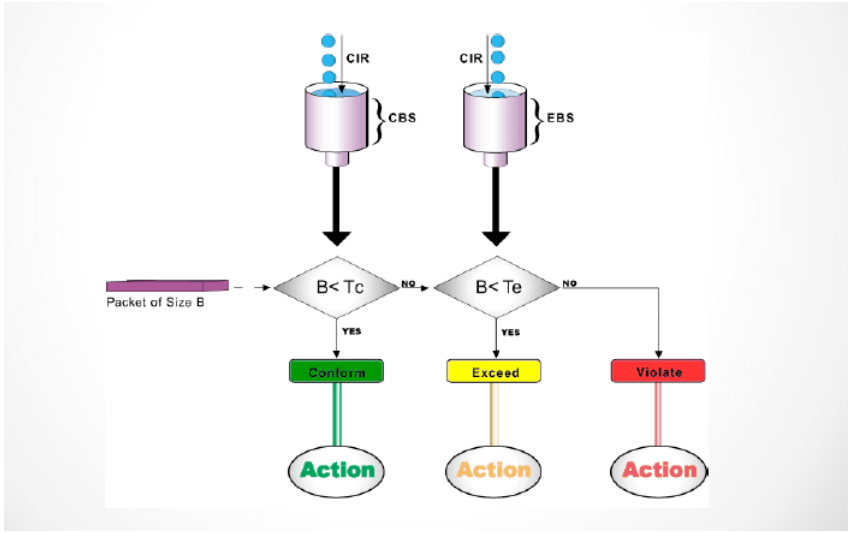
Профиль потока определяется интенсивностью пересылки его данных (скорость, всплески, …)

Shaping & Policing трафика позволяет придавать профилям потоков необходимую форму с помощью встроенных средств коммутатора.
Если интенсивность потока превышает спецификации профиля, то shaping задерживает обработку пакетов, policing сбрасывает пакеты.

Алгоритм Token Bucket
Есть ведро, в него помещается некоторое количество токенов (очевидно определяется объемом ведра). Новые токены поступают в ведро с заданной скоростью.
Если ведро заполняется, то поступающие токены игнорируются
При отправке пакета размером N байтов из ведра извлекается N токенов.
Если в ведре недостаточно токенов, то отправка пакета откладывается.



Использование нескольких алгоритмов текущего ведра



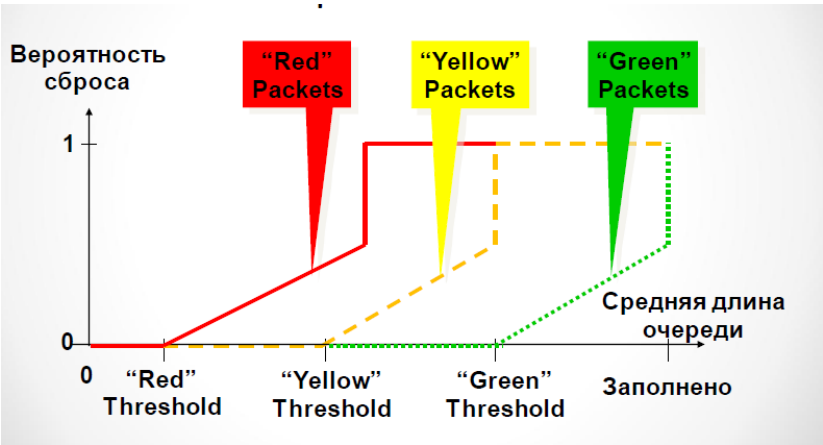
Суть: для передачи пакета занимаются маркеры из первого ведра, если их не хватает, то тогда занимаются из второго ведра, но пакет передаётся с пониженным приоритетом, если и во втором ведре маркеров не хватает, то пакет не отправляется.
Такая стратегия позволяет выделять типы траффика, применяя к разным типам траффика разные политики.

Дисциплины очередизации:
Дисциплина обслуживания очередей включает в себя планирование выборки пакетов из очереди и политику сброса пакетов.
Выбор дисциплины очередизации определяет распределение пропускной способности канала между потоками – какой пакет отправить следующим и распределение буферной памяти – какой пакет будет сброшен, если памяти на всех не хватает.
Правила обслуживания очередей значительно влияют на задержку.
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.

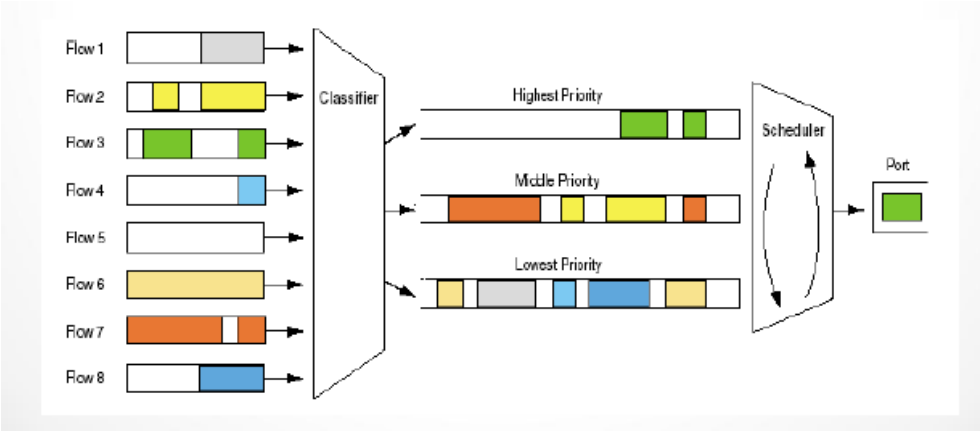
Распространённые дисциплины очередизации:
• First-In-First-Out (FIFO)
• Priority Queuing (PQ)
• Fair Queuing (FQ)
• Weighted Fair Queuing (WFQ)
• Weighted Round Robin (WRR)
• Shared Round Robin (SRR)
• Deficit Weighted Round Robin (DWRR)

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 к увеличению сквозной задержки
– Наступает эффект синхронизации TCP трафика—очередь то переполняется, то простаивает
• Потоки с большими всплесками ущемляются сильнее других
Проблему блокировки потоков можно обойти двумя способами:

1) Если приходит новый пакет, а очередь полна, то сбрасываем рандомный пакет из уже имеющихся в очереди (random drop)
2) Сбрасывать пакеты заранее, избегая переполнения (random early detection)
    a. Вычисляет среднюю загруженность очереди x
    b. Если $x < x_{min}$ - пакеты не сбрасываются
    c. Если $x > x_{max}$ - сбрасывается поступающий пакет
    d. Иначе, пакет сбрасывается с вероятностью, линейно зависящей от близости к пороговым значениям



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



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

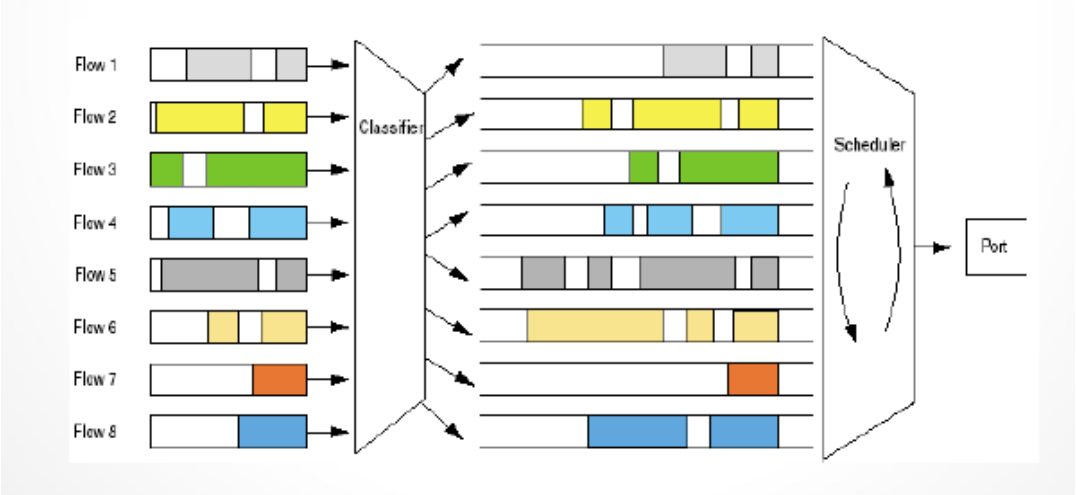
• Низкоприоритетный трафик может испытывать существенные задержки

• 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



• Потоки изолированы друг от друга

• Реализация на базе аппаратных очередей затруднительна

• Может быть использован совместно с другими дисциплинами обслуживания

• Дифференциация между потоками передачи данных не производится, потоки с разными требованиями к пропускной способности не поддерживаются

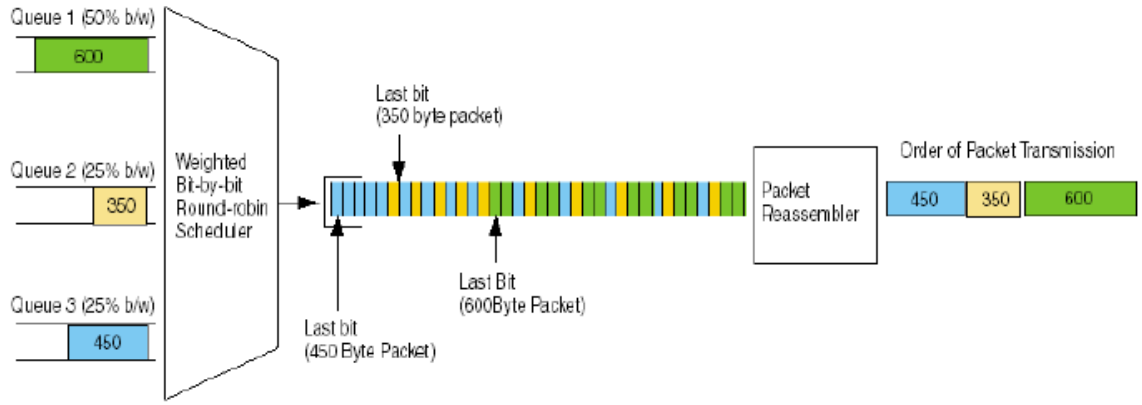
• Нет механизмов для передачи real-time трафика

• Потоки с пакетами больших размеров получают преимущество

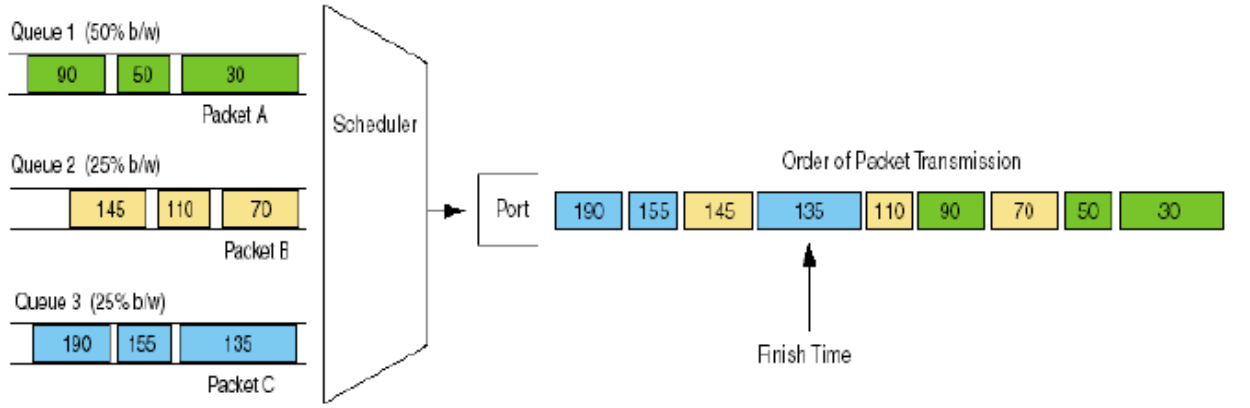
Взвешенная справедливая очередизация (WFQ):

• Использует веса для поддержки потоков с разными требованиями к пропускной способности

• Учитывает размер пакетов при планировании выборки пакетов из очереди

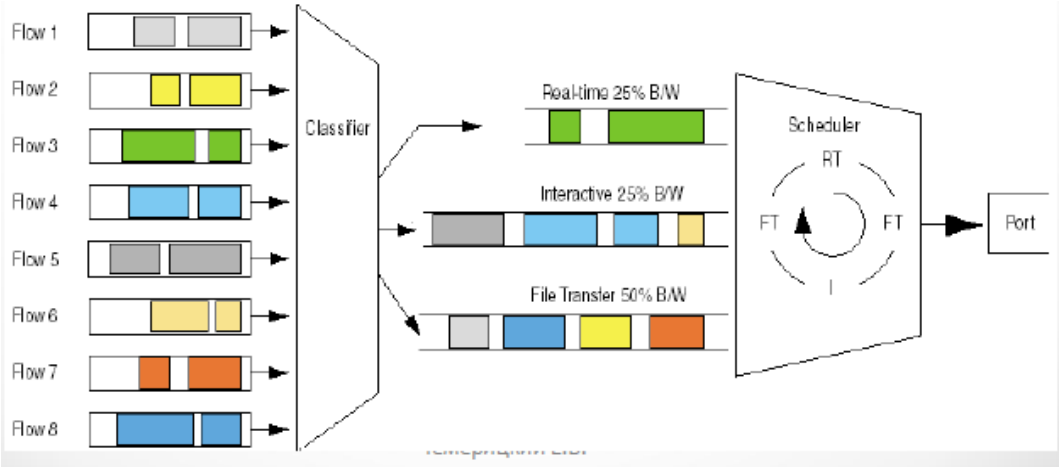


• Абстракция fluid модели приближается путём вычисления времени завершения передачи пакета
• Планировщик выбирает пакеты с наименьшим расчётным временем завершения передачи



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



• 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 – deficit weighted round robin
– Выборка осуществляется не по пакетам, а по байтам – решается проблема пакетов разного размера

Часто применяется комбинирование нескольких дисциплин очередизации.

4.7.Принципы функционирования протокола резервирования ресурсов RSVP. Модели управления качеством сервиса в сети Интернет: IntServ и DiffServ.

Мультимедийный трафик в сети:
Как оградить TCP трафик от мультимедийных данных, передающихся через UDP? Как обеспечить качество соединения?
Гарантированный уровень качества можно обеспечить лишь с помощью резервирования ресурсов – закрепления части ресурсов сети за конкретным потоком данных.

Модель Интегрированных Сервисов (IntServ):
Основная идея – прокладывание маршрутов с заданным качеством, за счёт предварительного резервирования ресурсов на оборудовании.
Модель лишь расширяет архитектуру Интернета, и способна работать по принципу best-effort. Модель особенно эффективна при многоадресной передаче данных. Допускаются накладные расходы на предварительное прокладывание маршрута. Модель или гарантирует соединение нужного качества, или отказывается предоставить какое-либо соединение.

Основные компоненты модели IntServ:
• Классификатор (classifier) - разделение пакетов на классы обслуживания
• Планировщик (scheduler) - обеспечение выполнения требований QoS
• Контроль доступа (admission control) - оценка возможности добавления потоков
• Протокол резервирования ресурсов - резервирование ресурсов вдоль маршрута

Применяется распределение по очередям на входящих и исходящих интерфейсах коммутатора, использование policing & shaping для формирования нужного профиля потоков, установка надлежащих дисциплин сброса пакетов при их постановке в очереди и выборки пакетов из очередей, настройка алгоритмов планирования коммутационной матрицы.

Протокол резервирования ресурсов RSVP:
Поддерживаемые типы трафика:
• Регулярный (best-effort) - передача файлов, просмотр почты, …
• Чувствительный к скорости (rate-sensitive) - потоковое вещание аудио и видео
• Чувствительный к задержке (delay-sensitive) - Voice Over IP, online игры
• Поток данных - последовательность пакетов, имеющих общего отправителя, единый набор получателей, и требующих одинакового уровня качества сервиса
• Спецификация потока (flowspec) - определяет поток данных, тип трафика, требования к качеству соединения
• Спецификация фильтра (filterspec) - определяет правила доступа пакетов к зарезервированным ресурсам
• RSVP работает на уровне сессий - множество потоков с одинаковым набором получателей

Стили резервирования FilterSpec:
• Заданным фильтром (Fixed Filter)
– Ресурсы выделяются отправителю для индивидуального пользования
– Обычный видеостриминг
• Шаблонное (WildcardedFilter)
– Ресурсы разделяются между группой отправителей по заданному предикату
– Во время аудио конференции одновременная передача данных маловероятна
• Разделяемое явно (Shared-Explicit)
– Ресурсы разделяются между группой отправителей
– Члены группы могут изменяться со временем

Пример работы RSVP:

1) Приложение-отправитель посылает RSVP path message получателям:
    • Одному (unicast)
    • Группе (multicast)

   Каждый маршрутизатор запоминает, откуда пришло сообщение
2) Приложение-получатель посылает RSVP resv message отправителю.
   Маршрутизаторы передают resv сообщения в точности повторяя путь path сообщений
   (а не по общим правилам маршрутизации пакетов)
3) При получении resv message каждый маршрутизатор:
    • Оценивает возможность резервирования
    • Устанавливает нужные настройки для планировщика и классификатора
    • Формирует новое сообщение resv и направляет его на следующий маршрутизатор
4) Если ресурсы удалось выделить на каждом из узлов, то маршрут с нужным качеством сервиса проложен - можно начинать передачу данных.
5) При поступлении новых запросов на резервирование маршрутизаторы пытаются их суммировать. Часто удаётся построить дерево передачи и увеличить эффективность передачи.

Особенности RSVP:
• RSVP – сигнальный протокол: обеспечивает лишь резервирование вдоль маршрута: первичный выбор маршрута – забота протоколов маршрутизации
• Протокол экономно расходует ресурсы при частичном совпадении маршрутов
• Инициирует резервирование получатель
• Одни и те же ресурсы могут использоваться сразу несколькими отправителями
• Резервирование поддерживается отправителем и получателем (Soft State)

Недостатки RSVP:
• Необходимо вести маршрутизацию на уровне потоков – для каждого потока нужно хранить его состояние (soft state)
• Выделение ресурсов в индивидуальном порядке слишком требовательно к железу
• Плохая масштабируемость решения (!!! В чём именно проблема?)
• Внутренняя фрагментация ресурсов при статическом резервировании – низкая утилизация оборудования, а чем выше утилизация, тем лучше отношение производительности сети к стоимости инфраструктуры.

Модель Дифференцированных Сервисов (DiffServ)
• Каждый маршрутизатор имеет несколько предопределённых классов обслуживания
• Пограничные маршрутизаторы определяют класс потока, маркируют его пакеты dscp меткамии, проводят traffic conditioning – используют инструменты policing & shaping для установки нужного профиля трафика
• На внутренних маршрутизаторах пакеты с более высоким приоритетом получают большую долю ресурсов
В заголовке IPv4 пакета есть поля PRE – приоритет пакета и ToS (type of service) – тип сервиса, который требует пакет. Возможные типы сервиса: minimize delay, maximize throughput, maximize reliability и minimize cost.
Также есть поле для указания Differentiated Services Codepoint: reserved for standardization, reserved for local use или open for local use, may be standardized later.

В IPv6 есть классы траффика, работающие аналогично DSCP полю IPv4.

Стандартные классы обслуживания DiffServ:
• Default Forwarding (DF)
– Обычно обслуживается по best-effort
• Expedient Forwarding (EF)
– Идёт через очередь с высшим приоритетом
– Маленькие delay, jitter & loss
• Assured Forwarding (AF)
– Идёт через менее приоритетную очередь
– Охватывает несколько классов с разной политикой сброса при заполнении очереди

Преимущества DiffServ:
• Отстутствие внутренней фрагментации
• Высокая степень утилизации оборудования
• Простота реализации в аппаратуре
• Хорошая масштабируемость

Недостатки DiffServ:
• Неразличает метрик качества
• Не предоставляет гарантий качества
• Распределяет ресурсы пропорционально

4.8. Связь задачи управления качеством с задачей распределения сетевых ресурсов. Управление качеством с помощью планирования маршрутов и многопоточной маршрутизации.

Качество Сервиса в сети Интернет:
Если управление качеством сервиса не поддерживается, то сетевые соединения обслуживаются по принципу best effort, т.е. предоставляется такой сервис, который можно предоставить на текущий момент, никаких гарантий не предоставляется.

(наверно, здесь везде не хватает лаконичных ответов)
***Методы управления качеством сервиса в сети**:*
• Выполнение потребностей приложений
– Какие параметры соединения можно задавать?
– Даётся ли гарантия выполнения требований?
• Сложность реализации и эксплуатации
– Какими дополнительными функциями должно обладать оборудование?
– Какая поддержка требуется от хостов?
• Накладные расходы на использование
– На сколько эффективна работа сети?
– Сколько ресурсов будет задействовано?
– Сколько будет простаивать?

Проблема эффективного распределения ресурсов
• Проблема обеспечения качества связана с проблемой распределения сетевых ресурсов между потоками данных. Проблему распределения ресурсов можно формализовать как задачу оптимизации
– Чем больше ресурсов вовлечено в обслуживание потока, тем выше качество его соединения
– Чем большее количество ресурсов позволяет задействовать модель распределения, тем выше эффективность сети, и тем больший уровень утилизации достигается
Почему высокий уровень утилизации – хорошо?
• Приложения не всегда способны задействовать все ресурсы сети

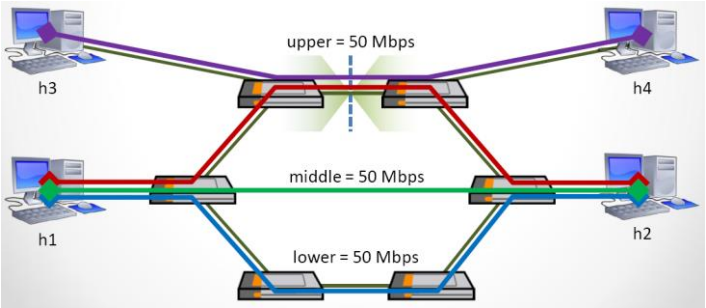
– Spanning Tree Protocol – в топологии Fat Tree
– Backbone networks – запас производительности
– Разница в характеристиках аппаратуры
• Чем выше уровень утилизации, тем лучше отношение производительности сети к стоимости сетевой инфраструктуры
– Уменьшение затрат провайдеров

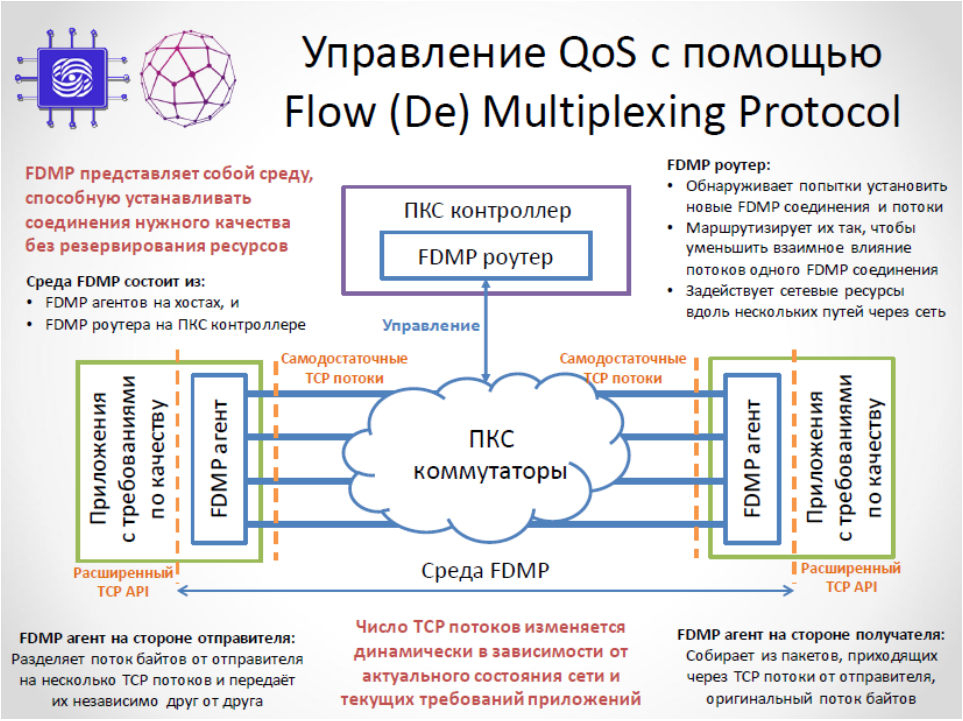
QoS Routing:
• Прокладывание индивидуальных маршрутов для потоков с учётом их требований качества сервиса
– Маршруты между двумя точками в сети могут быть построены разными способами, если их требования качества для потоков не совпадают
• Появляется возможность вовлечь ресурсы, расположенные вне основных маршрутов передачи данных
• Метод совместим с обеими моделями управления ресурсами (DiffServ и IntServ)

Проблемы QoS Routing:
• Необходим гранулярный контроль за коммутационными устройствами
- Управление на уровне потоков
• Необходимо централизованное управление
- Алгоритмы маршрутизации не будут сходиться
• Нужна возможность получения требований качества от приложений
• Построение маршрутов – тяжёлая задача
Проблема наилучшего распределения ресурсов
Управлять выполнением требований качества и уровнем утилизации сети можно за счёт особого алгоритма построения маршрутов

Критерии оптимизации:
• Увеличение устойчивости к пиковым нагрузкам – равномерное распределение
• Обслуживание потоков наименьшим количеством устройств – можно выключить часть оборудования
• Максимизация надёжности сети

Многопоточная оптимизация:
Имеется ввиду прокладка потоков при помощи FDMP (Flow De Multiplexing Protocol), который позволяет перепрокладывать потоки, в зависимости от ситуации.



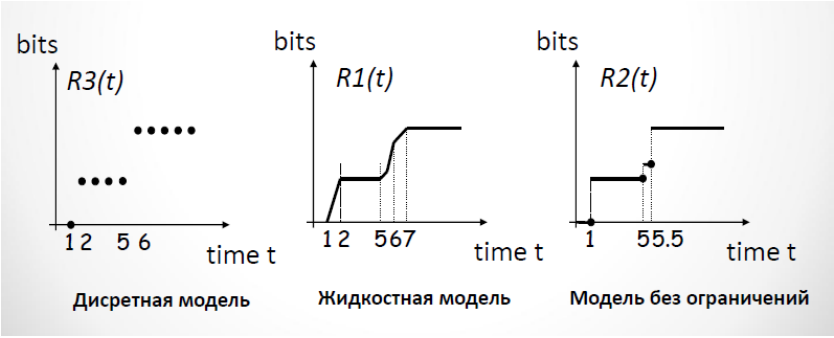


4.9.Основные понятия сетевого исчисления. Функции поступления и отправки, задержка и отставание, кривые нагрузки и сервиса. Оценки отставания, задержки и интенсивности выходного потока.

Queueing Network – предсказывает отставание и задержку для системы из обработчиков (вероятностная модель не годится для анализа систем реального времени)
Scheduling Theory – оценка худшего случая (worst-case analysis)
Network Calculus – расширение теории расписаний до системы из обработчиков и буферов.

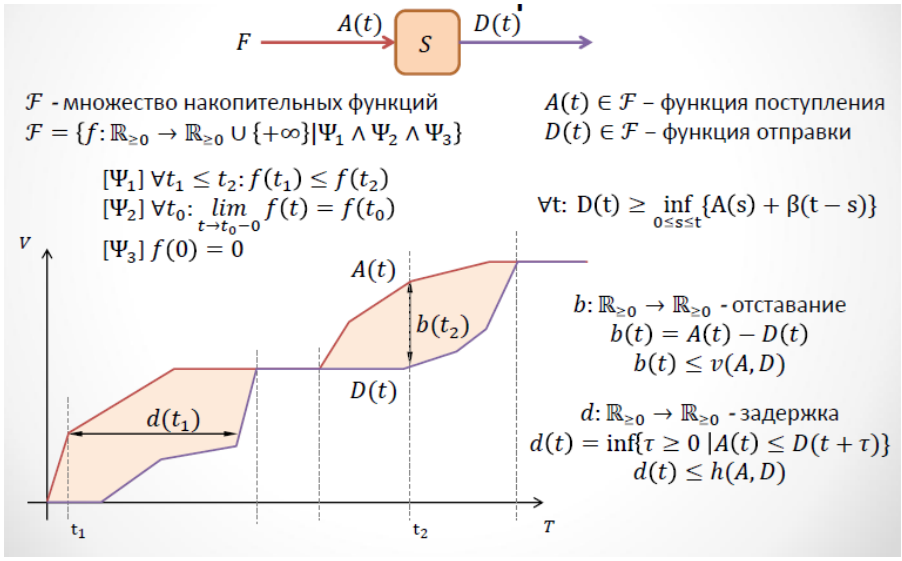
Компоненты модели:
• Представление в виде сети обработчиков
– Обработчик–логически целостный компонент, выполняющий преобразования потоков данных
• Описание нагрузки–множества потоков данных, поступающих в систему
– Маршрут передачи данных
– Характеристики интенсивности
• Описание обработчиков
– Характеристики производительности
– Принципы мультиплексирования

Накопительные функции:
• Зависимость количества переданных данных от времени



Основные определения:

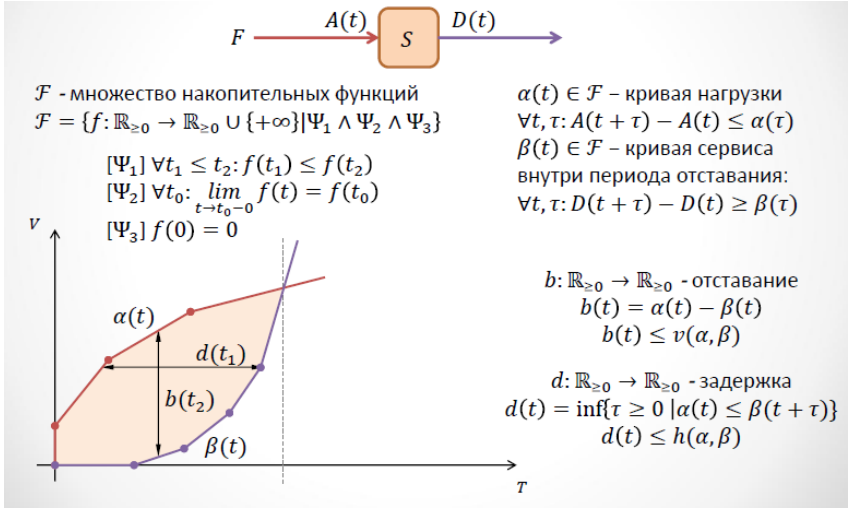
• Функция прибытия $A(t)$ описывает зависимость суммарного количества данных, поступивших на обработчик, от времени
• Функция отправки $D(t)$ – зависимость количества переданных данных потока от времени
• Каждый обработчик может быть описан перечислением пар вида $\langle A_t, D_t \rangle$
• Отставание (backlog) $b(t)$ – выражает количество данных, находящихся внутри обработчика
• Период отставания – промежуток в течение которого функция отставания строго положительна
• Задержка (delay) $d(t)$ – время прохождения через обработчик той порции данных, которая поступила на него в момент времени t



Кривые нагрузки и сервиса:
• Вид функций поступления и отправки в большинстве случаев неизвестен
- Необходима аппроксимация
• Кривая нагрузки $\alpha(t)$—накопительная функция, для которой выполнено условие $\forall t,\tau: A(t+\tau) - A(t) \leq \alpha(\tau)$
- за время τ поступает не больше $\alpha(\tau)$ данных
• Кривая сервиса $\beta(t)$ – накопительная функция, для которой внутри каждого периода отставания выполняется условие $\forall t,\tau: D(t+\tau) - D(t) \geq \beta(\tau)$
- за время τ обслуживается не меньше $\beta(\tau)$ данных

Теорема 1 (оценка отставания). Пусть поток данных с кривой нагрузки $\alpha \in \mathcal{F}$, обслуживается обработчиком с кривой сервиса β.
Тогда значение отставание обработчика не превышает вертикального отклонения между кривыми прибытия α и сервиса β: $\forall t \in \mathbb{R}: b(t) \leq v(\alpha,\beta)$, $v(\alpha,\beta) = \sup_{t \geq s \geq 0} \{\alpha(s) - \beta(s)\}$

Теорема 2 (оценка задержки). Пусть поток с кривой нагрузки $\alpha \in \mathcal{F}$, обслуживается обработчиком с кривой сервиса $\beta \in \mathcal{F}$ по дисциплине FIFO. Тогда $\forall t \in \mathbb{R}: d(t) \leq h(\alpha,\beta)$, $h(\alpha,\beta) = \sup_t \inf(\tau \geq 0 | \alpha(t) \leq \beta(t+\tau))$
Если же дисциплина обслуживания неизвестна, то для задержки $d(t)$ справедлива оценка: $(t) \leq \inf\{\tau \geq 0 \,|\, \alpha(\tau) \leq \beta(\tau)\}$



Теорема 3 (оценка выходного потока).
Если поток данных с кривой нагрузки $\alpha \in \mathcal{F}$, поступает на элемент с кривой сервиса $\beta \in \mathcal{F}$, то выходной поток, полученный в результате его обработки, ограничен кривой $\alpha' \in \mathcal{F}$: $\alpha'(s)=\sup\{\tau \geq 0 \mid \{\alpha(s+\tau)-\beta(\tau)\}\}$

Если заменить операции суммы и умножения на операции поточечной минимизации и суммы, то свёрткой будет называться: $(f \otimes g)(t) \stackrel{\text{def}}{=} \inf_{0 \leq s \leq t} \{f(s) + g(t-s)\}$
В min-plus алгебре операторы свёртки (convolution)$\otimes$и обратной свёрткой (deconvolution) $\oslash$ функций f и g имеют следующий вид: $(f \otimes g)(t) \stackrel{\text{def}}{=} \inf_{0 \leq s \leq t} \{f(s) + g(t-s)\}$
$(w \oslash g)(t) \stackrel{\text{def}}{=} \sup_{t \geq u \geq 0} \{wt + u - g(u)\}$

Кривые нагрузки и сервиса:

$$\forall t: \forall s \leq t: A(t) - A(s) \leq \alpha(t-s) \Leftrightarrow A = A \otimes \alpha$$
$$\forall t: D(t) \geq \inf_{0 \leq s \leq t} \{A(s) + \beta(t-s)\} \Leftrightarrow D \geq A \otimes \beta$$

Оценки для оставания и задержки:

$$b(t) \leq v(\alpha, \beta) = \sup_{s \geq 0} \{\alpha(s) - \beta(s)\} = (\alpha \oslash \beta)(0)$$
$$d(t) \leq h(\alpha, \beta) = \sup_{t} \{\inf\{\tau \geq 0 \mid \alpha(t) \leq \beta(t+\tau)\}\}$$

Оценка выходного потока:

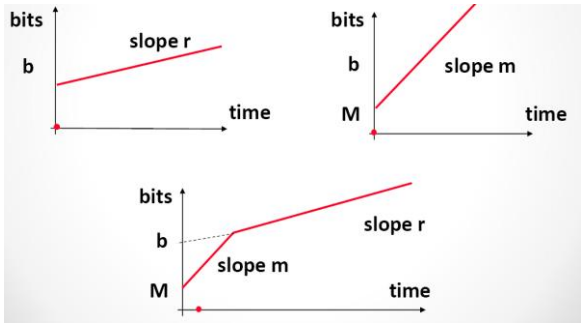
$$\alpha'(t) = \sup_{u \geq 0} \{\alpha(t+u) - \beta(u)\} = (\alpha \oslash \beta)(t)$$

Если обработчики S_1 и S_2 с кривыми сервиса β_1 и β_2 образуют ***тандем***, то их систему описывает кривая сервиса $\beta = \beta_1 \otimes \beta_2$

Доказательство:

$$D_2 \geq A_2 \otimes \beta_2 = D_1 \otimes \beta_2 \geq (A_1 \otimes \beta_1) \otimes \beta_2 = A_1 \otimes (\beta_1 \otimes \beta_2) = A_1 \otimes \beta$$

Комбинирование нескольких шейперов:

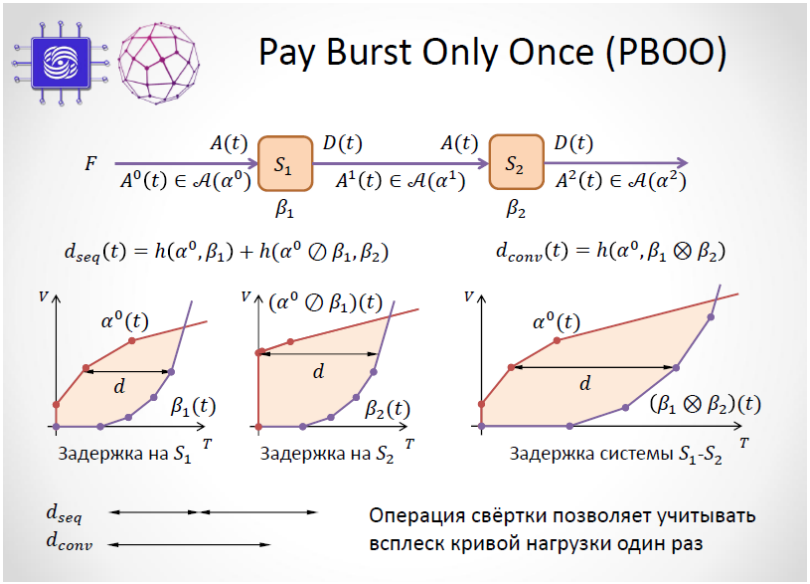


4.10. Построение оценки для сквозной задержки передачи данных через сеть с помощью алгоритма SFA. Причины низкой точности алгоритма SFA при его применении в предположениях модели DiffServ.

Если обработчики S_1 и S_2 с кривыми сервиса β_1 и β_2 образуют ***тандем***, то их систему описывает кривая сервиса $\beta = \beta_1 \otimes \beta_2$

Доказательство:

$$D_2 \geq A_2 \otimes \beta_2 = D_1 \otimes \beta_2 \geq (A_1 \otimes \beta_1) \otimes \beta_2 = A_1 \otimes (\beta_1 \otimes \beta_2) = A_1 \otimes \beta$$

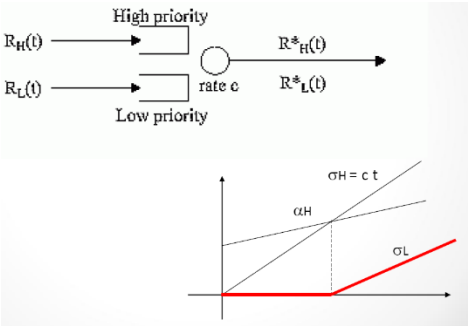


Модель ПКС:

• Сеть представляется графом обработчиков
• Известны кривые нагрузки для каждого из потоков и кривые сервиса для каждого из обработчиков
• Дисциплины мультиплексирования потоков неизвестны
• Нагрузки не превышают возможности обработчиков
• Граф обработчиков не содержит циклов

Пусть через один обработчик проходит несколько потоков. Зафиксируем один поток, он будет интересующим, остальные – пересекающие потоки.

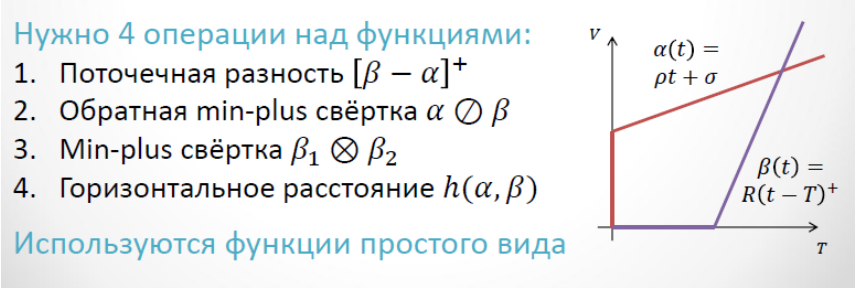
Остаточной кривой сервиса назовем кривую сервиса, которую предоставляет обработчик интересующему потоку, после того, как обработает все пересекающие потоки.

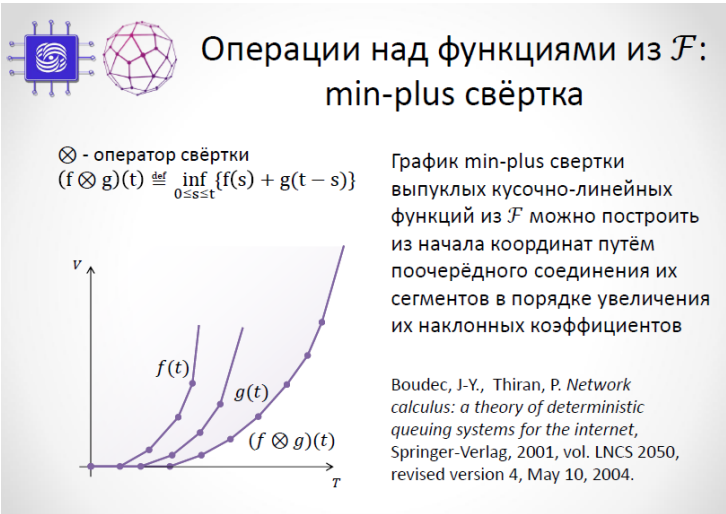


Separated Flow Analysis:

Для заданного целевого потока:

1. Построить остаточные кривые сервиса обработчиков
2. Вычислить общую кривую сервиса системы
3. Задержка потока равна максимальному расстоянию между кривой сервиса системы и его кривой нагрузки





В предположениях модели DiffServ SFA дает низкую точность из-за переучета мультиплексирования. Т.к. мы считаем, что интересующий нас поток является наименее приоритетным, то часто возникает такая ситуация, что на одном и том же обработчике мы, при вычислениях для одного потока, считаем, что он всем уступает, а при вычислениях для другого потока (но на том же обработчике), что теперь уже он всем уступает (хотя до этого уступали ему) (такое легко случается в, например, ромбовидной топологии). SFA подразумевает отсутствие циклов в потоках (что вполне адекватное требование).

Существует механизм, который позволяет избежать подобной ситуации, по крайней мере, в случае FIFO-мультиплексирования. А ещё есть теория расчётов через линейное программирование.