Тестирование QoS на экспериментальном стенде

НИУ ИТМО

Докладчик: Дмитрий Чугреев (chugreevd@gmail.com)

Соавторы: Д. В. Власов, В.А. Грудинин, А.Б. Каирканов, О.Л. Садов, Л.Н. Сомс, В.Б. Титов, С.Э. Хоружников, А.Е. Шевель, А.Е. Шкребец

Quality of Service

- Основные характеристики качества обслуживания
  - полоса пропускания (Bandwidth)
  - задержка (Delay)
  - джиттер (Jitter)
  - потеря пакетов (Packet loss)

  разные приложения – разные группы требований

- Задача: обеспечение **сетевыми устройствами** заданного приложением QoS согласно SLA (Service-level agreement)

Концепции Quality of Service

- **выделение ресурсов под приложения заранее**
- технология IntServ
  - RSVP: резервирование ресурсов под QoS-потоки

проблема: в чистом виде неприменимо, если слишком много потоков

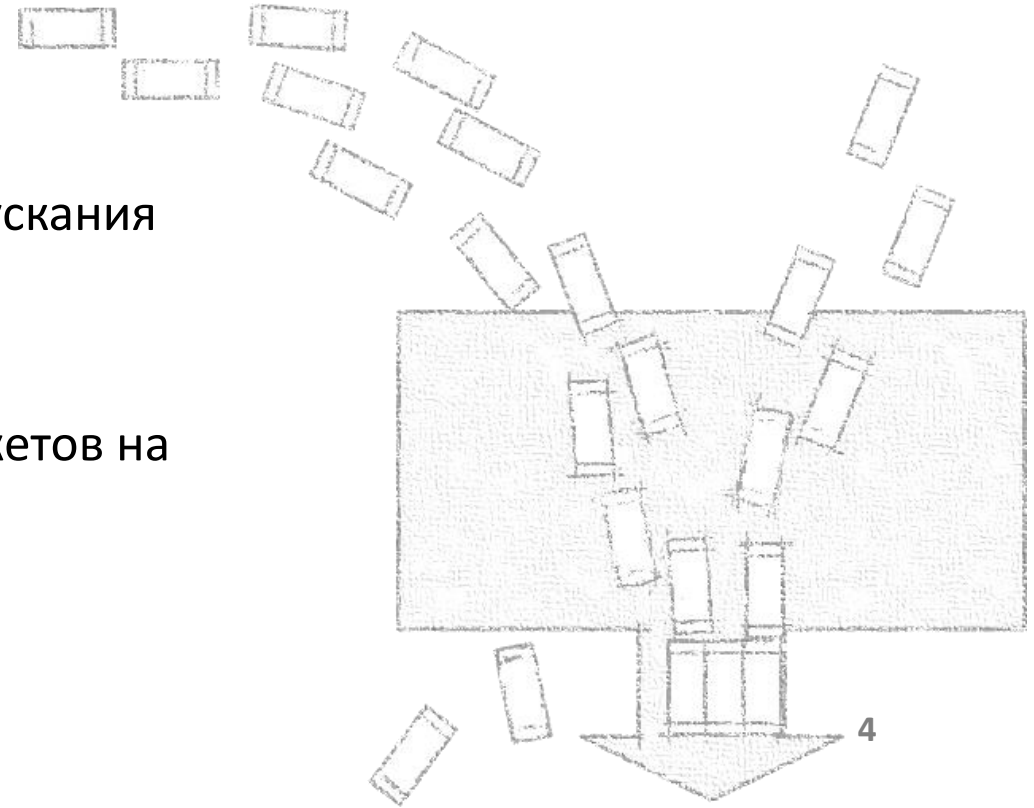
- **на основе анализа специальных полей в пакетах, характеризующих класс сервиса**
- технология DiffServ
  - поля DSCP + ECN в IP-пакетах
  - *могут анализироваться и другие поля*
    - Канальный уровень: VLAN PCP
    - Транспортный уровень: TCP-порты
    - …

проблема: нет сквозной (end-to-end) поддержки QoS

QoS и OpenFlow

- Основные плюсы
  - **централизованное управление** в рамках сети, управляемой одним контроллером
    - *теоретически, можно реализовать любой механизм*
  - **гибкость** в написании приложений, управляющих QoS

- Методы
  - ограничение полосы пропускания
    - OpenFlow 1.3 Meters
    - HP QoS Extensions
  - управление очередями пакетов на каждом интерфейсе

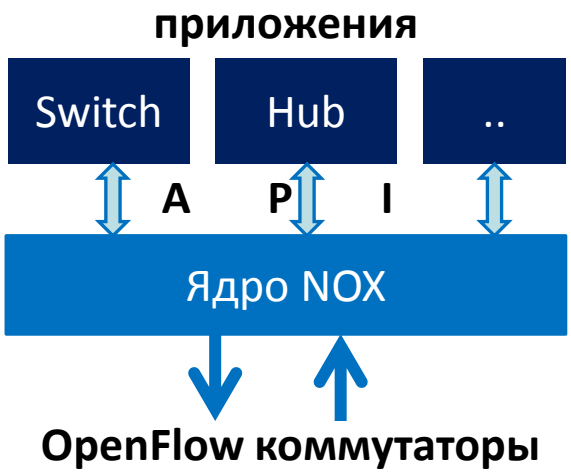


Очереди пакетов и OpenFlow

- Реализация
  - коммутаторы поддерживают выходные очереди пакетов на интерфейсах
  - приложения (applications) поверх контроллера определяют, как нужно привязывать трафик к ним
- Механизм 1
  - настраивается стандартными способами, не связанными с OpenFlow:
    - количество очередей и пропускная способность
    - привязка пакетов к очередям на основе полей (DSCP, PCP)
  - контроллер: **при задании нового потока** DSCP или PCP указывается в Flow Action
  - коммутатор: направляет трафик потока в эту очередь
  - пример: HP ProCurve 3500yl
- Механизм 2 (Slicing)
  - контроллер: настройка очередей на интерфейсах коммутатора
    - специальный протокол для конфигурации: OF-CONFIG
  - контроллер: при задании нового потока (flow) номер очереди указывается в Flow Action
  - коммутатор: направляет трафик потока в эту очередь
  - пример: CPqD OpenFlow SoftSwitch

Контроллеры nox и nox-Classic

- OpenFlow - контроллеры NOX
  - одни из наиболее популярных
  - простая и расширяемая архитектура
  - скорость (написаны на C/C++)



- NOX-Classic
  - 2008-2011
  - Приложения м.б. написаны на C++ и Python
  - основа для многих других проектов
    - *Компания CPqD поддерживает свою ветку, добавляя функционал новых версий OpenFlow*

- NOX: новое поколение
  - с 2012 – *полностью переписан*
  - только C++
  - заявлена лучшая производительность, более удобный API
  - **далеко не стабилен**

- Собраны пакеты (бинарные и с исходными кодами) для НауЛинукс

Коммутатор HP ProCurve 3500yl

- Современный «intelligent-edge» коммутатор
  - 24-port Gigabit Ethernet
  - широкие возможности по управлению QoS
  - поддержка OpenFlow

- QoS в коммутаторах HP
  - до 8 выходных очередей приоритетов с возможностью задания минимальных пропускных полос
  - гибкая схема привязки трафика к очереди на основе L2-L4 полей

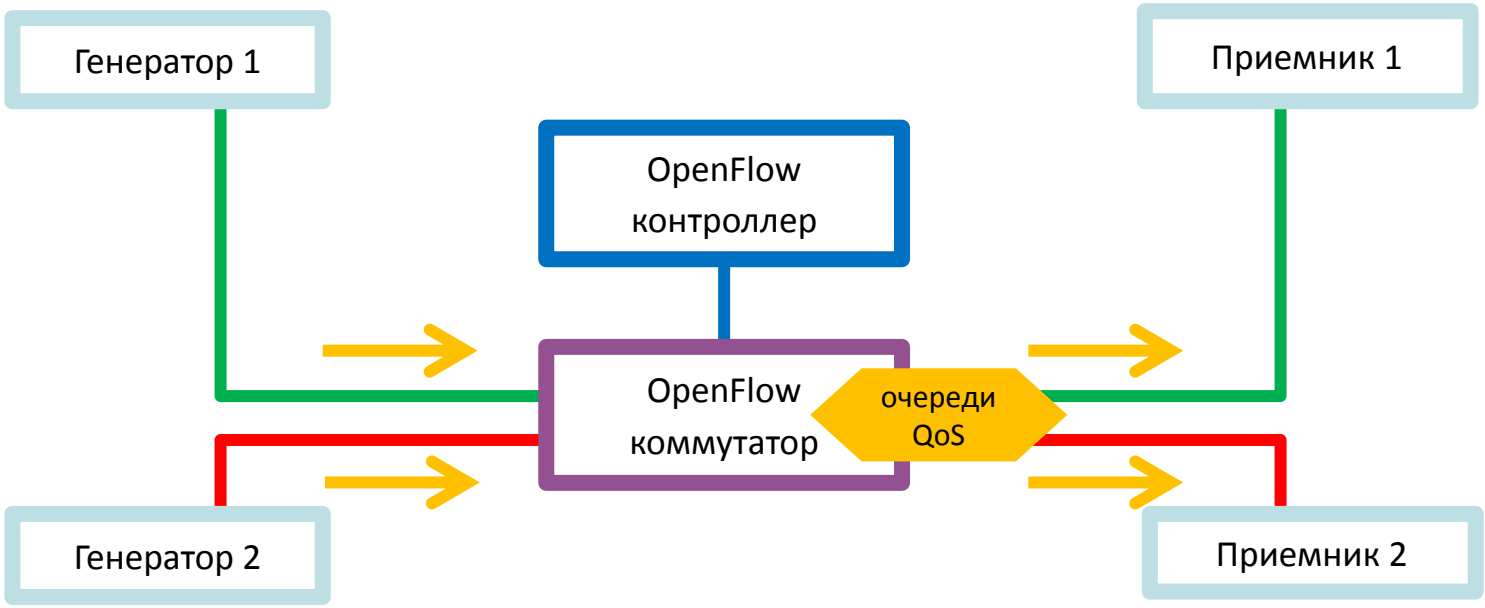
- OpenFlow в коммутаторах HP
  - только OpenFlow 1.0
  - нет поддержки добавления очередей через OF-CONFIG
    - *т.е. доступен только механизм 1*
  - высокоскоростная аппаратная обработка некоторых типов потоков
    - *остальные обрабатываются программно и медленно*

Метод тестирования

- Операционная система: НауЛинукс 6.3 (совместим с RedHat)
- Параллельный запуск
  - «Полезный трафик» - запросы к СХД по iSCSI и
  - «Нагрузочный трафик» генератора Iperf

  попадают в разные очереди QoS
- Задаем разные полосы пропускания очередей

  и замеряем реальные скорости передачи данных



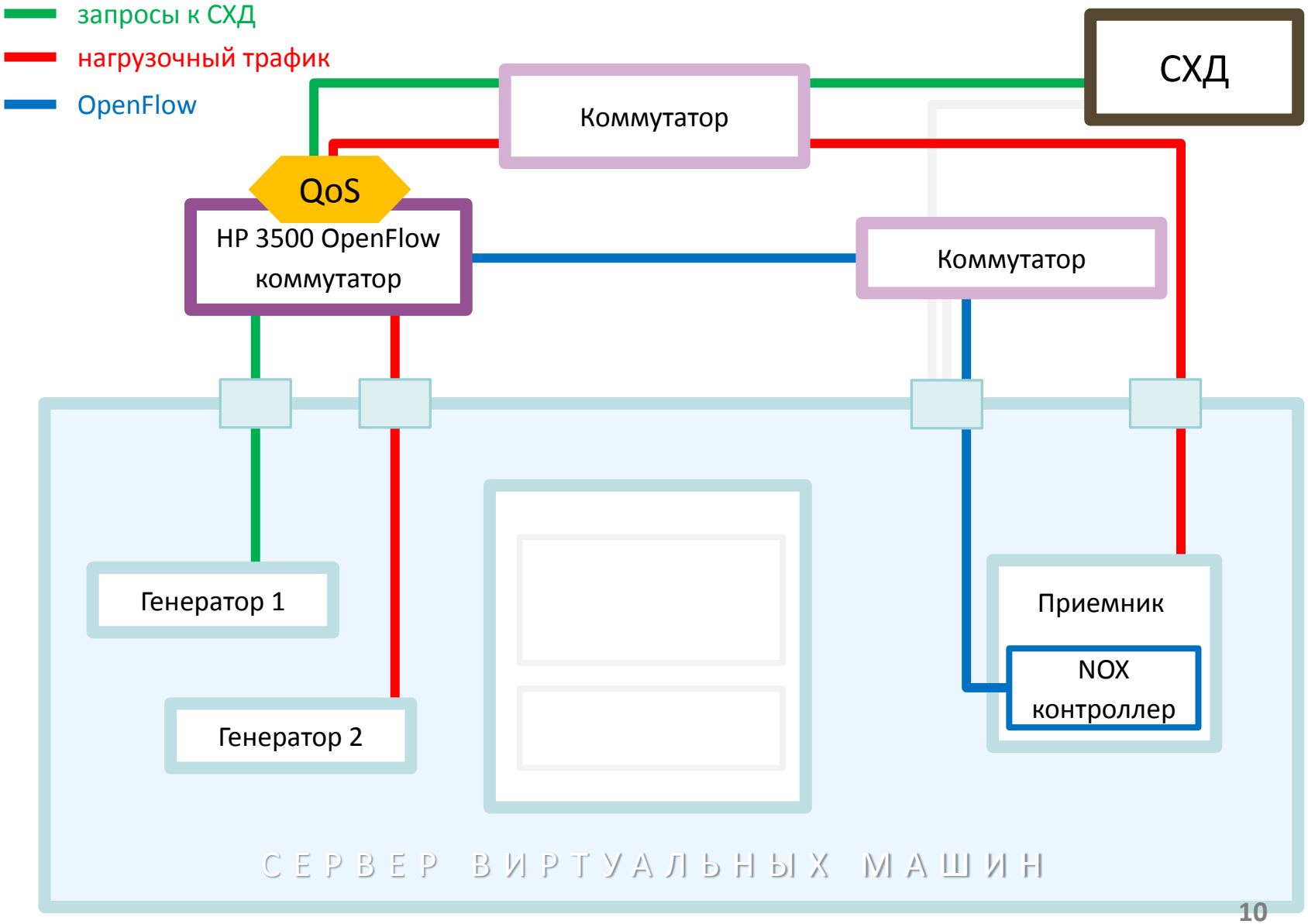
Тестирование с коммутатором HP ProCurve 3500yl и новым NOX

- Проблема: как автоматизировать настройку очередей?
  - Причина: не применить Механизм 2
  - Решение: скрипт на Python, производящий установки по SSH

Заданные полосы пропускания	Скорость запросов к СХД (Мбит/с)
100% / 0 %	10.0
80% / 20 %	8.40
20% / 80%	2.16
0% / 100%	0

- Схема работает, но
  - она в таком виде сложна и неочевидна
  - аппаратная поддержка ограничена
  - качество работы NOX оставляет желать лучшего

Общая схема тестирования



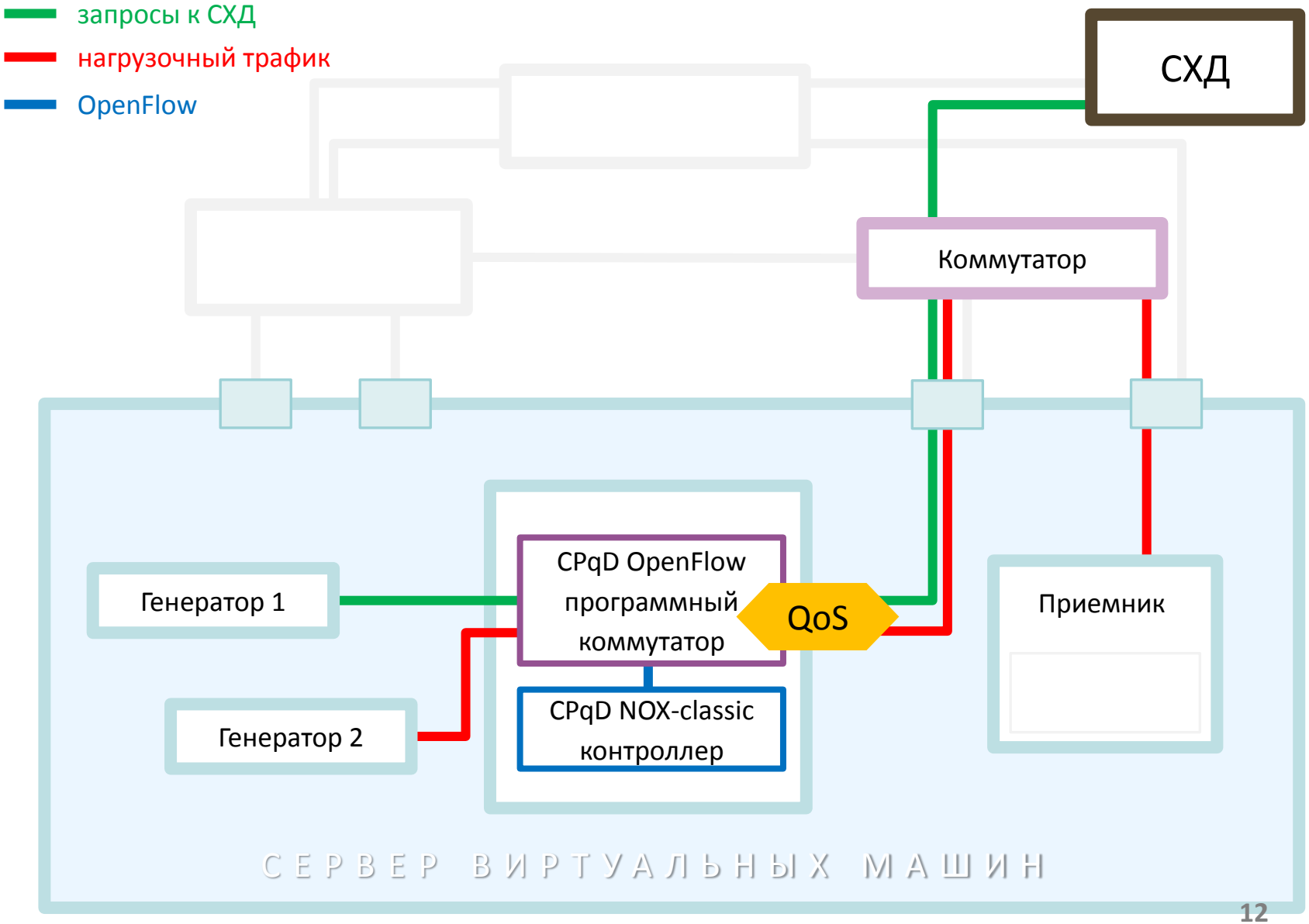
Тестирование с программным коммутатором

- Использовался программный комплекс от компании CPqD
  - программный коммутатор OpenFlow 1.2 Softswitch
  - модификация NOX-Classic (Zaku) – NOX 1.2 Oflib
- Проблема: очереди QoS не давали эффекта
  - Причина: динамическое изменение полосы пропускания Linux Traffic Control
  - Решение: фиксация ширины полосы пропускания

Заданные полосы пропускания	Скорость запросов к СХД (Кбайт/с)
100% / 0%	35.1
100% / 0.1%	31.6
100% / 100%	8.3
0.1% / 100%	5.4
0.1% / 0.1%	9.2

- Эффект очередей QoS виден, но
  - корреляция между задаваемыми полосами пропускания и реальными плохая
  - OpenFlow 1.2 Softswitch + NOX 1.2 Oflib – медленны, нестабильны и пригодны только для тестирования

Общая схема тестирования



Спасибо за внимание