Модели оценки качества сервиса

м.н.с. Степанов Евгений Павлович

Программа курса

Подходы:

1. ***Управление перегрузкой***
   - Современные протоколы управления перегрузкой TCP
2. ***Демультиплексирование/мультиплексирование***
   - Многопоточные транспортные протоколы
   - Маршрутизация на уровне интернет провайдеров
   - Network Coding
3. ***Сегментация***
   - TCP Proxy
4. ***Балансировка***
   - Балансировка нагрузки и управление трафиком

Модели:

- Сетевое исчисление: математический подход к качеству сервиса
- NS3: моделирование поведения сети с высокой точностью

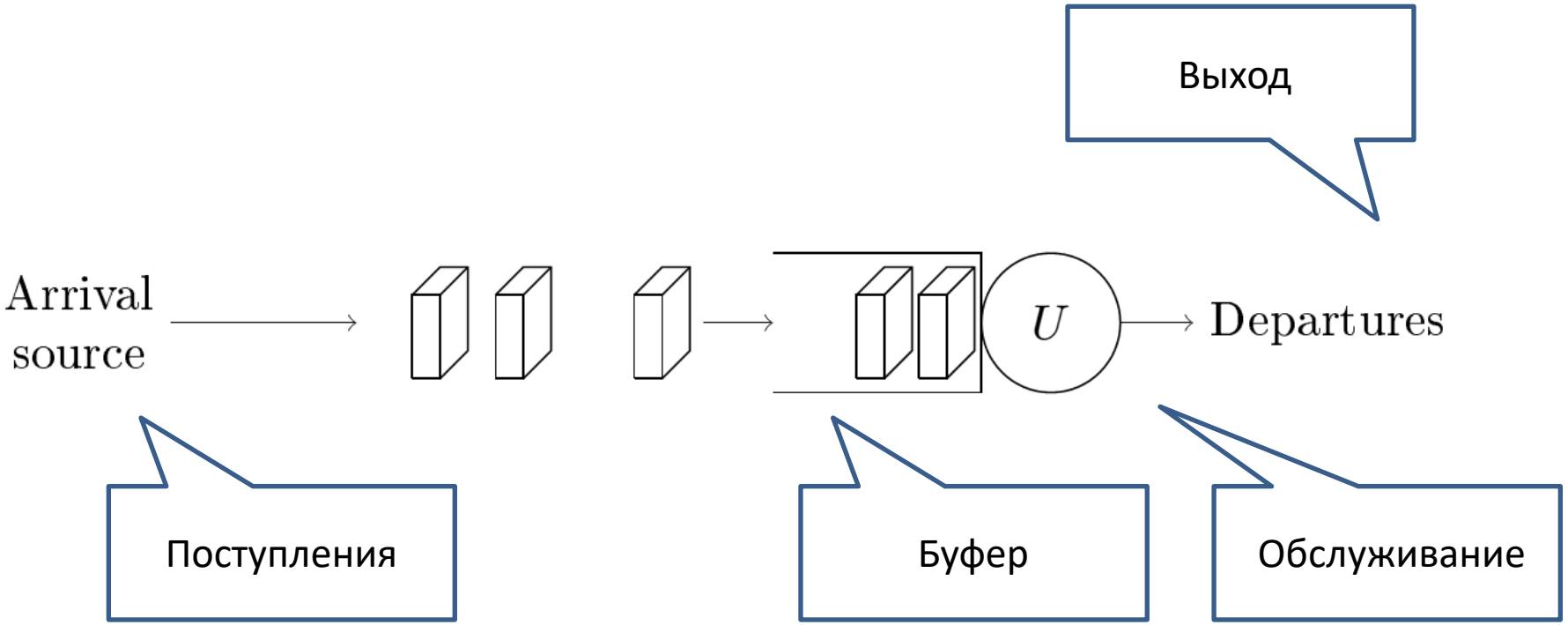
Примеры:

- HTTP3/QUIC
- Управление сетевыми ресурсами в Центрах Обработки Данных
- Обеспечение качества сервиса в сетях доставки контента
- Пропускная способность по требованию

Очереди


Очень долгое время обслуживания
Как долго это займет?
Ждать или нет?

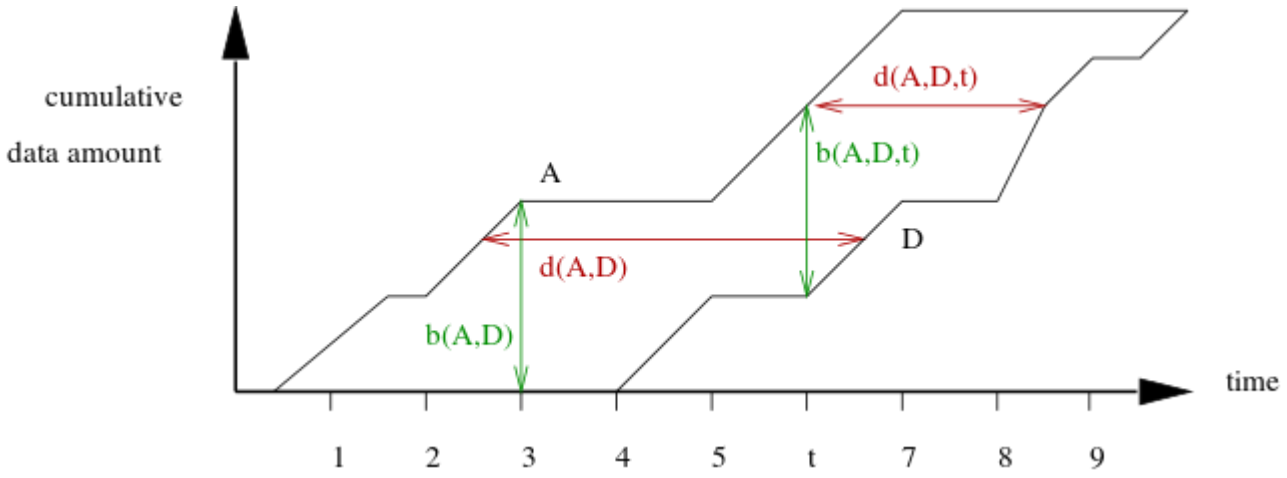
Очереди


Выход
Arrival source
U
Departures
Поступления
Буфер
Обслуживание

Анализ очередей

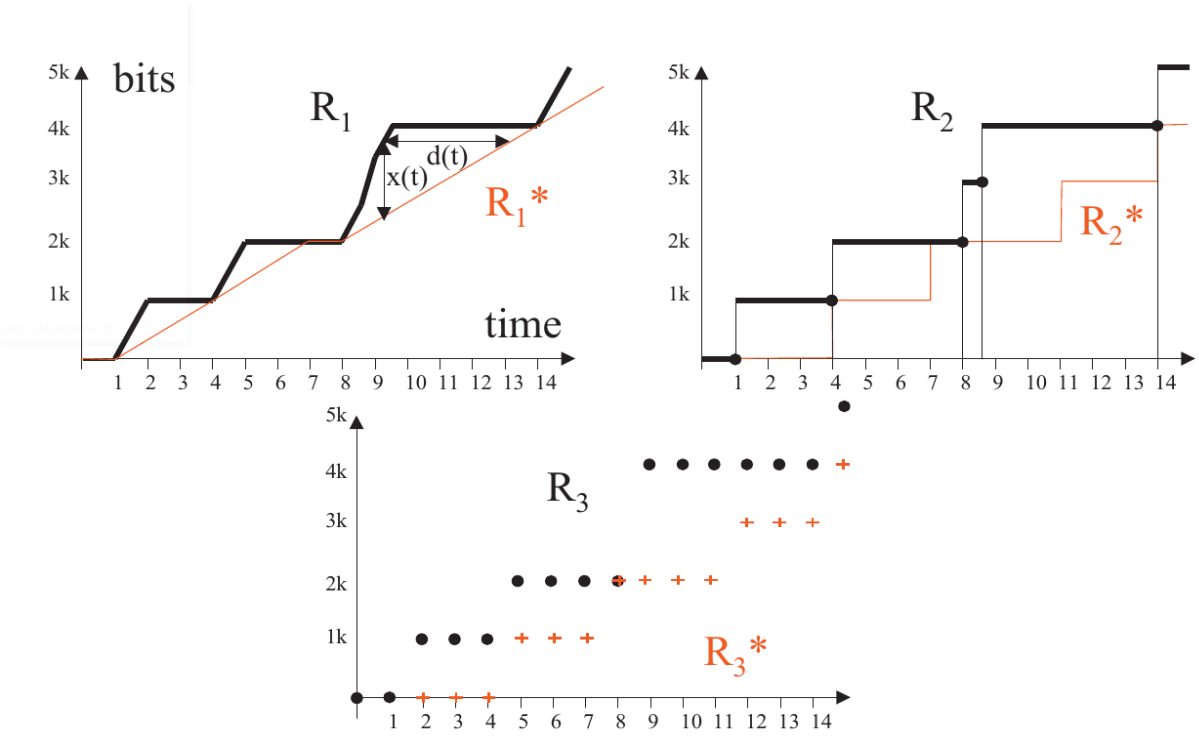
- Erlang – теория очередей, 1909
  - The Theory of Probabilities and Telephone Conversations Nyt Tidsskrift for Matematik, Vol. 20, No. B. (1909), pp. 33-39 by Agner K. Erlang
- Cruz – Сетевое исчисление, 1991
  - [1]*R. L. Cruz. A calculus for network delay, Part I: Network elements in isolation. IEEE Transactions on Information Theory, 37(1):114-131, January 1991.*
  - [2] *R. L. Cruz. A calculus for network delay, Part II: Network analysis. IEEE Transactions on Information Theory, 37(1):132-141, January 1991.*

Детерминированное сетевое исчисление



Модели потока данных

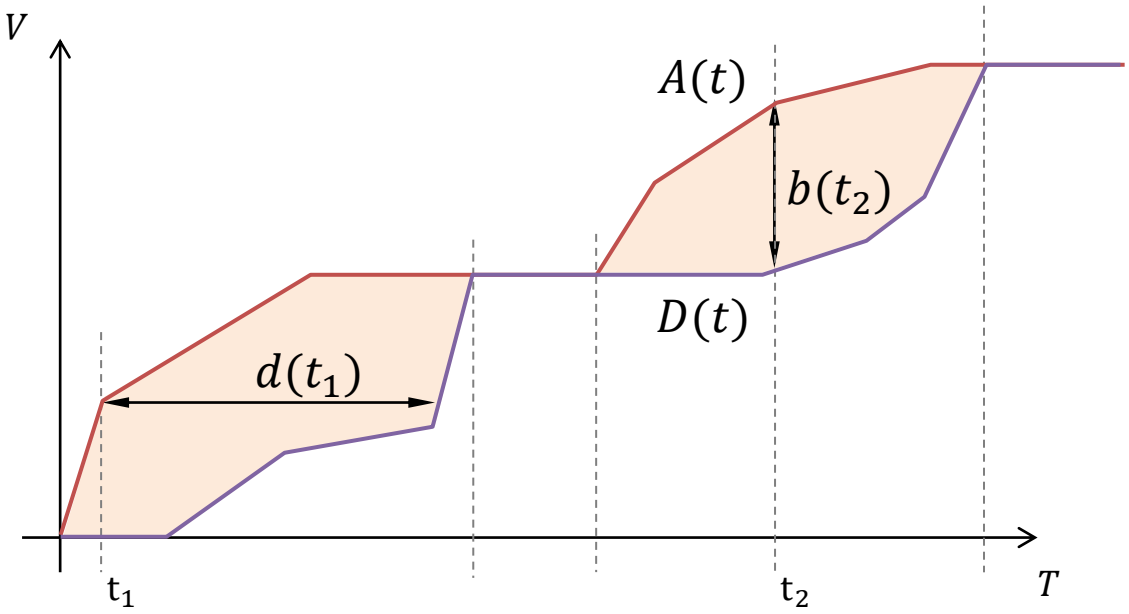
1. С дискретным временем (например, t = 0, 1, 2, 3, …).
2. С непрерывным временем, модель жидкости (R – непрерывная)
3. С непрерывным временем, общая модель (R – непрерывная справа или слева)



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

Функции поступления и отправки



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

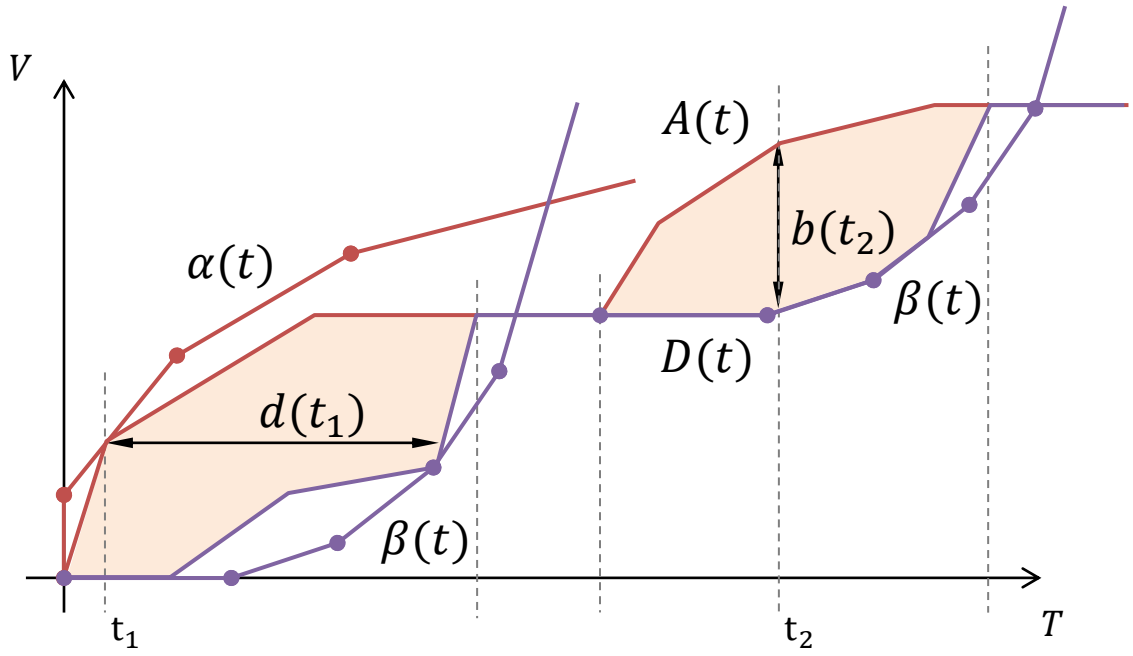
$$\forall t, \tau: A(t+\tau) - A(t) \leq \alpha(\tau)$$

  - за время τ поступает не больше $\alpha(\tau)$ данных
- ***Кривая сервиса*** $\beta(t)$ – накопительная функция, для которой внутри каждого периода оставание выполняется условие

$$\forall t, \tau: D(t+\tau) - D(t) \geq \beta(\tau)$$

  - за время τ обслуживается не меньше $\beta(\tau)$ данных

Кривые нагрузки и сервиса



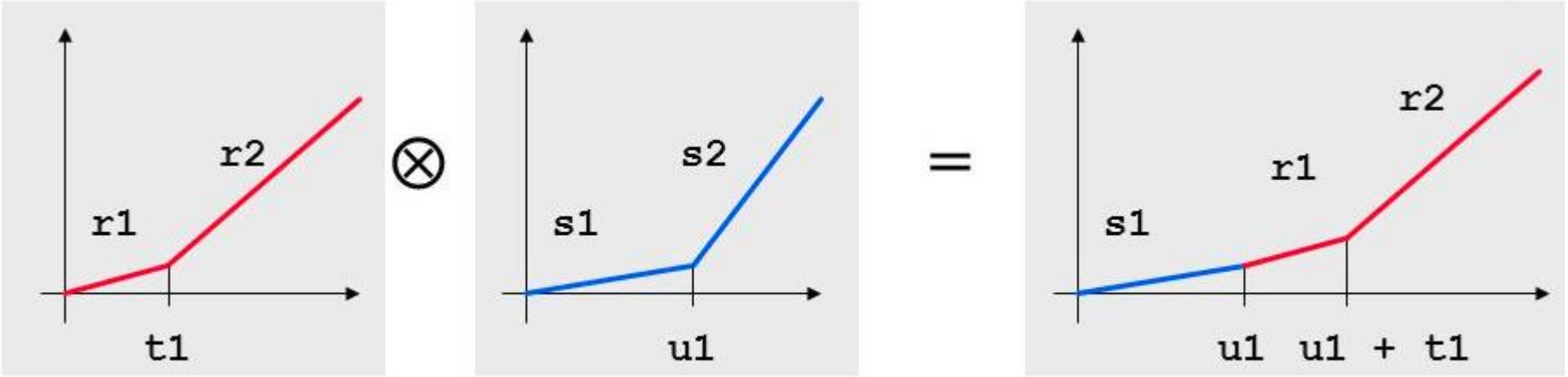
Min-plus алгебра

- Обычная алгебра R, +, *
  - Поле
  - a * (b + c) = a * b + a * c

- Min-plus алгебра R∪ {+∞}, ∧ (inf, min), +
  - Коммутативное идемпотентное полукольцо (диоид)
  - a + (b ∧ c) = (a + b) ∧ (a + c)

Min-plus свертка

- Обычная свертка
  - $(f \otimes g)(t) = \int_0^t f(t-s)g(s)ds$
- Min-plus свертка
  - $(f \otimes g)(t) = \inf_{0 \le s \le t} \{f(t-s) + g(s)\}$

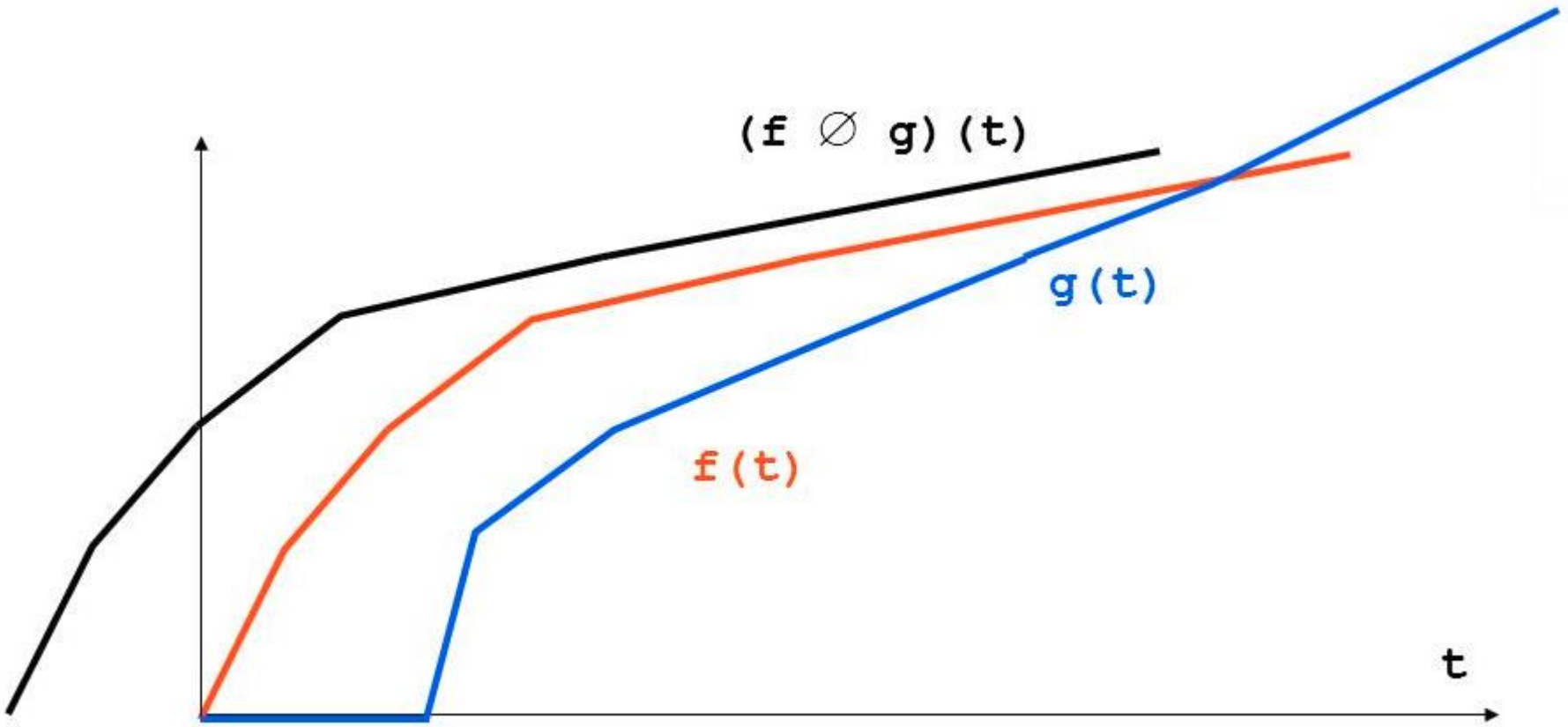


Min-plus свертка

- *Выпуклые функции:* если f и g выпуклые функции, тогда $f \otimes g$ тоже выпуклая. В частности, если f и g – выпуклые и кусочно-линейные функции, то $f \otimes g$ может быть получена соединением концов различных линейных кусков функций f и g в порядке увеличения наклона.

Обратная min-plus свертка

- $(f \oslash g)(t) = \sup_{u \geq 0} \{f(t+u) - g(u)\}.$



Полученные оценки

Оценка отставания:

$$b(t) \leq v(\alpha, \beta) = (\alpha \oslash \beta)(0)$$

Оценка задержки (FIFO):

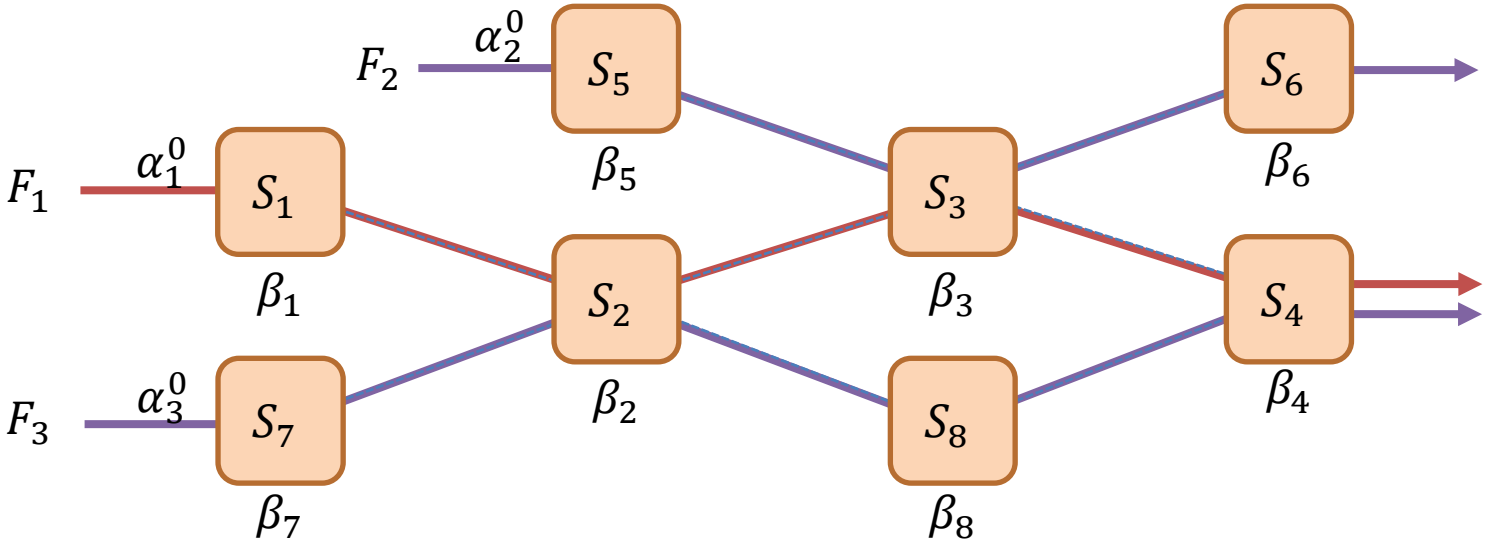
$$d(t) \leq h(\alpha, \beta) = \sup_{t}\{\inf\{\tau \geq 0 \mid \alpha(t) \leq \beta(t + \tau)\}\}$$

Композиция обработчиков:

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

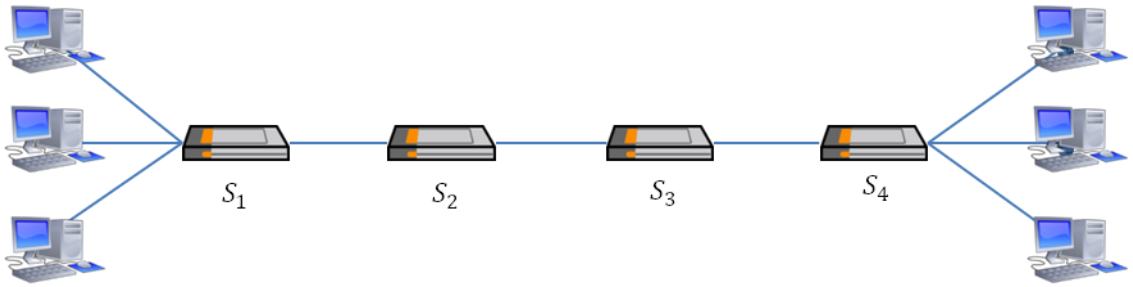
Применение Network Calculus

- Separated Flow Analysis
- Линейное программирование

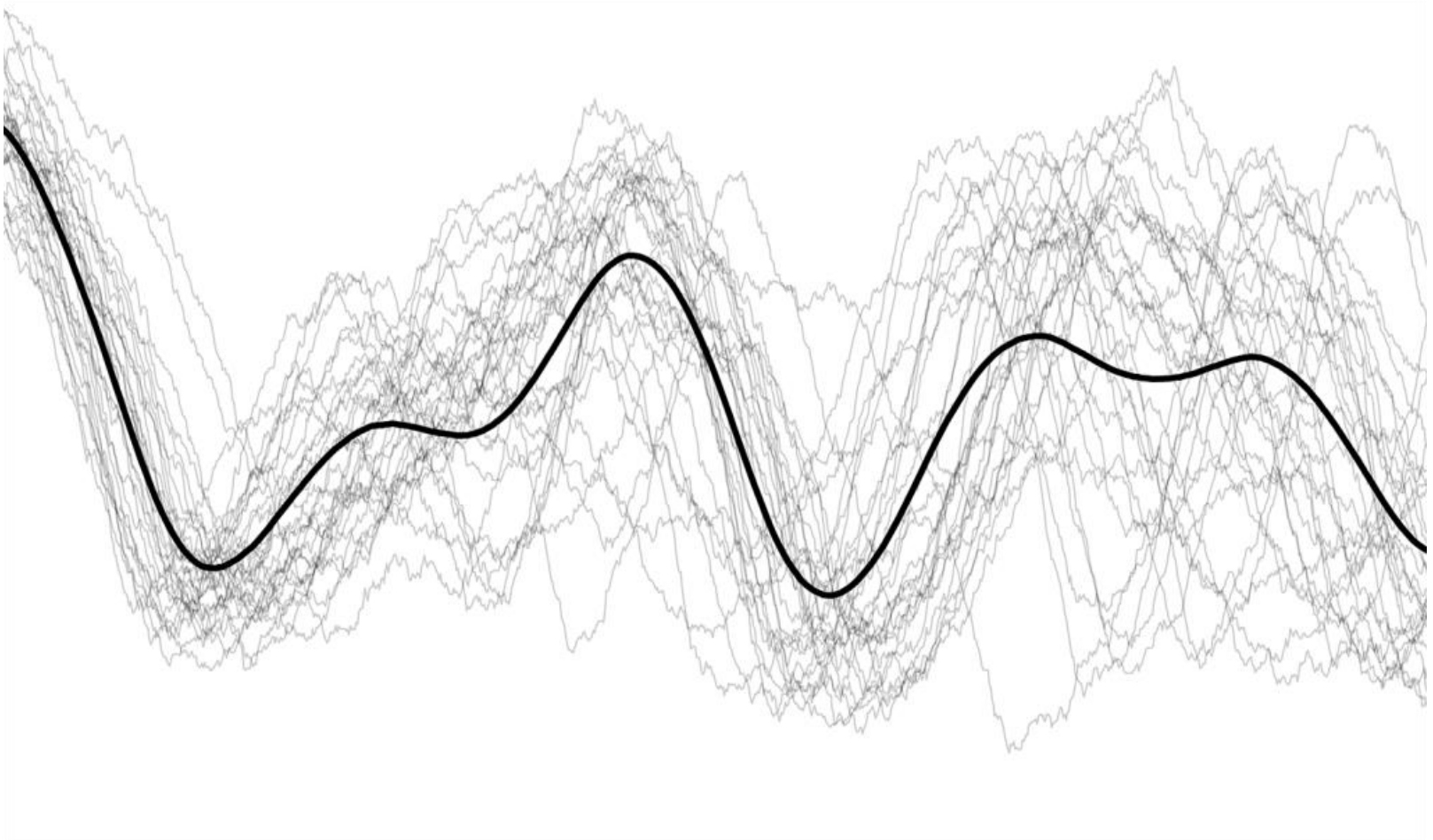


Применение Network Calculus

	Аудио			Видео			Данные		
Длина	NS3	SF	LP	NS3	SF	LP	NS3	SF	LP
2	159	12348	12348	244	11881	11881	244	9204	9204
3	281	16053	12519	459	23681	12045	366	18167	9334
4	403	23123	12689	488	35640	12209	515	27231	9457
5	525	33558	12860	692	47877	12373	639	36486	9584
6	647	47359	13031	733	60510	12537	733	46022	9711
7	769	64527	13201	891	73662	12701	962	55934	9838
8	891	85063	13372	998	87459	12866	977	66316	9964
9	1013	108972	13543	1099	102031	13030	1099	77270	10091



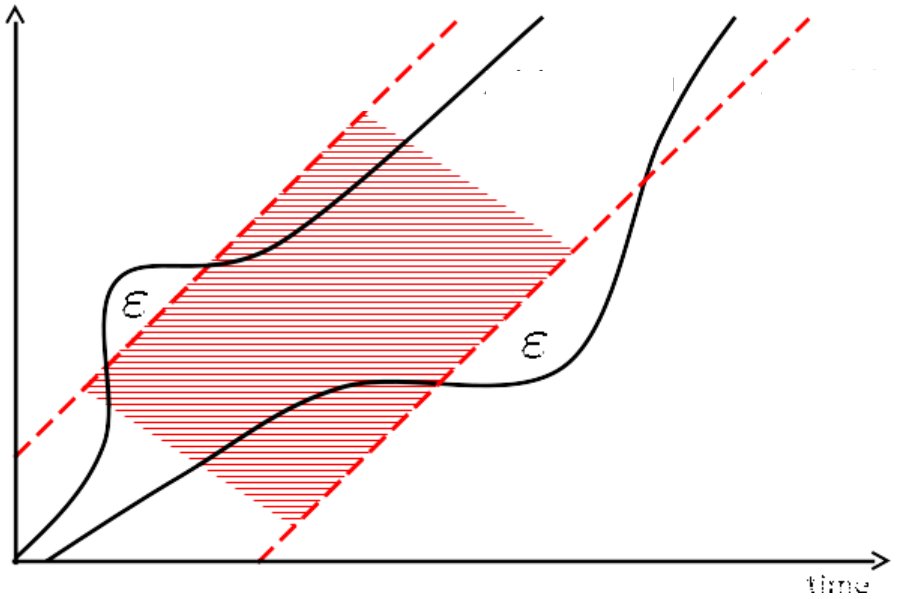
Стохастическое сетевое исчисление



Общая идея

Позволить нарушения кривых нагрузки и сервиса с некоторой вероятностью

-> теряем точность результата



Основные принципы

- Задать распределение для поступающего трафика и обслуживания обработчиков
- Получить афинную оценку
- Связать афинную оценку с погрешностью оценки
- Сохраняем подходы из детерминированного сетевого исчисления

Моделирование компьютерных сетей с помощью Network Simulator

Степанов Евгений Павлович

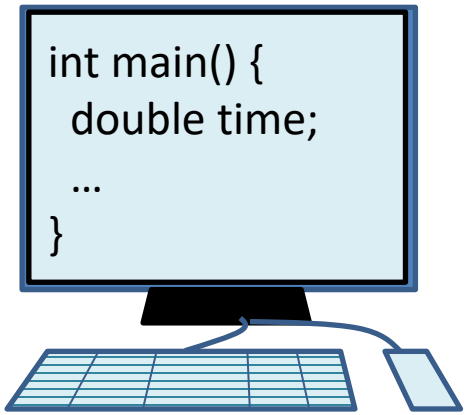
Network Simulator

- NS – среда дискретно-событийного моделирования компьютерных сетей
- NS ориентирована на исследования
  - Гибкость, модульность, расширяемость
  - Сопряжение с физическими устройствами
  - Поддержка стандартных методов IO
- Написана на C++ и Python
- OpenSource (GNU GPLv2)
- http://www.nsnam.org

Время в имитационном моделировании

Исследуемая система

Имитационная модель



1. *Физическое время* – время исследуемой системы
   [Интервал с 12:00 до 13:00 1 января 2017 года]
2. *Модельное время* – время исследуемое модели
   [C++ double в интервале [0.0, 1.0]]
3. *Инструментальное время* – время выполнения модели на аппаратуре
   [Интервал с 9:00 до 9:15 12 октября 2016 года]

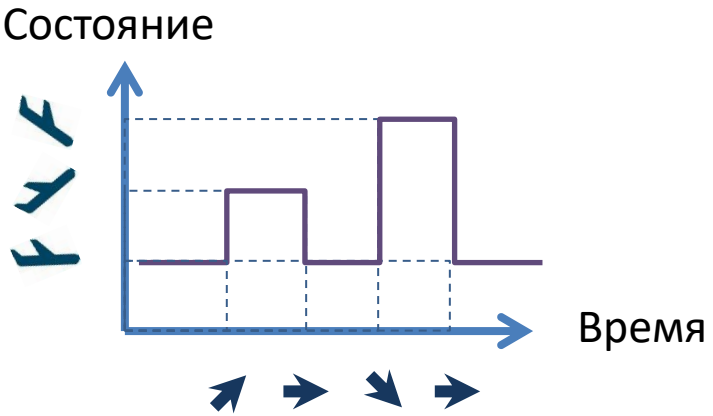
Способы продвижения модельного времени

- Максимально быстрое (As-Fast-As-Possible)
  - Продвижения модельного времени могут не соответствовать продвижениям физического времени
- В реальном времени
  - Изменения модельного времени и физического времени строго синхронизированы между собой
- В масштабируемом реальном времени
  - Модельное время движется в N раз быстрее/медленнее физического времени

Дискретно-событийное имитационное моделирование

Исследуемая система

Имитационная модель



Состояние системы – набор переменных

Событие – переход между состояниями системы

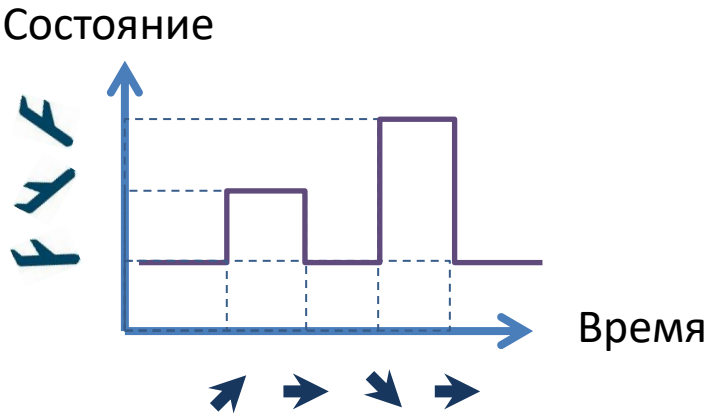
События происходят мгновенно в дискретные моменты модельного времени

Внутри имитационной программы поддерживается *список событий*

Дискретно-событийное имитационное моделирование

Исследуемая система

Имитационная модель



Обработка события:

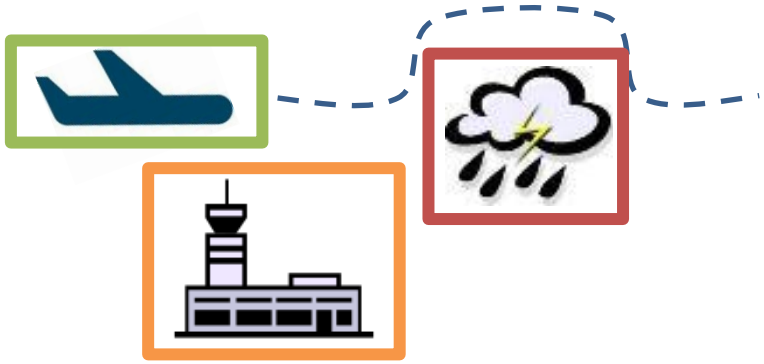
1. Изменение переменных состояния модели
2. Планирование новых событий в будущее
3. Исключение события из списка

Выполнение модели – обработка событий из списка в порядке увеличения их модельного времени

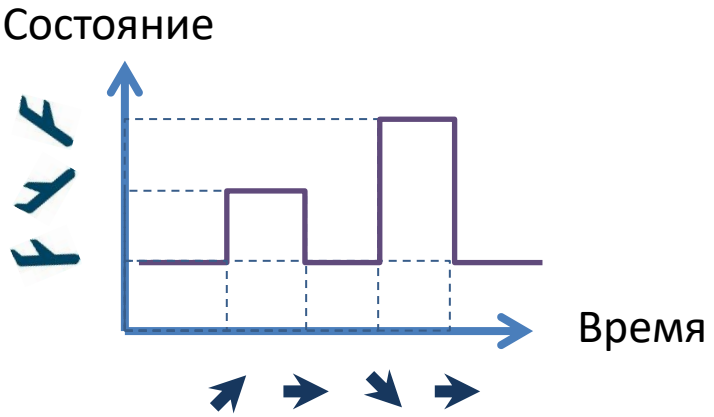
Моделирование завершается, когда список событий пуст

Дискретно-событийное имитационное моделирование

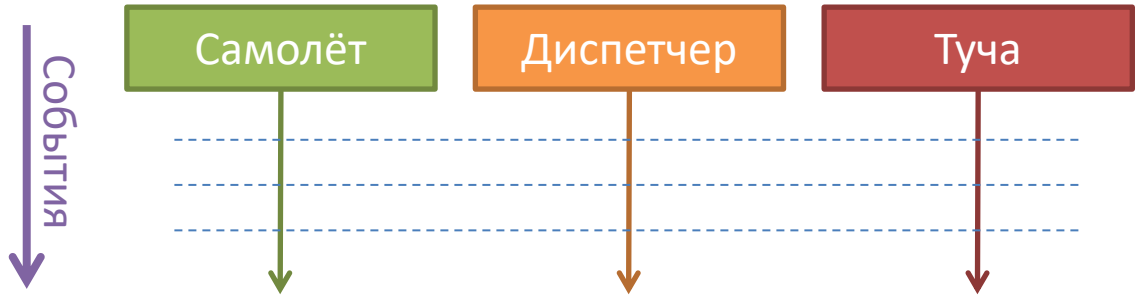
Исследуемая система



Имитационная модель



Модель представляет систему в виде набора независимых *логических процессов*, генерирующих собственные потоки событий



Необходима синхронизация компонентов модели

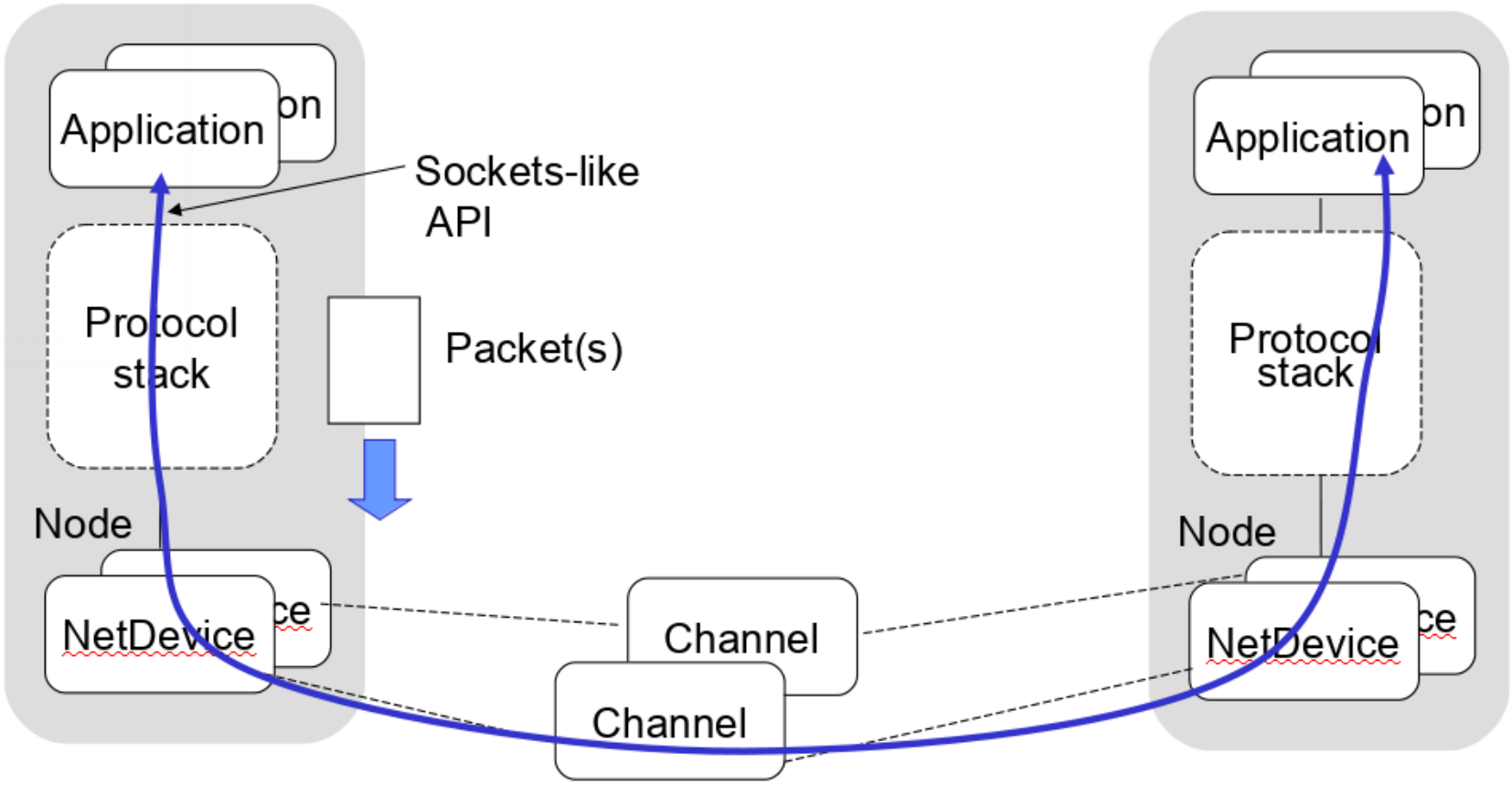
Подходы к синхронизация времени

- Общее модельное время для всех логических процессов
  - Простота реализации и использования
  - Синхронное, часто последовательное выполнение логических процессов
- Собственное модельное время для каждого логического процесса
  - Большая автономность процессов
  - Необходимость использования дополнительных алгоритмов синхронизации
  - Поддержка распределённого моделирования

Основные абстракции NS3

- Node – узел сети
- NetDevice – сетевая карта
- Interface – интерфейс сетевого уровня
- Application – приложение на узле
- Protocol Stack
- Channel – физическая линия связи
  - CSMAChannel, PointToPointChannel, WifiChannel

Основные абстракции NS3



Представление пакетов

- Virtual zero bytes
  - Вместо передачи случайных данных система моделирования может передавать их размер
  - Снижает требования по памяти
- Packet tags
  - В рамках среды моделирования к пакетам можно привязывать дополнительные атрибуты
  - Передача информации между не связанными между собой объектами моделирования

Приложения

- PacketSinkApplication
  - Получает пакеты по UDP или TCP
- OnOffApplication
  - Моделирует потоки с переменной активностью
  - Хорошо подходит для моделирования VOIP
- BulkSendApplication
  - Непрерывно передаёт данные
  - Хорошо подходит для моделирования FTP
- UDPEchoClientAppliction
  - Посылает одиночные пакеты с заданным интервалом
- UDPEchoServerAppliction
  - Отвечает на полученный пакеты

Построение модели сети

- Создать множество узлов
- Установить стек протоколов
- Добавить к узлам сетевые карты
- Соединить сетевые карты линиями связи
- Сконфигурировать сетевые интерфейсы
- Развернуть на узлах приложения
- Настроить время запуска и время остановки
- -> Запустить моделирование

Создание узлов и линий связи

```
  /* Build nodes. */
  NodeContainer h1, h2;
  h1.Create(1);
  h2.Create(1);

  NodeContainer router;
  router.Create(1);

  /* Build link. */
  CsmaHelper h1_link;
  h1_link.SetChannelAttribute ("DataRate", DataRateValue(100000000));
  h1_link.SetChannelAttribute ("Delay",  TimeValue(MilliSeconds(100)));

  CsmaHelper h2_link;
  h2_link.SetChannelAttribute ("DataRate", DataRateValue(100000000));
  h2_link.SetChannelAttribute ("Delay",  TimeValue(MilliSeconds(100)));
```

Добавление сетевых интерфейсов

```
→ /* Build link net device container. */
→ NodeContainer h1_and_router;
→ h1_and_router.Add(h1);
→ h1_and_router.Add(router);
→ NetDeviceContainer h1_link_devs = h1_link.Install(h1_and_router);

→ NodeContainer h2_and_router;
→ h2_and_router.Add(h2);
→ h2_and_router.Add(router);
→ NetDeviceContainer h2_link_devs = h2_link.Install(h2_and_router);
```

```
  /* Install the IP stack. */
→ InternetStackHelper StackHelper;
→ StackHelper.Install(h1);
→ StackHelper.Install(h2);
→ StackHelper.Install(router);
```

Конфигурирование сетевых интерфейсов

```
  /* IP assign. */
  Ipv4AddressHelper ipv4;
  ipv4.SetBase ("10.0.1.0", "255.255.255.0");
  Ipv4InterfaceContainer h1_iface = ipv4.Assign(h1_link_devs);
  ipv4.SetBase ("10.0.2.0", "255.255.255.0");
  Ipv4InterfaceContainer h2_iface = ipv4.Assign(h2_link_devs);

  /* Generate Route. */
  Ipv4GlobalRoutingHelper::PopulateRoutingTables();
```

Настройка приложений

```
→ /* Receiver application. */
→ PacketSinkHelper sinkHelper("ns3::TcpSocketFactory",
→ → → → → → InetSocketAddress(h1_iface.GetAddress(0), 1234));
→ ApplicationContainer sinkApp = sinkHelper.Install(h1.Get(0));
→ sinkApp.Start(Seconds(0.0));
→ sinkApp.Stop(Seconds(10.0));

→ /* Sender application. */
→ BulkSendHelper sendHelper("ns3::TcpSocketFactory",
→ → → → → → InetSocketAddress(h1_iface.GetAddress(0), 1234));
→ sendHelper.SetAttribute("MaxBytes", UintegerValue(1000000));
→ ApplicationContainer sendApp = sendHelper.Install(h2.Get(0));
→ sendApp.Start (Seconds (0.1));
→ sendApp.Stop (Seconds (10.0));
```

Запуск среды выполнения

```
  /* Simulation. */
  /* Pcap output. */
  CsmaHelper helper;
  helper.EnableAsciiAll("sample");
  helper.EnablePcapAll("sample");

  /* Stop the simulation after x seconds. */
  uint32_t stopTime = 11;
  Simulator::Stop (Seconds(stopTime));
  /* Start and clean simulation. */
  Simulator::Run ();
  Simulator::Destroy ();
```

Логирование

- Указание компонентов в коде

  LogEnableComponent(<log-component>, <option>|<option>)

- Установка переменной NS_LOG

  $ NS_LOG="<log-component>=<option>|<option>...:<log-component>... “

- Компоненты – модули NS3
  - Список компонентов выводится при указании неизвестного компонента
- Опции – фильтры для логов
  - Уровни логирования (error, log, debug, info, …)
  - Префиксы (function, time, node, …)
- Пример:
  - $ export NS_LOG=UdpEchoClientApplication=level_all
  - $ export ’NS_LOG=UdpEchoClientApplication=level_all|prefix_func: UdpEchoServerApplication=level_all|prefix_func’

Трассировка

- Модули определяют trace source – изменяющиеся параметры модели, которые может быть полезно отслеживать
- Модели могут подписываться на trace source, устанавливая собственные callback-функции, которые необходимо вызвать при изменении параметра
- Каждый trace-source может поддерживать множество callback-функций одновременно

Управление trace sources

- Все trace source в системе доступны через единый реестр на этапе конфигурации модели
- На этапе инициализации NS3 устанавливает связи между существующими trace source и подписавшимися на них компонентами модели

```
void CwndTracer (uint32_t oldval, uint32_t newval) {}
Config::ConnectWithoutContext(
        "/NodeList/0/$ns3::TcpL4Protocol/SocketList/0/CongestionWindow",
        MakeCallback(&CwndTracer)
);
```

Резюме

- NS-3 хорошо подходит для моделирования компьютерных сетей на низком уровне
- NS-3 имеет серьёзные ограничения по масштабируемости исследуемых моделей

Программа курса

Подходы:

1. ***Управление перегрузкой***
   - Современные протоколы управления перегрузкой TCP
2. ***Демультиплексирование/мультиплексирование***
   - Многопоточные транспортные протоколы
   - Маршрутизация на уровне интернет провайдеров
   - Network Coding
3. ***Сегментация***
   - TCP Proxy
4. ***Балансировка***
   - Балансировка нагрузки и управление трафиком

Модели:

- Сетевое исчисление: математический подход к качеству сервиса
- NS3: моделирование поведения сети с высокой точностью

Примеры:

- HTTP3/QUIC
- Управление сетевыми ресурсами в Центрах Обработки Данных
- Обеспечение качества сервиса в сетях доставки контента
- Пропускная способность по требованию