Моделирование компьютерных сетей с помощью Network Simulator

Чемерицкий Евгений Викторович

к.ф.-м.н.

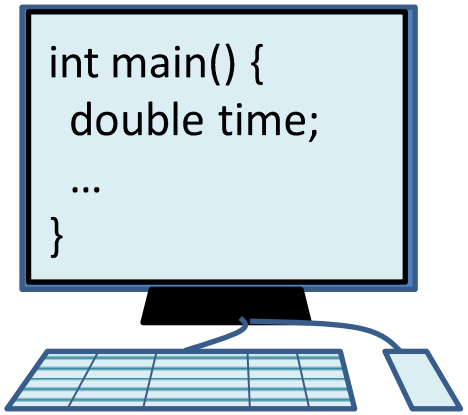
Network Simulator

- 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]

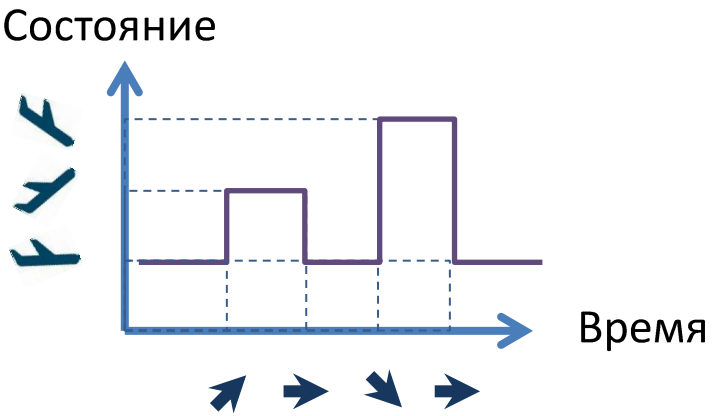
Способы продвижения модельного времени

- 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

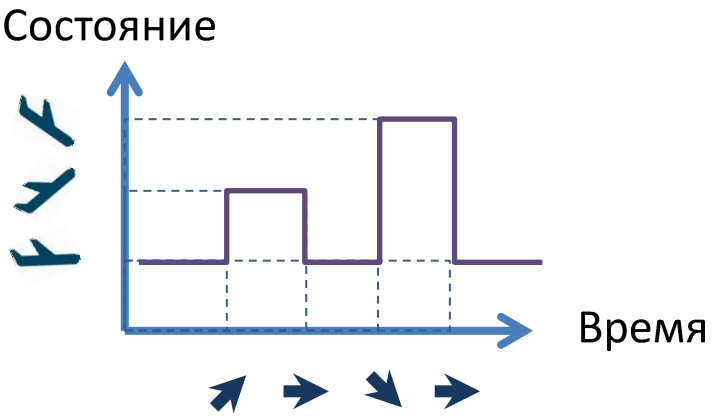
События происходят мгновенно в дискретные моменты модельного времени

Внутри имитационной программы поддерживается *список событий*

Дискретно-событийное имитационное моделирование

Исследуемая система

Имитационная модель



Обработка события:

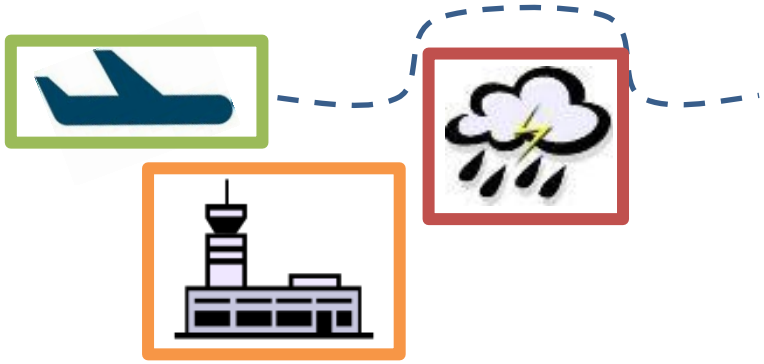
1. Изменение переменных состояния модели
2. Планирование новых событий в будущее
3. Исключение события из списка

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

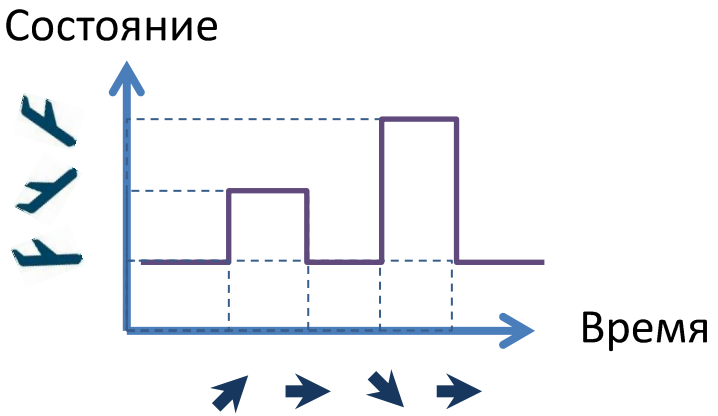
Моделирование завершается, когда список событий пуст

Дискретно-событийное имитационное моделирование

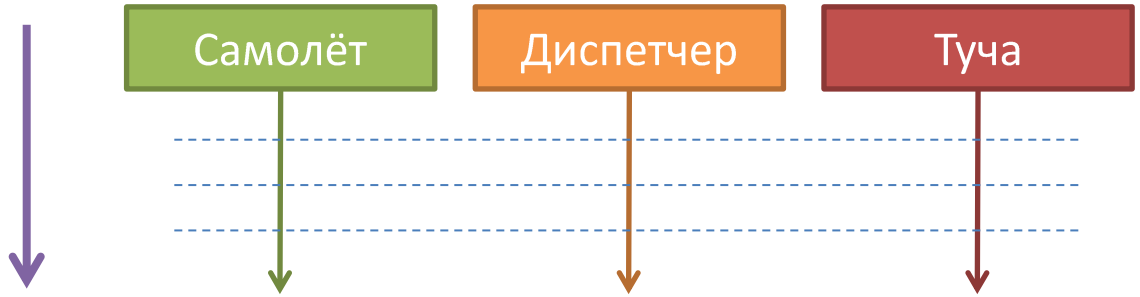
Исследуемая система



Имитационная модель



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



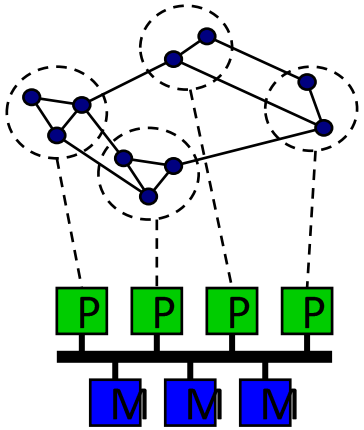
Необходима синхронизация компонентов модели

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

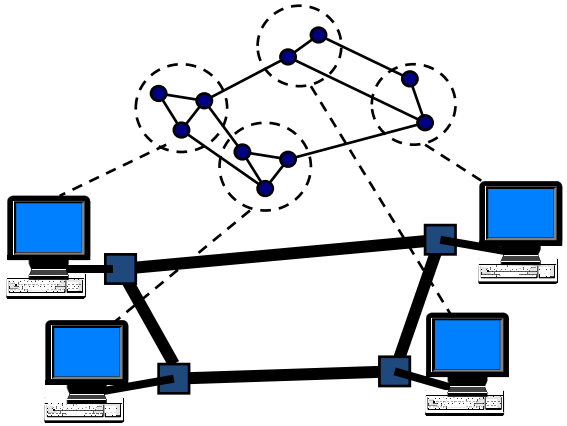
Многомашинные системы моделирования

Параллельные



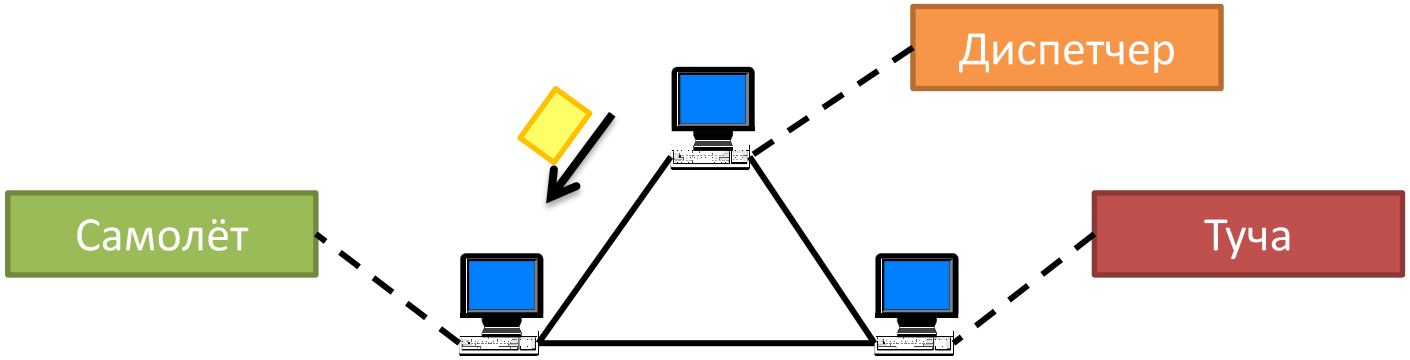
- Системы принятия решений
- Системы массового обслуживания
- Телекоммуникации

Распределённые



- Виртуальные среды
- Системы дистанционного обучения
- Игры

Алгоритмы временной синхронизации



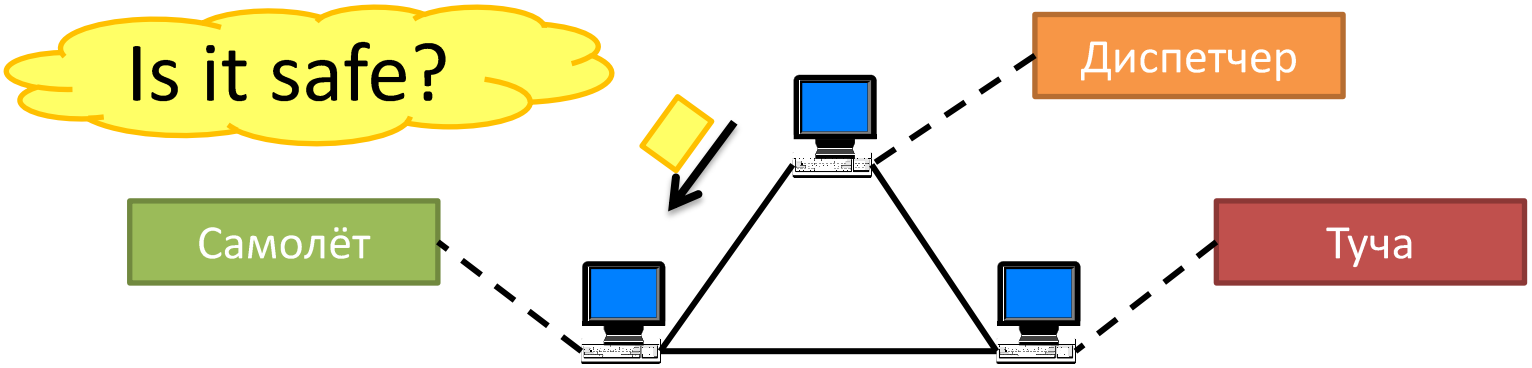
Распределённая модель – набор имитационных моделей, выполняющихся на разных машинах

Взаимодействия между логическими процессами модели происходят через посылку *сообщений*

Сообщения соответствуют событиям систем ДСИМ

Каждый логический процесс получает сообщения от других процессов строго в порядке увеличения их временных меток

Алгоритмы временной синхронизации



Консервативные алгоритмы не позволяют передавать сообщения, пока есть опасность появления события из прошлого

Оптимистические алгоритмы допускают события из прошлого, но корректно обрабатывают этот случай с помощью механизма откатов

Смешанные алгоритмы позволяют части машин использовать консервативные алгоритмы, а части – оптимистические.

Основные абстракции NS3

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

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, …)

Трассировка

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

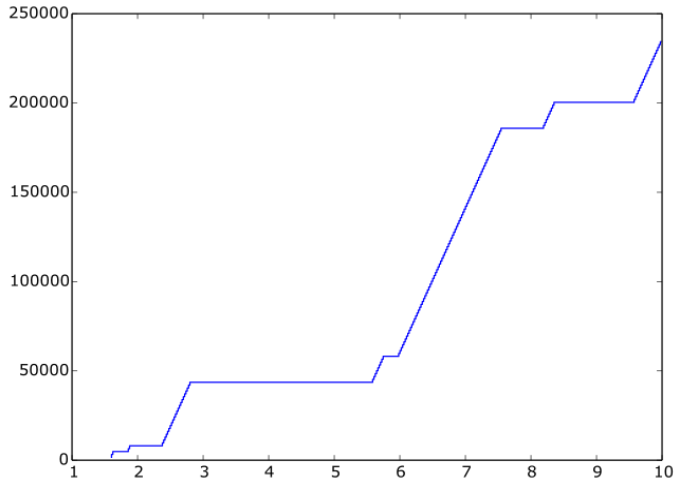
Управление trace sources

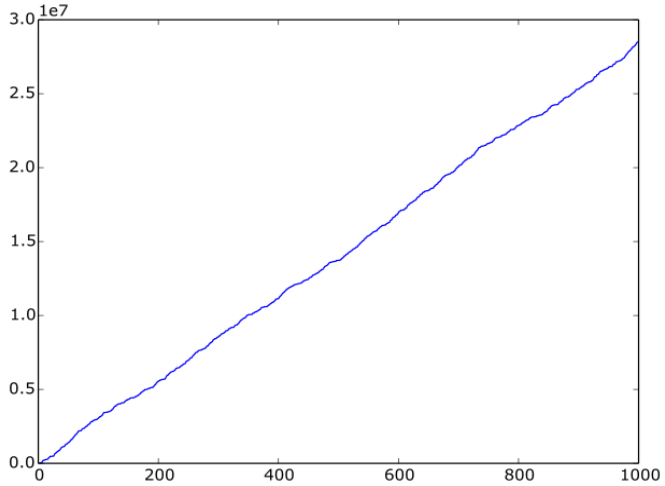
- Все trace source в системе доступны через единый реестр на этапе конфигурации модели
- На этапе инициализации NS3 устанавливает связи между существующими trace source и подписавшимися на них компонентами модели

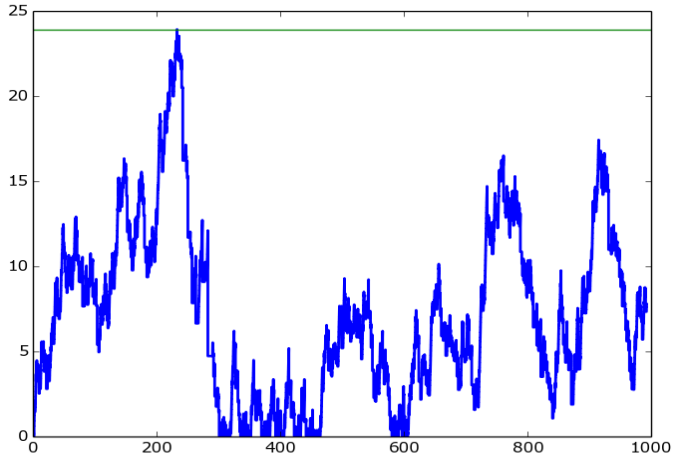
```
void CwndTracer (uint32_t oldval, uint32_t newval) {}
Config::ConnectWithoutContext(
            "/NodeList/0/$ns3::TcpL4Protocol/SocketList/0/CongestionWindow",
            MakeCallback(&CwndTracer)
);
```

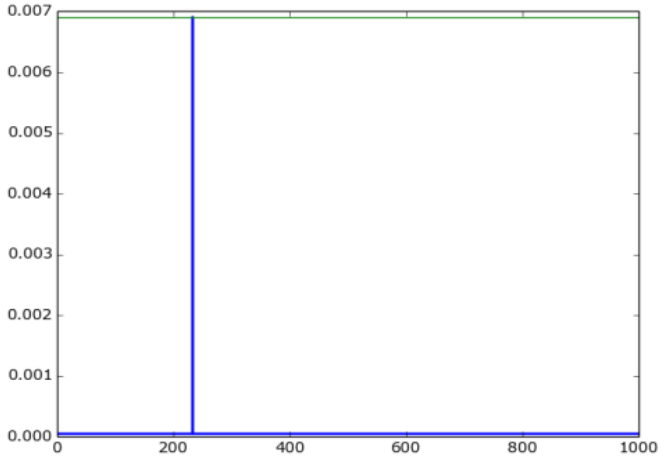
Профили потоков данных

Траффик VOIP



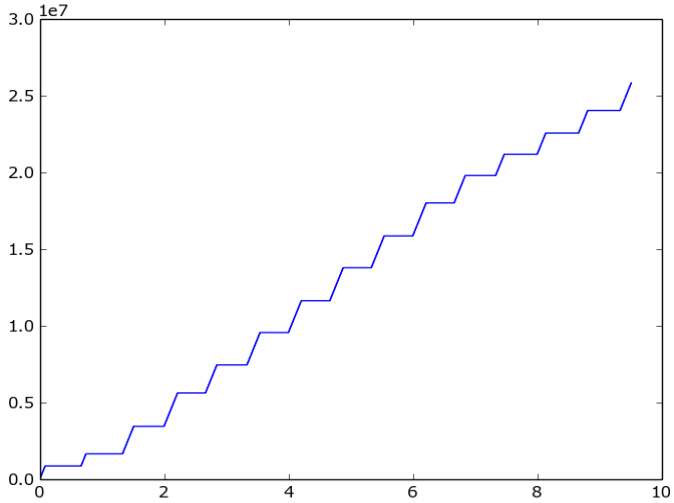


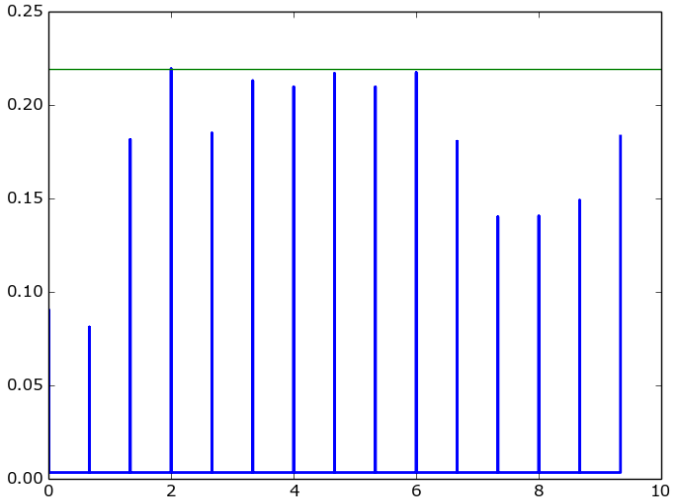


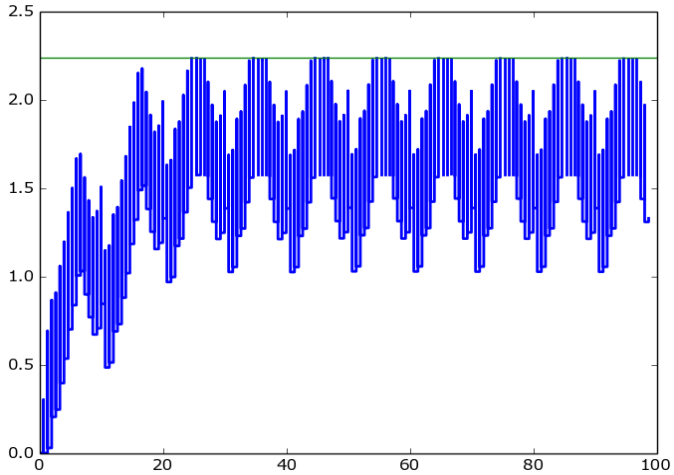


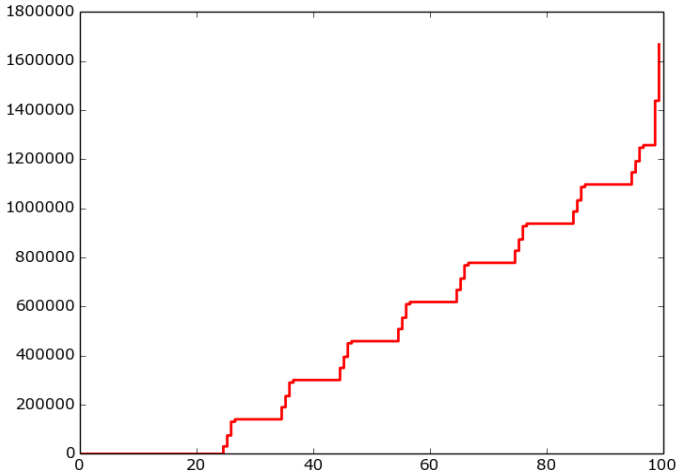
Профили потоков данных

Видео траффик



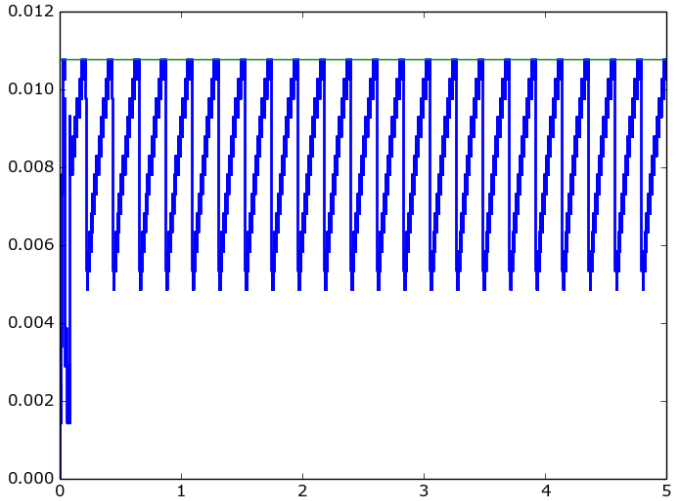


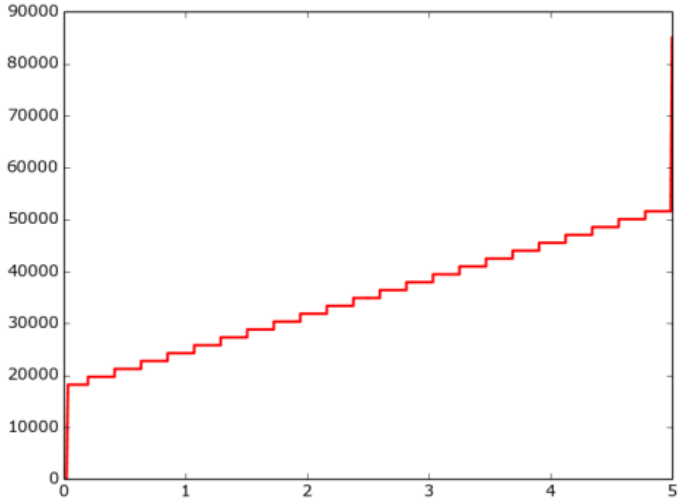


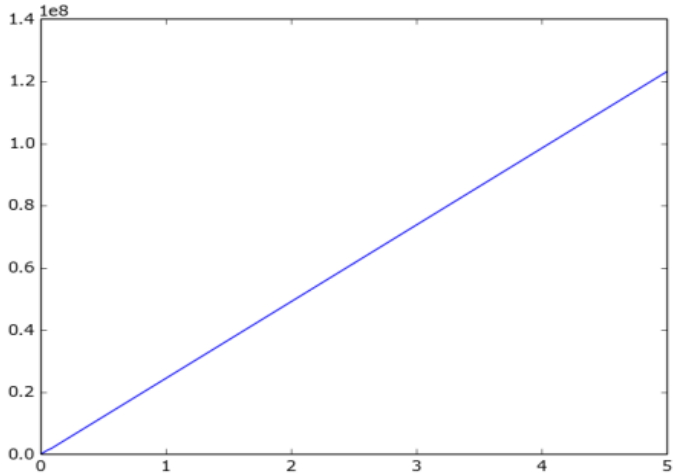


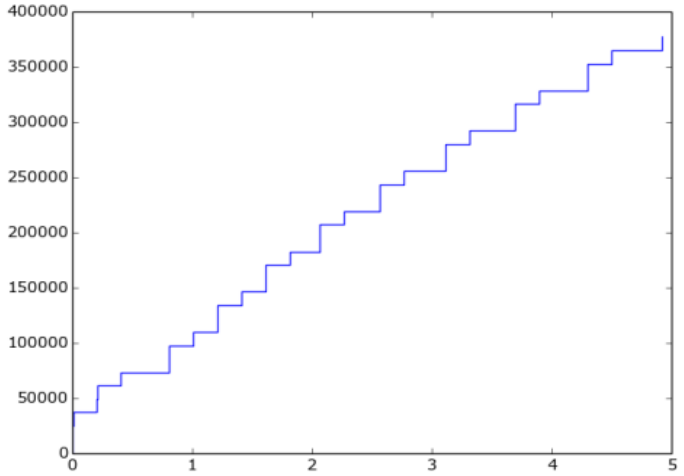
Профили потоков данных

Потоки данных









Резюме

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

Задание для получения автомата по курсу

- Построить модель сети из двух узлов, соединённых друг с другом через один маршрутизатор линиями связи с полосой пропускания 100Mbps и задержкой в 200ms
- Запустить TCP BulkSenderApplication и пострить график изменения CWND на протяжении 10с
- Исследовать несколько различных алгоритмов управления перегрузкой TCP:
  - NewReno, TCP Illinois, TCP Westwood
- Рассчитать процент утилизации канала для каждого из указанных алгоритмов

Проверка остаточных знаний =)

- Что такое эффект переподписки?
- Какие преимущества даёт интернет провайдерам использование IXP?
- Почему в сети сосуществует сразу много разных алгоритмов управления перегрузками TCP соединений?