Активные и пассивные измерения

к.ф.-м.н., м.н.с. Степанов Евгений Павлович

Оценка качества сервиса

- Зачем оценивать качество сервиса?
- Классификация: внутри сети, вне сети, на моделях
- Оценка показателей качества сервиса (B, D, L, J)

Качество сервиса наложенного канала

- Наложенные каналы – SD-WAN, CDN, NPC, CFN, CPN, …
- Пассивные измерения – есть статистика по наблюдаемым потокам (network-limited vs application-limited)
- Активные измерения – посылаем дополнительные пакеты

Активные/пассивные измерения: задержка

- *r – переменная, начальная оценка RTT*
- *m – измерение RTT для последнего подтвержденного пакета*
- *Вычисляем взвешенное среднее –*

$$r := a*r + (1- a)* m, \text{ где } 0< a <1 \text{ (обычно 7/8)}$$

Введение в сети ЭВМ: лекция «Перегрузки»

Jacobson V. Congestion avoidance and control //ACM SIGCOMM computer communication review. – 1988. – Т. 18. – №. 4. – С. 314-329.

Активные/пассивные измерения: вариация задержки

- RFC 1889 (RTP) – interarrival jitter

```
int transit = arrival - r->ts;
int d = transit - s->transit;
s->transit = transit;
if (d < 0) d = -d;
s->jitter += (1./16.) * ((double)d - s->jitter);
```

- RFC 3393 – IP packet Delay Variation
  - IPDV
  - PDV

Активные/пассивные измерения: потери

- Статистика по потерям
- ERL, EURL

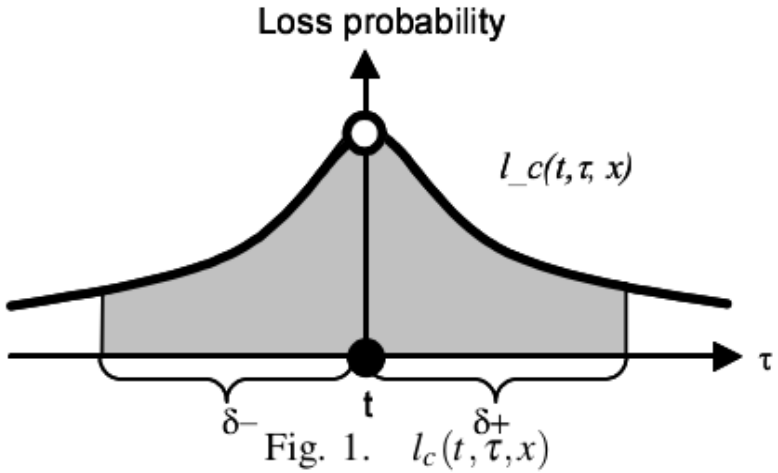
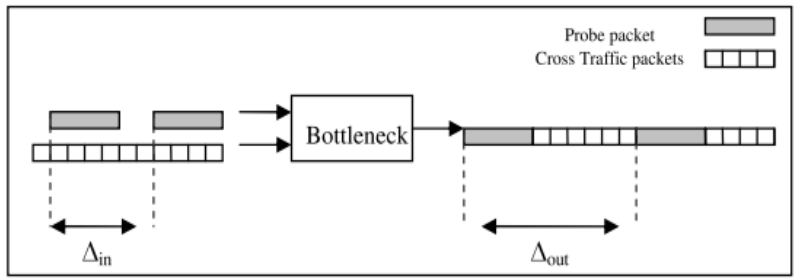


Fig. 1. $l_c(t,\tau,x)$

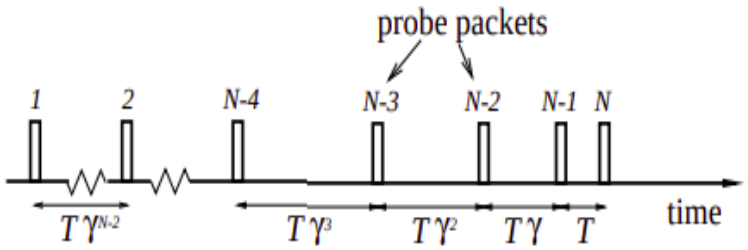
Активные измерения: пропускная способность

Доступная пропускная способность канала (ДПС) — неиспользованная часть пропускной способности канала *(которую может получить поток пользователя)*

Активные методы оценки ДПС



Probe gap model:
Spurce

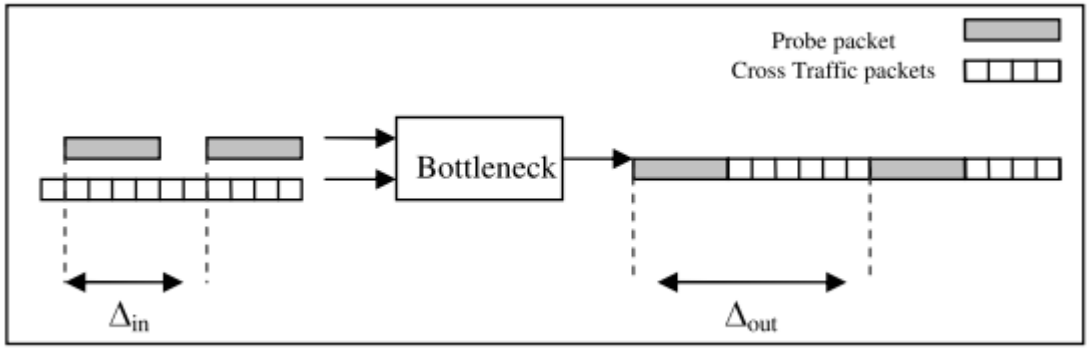


Probe rate model:
pathLoad, pathChirp, YAZ, pathQuick3, pathML

Burst Queue Recovery:
FABMon

Spurce

Probe gap model – модель пробирования зазора. Используется информация о временном промежутке между прибытием двух последовательных служебных пакетов на получателе.



Предполагается единственное узкое место с ПС равной C

Δ_{in} – временной промежуток между посылкой данных

Δ_{out}– временной промежуток между получением данных

A – искомая ДПС

$$A = C \times \left(1 - \frac{\Delta_{out} - \Delta_{in}}{\Delta_{in}}\right).$$

Spurce:
C – предполагается известным;

Δ_{in} – время прохождения 1500 байт по узкому месту;

ДПС в момент времени t – среднее последних K измерений (по умолчанию, K =100);

Межпарный интервал – по экспоненциальному закону со средним $\tau >> \Delta_{in}$

PathLoad

Probe rate model - модель пробирования скорости, основана на концепции самовызванной перегрузки. Общая идея: будем отправлять последовательности служебных пакетов,
r_{in} - скорость отправки пакетов, r_{out} — скорость на получателе, A - ДПС.

Тогда: $\frac{r_{in}}{r_{out}} = \begin{cases} \leq 1 & r_{in} \leq A \\ > 1 & r_{in} > A \end{cases}$

Перебираем r_{in} до некоторого порогового значения, после которого начинаются перегрузки. Это значение и будет искомой ДПС.

PathLoad: для скорости отправки R проводятся N измерений длиной K пакетов, R итеративно изменяется. Эти N потоков в совокупности называются флотом. Пакеты имеют одинаковый размер и в рамках одного потока отправляются через равные промежутки времени.

Алгоритм оценки ДПС

Упрощенный

Rmin, Rmax - нижняя и верхняя границы для A после потока n. При n = 0: Rmin=0, Rmax >> A

R(n) - скорость отправки потока n, для n = 1, 2, ...

- Если $R(n) > A$, то $R^{max} = R(n)$
- Если $R(n) <= A$, то $R^{min} = R(n)$

$R(n+1) = (R^{max} + R^{min})/2$

Выход при: $(R^{max} - R^{min}) <= \Omega$

В pathLoad

Если для флота из N потоков нельзя однозначно сказать R > A или R<=A, то считается, что R принадлежит "**серой зоне**": R><A . Ее границы: Gmin и Gmax

При R(n) < A, R(n) > A или если Gmin, Gmax неизвестны - действуем по упрощенному алгоритму.

Иначе, если известны Gmin, Gmax и R><A:
обновляется одна из границ Gmin, Gmax и затем
$R(n+1) = (G^{max} + R^{max})/2$, если $R(n) = G^{max}$
$R(n+1) = (G^{min} + R^{min})/2$, если $R(n) = G^{min}$

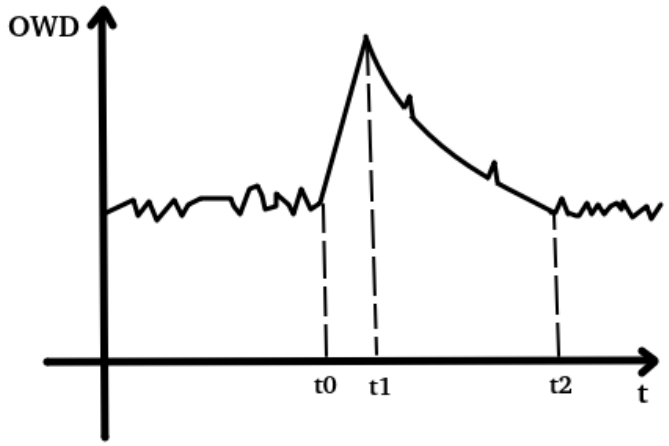
Выход при: $(R^{max} - R^{min}) <= \Omega$ либо $R^{max} - G^{max} <= \chi$ и $G^{min} — R^{min} <= \chi$ одновременно

FABMon

Новый метод: **Burst Queue Recovery** (BQR) - восстановление очереди пакетов.

Средство: FABMon. 2 фазы:

1. “Загрузка”: вызывается мгновенная перегрузка - loading train
2. “Инспекция”: 2 потока: вычисление ДПС - главный, подсчет OWD - вспомогательный



$\Delta t = t^{(2)} - t^{(0)}$

P - объем сгенерированного трафика за $[t^{(0)};t^{(2)}]$

$\hat{A} = P/\Delta t$ - оценка ДПС

Моделирование компьютерных сетей с помощью Network Simulator

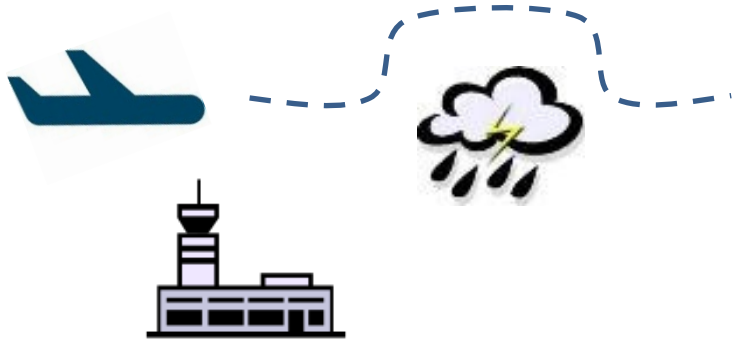
к.ф.-м.н., м.н.с. Степанов Евгений Павлович

Network Simulator

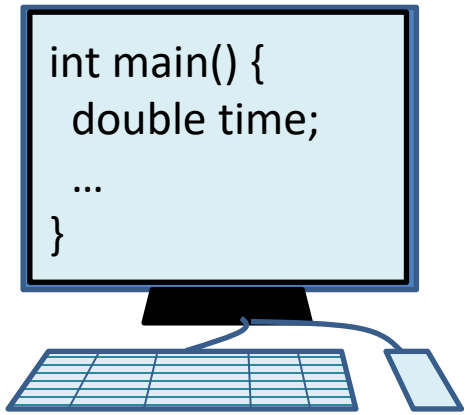
- NS – среда дискретно-событийного моделирования компьютерных сетей
- NS ориентирована на исследования
  - Гибкость, модульность, расширяемость
  - Сопряжение с физическими устройствами
  - Поддержка стандартных методов IO
- Написана на C++ и Python
- OpenSource (GNU GPLv2)
- http://www.nsnam.org

Время в имитационном моделировании

Исследуемая система



Имитационная модель



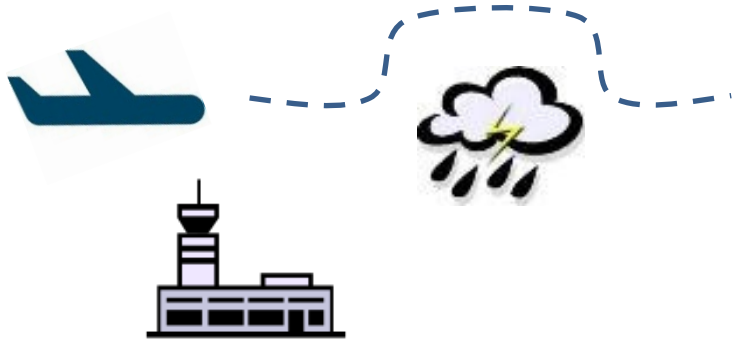
1. *Физическое время* – время исследуемой системы
   [Интервал с 12:00 до 13:00 1 января 2017 года]
2. *Модельное время* – время исследуемое модели
   [C++ double в интервале [0.0, 1.0]]
3. *Инструментальное время* – время выполнения модели на аппаратуре
   [Интервал с 9:00 до 9:15 12 октября 2016 года]

Способы продвижения модельного времени

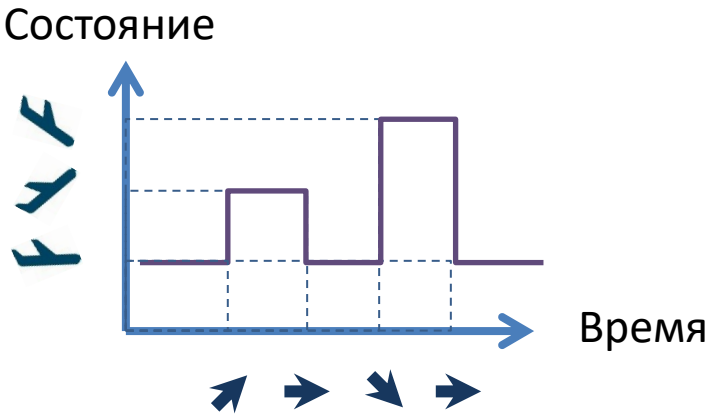
- Максимально быстрое (As-Fast-As-Possible)
  - Продвижения модельного времени могут не соответствовать продвижениям физического времени
- В реальном времени
  - Изменения модельного времени и физического времени строго синхронизированы между собой
- В масштабируемом реальном времени
  - Модельное время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

Дискретно-событийное имитационное моделирование

Исследуемая система



Имитационная модель



Состояние системы – набор переменных

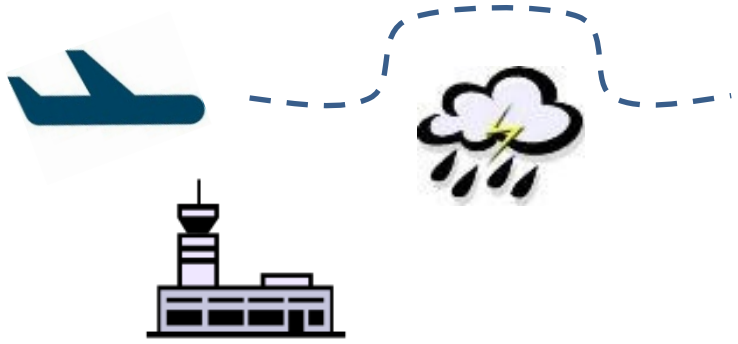
Событие – переход между состояниями системы

События происходят мгновенно в дискретные моменты модельного времени

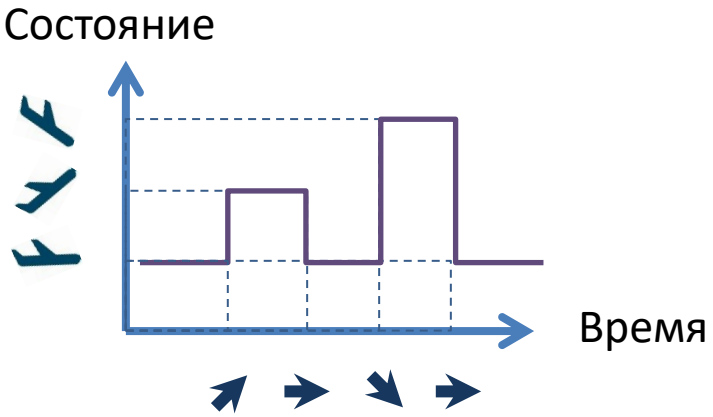
Внутри имитационной программы поддерживается *список событий*

Дискретно-событийное имитационное моделирование

Исследуемая система



Имитационная модель



Обработка события:

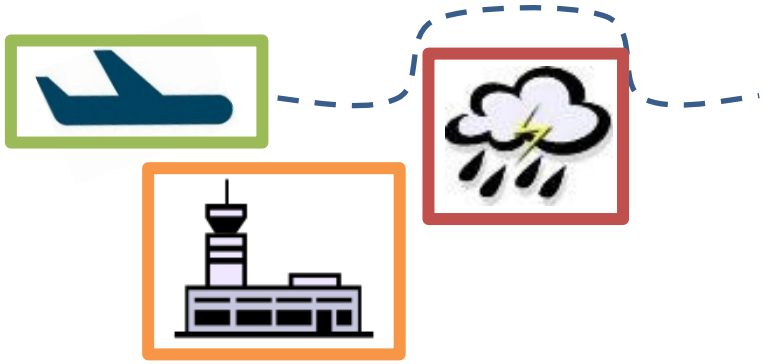
1. Изменение переменных состояния модели
2. Планирование новых событий в будущее
3. Исключение события из списка

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

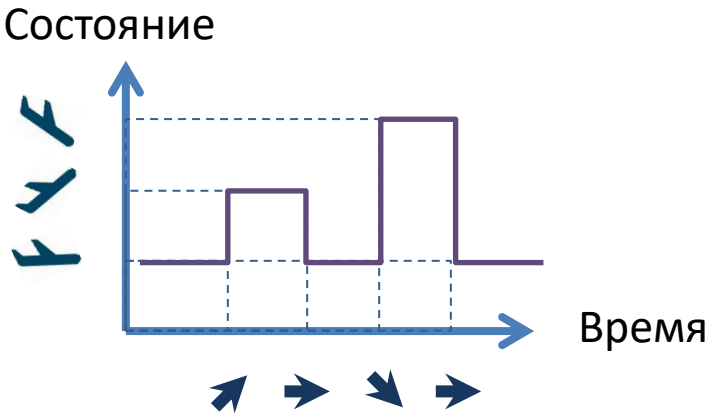
Моделирование завершается, когда список событий пуст

Дискретно-событийное имитационное моделирование

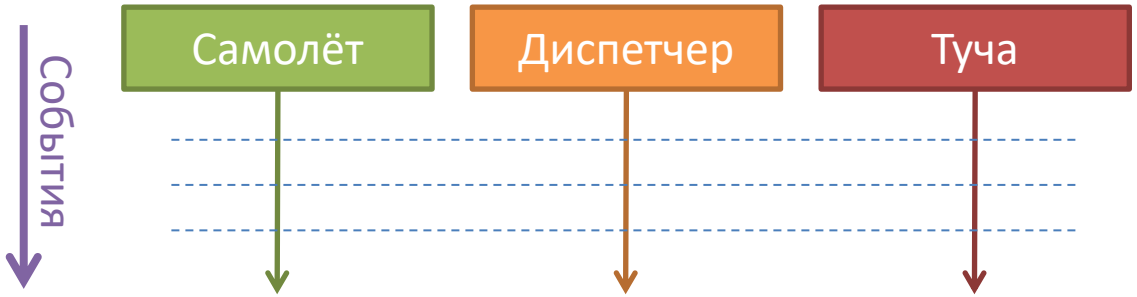
Исследуемая система



Имитационная модель



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



Необходима синхронизация компонентов модели

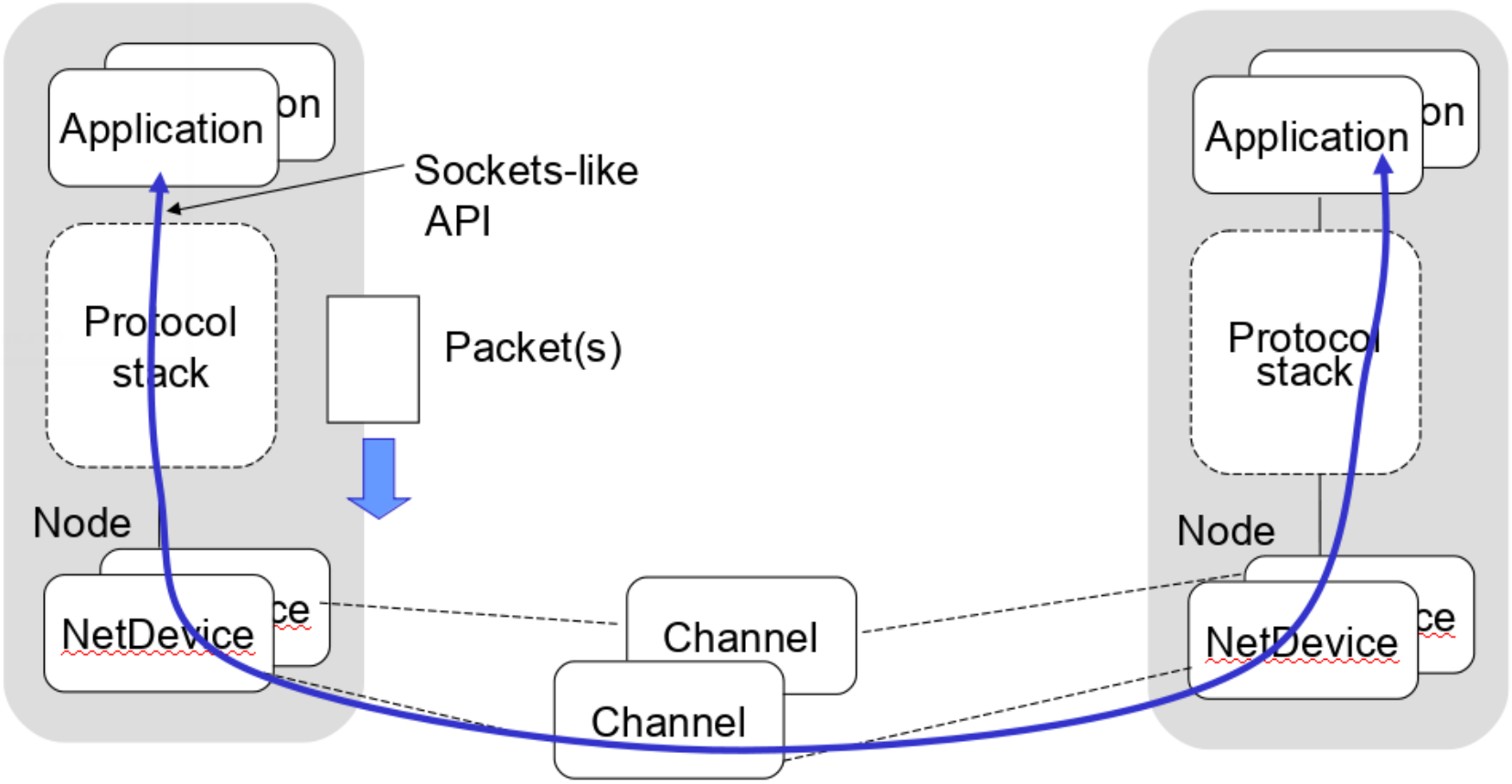
Подходы к синхронизация времени

- 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
5. ***Преобразование сообщений***
   - FEC
   - Сжатие

Модели оценки качества сервиса:

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

Примеры:

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