OMNeT++

Проф. Пламенка Боровска
Катедра Компютърни системи

Общ преглед

- OMNet++ (www.omnetpp.org)
- Скалируема модулна компонент-базирана работна рамка и библиотека за симулации (C++)
- Включва интегрирана развойна среда
- Графична runtime среда
- Функционалност, специфична за областта на мрежите – поддържа симулация на комуникационни мрежи, мрежи от опашки, оценка на комуникационна производителност, и др.
- Тази функционалност е имплементирана посредством модулни работни рамки под формата на независими проекти

Общ преглед

- *Съдържа разширения за*:
- симулации в реално време
- емулация на мрежи (емулаторът имитира поведението на компютърна или друга електронна система с помощта на друг тип компютър/система)
- поддръжка на алтернативни езици за програмиране – C++, Java
- интеграция с бази данни
- SystemC интеграция
- HLA

SystemC

- **SystemC** представлява множество от C++ класове и макроси, които осигуряват симулационно ядро на C++, управлявано от събития (event-driven simulation)
- Тези средства осигуряват възможност на проектанта да симулира паралелни процеси, всеки от които се описва с прост C++ синтаксис
- Процесите на SystemC могат да комуникират в среда на симулирано реално време, използвайки сигнали от всички типове данни, поддържани от C++ и допълнителни такива, предлагани от SystemC библиотека, както и дефинирани от потребителя
  - В някои аспекти SystemC е подобен на езиците за описание на хардуер като VHDL и Verilog, но по-скоро се определя като език за моделиране на системно ниво

SystemC

- SystemC се използва за моделиране на системно ниво, изследване на архитектурите, моделиране на производителност, развитие на софтуер, функционална верификация, и синтез на високо ниво.
- SystemC е дефиниран и промоциран от OSCI, the Open SystemC Initiative, and has been approved by the IEEE Standards Association
- SystemC version 1 включва качества на език за описание на хардуер като структурна йерархия и свързаност, и др.
  - След version 2 фокусът на SystemC се насочва към абстракция на комуникациите – моделиране на ниво транзакция и моделиране на виртуални платформи – добавени са абстрактни портове, динамични процеси и известяване на събития.

HLA – High Level Architecture (simulation)

- Универсална архитектура за разпределени компютърни симулации
- Използвайки HLA, компютърните симулации могат да си взаимодействат независимо от изчислителните платформи (комуникация на данни или синхронизация на дейности)
- Взаимодействието между симулациите се управлява от Run-Time Infrastructure (RTI)

Подход за моделиране при OMNet++

- Повечето мрежови симулатори използват фиксирани начини за представяне на мрежовите компоненти в модела
- OMNet++ използва обща компонент-базирана архитектура, която дава възможност на проектанта на модела да изгради съответствие между концепциите (мрежови устройства, протоколи или безжични канали) от една страна, и компонентите на модела, от друга страна

Подход за моделиране при OMNet++

- Компонентите на модела се наричат модули
- Модулите могат да се използват в различни среди и да бъдат комбинирани по различни начини (като блокове на LEGO)
- Модулите комуникират предимно посредством обмен на съобщения, директно или през предефинирани комуникационни връзки
- Съобщенията могат да представят събития, пакети, команди, или други обекти, в зависимост от контекста на областта

Средства на OMNet++

- C++ ядро и клас библиотека за изграждане на модулите
- Инфраструктура за асемблиране на симулациите от модулите и конфигурирането им (език NED, ini файлове)
- Графичен и batch mode runtime интерфейси
- Симулационна интегрирана среда за развитие – IDE (Integrated Development Environment) за проектиране, изпълнение и оценка на симулациите
- Интерфейси за разширения – симулация в реално време, емулация, MRIP (Multiple Replications In Parallel –stochastic simulation), паралелна разпределена симулация, свързване с база данни, и др.
  - http://www.omnetpp.org

Симулационна интегрирана среда за развитие - IDE (Integrated Development Environment)

- Eclipse – базирана симулационна IDE
- Eclipse (www.eclipse.org) е известна като Java IDE http://www.eclipse.org/downloads/
- По своята същност представлява платформа за интеграция за всички видове приложения
- Базовите проекти на Eclipse предлагат развитие на приложения на C++, развитие на Web приложения и др.
- Портал Eclipse Marketplace http://marketplace.eclipse.org/

и Eclipse Plugins www.eclipse-plugins.2y.net – хиляди plug-ins за широк спектър от приложения – от UML-проектанти до database browsers и езикови среди за развитие

Симулационна интегрирана среда за развитие - IDE

- Персонализирана Eclipse инстанция, която разширява средата с:
- графичен и текстов двупосочен редактор за проектиране на симулационни модели
- Конфигуриращ редактор на симулацията
- C++ build support
- Стартер на симулациите, вкл. за пакетна обработка
- Tool за графично извеждане и анализ на резултатите
- Анализатор на следи (trace analyzer) за визуализация на изпълнението на симулацията на диаграма на последователността (sequence chart)
- Генератор на документация

http://www.omnetpp.org/webdemo/ide/

Работни рамки за мрежова симулация

- INET Framework (www.inet.omnetpp.org) – open source софтуерен пакет за мрежова симулация, съдържащ модели за жични и безжични протоколи (UDP, TCP, IP, IPv6, Ethernet, IEEE 802.11, MPLS, OSPF и др.)
  - Осигурява разширения за емулация като
    - INET порт на Quagga routing daemon (www.quagga.net) и
    - пакета NSC (Network Simulation Cradle) – WAND research group http://research.wand.net.nz/software/nsc.php

Работни рамки за мрежова симулация

- INETMANET е клон на INET Framework за поддръжка на мобилни ad-hoc мрежи, поддържа протоколите AODV, DSR, OLSR, DYMO и др.
- OverSim – симулация на P2P мрежи, базира се на работната рамка на INET
- MiXiM – симулация на безжични и мобилни мрежи, детайлизирани модели на безжични канали (fading, и т.н.), безжично свързване, модели на мобилност и комуникационни протоколи на ниво MAC (Medium Access Control), графично представяне на безжични и мобилни мрежи и сложни сценарии
  - Castalia – симулатор за безжични сензорни мрежи (WSN – Wireless Sensor Networks
    http://castalia.npc.nicta.com.au/

Компонент-базиран OMNeT++ модел

- Съдържа модули, които комуникират посредством обмен на съобщения
- Активните модули се наричат прости модули, имплементирани са на C++ като е използвана клас бибиотеката за симулации
- Простите модули могат да представят потребителски агенти, източници и консуматори на трафик, мрежови устройства като интерфейсни платки IEEE 802.11, структури данни като таблици за маршрутизация, или потребителски агенти, генериращи трафик
- Функции при симулацията като управление на движението на мобилни възли или авто-присвояване на IP адреси в мрежата също използват прости модули

Компонент-базиран OMNeT++ модел

- Групи от модули могат да бъдат капсулирани в рамките на сложни (съставни) модули като нивата на йерархията не са ограничени
- Мрежовите възли като хостове и рутери типично се представят със сложни модули, асемблирани от прости модули
- Допълнителни йерархични нива над нивото на мрежовия възел могат да се използват за представяне на подмрежи или в рамките на мрежовите възли (напр., група прости модули представящи ротоколите IPv6)
- INET Framework съдържа повече от 150 типове модули

Компонент-базиран OMNeT++ модел

- Простите и сложните модули, както и симулираната мрежа са инстанции на *module types*
- Съобщенията, обменяни между модулите, могат да съдържат произволи данни и предефинирани атрибути като напр., етикети за време (timestamp)
- Простите модули си обменят съобщения през портове, наречени gates, но също така могат да изпращат съобщения и директно към модулите-дестинации
- Съществуват 3 типа gates: входни, изходни и входно/изходни (inout)

Компонент–базиран OMNeT++ модел

- Комуникационната връзка (*connection)* свързва входен и изходен гейтове или два inout гейта
- Комуникационните връзки се дефинират като част от сложен (съставен) модул, могат да свързват два подмодула, подмодул с родител, или два гейта на родителя
- Не се допускат комуникационни връзки, пресичащи йерархичните нива
- Поради йерархичната структура на симулационния модел, в общия случай, съобщенията преминават през верига от конекции, така че се изпращат и приемат от прости модули
- Сложните модули се оприличават на “кашони” (cardboard boxes) в модела, като те препредават съобщенията
- На конекциите се присвояват свойства като propagation delay (закъснение при трансфер), data rate (скорост на предаване на данните), bit error rate
- Могат да се дефинират типове връзки със специфични свойства (*termed channels),* които да се използват на няколко места

Компонент-базиран OMNeT++ модел

- Модулите могат да имат параметри
- Параметрите се използват за предаване на конфигурационни данни към простите модули и за подпомагане на дефинирането на топологията на модела
- Променливи параметри *(volatile parameters)* – изчисляват се всеки път, когато се чете тяхната стойност
- Променливите параметри се използват за вкарване на стохастични входни данни в модулите на симулационния модел

The NED Language

- DSL (Domain Specific Language) за описание на OMNeT++ компонент-базирания модел
- *Типични елементи на описанието с NED са декларации на простите модули, дефиниции на сложните модули и дефиниции на мрежата*
- Декларациите на простите модули описват интерфейса на модулите: гейтове и параметри
- Дефинициите на сложните модули включват декларация на външния интерфейс на модула (гейтове и параметри), и дефиниции на подмодулите и техните взаимни връзки

The NED Language

- Дефинициите на мрежата се окачествяват като самостоятелни симулационни модели
- Ограничени програмни конструктори като loop, conditional, осигуряват възможност за създаване на параметризирани топологии като напр., рутер с неограничен брой портове, или хексагонална решетка с параметризирани измерения
- Поддържат се анотации с метаданни за типове, параметри, подмодули, конекции, и др., които се използват за съхраняване на графични атрибути (позиция, икона, и др.), за маркиране на гейтове, за които се очаква да не бъдат осъществявни връзки, за означаване на сложни модули, представящи физически мрежови възли в работната рамка INET, и др.

Програмиране

- Простите модули се имплементират като C++ класове (*cSimple Module library class*)
- Съобщенията се представят със *cMessage class*, и могат да бъдат изпращани през изходни гейтове или директно към други модули
- Симулационното ядро доставя съобщенията на модула – handleMessage(cMessage*)
- Алтернатива – blocking receive calls
- Не се препоръчва използването на activity() – не се скалира

Програмиране

- Таймери и timeouts – *self-messages* – нормални съобщения, които модулът изпраща сам на себе си, изпращат се със schedule call и се връщат на изпращащия модул по същия начин като съобщенията от другите модули
- *Self-messages* също могат да бъдат канселирани
- За целта не съществува отделен клас събития, като ролята се изпълнява от *cMessage*

Инициализация и финализиране

- Програмистът може да прибави код за инициализация и финализиране като отмени (override) съответните методи на класа модули
- OMNeT++ поддържа многостепенна инициализация, което е от особено значение за големи модели като INET Framework
  - До финализиране се стига само при успешно завършване (терминиране) на симулацията и финализиращият код се използва най-често за регистрация на резултатите от симулацията

Симулационно време

- OMNeT++ използва цяло 64-битово число с фиксирана точка и експонента с основа 10
- Експонентата се съхранява в глобална променлива за да се елиминира необходимостта от нормализация и за икономия на паметта
- Осигуряваният обхват е приблизително 292 години при точност наносекунди, и 107 дни при точност пикосекунди

Library classes

- Повечето класове в симулационната библиотека на OMNeT++ представят различни части на компонент-базирания модел: модули, канали, гейтове, параметри на модулите, обекти, и т.н.
- Съобщенията и пакетите се представят от класа *cMessage* и неговите подкласове *cPacket*
- Често използван е контейнер класа cQueue, който може да изпълнява ролята на приоритетна опашка

Library classes

- Библиотеката съдържа клас за откриване на топология (*topology discovery class*), който може да извлече мрежовата топология от модела в съответствие със спецификацията на потребителя
- Топологията може да се представи под формата на граф
- Поддържа алгоритми като напр., алгоритъма на Дийкстра за намиране на най-кратък път в графа

Library classes

- Случайните числа се осигуряват от потоци, генерирани от архитектурата за случайни числа на симулационната работна рамка (*random number architecture*)
- Поддържат се множество случайни разпределения
- Непрекъснати разпределения – равномерно, експоненциално, нормално, гама, бета, Erlang, chi-square, Student-t, Cauchy, triangular, lognormal, Weibull, Pareto
- Дискретни разпределения – равномерно, Бернули, биномно, геометрично, negative binomial, Poisson

Проследяване на собственост Ownership tracking

- Инстанции на няколко класове в клас библиотеката на OMNeT++ (най-вече *cMessage*) поддържат указатели обратно към техните собственици
- Собственик е модулът, който е създал конкретното съобщение, опашка или друг контейнер обект в модула, или симулационното ядро (по-точно FEL – списъкът на предстоящите събития)
- Указателя към собственика дава възможност на симулационното ядро да открие грешки като изпращане на едно и също съобщение два пъти, изпращане на съобщение, което още е в опашката, или достъп до съобщение, което се държи от друг модул

Управление на собствеността
Ownership management

- Механизмът за управление на собствеността е прозрачен през по-голямата част от времето
- Когато модул предава обекта на съобщението на друг модул като използва C++ method call – целевият модул (target module) трябва явно да вземе обекта от настоящия му собственик
- Модулите са *soft owners* и изпълняват такива заявки
- Опашката (queue) е *hard owner* и в отговор на такава заявка ще генерира съобщение за грешка

Управление на собствеността Ownership management

- Модулите поддържат списък на собствените обекти
- Възможно е рекурсивно изброяване на всички обекти при симулацията по общ начин без да се използват полета за указателите, декларирани в подкласовете на простите модули
- Този механизъм дава възможност на потребителя да инспектира симулацията в графична runtime среда на ниво обекти и да открива пропуснати обекти

Представяне на мрежовите пакети

- Важен аспект на мрежовата симулация
- В OMNeT++ пакетите са C++ класове (от *cPacket*, който е подклас на *cMessage*)
- Полетата на *cPacket* включват: дължина на пакета, флаг за грешка индициращ повреден пакет, и указател към капсулирания пакет
- Полетата с указателя към капсулирания пакет се използват от методите *encapsulate()* и *decapsulate()* когато съобщението се предава нагоре или надолу по слоевете на протокола
- Тези методи автоматично актуализират дължината на външния пакет
- Указателя на капсулирания пакет дава възможност на OMNeT++ да редуцира броя на дубликатите на обекта на пакета чрез отчитане на референциите и копиране при достъп (copy-on-access) на капсулирания обект

Компилатор на съобщенията Message compiler

- OMNeT++ осигурява прост език за описание на съобщенията
- Build системата автоматично създава C++ класове от съобщенията в процеса на изграждане на модела
- Общи класове и structs могат да бъдат създавани аналогично

Управляваща информация Control info

- В OMNeT++ слоевете на протокола обикновено се имплементират като модули, които си обменят съобщения
- Комуникацията между слоевете на протокола обикновено изисква изпращането на допълнителна информация, която трябва да се прикрепи към пакетите
- Напр., когато TCP имплементация изпраща надолу TCP пакет към IP, е необходимо да се специфицира IP адреса на дестинацията или когато IP изпраща нагоре пакет към TCP след декапсулиране на IP дейтаграмата, е необходимо да се добавят IP адресите на сорса и дестинацията
- Тази допълнителна инфо се представя с обектите с управляваща информация (control info objects), които се прикрепят към пакетите

Предаване на пакети
Wired packet transmission

- Пакетите се изпращат от предавателя на един мрежов възел към приемника на друг възел посредством комуникационен път, съдържащ точно един обект “канал”
- Аналогично на модулите, каналите се програмират на C++
- Каналите дават възможност за моделирането на закъснението при трансфер, изчисляване и моделиране на времетраенето на предаването на съобщението, и моделирането на грешките
- Default модел на канал DatarateChannel

Предаване на пакети
Wired packet transmission

- В общия случай, обекта на пакета се предава на модула приемник в симулационно време, съответстващо на края на приемането на пакета
- Модулът на приемника, обаче, може да изиска пакетите да му се предават в началото на тяхното приемане като се препрограмира гейта на приемника с подходящ API call
- Времетраенето на последното предаване е регистрирано в отделно поле на обекта на пакета и може да се използва от приемника да определи колко време (моделни единици) каналът ще бъде зает

Сигнали

- Симулационната библиотека на OMNeT++ съдържа вграден механизъм за известяване (built-in notification mechanism), който осигурява комуникация между симулационните компоненти в стила “публикуване-абониране (publish-subscribe)
- Сигналите се генерират от компонентите (модули и канали) и се предават нагоре по йерархията от модули към корена
- На всяко ниво могат да бъдат регистрирани слушатели (listeners – callback objects), които ще бъдат известени (called back) при издаването на сигнал

Сигнали

- Слушателите, регистрирани на нивото на даден модул, ще получават сигнали от всички компоненти на дървото на подмодулите в рамките на този модул
- Слушателите, регистрирани на топ нивото ще получават сигнали от цялата симулация
- Сигналите се идентифицират посредством имена, но за ефективност, извикванията (calls) използват динамично присвоявани числови идентификатори на сигналите
- Имената и идентификаторите са валидни глобално през цялото време на симулацията

Сигналите се използват за:

- Имплементиране на комуникация между модулите в стила публикуване-абониране, предимство – източникът и консуматорът на информацията не знаят нищо един за друг
- Известяване на модулите относно промени в симулационния модел като напр., създаване или изтриване на модул, създаване или изтриване на връзка, промяна на параметрите и т.н.; тези сигнали се генерират от симулационното ядро

Сигналите се използват за:

- Предаване на променливи, които трябва да бъдат записани като резултати от симулацията, като напр., дължини на опашките, пропускане на пакети, закъснения “от край до край” (end-to-end) – работната рамка на симулатора прибавя слушатели, които записват селектираните данни в специфицирана форма
- Предаване на примитиви за анимация или допълнителна информация за анимация
- Предаване на pcap следи (traces), които могат да бъдат уловени и записани във файл от специални модули или от симулационната рамка

Архитектура за генериране на случайни числа

- OMNeT++ използва предимно генератора на псевдослучайни числа Mersenne Twister (default), създаден през 1997 г. от Makoto Matsumoto и Takuji Nishimura, базиран на матрично линейна рекурентност върху крайно двоично поле
- Бърза генерация на високо качествени псевдо случайни числа
- Практически неограничен период на повторение
- Името произлиза от факта, че дължината на периода се избира да бъде просто число по Mersenne
- Marin Mersenne е френски монах, започнал да изучава свойствата на тези числа в началото на 17-ти век

Mersenne Twister

- Число на Мерсен се нарича положително цяло число, за което е изпълнено $M_p = 2^p - 1$
- Просто число по Мерсен, е число на Мерсен, което е просто т.е. се дели само на себе си и на 1
- До октомври 2009г. са били известни само 47 прости числа на Мерсен
- Най-голямото известно просто число ($2^{43,112,609}-1$) е просто число на Мерсен
- След 1997г. всички новооткрити прости числа по Мерсен са открити в рамките на проекта "Great Internet Mersenne Prime Search" (GIMPS)

Архитектура за генериране на случайни числа

- Осигурени са множество глобални потоци от случайни числа за симулацията
- Глобалните потоци се трансформират в локални потоци от случайни числа за модулите
- Трансформирането на глобалните потоци в локални може да бъде конфигурирано по гъвкав начин
- Автоматичен seeding, но е възможно използването на ръчно въведени seeds
- Освен default генератора на случайни числа RNG (Random Number Generator) Mersenne Twister, OMNeT++ осигурява и други два RNG: LCG-32 и случаен генератор с опаковане на случайни числа от Akaroa library