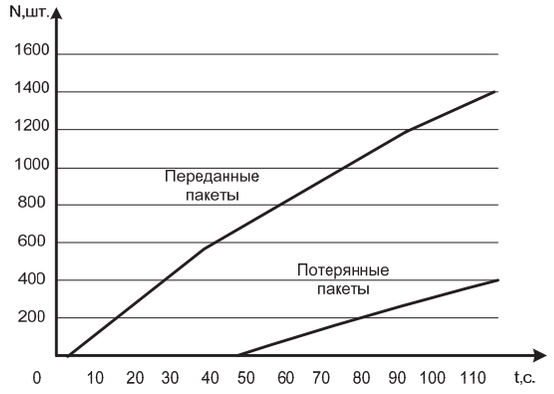

Рис. 5. Результаты исследования работы коммутатора в стационарном режиме с применение сервиса RED (эксперимент №3)

Выводы

Таким образом, результаты исследований доказали адекватность модели и позволили выявить диапазоны параметров нагрузки, при которых коммутатор работает в стационарном режиме и в условиях пиковой нагрузки.

Последняя возникает, если время коммутации в два и более раз превышает интервал между моментами поступления пакетов. При этом количество потерянных пакетов возрастает лавинообразно и коммутатор практически теряет работоспособность.

Достоинством предлагаемого авторами подхода является возможность анализа наиболее важных факторов функционирования устройства и исключения влияния второстепенных.

Литература

1. Олифер В.Г., Олифер Н.А. Компьютерные сети. Принципы, технологии, протоколы: СПб.: Питер, 2008. – 958 с.
2. Горелов Г.В. Ромашкова О.Н. Оценка качества обслуживания в сетях с пакетной передачей речи и данных. СПб.: ИТМО, 2003. – 45 с.
3. Тютин В.А. Проблемы создания средств проектирования телекоммуникационных сетей. М.: Наука, 2005. – 130с.
4. Башарин Г.П., Бочаров П.П., Коган Я.А. Анализ очередей в вычислительных сетях. Теория и методы расчета. М.: Наука, 2001. – 453с.
4. Ефимушкина Н.В., Миронов А.А. Модели вычислительных систем с телекоммуникационным доступом // Труды 7 РНПК «Компьютерные технологии в науке, практике и образовании». Самара, СГТУ, 2008. – С. 216-219.

RESEARCH OF THE BASIC CHARACTERISTICS OF MODERN NETWORK SWITCHS

Voronzov I.V., Efimushkina N.V.

The basic structures and principles of work of the modern switchs used in computing systems with telecommunication access and networks, and principles of construction of their models are considered. By means of imitating models the analysis of the most important characteristics of these devices is made and effective modes of their work are defined.

Keywords: *imitating model, switch, a package, a shot, the discipline of service, entering and leaving turn.*

Воронцов Игорь Васильевич, доцент Кафедры вычислительной техники (ВТ) Самарского государственного технического университета (СГТУ). Тел. (8-846) 337-12-86; 8 -927-685-74-15. E-mail: vt@vt.samgtu.ru

Ефимушкина Наталья Владимировна, к.т.н., доцент, заместитель заведующего Кафедрой ВТ СГТУ. Тел. (8-846) 337-12-86; 8-904-730-56-73. E-mail: efimushkina@vt.samgtu.ru

УДК 004.4, 621.391

АНАЛИЗ СОВРЕМЕННЫХ ТЕХНОЛОГИЙ ТРАНСЛЯЦИИ И ПАРАДИГМ ПРОГРАММИРОВАНИЯ И ИХ ПРИМЕНИМОСТИ К ЗАДАЧЕ РАЗРАБОТКИ ФОРМАЛЬНОГО ЯЗЫКА ОПИСАНИЯ ПРОТОКОЛОВ МАРШРУТИЗАЦИИ

Милюткин М.Г., Полукаров Д.Ю.

Поставлена задача систематизации современных парадигм программирования. Проведен анализ проблемы формального описания протоколов маршрутизации. Рассмотрена возможность применения декларативной парадигмы программирования для решения задачи маршрутизации.

Ключевые слова: парадигмы программирования, языки программирования, протокол маршрутизации.

Введение

Современное программное обеспечение – начиная от простых любительских программ и заканчивая крупными промышленными enterprise-системами – создается с помощью известных стандартизированных методик и языков программирования.

Компьютер (вычислительное устройство) представляет собой микропроцессор, в сочетании с сопутствующими устройствами способный выполнять ограниченный набор команд. Современные языки программирования являются многоуровневыми и сложными. Для перевода высокоуровневых программ в команды процессора существуют различные технологии трансляции (компиляторы, интерпретаторы).

Создание языка программирования представляет собой нетривиальную задачу, так как требует создания соответствующего транслятора.

Многообразие языков программирования

Задачи, которые приходится решать с помощью компьютерных программ, многообразны по своим предметным областям, и для их решения требуются разные методики и подходы, что нашло свое непосредственное отражение в конкретных языках программирования. Каждый язык программирования основывается на одной или нескольких таких методиках. Эти методики (подходы) также называются парадигмами программирования. Разные парадигмы программирования закладывали основу в разные языки программирования, каждый из которых создавался для решения своего типа задач.

В данной работе анализируются и систематизируются современные технологии трансляции и парадигмы программирования, а также делается анализ их применимости к решению задачи разработки формального языка описания протоколов маршрутизации.

Транслятор – это компьютерная программа, которая осуществляет перевод исходного кода исследуемой компьютерной программы с одного языка программирования на другой язык программирования [1]. Есть множество примеров таких общих трансляторов: FORTRAN-to-Ada, CHILL-to-C++, PASCAL-to-C, COBOL(DialectA)-to-COBOL(DialectB) и т.п. В науке о вычислительной технике и языках программирования принято классифицировать трансляторы на конкретные типы по их внутренней функциональности [2] – см. таблицу 1.

Таблица 1. Классификация трансляторов по их внутренней функциональности

Компилятор	Транслятор, который переводит исходный код программы, написанной на некотором высокоуровневом языке программирования, на язык ассемблера или машинный язык процессора (заметим, что любой язык программирования можно транслировать либо в другой высокоуровневый язык, обладающий свойством полноты по Тьюрингу, либо в язык ассемблера).
Интерпретатор	Транслятор, который переводит язык высокого уровня в некий промежуточный язык, команды которого подлежат немедленному пошаговому выполнению.
Декомпилятор	Транслятор, который осуществляет обратный процесс - транслирует целевой или машинный код в исходный код некоторого языка.
Ассемблер	Транслятор, который переводит язык ассемблера в машинный код.
Дизассемблер	Транслятор, который осуществляет обратный процесс – перевод машинного кода в язык ассемблера.

Исторически технологии трансляции развивались одновременно с парадигмами программирования, таким образом, одно является продолжением другого. С ростом объемов исходного кода начали появляться проблемы с поддержанием работы существующей функциональности и добавлением новой. Назревала потребность в переосмыслении философии, базовых фундаментальных подходов к программированию. По этой причине стали появляться парадигмы программирования, на базе которых стали строиться новые языки. У каждого языка были свои трансляторы, с присущими только им особенностями. Методики построения трансляторов различаются по своей реализации для каждой конкретной парадигмы.

Парадигма программирования (подход к программированию) в самом общем смысле – это фундаментальный стиль программирования вычислительных систем. На сегодняшний день существует множество различных парадигм [3], однако среди них можно выделить фундамен-

тальные парадигмы, являющиеся базовыми для всех остальных, основанные каждая на своей математической теории. Фундаментальные парадигмы представлены в таблице 2.

Таблица 2. Фундаментальные парадигмы

Парадигма	Теоретическое основание
Объектно-ориентированное программирование	Машина Тьюринга
Императивное программирование	
Функциональное программирование	Лямбда-исчисления
Логическое программирование	Исчисление предикатов (логика первого порядка)

Модель программирования – это абстракция некоторой вычислительной системы. Одна из известных моделей – модель Фон-Неймана [4] используется для описания традиционных последовательных компьютеров. Для параллельных вычислений существует множество своих моделей, обычно отражающих разные способы взаимодействия между собой процессоров, например, модель общей памяти, модель распределенной памяти с передачей сообщений и др. [5]. В [6] проведен анализ парадигм программирования и их зависимости между собой. Была предложена наглядная диаграмма, отражающая взаимосвязь парадигм и показывающая причинно-следственные связи (см. рис. 1). Один язык программирования может поддерживать множество парадигм одновременно. Например, программы, написанные на языках C++ и Object Pascal, могут быть как чисто процедурными [3], так и чисто объектно-ориентированным или содержать смесь двух парадигм.

Аналогично тому как на практике используются различные методологии [7], так и языки программирования используют разные парадигмы. Некоторые языки созданы для поддержки только одной конкретной парадигмы (Smalltalk – ООП, Haskell – функциональное), в то время как другие поддерживают множество парадигм (C++, C# и др.).

Есть парадигмы, которые известны больше теми методиками, которые они запрещают, чем теми, которые они используют. Например, функциональное программирование запрещает так называемые побочные эффекты (говорят, что функция или выражение имеет побочный эффект, если вдобавок к тому, что оно возвращает некое значение, также модифицируется некое состояние или есть очевидное взаимодействие с вызывающей функцией или внешним миром) [8], а структурное программирование запрещает использование оператора goto. Запрет некоторых техник позволяет проще доказывать теоремы о корректности программ и быстрее понимать их суть.

Самой древней и низкоуровневой парадигмой программирования является программирование в машинных кодах – прямое представление инструкций для процессора (содержимого ячеек памяти) в виде последовательности цифр. Близка к ней и парадигма программирования на языке ассемблера – инструкции для процессора выражены мнемоническими аббревиатурами и адресам памяти можно давать символьные обозначения. Такие языки программирования принято называть языками программирования первого и второго поколения. Ассемблер используется до сих пор для систем критичных ко времени выполнения и встроенных систем (микроконтроллеры, микропроцессоры, обрабатывающие цифровые сигналы и т.п.), так как дает прямой контроль над фактическими действиями машины. Ассемблер очень полезен для понимания того, как на самом деле работает компьютер и что происходит в памяти, особенно на современном этапе развития программирования, когда стали популярны высокоуровневые многослойные фреймворки. Любой язык программирования, технология, фреймворк – все в конечном итоге есть машинный код в памяти компьютера, и понимание этого позволяет прояснить многие скрытые от программиста особенности.

Следующей парадигмой стал процедурный подход. Такие языки программирования стали называть языки третьего поколения, их также впервые стали называть языками высокого уровня. Эти языки уже стали использовать для решения задач команды, которые являются словами естественного языка. Известными представителями этой парадигмы являются C, COBOL, FORTRAN, ALGOL, PL/I, BASIC.

В связи с увеличением сложности программного обеспечения и разнообразием предметных областей решаемых задач процедурная парадигма логично переросла в объектно-ориентированную. Объектно-ориентированное программирование является наиболее популярной на сегодняшний день парадигмой программирования.

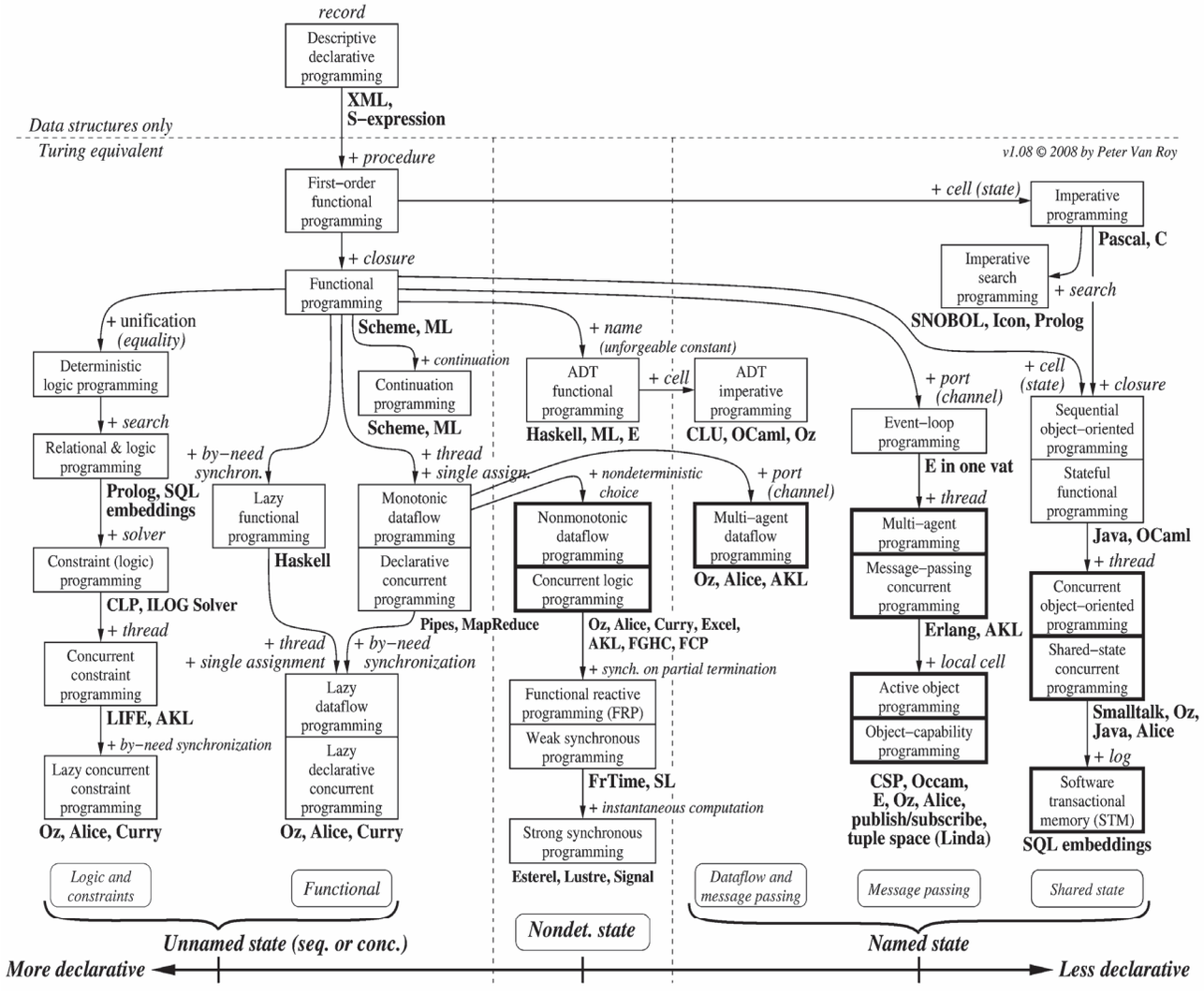

Рис. 1. Диаграмма парадигм программирования [6]

Заметим, что объектно-ориентированный подход является именно парадигмой, а не конкретным языком, поэтому возможно создать даже объектно-ориентированный язык ассемблера (примером этому является – High Level Assembly). Высокоуровневые процедурные и объектно-ориентированные языки принято называть языками третьего поколения.

Процедурный подход, объектно-ориентированный подход и их разновидности (см. рис. 1) имеют одну общую идею – программа (то есть вычисление) описывается в терминах явных инструкций (steatements), меняющих состояние (state) этой программы. Чтобы выполнить некое действие, мы описываем четким набором команд, как надо выполнить это действие. То есть в таких парадигмах программист описывает, как надо сделать задачу. Такой подход (идею) в самом общем смысле принято называть императивной парадигмой программирования.

Независимо и параллельно императивному подходу развивался с течением времени идейно противоположный подход – декларативная парадигма. В декларативном подходе программист должен указать, что надо сделать, не указывая при этом, как это надо сделать (то, как будет реализована команда, полностью зависит от внутренней реализации транслятора такого языка). Компьютеру даются инструкции о том, какую проблему надо решить, а не как ее надо решить. Программа на декларативном языке – это не последовательность действий, которой надо строго следовать, это скорее набор свойств, которые надо обнаружить в ожидаемом результате. На вход подается набор неких условий, а программа пытается найти решение, соблюдающее эти условия. Ярким примером декларативного языка является язык SQL, который относится уже к языкам четвертого поколения. Разновидностями декларативной парадигмы являются упомянутые фундаменталь-

ные функциональная и логическая парадигма [3] (также см. рис. 1).

Задача формального описания протоколов маршрутизации

В настоящее время маршрутизаторы производятся с аппаратно-фиксированным количеством протоколов маршрутизации. Для решения задач IP-маршрутизации используется только статическая маршрутизация вместе со стандартизированными динамическими протоколами маршрутизации (RIP, OSPF, IS-IS, BGP) или коммерческим протоколом Cisco Systems EIGRP. Хотя BGP изначально был разработан как междоменный протокол маршрутизации, в настоящее время он используется в больших корпорациях в качестве внутридоменного протокола маршрутизации (не путать с использованием внутреннего BGP – IBGP, являющимся основным компонентом междоменной маршрутизации).

BGP используется в качестве протокола внутренней маршрутизации потому, что он доступен, имеет ярко выраженные механизмы контроля политик, может быть использован для реализации административных границ, и, что не менее важно, он хорошо масштабируется.

Однако BGP не имеет гарантий сходимости. Более того, когда BGP используется в качестве протокола внутренней маршрутизации, политики маршрутизации имеют меньше ограничений по сравнению с внутридоменной маршрутизацией, где стандартные политика типа «отдавать приоритет сначала маршрутам заказчика, затем маршрутам узлов и только затем маршрутам провайдеров» обеспечивают как минимум частичную защиту от расхождения протоколов [9].

Другим способом решения сложившейся проблемы может быть повышение гибкости протокола внутренней маршрутизации [10]. Для повышения гибкости протокола маршрутизации можно предложить возможность программно его менять. Это можно реализовать несколькими способами.

Первый способ заключается в том, чтобы выделить элементарные операции маршрутизатора и создать поверх множества элементарных операций библиотеку на одном из высокоуровневых языков – для такой системной задачи подходит язык C. В дальнейшем для программирования на каком-либо языке непосредственно логики протокола можно использовать эту библиотеку. В итоге имеем высокую гибкость настройки протоколов маршрутизации. Недостаток данного способа в том, что здесь необходимо иметь знания языков программирования C/C++ то есть этот подход пригоден скорее для программиста, чем для сетевого инженера.

Второй способ заключается в использовании различных сценариев. С помощью языка сценария описываются параметры протокола маршрутизации. Затем этот сценарий обрабатывается с помощью библиотеки, создающей непосредственно логику протокола маршрутизации, как в предыдущем случае. Данный способ более удобен практикам, которые не являются специалистами в области программирования.

Третий способ предполагает создание собственного языка программирования, ориентированного на протоколы маршрутизации. Создание языка программирования в первую очередь предполагает создание соответствующего транслятора.

Таким образом, возможно определить протокол маршрутизации в декларативном высокоуровневом стиле. В данном случае необходимо лишь реализовать на маршрутизаторе транслятор для декларативного языка маршрутизации, чтобы запускать в действие таким образом заданный протокол. Это позволит сетевому инженеру определять новый протокол маршрутизации самостоятельно и затем использовать его. Транслятор будут преобразовывать код такого языка напрямую в машинный код микропроцессора, под управлением которого работает маршрутизатор. Такой подход даст максимальную гибкость в управлении маршрутизатором, однако он является наиболее затратным с точки зрения практического воплощения.

В рамках данного способа более предпочтительна декларативная парадигма, так как это освобождает сетевого инженера от необходимости прорабатывать детали выполнения нужной операции.

Заключение

В статье проанализированы современные технологии трансляции и парадигмы программирования. Выяснено, что технологии создания языков программирования и их трансляторов зависят от той парадигмы, которую планируется реализовать в этом языке. Также было показано, что несмотря на многообразие существующих парадигм [3] программирования, их можно свести в упорядоченную иерархию (см. рис. 1) и разделить на два различных по смыслу подхода – императивный и декларативный.

Кратко рассмотрены некоторые общие проблемы технологий маршрутизации в компьютерных сетях. Решение этих проблем требует введе-

ния возможности программирования протоколов маршрутизации. Рассмотрены способы введения этой возможности. Была рассмотрена применимость декларативной парадигмы программирования к решению проблемы гибкости протоколов маршрутизации.

Литература

1. Таненбаум Э. Архитектура компьютера, 5-е изд. - СПб.: Питер, 2007. – 844 с.
2. Translator (computing) URL: http://en.wikipedia.org/wiki/Translator_(computing) – 26.12.2012.
3. Programming paradigm URL:http://en.wikipedia.org/wiki/Programming_paradigm – 26.12.2012.
4. First draft of a report on the EDVAC : Von Neumann, John URL:web.mit.edu/sts.035/www/PDFs /edvac.pdf – 26.12.2012.
5. Гергель В.П., Фурсов В.А. Лекции по параллельным вычислениям. Самара: Изд. СГАУ, 2009. – 164 с.
6. Concepts, Techniques, and Models of Computer Programming:Textbook and Reference Work. Peter Van Roy & Seif Haridi, MIT Press, 2004. – 900 p.
7. Гамма Э., Хелм Р., Джонсон Р. и др. Приемы объектно-ориентированного проектирования. Паттерны проектирования. Пер. с англ. СПб.: Питер, 2007. – 366 с.
8. Research Topics in Functional Programming ed. Turner D. Addison-Wesley, 1990. – 374 p.
9. Lixin Gao, Rexford J. Stable internet routing without global coordination // IEEE/ACM Transactions on Networking. December, 2001. – P. 681-692.
10. Полукаров Д.Ю. Нечеткая аппроксимация метрики протокола IGRP // ИКТ. Т.4, №4, 2006. – С. 51-54.

ANALYSIS OF MODERN TRANSLATION TECHNOLOGIES, PROGRAMMING PARADIGMS AND THEIR APPLICABILITY TO THE TASK OF DEVELOPING A FORMAL LANGUAGE DESCRIPTION OF ROUTING PROTOCOLS

Milutkin M.G., Polukarov D.Y.

The task of programming paradigms systematization was given. Analysis of the problem of formal description of routing protocols has been made. The applicability of declarative programming paradigm for solving the routing has been considered.

Keywords: *programming paradigms, programming languages, routing protocol.*

Милюткин Михаил Григорьевич, аспирант Кафедры информационных систем и технологий (ИСТ) Поволжского государственного университета телекоммуникаций и информатики (ПГУТИ). Тел. 8-927-906-75-46. E-mail: m.milutkin@gmail.com

Полукаров Данил Юрьевич, к.т.н., доцент Кафедры ИСТ ПГУТИ. Тел. (8-846) 228-00-21. E-mail: plkw@mail.ru

НОВЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

УДК 004.021

КОМПЛЕКСНЫЙ АНАЛИЗ И ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ ЗАГРЯЗНЕНИЙ АТМОСФЕРНОГО ВОЗДУХА ГОРОДА САМАРЫ

Гаврилова А.А., Иванова Д.В., Салов А.Г., Саксонова В.В.

В статье предложено системное оценивание эффективности природоохранных мероприятий методами математического моделирования. Проведено имитационное моделирование и комплексный анализ эффективности финансовых вложений, обеспечивающих снижение уровня загрязнения окружающей среды на примере крупнейшего промышленного региона России – города Самары.

Ключевые слова: системный анализ, выбросы, математическая модель, производственная функция, имитационная модель, показатели эффективности, валовый региональный продукт, природоохранные мероприятия.

Введение

Одним из важнейших условий, обеспечивающих разработку эффективных мероприятий по совершенствованию управления социально-экономическими системами, является разработка ме-