Содержание

Об авторах 11
О технических рецензентах 12
Посвящения 13
Благодарности 14
Условные обозначения 16

Предисловие 17

Введение 19
Промышленные тенденции 19
Что такое приложение? 20
Необходимость абстракции 21
Что такое Cisco ACI? 22
Новшества Cisco ACI 23
Структура книги 24
Терминология 25

Глава 1. Основы архитектуры центров обработки данных 27
Приложение и системы хранения данных 27
Виртуализированный центр обработки данных 28
Введение 28
Большие данные 34
Высокопроизводительные вычисления 41
Сверхмалое время задержки 44
Высокомасштабируемый центр обработки данных 49
Примеры топологий 53
Структуры, основанные на концепции POD 54
Модель POD или модель данных для совместно используемой инфраструктуры и облачных вычислений 54
Архитектура FlexPod 57
Проектирование центров обработки данных 57
Конец ряда 58
Логическая структура центра обработки данных с архитектурой Spine-Leaf ACI Foundation 62
Резюме 65

Глава 2. Структурные элементы облачной архитектуры 67
Введение в облачные структуры 67
Сетевые требования к облакам и решение ACI 69
Модель Amazon Web Services 71
Автоматизация настройки сервера 73
PXE-загрузка 74

Развертывание операционной системы с программами Chef, Puppet, CFengine и аналогичными инструментами 74
Оркестрация инфраструктуры как услуги 77
Платформа OpenStack 78
Платформа UCS Director 82
Платформа Cisco Intelligent Automation for Cloud 83
Согласование разных моделей абстракции 85
Резюме 87

Глава 3. Центры обработки данных, управляемые политиками 89
Зачем нужна модель, управляемая политиками? 89
Теория политик 91
Объектная модель политики Cisco APIC 93
Применение политики Cisco APIC 98
Принципы работы контроллера Cisco APIC 112
Операционная система Cisco ACI (Cisco ACI Fabric OS) 112
Структура: компоненты и функция контроллера Cisco APIC 112
Администратор политики 113
Администратор топологии 113
Система мониторинга (Observer) 114
Администратор загрузок 114
Администратор устройств (контроллер кластера) 115
Администратор VMM 116
Администратор событий 116
Администратор сервера APIC 116
Структура: управление данными с помощью сегментирования 116
Пользовательский интерфейс: рафический пользовательский интерфейс 119
Пользовательский интерфейс: командная строка 120
Пользовательский интерфейс: RESTful API 120
Доступ к системе: аутентификация, авторизация и RBAC 121
Резюме 122

Глава 4. Операционная модель 123
Введение в основные технологические и инструментальные средства центров обработки данных 124
Введение в основные технологические... 124
Возможности сетевого управления 124
Протокол REST 125
Форматы XML, JSON и YAML 126
Язык Python 128
Система Git и хранилище GitHub 135
Операции с контроллером Cisco APIC 139
Дерево объектов 141
Применение интерфейса REST для программирования сети 146
Комплект ACI SDK 154

Дополнительные сведения 157
Резюме 157

Глава 5. Проектирование центров обработки данных, использующих гипервизоры 159

Организация сетей на основе виртуализированных серверов 160
Назначение программных компонентов коммутации на сервере 162
Краткий обзор сетевых компонентов 164
Перенос активных виртуальных машин 166
Варианты сегментации 167
Обычные виртуальные локальные сети (VLANs) 167
Расширяемые виртуальные локальные сети (VXLANs) 167
Организация сети с использованием гипервизора Microsoft Hyper-V 171
Организация сети с использованием гипервизора Linux KVM 174
Мостовое соединение в Linux 176
Коммутатор Open vSwitch 177
Организация сети с использованием гипервизора VMware ESX/ESXi 183
Коммутаторы VMware vSwitch и Distributed Virtual Switch 184
Требования к сетевому трафику на сервере VMware ESXi Server 186
Компоненты vCloud Director и vApps 187
Коммутатор Cisco Nexus 1000V 189
Расширение портов с помощью маркера VN-TAG 193
Моделирование связи с виртуальным сервером средствами инфраструктуры Cisco ACI 195
Нормализация оверлейных сетей 196
Домен VMM 196
Обнаружение конечных устройств 197
Немедленное применение политик 198
Интеграция инфраструктуры Cisco ACI с гипервизором Hyper-V 198
Интеграция инфраструктуры Cisco ACI с гипервизором KVM 199
Интеграция инфраструктуры Cisco ACI с гипервизором VMware ESX 200
Резюме 201

Глава 6. Платформа OpenStack 203

Назначение платформы OpenStack 203
Nova 204
Neutron 205
Swift 209
Cinder 209
Horizon 210
Heat 210
Ironic 211
Развертывание платформы OpenStack в масштабах предприятия 212
Преимущества совместно применения инфраструктуры Cisco ACI и платформы OpenStack 214
Модель политик инфраструктуры Cisco ACI 215

Интеграция физических и виртуальных сетей 215
Туннели внутри фабрики 216
Связывание служб в цепочку 216
Телеметрия 216
Архитектура и работа драйвера OpenStack APIC 217
Порядок интеграции 218
Пример развертывания 219
Установка версии Icehouse 220
Конфигурирование драйвера контроллера Cisco APIC 222
Выявление неисправностей 227
Проект групповой политики для платформы OpenStack 228
Резюме 230

Глава 7. Методология проектирования фабрики ACI 233
Перечень важнейших функциональных возможностей фабрики ACI 234
Принципы пересылки пакетов в инфраструктуре ACI 234
Сегментация на основе терминальных групп 243
Модель управления 245
Оборудование и программное обеспечение 248
Физическая топология 250
Соображения, касающиеся структуры Cisco APIC 250
Соображения, касающиеся структуры ствола 253
Соображения, касающиеся структуры листьев 254
Конфигурации с множеством арендаторов (multi-tenancy) 261
Этапы первоначального конфигурирования 262
Ускоренный вариант инициализации 263
Управление сетью 264
Конфигурирование портов доступа на основе политик 266
Группы политик интерфейса и PortChannels 271
Домены Virtual Machine Manager (VMM) 275
Конфигурирование виртуальной топологии 277
Широковещательный домен 278
Внешние соединения 282
Резюме 283

Глава 8. Подключение сетевых служб с помощью ACI 285
Обзор структуры ACI с сетевыми службами 4–7 уровней 286
Преимущества 286
Подключение терминальных групп с сервисным графом 287
Расширение на виртуализированные серверы 287
Модель управления 287
Сервисные графы, функции и интерпретация 289
Аппаратная и программная поддержка 290
Моделирование подключения службы в Cisco ACI 291

Определение сервисного графа 292
Физические и логические устройства 293
Селектор (или контекст) логического устройства 294
Расщепление широковещательных доменов 294
Этапы конфигурирования 295
Определение сервисного графа 296
Определение конкретных устройств и кластера конкретных устройств 303
Резюме 310

Глава 9. Усовершенствованные механизмы телеметрии 311
Атомарные счетчики 311
Принцип действия атомарных счетчиков 311
Подробное пояснение и пример 313
Атомарные счетчики и APIC 314
Показатели задержки 315
Мониторинг рабочего состояния ACI 316
Статистическая информация 317
Сбои 319
События, записи в журнале, диагностика 324
Оценка рабочего состояния в баллах 325
Централизованный ACI-подход с использованием команды **`show tech-support`** **326**
Резюме 327

Глава 10. Особенности построения коммутаторов для центров обработки данных 329
Модули данных, управления и контроля 329
Отделение модулей передачи данных, контроля и управления друг от друга 329
Взаимодействие модулей контроля, данных и управления 330
Защита модуля контроля с помощью политики CoPP 332
Архитектура коммутаторов центра обработки данных 335
Сквозная коммутация: производительность центра обработки данных 336
Архитектура фабрики перекрестной коммутации 340
Централизованная общая память (SoC) 351
Многоуровневая система SoC 353
Основы качества обслуживания 355
Возможности центра обработки данных QoS 361
Реализация Nexus QoS: модель MQC 370
Резюме 373

Заключение 375
Предметный указатель 377

Глава 3

Центры обработки данных, управляемые политиками

Цель этой главы — помочь читателям понять подход Cisco Application Centric Infrastructure (ACI) к моделированию бизнес-приложений с помощью сетевой фабрики Cisco ACI и показать, как применять непротиворечивые, отлаженные политики к этим приложениям. Подход Cisco ACI обеспечивает уникальное применение сочетания аппаратных и программных возможностей при развертывании приложений либо графически посредством контроллера Cisco GUI Application Policy Infrastructure Controller (APIC), либо программно с помощью модели Cisco APIC API, представляющей собой интерфейс передачи репрезентативного состояния (Representational State Transfer — RESTful). Модель APIC предлагает тип контроллера, который уникален в своей отрасли. Концепции и принципы APIC подробно объясняются в этой главе. Наконец, фабрика Cisco ACI предназначена не только для развертывания с нуля. Много пользователей интересуются развертыванием фабрики ACI в существующей среде. В конце настоящей главы мы объясняем, как интегрировать фабрику ACI в существующую сеть.

Зачем нужна модель, управляемая политиками?

Современные предприятия, поставщики услуг, провайдеры облачной инфраструктуры и (более широко) клиенты центра обработки данных стремятся все быстрее развертывать приложения в своем рабочем окружении. Количество приложений, размещаемых в центрах обработки данных, стремительно растет. Одновременно увеличивается аппаратная сложность сетевых устройств, поскольку все больше функций должно быть реализовано на уровне микросхем при увеличении плотности, пропускной способности и функциональных возможностей порта. Сетевая среда также становится более разнообразной, потому что есть различные устройства, используемые для различных уровней агрегации, а также несколько поколений продуктов в той же среде центра обработки данных. Это еще больше обостряет проблемы, связанные с потребностями владельцев приложений и возможностями администраторов сетей своевременно внедрять новые приложения. Барьер коммуникации, с которым сталкиваются администраторы сети и владельцы приложений при развертывании нового решения, изображен на рис. 3.1.

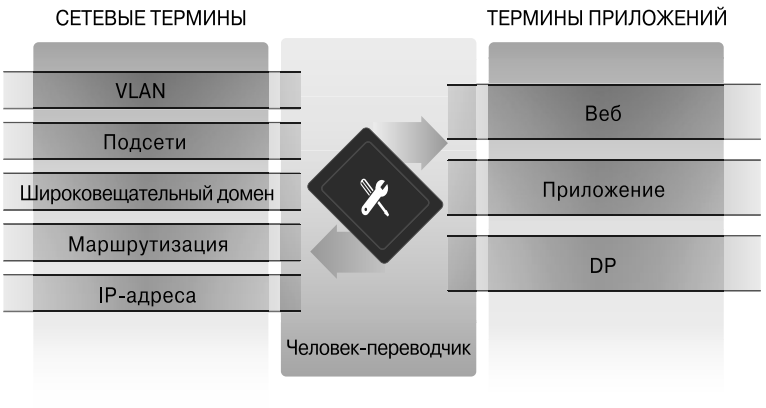

Рис. 3.1. Различие между терминами сетей и приложений

Для развертывания нового приложения в сети персонал должен выполнить следующие действия.

- Локализовать сеть VLAN и подсеть.
- Обеспечить безопасность с помощью списков управления доступом (ACL).
- Определить, нужна ли карта качества обслуживания (QoS) для нового приложения в законченной сетевой модели QoS, возможности которой будут изменяться в зависимости от аппаратного обеспечения, на котором она будет развертываться.

Итак, чтобы обеспечить успешное развертывание нового приложения, администраторы сети должны стать истинными экспертами по сетевой инфраструктуре, а также хорошо понимать длительные процессы сертификации сетевых изменений и среды тестирования. Естественно, те же проблемы возникнут при поиске и устранении неисправностей приложений. В крупномасштабных средах сопоставление событий, связанных с работой приложения (таких, как задержка, изменение полосы пропускания и отбрасывание пакетов), со сбоями при выполнении сетевых операций (при возникновении которых необходимо проследить путь пакетов вплоть до каждого аппаратного устройства, чтобы подтвердить, является ли это сетевой проблемой) — очень долгое и трудоемкое занятие.

Это приводит нас к идее об уровне абстракции: необходимо избежать участия человека при переводе терминов с сетевого языка на язык приложения. Кроме того, необходимо оптимизировать ресурсы. Системы должны допускать развертывание приложений без нарушения трафика других приложений и обеспечивать оптимизацию доступных аппаратных сетевых ресурсов. Именно в этом заключается значение подхода к управлению центром обработки данных с помощью политик (Policy Driven Data Center approach).

Центры обработки данных, управляемые политиками, основаны на декларативной модели, в которой старая императивная модель управления заменена политикой (рис. 3.2). В декларативной модели управления коммутаторы настраиваются

на основе требований приложений (которые на платформе Cisco ACI называются *терминальными группами* (endpoint groups)) и развертывают приложения там, где и когда это возможно, вместо жесткого кодирования с помощью базовых инструкций. Например, контрольно-диспетчерский пункт сообщает самолету, где взлететь. Но ведь пилот самолета, а не контрольно-диспетчерский пункт выполняет взлет. В этом заключается сущность использования декларативной модели управления, основанной на теории обещаний (promise theory) — активно развивающемся подходе к управлению сложными системами.

Операторы по обработке багажа выполняют последовательность простых основных инструкций

Центр управления полетами указывает, откуда взлетать, а не как лететь

Рис. 3.2. Декларативная модель управления центром обработки данных на основе политик

В рамках стратегического подхода к управлению центрами обработки данных возникает новый уровень абстракции между аппаратными средствами и программным обеспечением, а также методология, позволяющая адаптировать сетевые настройки к разным аппаратным платформам, сетевым возможностями и будущим технологиям. Это обеспечивает автоматизацию коммуникации между персоналом сетей и командами, эксплуатирующими приложения, и уменьшает время развертывания для приложений с месяцев до секунд, а то и меньше.

Теория политик

Модель политики Cisco APIC определяется как механизм установления правил, ориентированный на приложение, сверху и абстрагирования сетевой функциональности снизу. Модель политики Cisco APIC — это объектно-ориентированная модель, основанная на теории обещания. *Теория обещания* (promise theory) основывается на декларативном, масштабируемом управлении интеллектуальными объектами в противоположность устаревшим императивным моделям управления. Императивная модель управления — это централизованная система управления в стиле сверху вниз. В этих системах центральный администратор должен знать и команды настройки базовых объектов, и текущее состояние этих объектов (рис. 3.3).

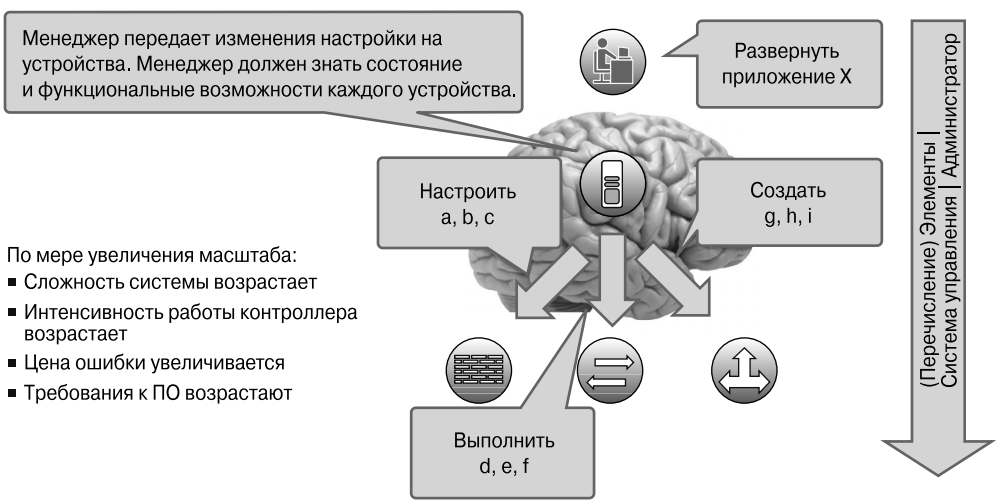

Рис. 3.3. Конфигурация базовых компонентов

Теория обещания, наоборот, полагается на действия конечных устройств (объектов) для достижения требуемого состояния конфигурации, инициированного системой управления. Объекты в свою очередь отвечают за передачу исключений или отказов назад к системе управления. Это подчеркивает сложность системы управления и допускает масштабирование системы. Эти системы масштабируются далее, разрешая методам базовых объектов поочередно запрашивать изменения состояния друг у друга и/или у объектов низшего уровня. Теория обещания схематически изображена на рис. 3.4.

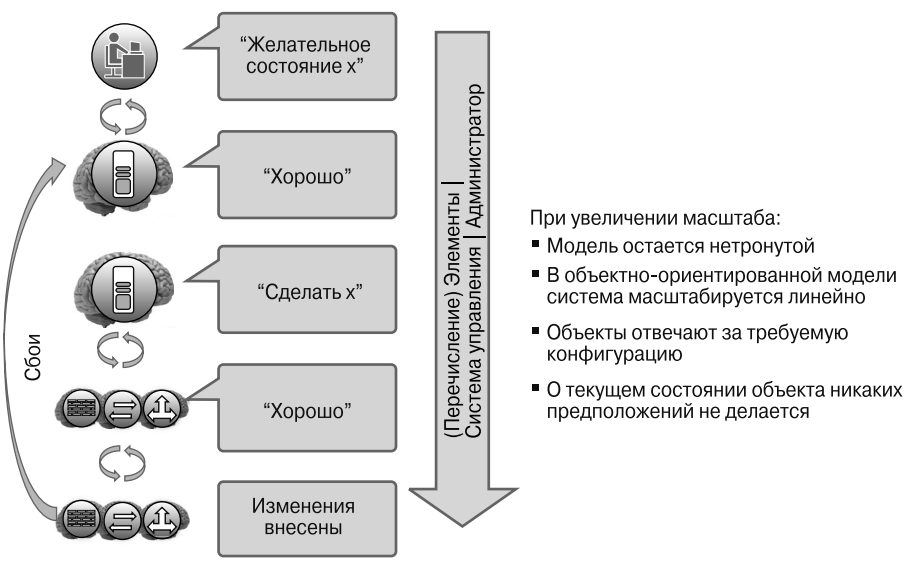

Рис. 3.4. Применение теории обещания для управления крупномасштабной системой

Объектная модель политики Cisco APIC

Традиционно приложения были ограничены возможностями сети. Такие понятия, как адресация, VLAN и безопасность, были связаны друг с другом, ограничивая масштаб и мобильность самого приложения. Поскольку современные приложения проектируются с учетом мобильности и веб-масштаба, такие ограничения затрудняют их быстрое и согласованное развертывание. Физическая фабрика Cisco ACI основана на топологии сети Клоса (рис. 3.5). Она использует двудольный граф, состоящий условно из уровней листьев и ствола, в котором каждый лист (коммутатор) соединяется с каждым коммутатором ствола, а внутри уровней никакие прямые подключения между коммутаторами не разрешены. Листья действуют как точки подключения для всех внешних устройств и сетей, а ствол действует как быстродействующий механизм переадресации между листьями. Фабрика Cisco ACI управляется, контролируется и администрируется контролером Cisco APIC.

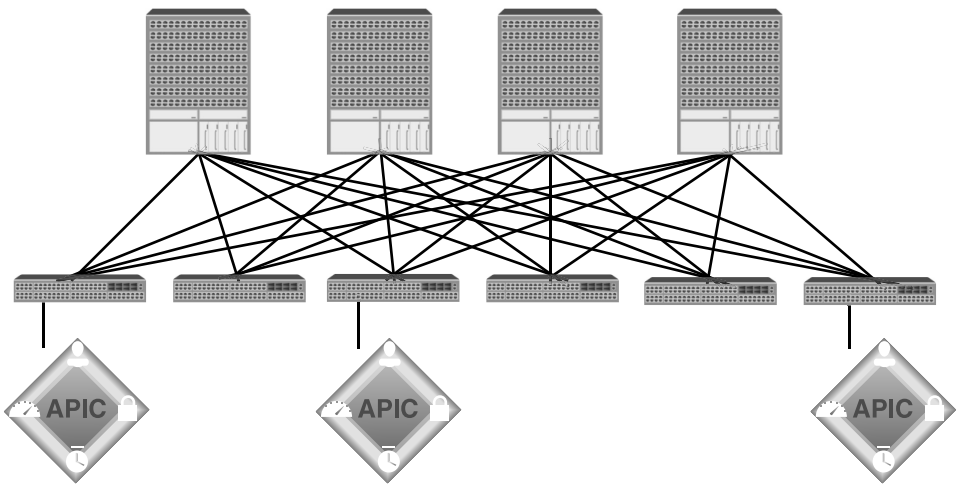

Рис. 3.5. Схема фабрики Cisco ACI

На верхнем уровне модели политики Cisco APIC находятся один или несколько арендаторов, которые допускают сегрегацию администрирования сетевой инфраструктуры и потоков данных. Эти арендаторы могут быть клиентами, подразделениями или группами, в зависимости от организационных потребностей. Например, конкретное предприятие могло бы использовать одного арендатора для всей организации, в то время как у поставщика облачной инфраструктуры могли бы быть клиенты, использующие одного или более арендатора, представляющего свою организацию.

Арендаторы далее разделяются на частные сети третьего уровня, которые непосредственно связаны с экземпляром виртуального маршрутизатора (Virtual Route Forwarding — VRF) или отдельным IP-пространством. У каждого арендатора может быть одна или несколько частных сетей третьего уровня в зависимости от потребностей их бизнеса. Частные сети третьего уровня позволяют далее разделить

организационные требования и требования переадресации вниз по иерархии. Поскольку контексты используют разные экземпляры виртуальных маршрутизаторов, IP-адресация может дублироваться в отдельных контекстах, что обеспечивает поддержку многопользовательского режима.

Арендатор (tenant) — это логический контейнер или папка для политик приложения. Эта сущность может представлять собой физическое лицо, организацию или домен, а может просто использоваться для удобства организации информации. С точки зрения политики обычный арендатор представляет собой изолированную среду настроек, а не выделенную сеть. Специальный арендатор, который называется *общим* (common), содержит общие политики, которые могут использоваться другими арендаторами. *Контекст* (context) — это представление выделенного пространства сети третьего уровня. Это еще один тип изолированной среды на платформе Cisco ACI. Арендатор может использовать несколько контекстов. Контексты могут принадлежать арендатору (содержаться в настройках арендатора) или быть частью настроек общего арендатора. Этот подход позволяет разворачивать как несколько частных сетей третьего уровня для каждого арендатора, так и общие сети третьего уровня, используемые несколькими арендаторами. Таким образом, не вы диктуете определенную жестко ограниченную модель аренды, а политика конечного устройства определяет общее поведение платформы Cisco ACI для всех конечных устройств, определенных в данном виртуальном контексте ACI.

Ниже контекста модель содержит серию объектов, которые определяют само приложение. Эти объекты называются *терминальными группами* (EPG). Терминальные группы — это набор подобных конечных устройств, представляющих уровень приложения или набор служб. Терминальные группы соединены друг с другом посредством политики. Важно отметить, что политика в этом случае — нечто большее, чем просто ряд списков управления доступом, и включает в себя набор входящих/исходящих фильтров, качественных параметров настройки трафика, правила разметки и переадресации, а также графы служебных устройств (сервисные цепочки) уровней 4–7. Это отношение показано на рис. 3.6.

На рис. 3.6 показаны два контекста под конкретным арендатором и ряд приложений, образующих этот контекст. На этом рисунке терминальные группы образуют уровень приложения или другую логическую группировку приложения. Например, Приложение B (раскрытое в правой части рисунка) может состоять условно из синего уровня веб-сайтов, красного уровня приложения и оранжевого уровня базы данных. Комбинация терминальных групп и политики, которая определяет их взаимодействие, является профилем сети приложения в фабрике Cisco ACI.

Терминальные группы

Терминальные группы обеспечивают логическую группировку объектов, требующих применения сходной политики. Например, терминальная группа может быть группой компонентов, составляющих уровень веб-сайтов приложения. Сами конечные устройства определяются с помощью NIC, vNIC, IP-адресов или имен DNS с возможностью расширения для будущих методов идентификации компонентов приложения.

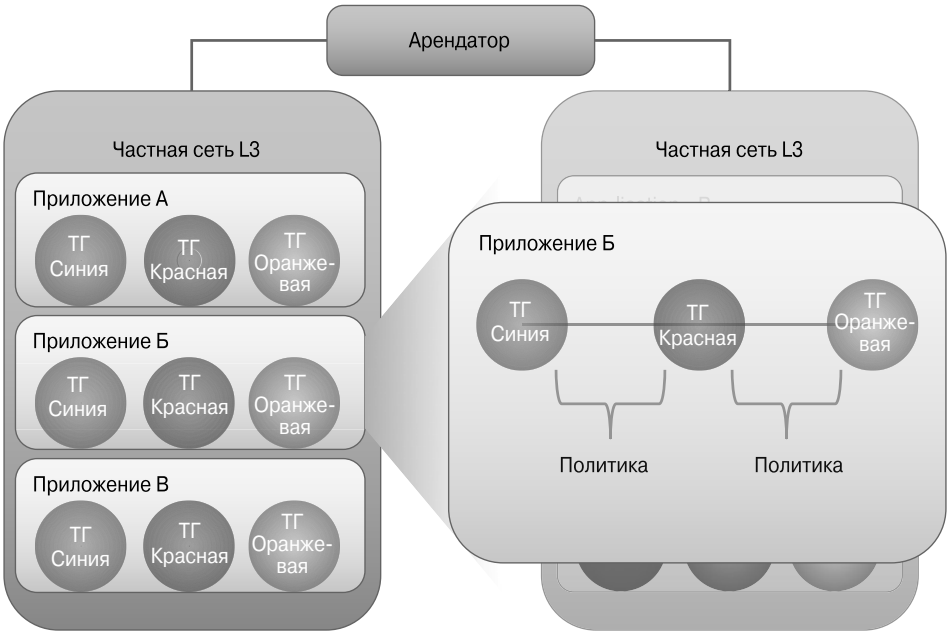

Рис. 3.6. Логическая объектная модель Cisco APIC

Терминальные группы также используются для представления других объектов, таких как внешние сети, сетевые службы, устройства безопасности, сетевого хранения и т.д. Они представляют собой коллекции одного или нескольких конечных устройств, обеспечивающих одинаковую функцию, и являются логической группировкой с переменными опциями использования в зависимости от модели развертывания используемого приложения. Отношения между конечными устройствами, терминальными группами и приложениями показаны на рис. 3.7.

Терминальные группы предназначены для обеспечения гибкости. Их можно настраивать на одну или несколько моделей развертывания по выбору клиента. После этого терминальные группы используются для того, чтобы определить, где применяется политика. Политика применяется к терминальным группам в фабрике Cisco ACI, тем самым определяя, как они будут связываться друг с другом. Это позволяет расширить применение политики в будущем в самой терминальной группе.

Приведем несколько примеров использования терминальных групп.

- Терминальная группа, определенная традиционными сетями VLAN: все конечные устройства, соединенные с конкретной сетью VLAN, помещаются в терминальную группу.
- Терминальная группа, определенная сетью VxLAN: все конечные устройства, соединенные с конкретной сетью VLAN, помещаются в терминальную группу.
- Терминальная группа, отображенная в группу портов VMware.

- Терминальная группа, определенная IP-адресами или подсетью: например, `172.168.10.10` or `172.168.10*`.
- Терминальная группа, определенная именами или диапазонами DNS: например, `example.web.networks.com` или `*.web.networks.com`.

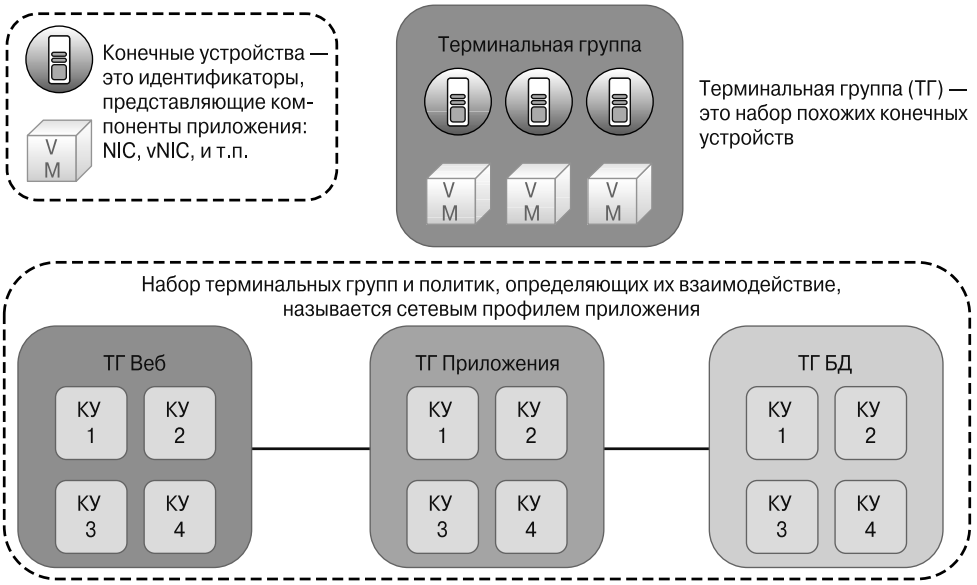

Рис. 3.7. Отношения между терминальными группами

Использование терминальных групп преднамеренно сделано гибким и расширяемым. Эта модель предоставляет инструменты для создания сетевого представления приложений, которое затем будет отображено в модель развертывания фактической среды. Кроме того, определение конечных устройств сделано расширяемым, чтобы обеспечить поддержку будущих модификаций продукта и новых отраслевых требований.

Реализация терминальных групп в фабрике имеет несколько ценных преимуществ. Терминальная группа действует как единственная точка осуществления политики для группы внутренних объектов. Это упрощает настройку этой политики и гарантирует ее непротиворечивость. Любая дополнительная политика применяется не к подсети, а к самой терминальной группе. Это означает, что изменения IP-адресации в конечном устройстве не изменяет применяемую к нему политику, как это часто бывает в традиционных сетях (исключение составляет конечное устройство, определенное его IP-адресом).

И наоборот, перенос конечного устройства в другую терминальную группу приводит к применению новой политики к коммутатору, в который включено конечное устройство, и тем самым определяет поведение этого конечного устройства с учетом новой терминальной группы.

Отношения между конечными устройствами (КУ), терминальными группами (ТГ) и политиками показаны на рис. 3.8.

Кроме того, терминальные группы предоставляют способ, которым применяется политика для терминальных групп. *Физическая троичная ассоциативная память* (ternary content-addressable memory — TCAM), в которой необходимо хранить политику для ее применения, является дорогим компонентом коммутаторов, поэтому производители заинтересованы в уменьшении числа необходимых политик, в противном случае стоимость коммутаторов будет расти. В фабрике Cisco ACI политика применяется на основе терминальной группы, а не самого конечного устройства. Размер политики можно выразить как n Чm Чf, где n — количество источников; m — количество адресатов; f — количество фильтров политики. В фабрике Cisco ACI источники и адресаты становятся одной записью для данной терминальной группы, что сокращает количество общих требуемых записей. Терминальная группа отличается от сети VLAN: терминальная группа может быть ограничена сетью VLAN в определенном широковещательном домене. Однако терминальная группа может быть определена шире, чем сети VLAN, — она может быть набором виртуальных сетевых адаптеров (vNIC), MAC-адресов, подсетей и т.д.

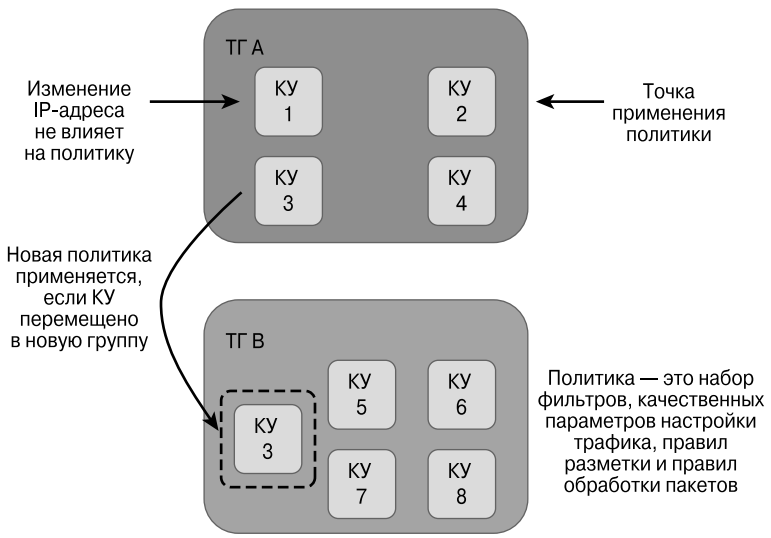

Рис. 3.8. Отношения между терминальными группами и политиками

Роль терминальных групп в сокращении размера таблицы политики продемонстрирована на рис. 3.9.

Как уже указывалось, политика внутри фабрики Cisco ACI определяет передачу данных между двумя терминальными группами. Они могут применяться как в однонаправленном, так и в двунаправленном режиме для любой заданной пары терминальных групп. Эти политики определяют разрешенную связь между терминальными группами, как показано на рис. 3.10.

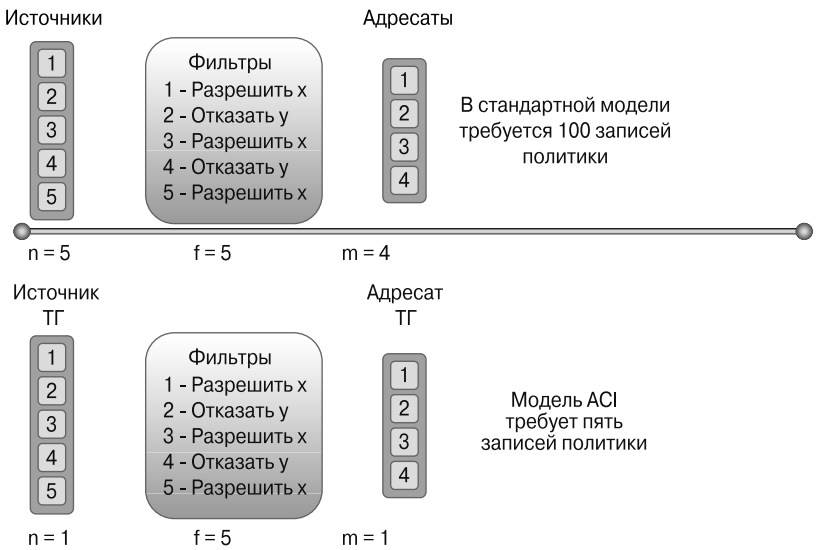

Рис. 3.9. Роль терминальной группы в сокращении размера таблицы политики

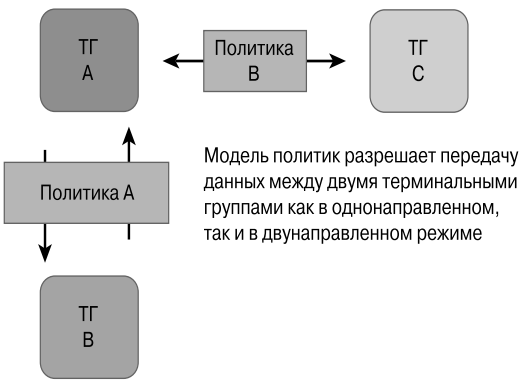

Рис. 3.10. Одно- и двунаправленный режим политики

Применение политики Cisco APIC

В этом разделе описывается концепция применения политики Cisco APIC, включая одно- и многоадресную передачу.

Применение одноадресной политики

Отношение между терминальными группами и политиками можно представить в виде матрицы, в которой одна ось соответствует терминальным группам источника (source EPG — sEPG), а другая — терминальным группам адресата (destination EDP — dEPG), как показано на рис. 3.11. Точке пересечения

конкретных терминальных групп источника (ТКИ) и адресата (ТКА) соответствует одна или несколько политик. Чаще всего эта матрица оказывается разреженной, потому что многие терминальные группы источника не взаимодействуют друг с другом.

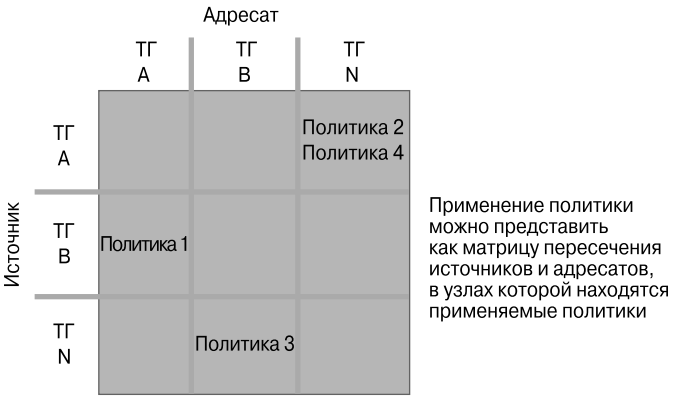

Рис. 3.11. Матрицы применения политики

Каждая политика состоит из последовательности фильтров по качеству обслуживания, управления доступом и т.д. Фильтры — это определенные правила, образующие политику между двумя терминальными группами. Фильтры состоят из входа и выхода: функций разрешения, отказа, переадресации, регистрации, копирования (не путать с функцией SPAN) и помечающей функции. Политики допускают применение символов подстановки при определении отдельных полей. Применение политики, как правило, основано на поиске первого наилучшего соответствия. Правила применения символа подстановки приведены на рис. 3.12.

ТГИ (Терминальная группа источника)	ТГН (Терминальная группа назначения)	ID приложения	Комментарии
Полностью определенно	Полностью определенно	Полностью определенно	Полностью определенно (S, D, A) правила
Полностью определенно	Полностью определенно	*	(S, D, *) правила
Полностью определенно	*	Полностью определенно	(S, *, A) правила
*	Полностью определенно	Полностью определенно	(*, D, A) правила
Полностью определенно	*	*	(S, *, *) правила
*	Полностью определенно	*	(*, D, *) правила
*	*	Полностью определенно	(*, *, A) правила
*	*	*	По умолчанию (например, неявный отказ)

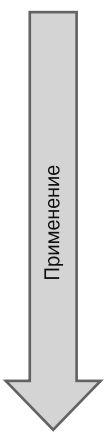

Рис. 3.12. Правила применения шаблонного символа

Применение политики в пределах фабрики гарантируется всегда; однако политика может быть применена в одном из двух мест: либо на входном, либо на выходном листе. Политика может быть применена на входе, только если известна терминальная группа адресата. Терминальная группа источника известна всегда. Во время присоединения нового конечного устройства определяются его терминальная группа EPG и все политики, для которых эта группа является источником или адресатом, всегда переносятся на соответствующий листовой коммутатор. После того, как политика перенесена на лист, она сохраняется и реализуется в аппаратных средствах. Поскольку контроллер APIC Cisco знает обо всех терминальных группах и включенных в них конечных устройствах, лист, к которому всегда присоединяется терминальная группа, имеет все требуемые политики и не обязан направлять трафик на контроллер, как это могло бы быть в других системах. Применение политики к листовым узлам продемонстрировано на рис. 3.13.

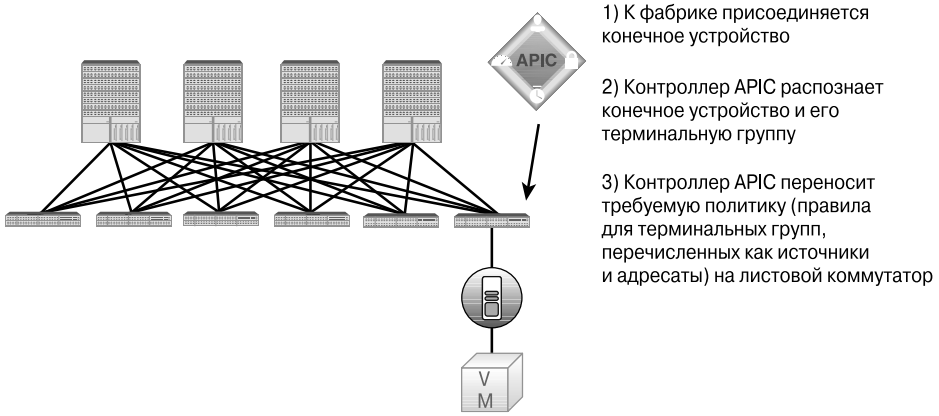

Рис. 3.13. Применение политики к листовым узлам

Как уже упоминалось, если терминальная группа адресата не известна, политика не может быть применена на входе в фабрику. Вместо этого биты терминальной группы источника в зарезервированной части заголовка VxLAN помечаются дескриптором, а биты применяемой политики остаются не помеченными. После этого пакет отправляется на узел переадресации, который обычно располагается в стволе. Стволу известны все точки назначения внутри фабрики, поэтому, если место назначения неизвестно, пакет отбрасывается. Если место назначения распознается, то пакет направляется на лист назначения.

Ствол никогда не применяет политику; она всегда обрабатывается на конечном листе. Когда конечный коммутатор листа получает пакет, он считывает биты записи, относящиеся к терминальной группе источника и политике (они были помечены на входе). Если метки битов политики означают, что она уже была применена, то пакет выходит из фабрики без дополнительной обработки. Если метки битов политики означают, что она еще не применялась, то терминальная группа источника, указанная в пакете, сопоставляется с терминальной группой адресата, всегда известной на листе выхода, и применяется соответствующая политика. Применение политики ко всей фабрике показано на рис. 3.14.

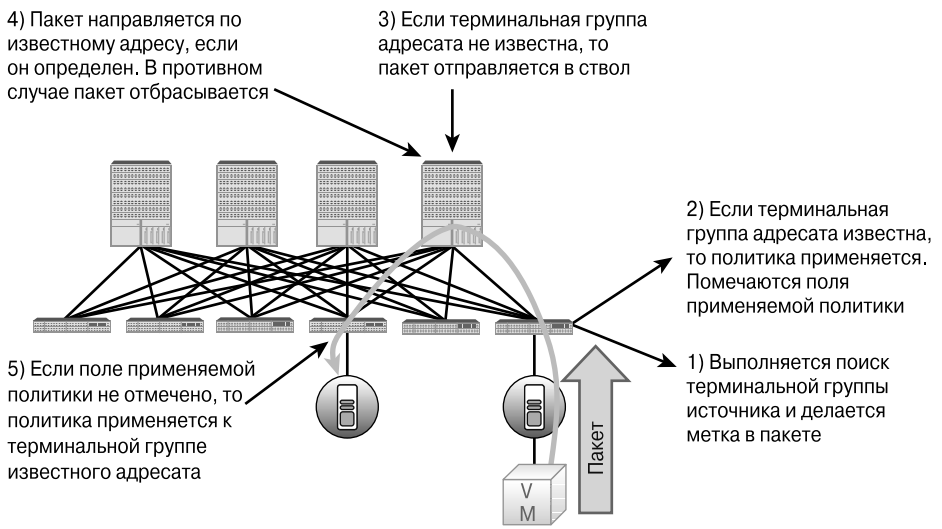

Рис. 3.14. Применение политики к фабрике

Механизм применения политики на входном коммутаторе при известных группах источника и назначения предназначен для повышения эффективности обработки трафика внутри фабрики (рис. 3.15).

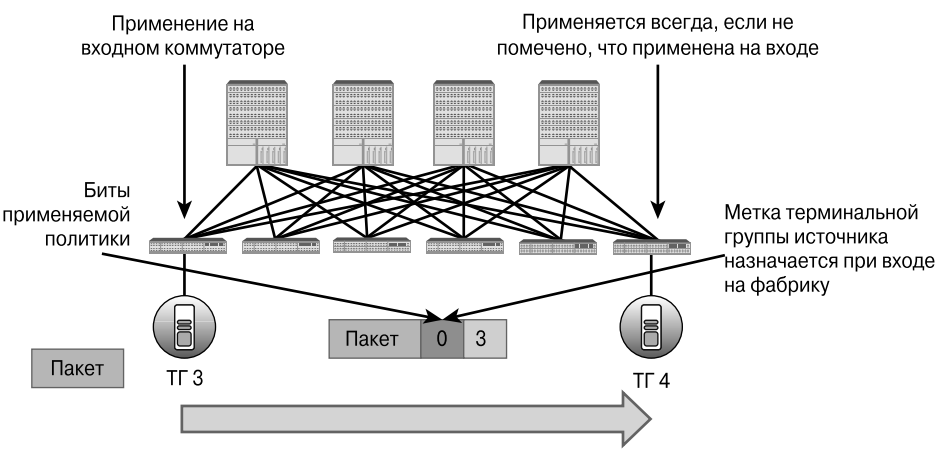

Рис. 3.15. Оппортунистическое применение политики на входе

Применение многоадресной политики

Природа многоадресной передачи выдвигает к применению политики немного другие требования. Несмотря на то что терминальная группа источника легко определяется на входе, потому что она никогда не бывает многоадресной, место назначения является абстрактным объектом; группа многоадресной передачи может состоять из конечных устройств, относящихся к разным терминальным группам.

В многоадресных случаях для применения политики фабрика Cisco ACI использует группу многоадресной передачи. Эти группы определяются диапазоном или диапазонами многоадресной передачи. В таком случае политика встраивается между терминальной группой источника и группой многоадресной передачи, как показано на рис. 3.16.

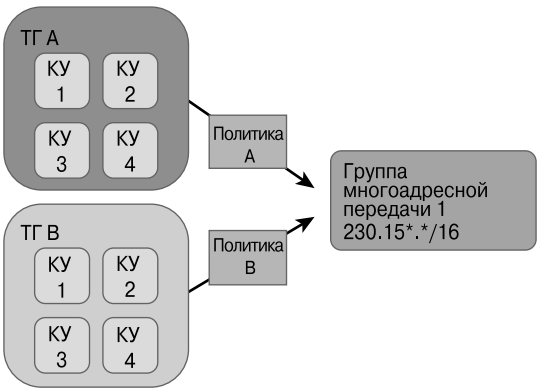

Рис. 3.16. Группа многоадресной передачи (специализированная терминальная группа многоадресной передачи)

Группа многоадресной передачи (терминальная группа, соответствующая многоадресному потоку) всегда является местом назначения и никогда не используется в качестве источника. Трафик, отправленный в группу многоадресной передачи, идет либо от многоадресного источника, либо от получателя, присоединяющегося к потоку через соединение *IGMP* (Internet Group Management Protocol — протокол управления группами Интернета). Поскольку многоадресные потоки не являются иерархическими, а сам поток уже находится в таблице переадресации (благодаря IGMP), многоадресная политика всегда осуществляется на входе. Это предотвращает необходимость записывать многоадресную политику на выходных листах, как показано на рис. 3.17.

Сетевые профили приложений

Как указывалось ранее, *сетевой профиль приложения* (application network profile — ANP) внутри фабрики представляет собой набор терминальных групп, их соединений и политик, определяющих эти соединения. Сетевые профили приложений становятся логическим представлением всего приложения и его взаимозависимостей внутри фабрики Cisco ACI.

Сетевые профили приложений логически моделируют процесс создания и развертывания сетевых связей приложений. Настройка и применение политик, а также связность обрабатываются самой системой с помощью контроллера Cisco APIC, а не администратором. Концепция сетевого профиля приложения приведена на рис. 3.18.

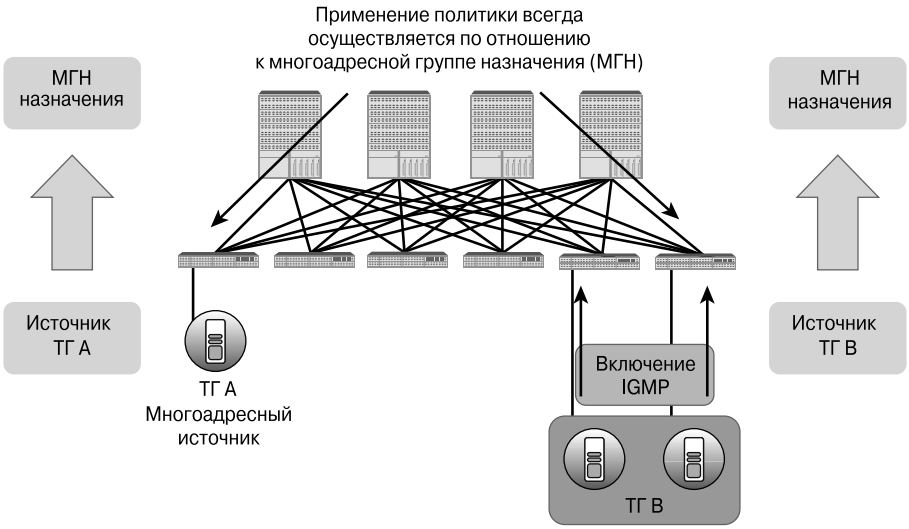

Рис. 3.17. Применение многоадресной политики

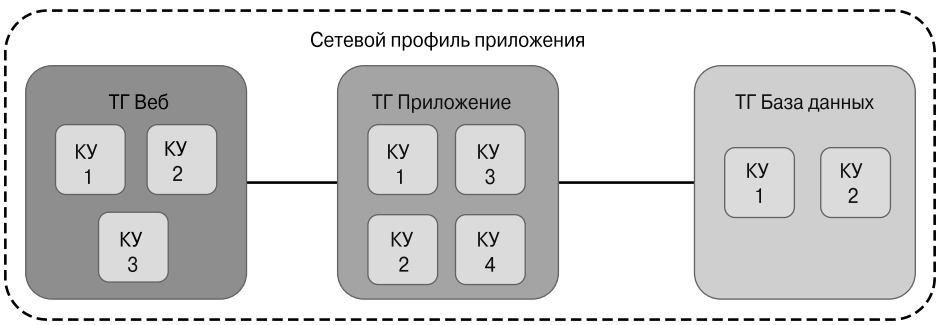

Рис. 3.18. Сетевой профиль приложения

Процесс создания сетевого профиля состоит из трех этапов.

- Создание терминальных групп (как указывалось ранее).
- Создание политик, определяющих связность, в том числе:
  - Разрешение
  - Отказ
  - Регистрация
  - Маркировка
  - Перенаправление
  - Копирование
  - Графы услуг
- Создание точек соединения между терминальными группами с помощью составных элементов политики, известных как *контракты*.

Контракты

Контракты определяют входящие и исходящие разрешения, отказы, QoS, перенаправление трафика и графы услуг. Они допускают как простое, так и сложное определение того, как данная терминальная группа связывается с другой терминальной группой с учетом требований конкретного окружения.

На рис. 3.19 показано отношение между тремя уровнями веб-приложения, определенного связью между терминальными группами и контрактами, определяющими их взаимодействие. Сумма этих частей образует сетевой профиль приложения. Контракты также устанавливают политику многократного использования и согласованность служб, которые обычно взаимодействуют с многочисленными терминальными группами. На рис. 3.20 показано использование ресурсов управления (MGMT) и сетевой файловой системы (NFS).

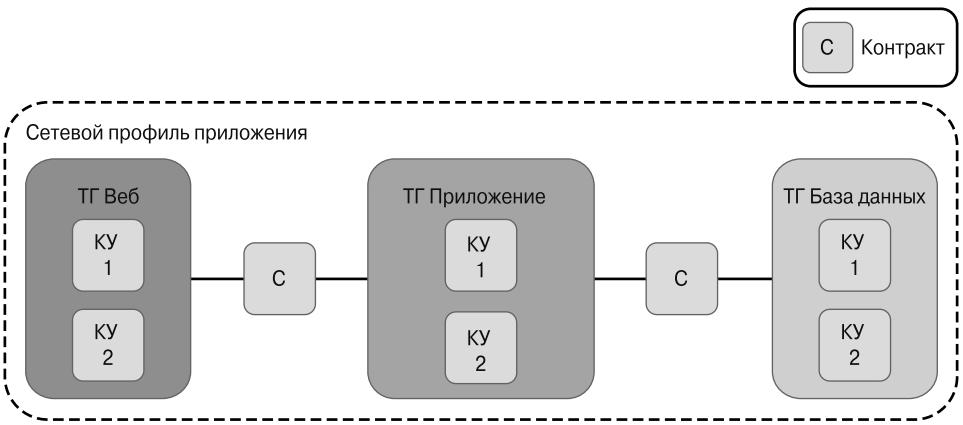

Рис. 3.19. Контракты в сетевых профилях приложений

На рис. 3.20 показано типичное трехуровневое веб-приложение, рассмотренное ранее, с некоторыми дополнительными связями. Обратите внимание на появление совместно используемых сетевых служб: файловой системы NFS и системы управления, которые используются всеми тремя уровнями, совместно с другими терминальными группами внутри фабрики. В этих случаях контракт обеспечивает повторно используемую политику, определяющую, каким образом терминальные группы файловой системы NFS и системы управления создают функции или службы, которые могут быть использованы другими терминальными группами.

Внутри фабрики Cisco ACI ответы на вопросы “что” и “где” по отношению к применению политики преднамеренно отделены друг от друга. Это позволяет создавать политику независимо от того, как она будет применяться, и будет ли она снова использована при необходимости. Фактическая политика, настроенная внутри фабрики, определяется на основе политики, определенной как *контракт* (ответ на вопрос “что”) и пересечение терминальных групп и других контрактов с той же политикой (ответ на вопрос “где”).

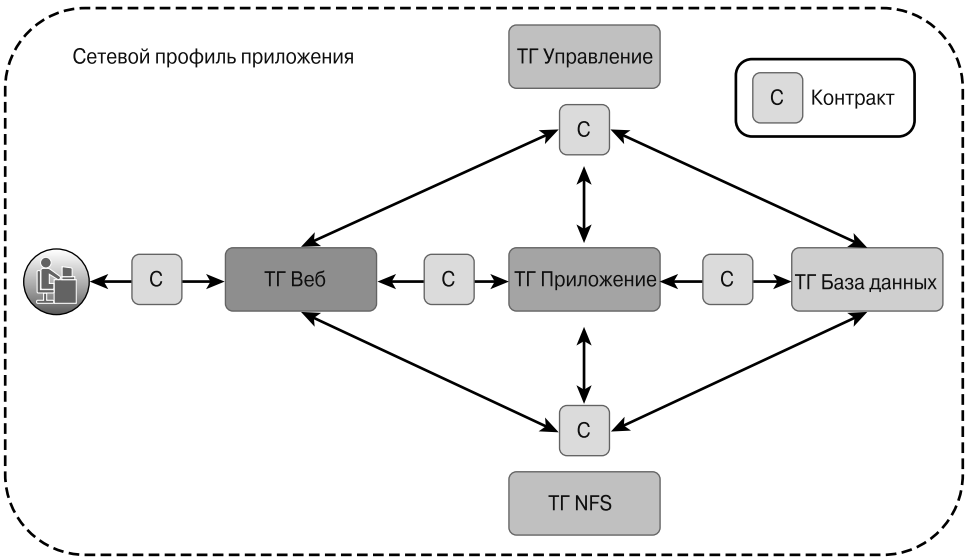

Рис. 3.20. Полный сетевой профиль приложения

В более сложных средах развертывания приложения контракты могут подразделяться на части с помощью *субъектов*, которые трактуются как приложения или части приложений. Для того чтобы лучше понять это деление, рассмотрим веб-сервер. Несмотря на то что его можно классифицировать как узел веб, он допускает обращение по протоколам HTTP, HTTPS, FTP и т.д., каждый из которых может требовать своей политики. В модели Cisco APIC эти отдельные функции или службы определяются с помощью субъектов, а субъекты объединяются в контракты, чтобы представлять набор правил, которые определяют, как терминальная группа связывается с другой терминальной группой (рис. 3.21).

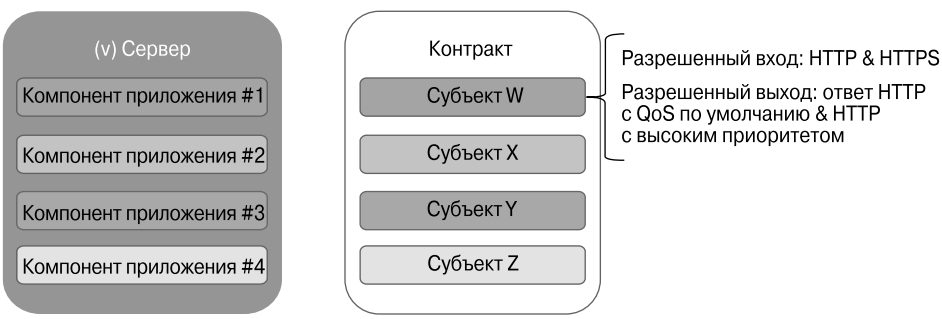

Рис. 3.21. Субъекты внутри контракта

Субъекты описывают функции, которые приложение представляет другим процессам в сети. Нужно рассматривать этот процесс как создание ряда функций: иначе говоря, веб-сервер предоставляет протоколы HTTP, HTTPS и FTP. Затем другие

терминальные группы используют одну или несколько из этих функций; какие терминальные группы будут использовать эти службы, определяется отношениями между терминальными группами и контрактами, которые содержат субъекты, определяющие приложения или части приложения. Полная политика устанавливается администраторами, определяющими совокупности терминальных групп, использующих функции, предоставляемые другими терминальными группами. На рис. 3.22 показан способ, которым эта модель обеспечивает функциональные возможности иерархических или обычных терминальных групп, которые являются группами приложений и частями приложений.

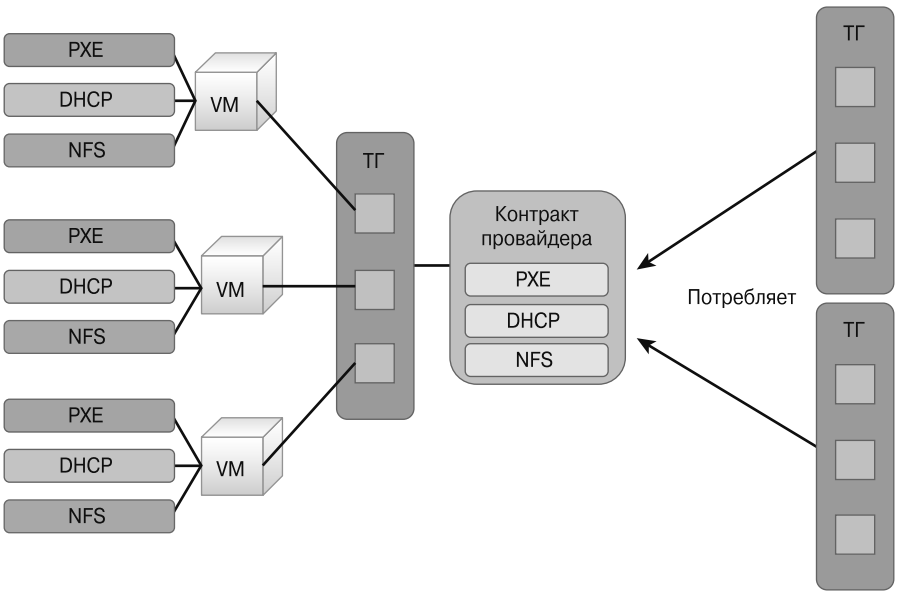

Рис. 3.22. Подробное представление субъектов внутри контракта

Кроме того, эта модель обеспечивает возможность определить черный список запрещенных адресов в каждой терминальной группе. Эти запрещения, известные как *табу*, имеют приоритет над контрактом, гарантируя, что определенные связи будут запрещены для некоторых терминальных групп. Эта возможность обеспечивает модель черного списка внутри фабрики Cisco ACI, как показано на рис. 3.23.

На рис. 3.23 показано, что контракт может быть определен так, чтобы разрешать весь трафик от всех терминальных групп. Затем это разрешение уточняется с помощью списка портов или диапазонов, которые являются нежелательными. Эта модель предоставляет клиентам способ постепенного перехода от модели черного списка, которая обычно используется в настоящее время, к более желательной модели белого списка. В модели черного списка все связи открыты, если они не были явно запрещены, тогда как модель белого списка требует, чтобы связь была явно определена, прежде чем стать разрешенной. Важно помнить, что списки запретов являются необязательными и в полной модели белого списка используются редко.

Рис. 3.23. Использование pfghtnjd для создания черного списка

Контракты обеспечивают группировку описаний и связанной с ними политики, определяющей прикладные службы. Они могут содержаться в заданной области видимости, арендаторе, контексте или терминальной группе как локальный контракт. Терминальная группа также допускает подписку на многочисленные контракты, которые обеспечивают надмножество политик, определенных внутри фабрики.

Несмотря на то что контракты могут использоваться для определения отношений между сложными реальными приложениями, их также легко использовать для традиционных моделей развертывания приложения. Например, если для определения отдельных служб используются единственные сети VLAN или VxLAN, связанные с группами портов в виртуальном коммутаторе VMware, можно определить простую модель контракта, избежав ненужной сложности.

Однако в более сложных моделях развертывания приложений, таких как PaaS, SOA 2.0 и Web 2.0, где требуется более высокая степень модульности приложений, используются более сложные отношения между контрактами. Реализация этих отношений позволяет определить подробные отношения между компонентами в единственной терминальной группе, а также между ней и многими другими терминальными группами.

Несмотря на то что контракты обеспечивают средства для поддержки более сложных прикладных моделей, они не навязывают дополнительную сложность. Как уже указывалось, для отношений между простыми приложениями можно использовать простые контракты. В то же время контракты позволяют создавать и повторно использовать отношения между сложными приложениями.

Контракты разделяются на следующие компоненты.

- **Субъекты.** Группы фильтров, применяемых к определенному приложению или службе.
- **Фильтры.** Применяются для классификации трафика.

- **Действия.** Разрешение, отказ, маркировка и другие действия, соответствующие фильтрам.
- **Метки.** Необязательное средство для выборочной группировки объектов, таких как субъекты и терминальные группы, для последующего применения политик.

В простой системе отношение между двумя терминальными группами похоже на отношения, показанные на рис. 3.24. Здесь терминальные группы WEB и APP образуют простое приложение, которое определяется заданным набором фильтров. Это очень общий сценарий развертывания. Даже в сложной среде эта модель оказывается предпочтительной для многих приложений.

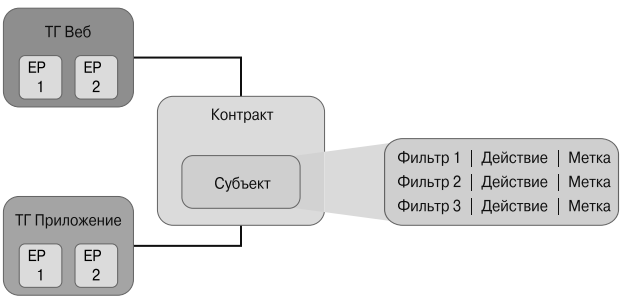

Рис. 3.24. Простые отношения между контрактами политик

Во многих средах требуются более сложные отношения; приведем несколько примеров.

- Среды, в которых используются сложные промежуточные программные системы.
- Среды, в которых одно множество серверов обеспечивает функциональные возможности многих приложений или групп (например, ферма баз данных для нескольких приложений).
- Среды PaaS, SOA и Web 2.0.
- Среды, в которых многие службы функционируют в одном и том же экземпляре операционной системы.

В этих средах фабрика Cisco ACI обеспечивает более устойчивый набор дополнительных функций, позволяющих логически моделировать физическое развертывание приложений. В обоих случаях контроллер Cisco APIC и программное обеспечение фабрики отвечают за распространение политики внутри фабрики и ее аппаратную реализацию. Автоматическое преобразование логической модели, которая используется для построения отношений между приложениями, в физическую модель, которая используется для реализации отношений между приложениями внутри фабрики, упрощает проектирование, развертывание и модификацию фабрики.

Примером такой реализации является ферма баз данных SQL, предоставляющая услуги базы данных многим группам разработчиков в организации (например,

красной, синей и зеленой командам), каждая из которых использует отдельную базу данных, предоставленную той же фермой. В этом экземпляре для доступа каждой команды к ферме базы данных могла бы использоваться отдельная политика, как показано на рис. 3.25.

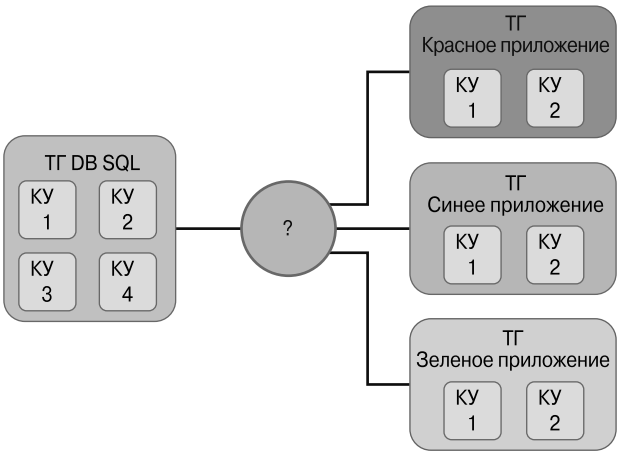

Рис. 3.25. Отдельная ферма баз данных, обслуживающая три отдельные группы, требующие отдельной политики контроля

Простые модели, рассмотренные ранее, не полностью охватывают сложные отношения между терминальными группами. В этом примере нужна возможность разделить политику для трех отдельных экземпляров базы данных в терминальной группе DB SQL, которые могут считаться *частями приложения* и рассматриваться внутри фабрики Cisco ACI как *субъекты*.

Фабрика Cisco ACI обеспечивает много способов для моделирования такого поведения приложения в зависимости от его сложности и пользовательских предпочтений. Первый способ заключается в том, чтобы использовать три контракта, по одному для каждой команды. Помните, что терминальная группа может предоставлять несколько контрактов и использовать надмножество правил, определенных в этих контрактах. На рис. 3.26 каждая терминальная группа команды приложения связывается с терминальной группой DB SQL, используя свой собственный контракт.

Как показано на рисунке, терминальная группа DB SQL наследует надмножество политик от трех отдельных контрактов. Затем терминальная группа каждой команды приложения использует свой контракт. Контракт определяет политику, в то время как отношение, определенное стрелками, обозначает, где политика будет применена или кто и какую службу предоставляет/использует. В этом примере терминальная группа красного приложения пользуется услугами служб SQL-DB и QoS, списком управления доступом, маркировкой, перенаправлением и другими функциями, определенными в контракте приложения красной команды. То же относится к синей и зеленой командам.

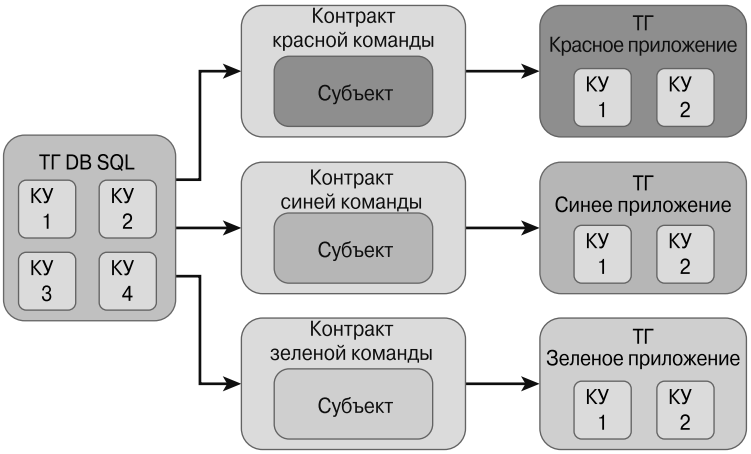

Рис. 3.26. Использование трех контрактов для разделения отношений между потребителями

Во многих подобных случаях применяются группы контрактов. Например, если создаются фермы из нескольких баз данных, доступ к которым в этом примере необходим всем трем командам, могут использоваться, например, фермы разработки, тестирования и производства. В этих случаях для логической группировки контрактов может использоваться пакет. Пакеты не являются обязательными; пакет может считаться контейнером для одного или нескольких контрактов в целях простоты использования. Использование пакетов показано на рис. 3.27.

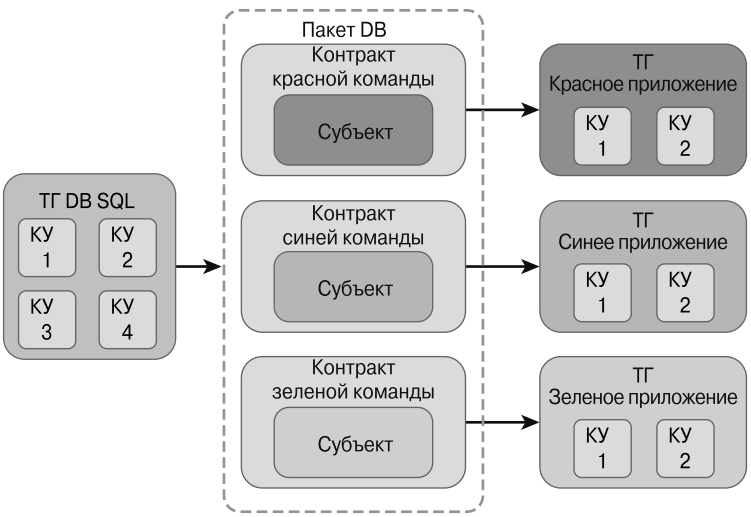

Рис. 3.27. Использование пакетов для группировки контрактов

На рис. 3.27 очень важно отметить точки присоединения стрелок, демонстрирующих отношение. В этом примере мы хотим, чтобы терминальная группа DB SQL

обеспечила все контракты из пакета контрактов, поэтому к терминальной группе присоединяется весь пакет. Для каждой из трех команд приложения мы хотим обеспечить доступ, определенный только ее конкретным контрактом, поэтому присоединили каждую команду к своему контракту в пакете.

Это же отношение можно смоделировать иначе, используя метки. Метки обеспечивают альтернативную функцию группировки, которая используется в определении политики приложения. В большинстве сред метки не требуются, но они доступны для развертывания с более сложными моделями приложений. Разумеется, члены команды должны быть знакомы с этим понятием.

При использовании меток единственный контракт может использоваться для представления нескольких служб или компонентов приложений, обеспечиваемых данной терминальной группой. В данном случае метки представляют терминальную группу DB, предоставляющую услуги базы данных трем отдельным командам. Маркируя субъекты и терминальные группы, которые их используют, мы применяем в данном контракте отдельную политику, даже если типы трафика или другие классификаторы являются идентичными. Это отношение показано на рис. 3.28.

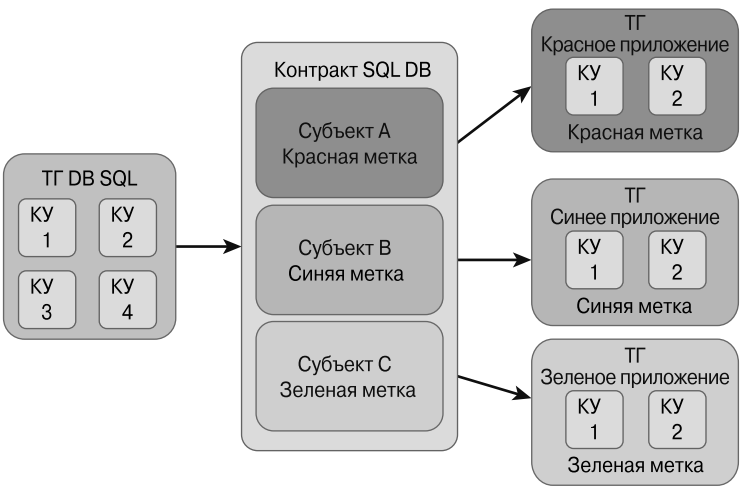

Рис. 3.28. Использование меток для группировки объектов в модели политики

На рис. 3.28 терминальная группа DB SQL предоставляет сервис, используя единственный контракт с именем *SQ-DB*, определяющий услуги базы данных, которые эта группа предоставляет трем различным командам. Затем каждая из терминальных групп трех команд, которые пользуются этими услугами, подключается к одному и тому же потоку данных. Конкретные правила для каждой из команд устанавливаются с помощью меток на субъектах и терминальных группах. К каждой терминальной группе применяются только те правила из контракта, который соответствует ее метке, даже если их классификация в структурном элементе совпадает (например, одни и те же порты четвертого уровня и т.д.).

Метки представляют собой очень мощный инструмент классификации, позволяющий группировать объекты для применения политики. Они также позволяют

быстро проводить приложения через разные этапы разработки. Например, если службу с красной меткой, представляющую среду разработки, необходимо заменить средой тестирования, представленной синей меткой, то достаточно просто изменить метку, присвоенную этой терминальной группе.

Принципы работы контроллера Cisco APIC

В этом разделе объясняются структура и компоненты контроллера Cisco APIC (Application Policy Infrastructure Controller).

Операционная система Cisco ACI (Cisco ACI Fabric OS)

Компания Cisco взяла традиционную операционную систему Cisco Nexus OS (NX-OS), разработанную для центров обработки данных, и оставила в ней только самые главные функции, необходимые для развертывания современных центров обработки данных Cisco ACI. Cisco также осуществила более глубокие структурные изменения, чтобы операционная система Cisco ACI Fabric OS могла легко отображать политику из контроллера APIC в физическую инфраструктуру. *Механизм управления данными* (Data Management Engine —DME) в операционной системе Cisco ACI Fabric OS создает каркас для запросов на чтение и запись данных из совместно используемой системы хранения без блокировки. Система хранения данных является объектно-ориентированной. Каждый объект в ней хранится в виде блока данных. Блок принадлежит одному из процессов операционной системы Cisco ACI Fabric OS, и только владелец этого процесса может записывать данные в этот блок. Однако любой процесс операционной системы Cisco ACI Fabric OS может считывать любые из данных одновременно с помощью интерфейса командной строки (CLI), *простого протокола сетевого управления* (Simple Network Management Protocol — SNMP) или вызова API. *Локальный элемент политики* (policy element — PE) позволяет контроллеру APIC реализовать модель политики непосредственно в операционной системе ACI Fabric OS, как показано на рис. 3.29.

Рис. 3.29. Операционная система Cisco ACI Fabric OS

Структура: компоненты и функция контроллера Cisco APIC

Контроллер APIC состоит из набора основных функций управления (рис. 3.30), который включает в себя следующие элементы.

- Администратор политики (система хранения политики).
- Администратор топологии.
- Система мониторинга (observer в терминологии ACI).
- Диспетчер загрузки.
- Администратор устройств (контроллер кластера).
- Администратор VMM.
- Администратор событий.
- Администратор сервера APIC.

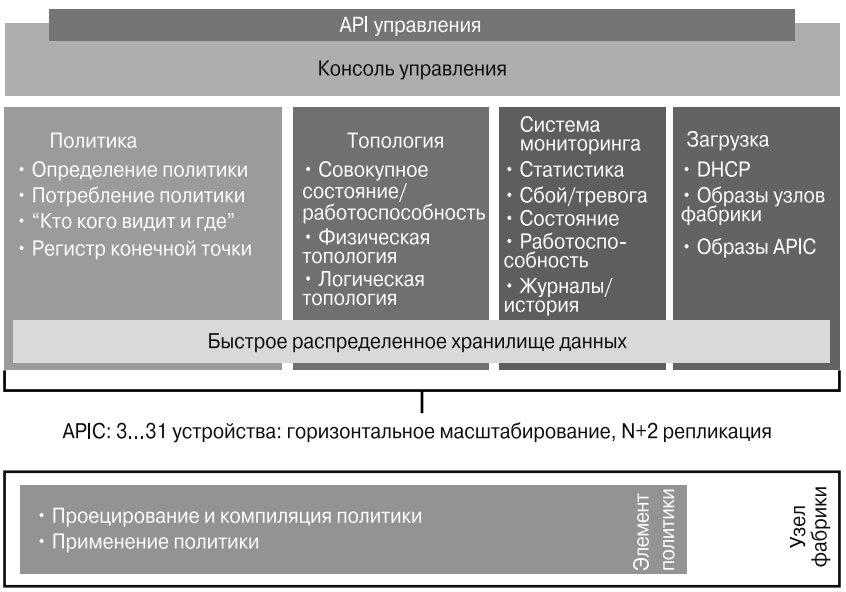

Рис. 3.30. Структура контроллера Cisco APIC

Администратор политики

Администратор политики — это распределенная система хранения, ответственная за определение и конфигурацию политик в фабрике Cisco ACI. Это набор политик и правил, которые применяются к существующим или гипотетическим (еще не созданным) конечным устройствам. Реестр конечных устройств — это подмножество администратора политики, который отслеживает конечные устройства, соединяющиеся с инфраструктурой Cisco ACI, и назначает политики терминальным группам, как определено системой хранения политик.

Администратор топологии

Администратор топологии поддерживает актуальность топологии Cisco ACI и информацию о входящих в нее устройствах. Данные о топологии передаются

контроллеру APIC листовыми и стволовыми коммутаторами. Физическая топология основывается на информации, обнаруженной протоколом *LLDP* (Link Layer Discovery Protocol — протокол канального уровня), и топологии маршрутизации фабрики, которая сообщается протоколами модифицированной системы *IS-IS* (Intermediate System-to-Intermediate System — протокол маршрутизации промежуточных систем), выполняемыми в пространстве инфраструктуры фабрики.

Администратор топологии отвечает за поддержание актуальной информации о глобальном состояния топологии. К ней относятся:

- физическая топология (уровень 1; физические соединения и узлы);
- логическая топология (отражение уровней 2 и 3).

Данные о топологии и соответствующее агрегированное рабочее состояние асинхронно обновляются в администраторе топологии после обнаружения изменений и доступны для запросов через интерфейсы APIC API, CLI и UI. Подфункция администратора топологии выполняет управление списком устройств контроллера APIC и поддерживает полную информацию об устройствах всей инфраструктуры Cisco ACI. Подфункция управления устройствами APIC обеспечивает полную идентификацию, включая номер модели и серийный номер, а также определяемые пользователем дескрипторы устройств (для простоты корреляции с устройствами и системами управления устройствами) для всех портов, линейных плат, коммутаторов, шасси и т.д.

Информация в базе устройств автоматически обновляется встроенным в коммутатор элементом/агентом политики, основанной на механизме управления данными DME, как только в фабрике Cisco ACI обнаруживаются или удаляются новые элементы или изменяются состояния устройств.

Система мониторинга (Observer)

Система мониторинга контроллера APIC служит системой хранения данных о рабочем состоянии, работоспособности и производительности инфраструктуры Cisco ACI. К ней относятся:

- состояние аппаратного и программного обеспечения и работоспособность компонентов ACI;
- рабочее состояние протоколов;
- данные о производительности (статистика);
- данные о сбоях, ожидающие обработки и или уже обработанных;
- записи о событиях.

Данные мониторинга можно получить по запросам с помощью интерфейсов APIC API, CLI и UI.

Администратор загрузок

Администратор загрузок управляет начальной загрузкой и микропрограммными обновлениями коммутаторов фабрики, а также элементов контроллера APIC. Он также функционирует как механизм распределения адресов для сети инфраструктуры,

который обеспечивает взаимодействие между контроллером APIC, коммутаторами ствола и листьев. Следующий процесс описывает перевод контроллера APIC в рабочее состояние и распознавание кластера.

Каждый контроллер APIC в инфраструктуре Cisco ACI использует внутренний частный IP-адрес, чтобы связаться с узлами ACI и другим контроллерами APIC в кластере. Контроллеры APIC обнаруживают IP-адрес другого контроллера APIC в кластере, используя процесс распознавания, основанный на протоколе LLDP.

Контроллеры APIC поддерживают вектор устройства (AV), который обеспечивает отображение идентификатора контроллера APIC в IP-адрес контроллера APIC и универсальный уникальный идентификатор (UUID) контроллера APIC. Первоначально каждый контроллер APIC запускается из вектора устройств AV, содержащего его локальный IP-адрес, а все другие слоты APIC помечаются как неизвестные.

После загрузки коммутатора, подключенного к контроллеру, элемент политики PE на листе получает свой вектор AV от контроллера APIC. Затем коммутатор передает этот вектор AV всем своим соседям и сообщает о любых несоответствиях между его локальным вектором AV и векторами AV соседей всем контроллерам APIC в его локальном векторе AV.

Используя этот процесс, контроллеры APIC узнают о другом контроллере APIC в инфраструктуре Cisco ACI через коммутаторы. После проверки вновь обнаруженных контроллеров APIC в кластере контроллеры APIC обновляют свой локальный вектор AV и программируют коммутаторы с новым вектором AV. После этого коммутаторы начинают распространять новый вектор AV. Этот процесс продолжается, пока все коммутаторы не будут иметь идентичные векторы AV и все контроллеры APIC не будут знать IP-адреса всех других контроллеров APIC.

Администратор устройств (контроллер кластера)

Администратор устройства отвечает за формирование кластера устройств APIC и управление им. Контроллер APIC работает на аппаратных средствах сервера (физическом сервере). Для управления горизонтальным масштабированием инфраструктуры ACI изначально инсталлируются как минимум три контроллера. Окончательный размер кластера APIC прямо пропорционален размеру инфраструктуры ACI и управляется требованиями скорости транзакций. Любой контроллер в кластере в состоянии обслужить любого пользователя для любой работы, причем контроллер может быть беспрепятственно добавлен в кластер или удален из кластера APIC. Важно понимать, что, в отличие от контроллера OpenFlow, ни один из контроллеров APIC никогда не находится в канале передачи данных. Администратор устройств показан на рис. 3.31.

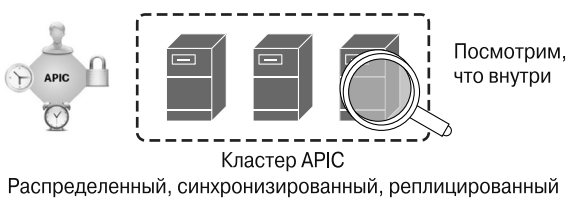

Рис. 3.31. Администратор устройств

Администратор VMM

Администратор VMM действует как агент между системой хранения политик и гипервизором. Он отвечает за взаимодействие с системами управления гипервизора, такими как vCenter, и облачными программными платформами, такими как OpenStack и CloudStack. Администратор VMM управляет виртуальными устройствами всех элементов гипервизора (pNIC, vNIC, имен VM и т.д.), а также перемещает политику в гипервизор, создавая группы портов и т.д. Он также прослушивает события гипервизора, такие как перемещение VM.

Администратор событий

Администратор событий — это система хранения всех событий и сбоев, инициированных контроллером APIC или узлами фабрики. Подробно эта тема раскрыта в главе 9.

Администратор сервера APIC

Администратор сервера — это монитор локального устройства. Он управляет ресурсами и состоянием локальных устройств контроллера APIC.

Структура: управление данными с помощью сегментирования

Кластер Cisco APIC использует технологию больших баз данных, которая называется *сегментированием* (sharding). Для того чтобы понять концепцию *сегментирования*, рассмотрим концепцию разделения базы данных. Сегментирование — это результат *горизонтального разделения базы данных*. В этой схеме строки базы данных сохраняются отдельно, а не нормализуются и не разделяются вертикально на столбцы. Сегментирование идет еще дальше, чем горизонтальное разделение, также разделяя базу данных на многочисленные экземпляры схемы. В дополнение к увеличивающейся избыточности сегментирование повышает производительность, потому что поисковая загрузка для большой разделенной таблицы может быть распределена среди нескольких серверов баз данных, а не просто среди нескольких индексов на том же логическом сервере. При сегментировании бо́льшие разделенные таблицы распределяются по серверам, а меньшие тиражируются полностью. После того как таблица сегментирована, каждый ее сегмент может находиться на совершенно разных логическом и физическом серверах, в центре обработки данных, в другом физическом месте и т.д. Нет никакой необходимости постоянно обеспечивать совместный доступ сегментов к другим неразделенным таблицам, расположенным в других сегментах.

Сегментирование упрощает репликацию среди нескольких серверов в отличие от горизонтального разделения. Это полезная концепция для распределенных приложений. Иначе требуется много больше связей между серверами баз данных,

потому что информация не была бы расположена на отдельных логическом и физическом серверах. Например, сегментирование сокращает количество соединений в центре обработки данных, необходимых для запросов к базам данных. Ему необходим механизм уведомления и репликации экземпляров схемы, чтобы неразделенные таблицы оставались синхронизируемыми в той степени, какой требуют приложения. Сегментирование дает большое преимущество в ситуациях, когда распределенные вычисления используются для того, чтобы разделить рабочую нагрузку между несколькими серверами.

Влияние репликации на надежность

На рис. 3.32 показана диаграмма, иллюстрирующая долю данных, которые будут потеряны, когда выйдет из строя n из пяти устройств, а коэффициент репликации K является переменным. Если $K = 1$, то никакой репликации не происходит и у каждого сегмента есть одна копия; если $K = 5$, то происходит полная репликация и все устройства содержат копию. N — это количество устройств Cisco APIC, вышедших из строя. Если $n = 1$, то вышло из строя одно устройство; если $n = 5$, значит, последнее устройство было потеряно.

Рассмотрим вариант, когда $K = 1$, т.е. хранится всего одна копия. Следовательно, при выходе из строя каждого устройства (n от одного до пяти) будет потерян один и тот же объем данных. По мере роста коэффициента репликации K не происходит никакой потери данных, если по крайней мере K устройств не вышли из строя; кроме того, потеря данных носит постепенный характер и начинается с меньшего значения. Например, с тремя устройствами ($K = 3$) никакие данные не потеряны, пока не вышло из строя третье устройство ($n = 3$). При этом теряются только 10% данных. По этой причине контроллер Cisco APIC использует минимум три устройства ($n = 3$).

Влияние сегментирования на надежность

На рис. 3.33 число L представляет собой количество устройств, начинающееся с минимума, равного трем. Поддерживая коэффициент репликации K равным трем, мы вообще избегаем потери данных, до тех пор, пока эти три устройства не выйдут из строя одновременно. Только когда выйдет из строя третье устройство Cisco APIC, произойдет полная потеря данных. Увеличение количества устройств значительно повышает гибкость системы. Например, с четырьмя устройствами, как показано на рис. 3.32, выход из строя третьего устройства означает потерю 25% данных. При 12 устройствах потеря третьего устройства означает потерю только 0,5% данных. При сегментировании, увеличивая количество устройств, можно очень быстро уменьшить вероятность потери данных. Полная репликация не является необходимой для достижения очень высокого показателя защиты данных.

Технология сегментирования

Технология сегментирования обеспечивает возможность масштабирования и надежность наборов данных, генерируемых и обрабатываемых распределенной системой хранения политики, реестром конечных устройств, системой мониторинга

и менеджером топологии. Данные для этих функций контроллера Cisco APIC разделяются на логически связанные подмножества, называемые *сегментами* (shards) (аналогично сегментам базы данных). Сегмент — это единица управления информацией, и все вышеупомянутые наборы данных содержатся в сегментах.

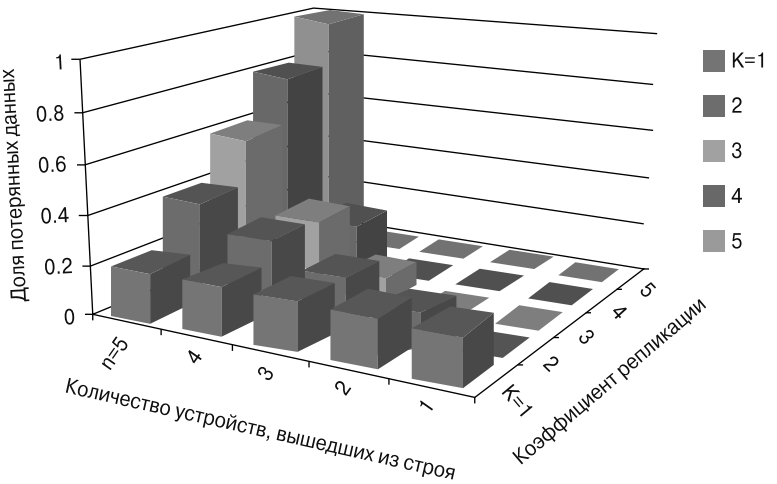

Рис. 3.32. Влияние репликации на надежность

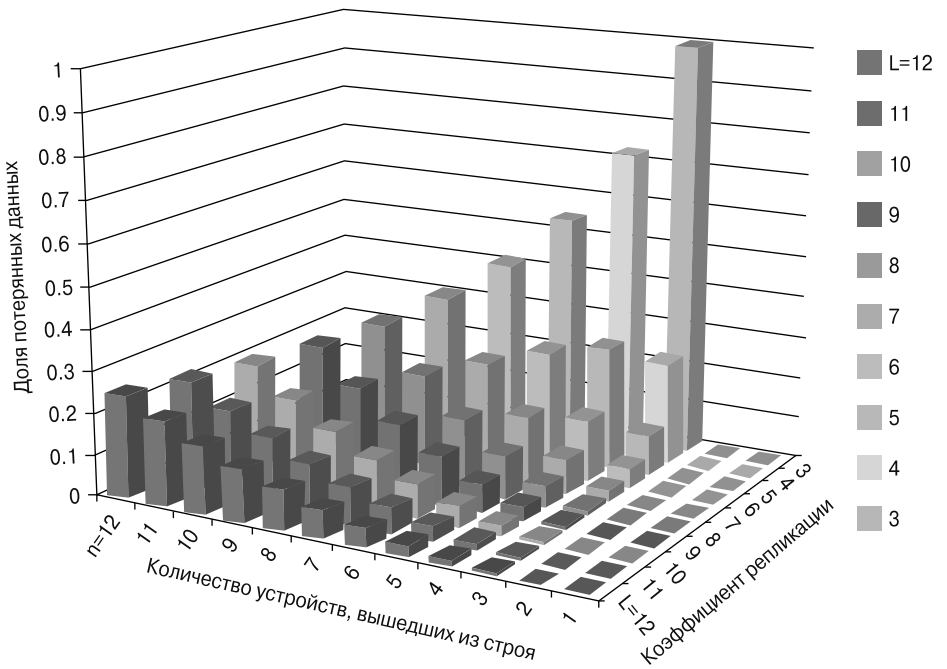

Рис. 3.33. Влияние сегментирования на надежность

Технология сегментирования (рис. 3.34) имеет следующие характеристики.

- Каждый сегмент имеет три реплики.
- Сегменты распределены равномерно.
- Сегменты распределены по принципу горизонтального масштабирования.
- Сегменты уменьшают объем данных для репликации.

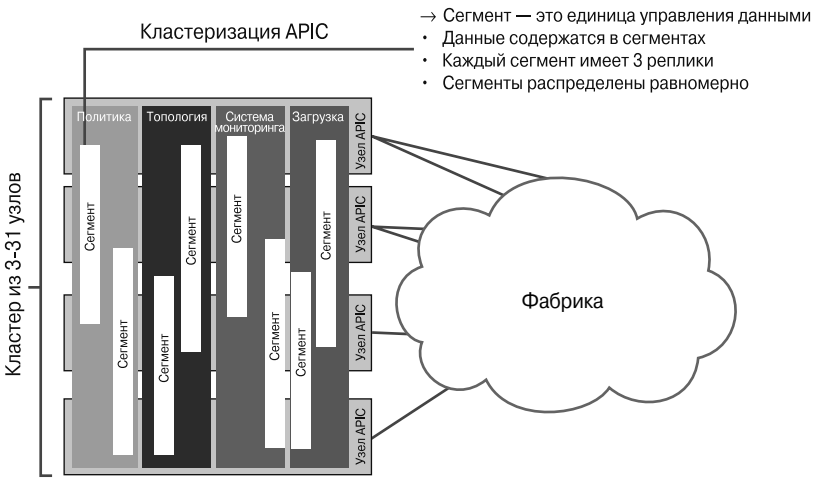

Рис. 3.34. Сегментирование

Один или несколько сегментов располагаются на каждом устройстве Cisco APIC и обрабатываются экземпляром контроллера, расположенным на том же устройстве. Распределение данных по сегментам основано на предопределенной хеш-функции, а назначение сегментов для устройств определяется статической схемой размещения. У каждой копии в сегменте есть приоритет, и записи происходят на копии, которая имеет наибольший приоритет. Остальные копии не допускают записи. При возникновении состояния “разделенной фабрики” (split-brain) автоматическое пересогласование выполняется на основе временны*х* меток. У каждого контроллера Cisco APIC есть все функции; однако обработка равномерно распределяется по всему кластеру Cisco APIC.

Пользовательский интерфейс: графический пользовательский интерфейс

Графический пользовательский интерфейс представляет собой веб-интерфейс, основанный на языке HTML5, который работает на большинстве современных веб-браузеров. Графический пользовательский интерфейс обеспечивает удобный доступ как к контроллеру APIC, так и к отдельным узлам.

Пользовательский интерфейс: командная строка

Интерфейс командной строки обеспечивает полную стилистическую и семантическую (где это возможно) совместимость с интерфейсом Cisco NX-OS CLI. Интерфейс командной строки для всего Cisco ACI доступен через APIC и поддерживает режим транзакций. Есть также возможность получить доступ к определенным узлам Cisco ACI с помощью интерфейса командной строки, предназначенного только для чтения и поиска неисправностей. Поддерживается интегрированный язык сценариев, основанный на языке Python, который позволяет присоединять пользовательские команды к дереву команд так, как будто они были собственными командами, поддерживаемыми платформой. Кроме того, контроллер APIC предоставляет библиотеку для любых пользовательских сценариев.

Пользовательский интерфейс: RESTful API

Контроллер Cisco APIC поддерживает полноценный интерфейс RESTful API на основе языка HTTP(S) с привязками к языку XML и формату JSON. Интерфейс прикладного программирования обеспечивает как древовидный доступ к данным, так и доступ на уровне классов. Стиль REST является стилем архитектуры программного обеспечения для распределенных систем, таких как всемирная паутина. Он появился несколько лет назад и стал основной моделью при разработке веб-сервисов. Со временем стиль REST все больше и больше вытеснял другие модели проектирования, такие как SOAP и WSDL (Web Services Description Language — язык описания веб-сервисов), благодаря своей простоте. Универсальный интерфейс, который должен обеспечивать любой интерфейс REST, считают основным при проектировании любой службы REST. Перечислим основные принципы стиля REST.

- **Идентификация ресурсов.** Индивидуальные ресурсы в веб-системах REST идентифицируются в запросах, например, с помощью универсальных кодов ресурса. Ресурсы концептуально отделяются от представлений, который возвращаются клиенту.
- **Манипулирование ресурсами с помощью представлений.** Когда клиент владеет представлением ресурса, включая присоединенные метаданные, у него достаточно информации для того, чтобы модифицировать или удалить ресурс на сервере, если у него есть разрешение.
- **Самодокументированные сообщения.** Каждое сообщение содержит информацию, которой достаточно для его обработки. Ответы также содержат явную информацию об их доступности в кеше.

Важное понятие в стиле REST — существование ресурсов (источников определенной информации), на каждый из которых ссылаются с помощью глобального идентификатора (такого, как URI в языке HTTP). Для того чтобы управлять этими ресурсами, компоненты сети (агенты пользователя и серверы источника) связываются через стандартизированный интерфейс (такой как HTTP) и обмениваются представлениями этих ресурсов (фактическими документами, содержащими информацию).

Любое количество подключенных объектов (клиентов, серверов, кешей, туннелей и т.д.) может обрабатывать запрос, но каждый делает это, “не видя” основного

запроса (это называется *иерархическим представлением* (layering), являющимся еще одним ограничением стиля REST и общим принципом во многих других стилях информационной и сетевой архитектуры). Таким образом, приложение может взаимодействовать с ресурсом, зная два факта: идентификатор ресурса и требуемое действие. Приложение не должно знать, есть ли кеши, прокси, шлюзы, брандмауэры, туннели или что-либо еще между ним и сервером, фактически содержащим информацию. Приложение должно понимать формат возвращаемого представления, которым обычно является документ HTML, XML или JSON, хотя он может содержать изображение, простой текст или любую другую информацию. Модель документа в формате XML и JSON описана в главе 4.

Доступ к системе: аутентификация, авторизация и RBAC

Контроллер Cisco APIC поддерживает и локальную, и внешнюю аутентификацию и авторизацию (TACACS+, RADIUS, *LDAP* (Lightweight Directory Access Protocol — облегченный протокол доступа к каталогам)), а также *ролевой административный контроль* (Role Based Administrative Control — RBAC), чтобы управлять доступом для чтения и записи на *всех* управляемых объектах, а также осуществлять административное разделение пользователей (как для всей системы, так и для сегментов отдельных арендаторов), как показано на рис. 3.35. Контроллер APIC также поддерживает доменное управление доступом, которое применяется там, где у пользователя есть права доступа (в соответствующих поддеревьях).

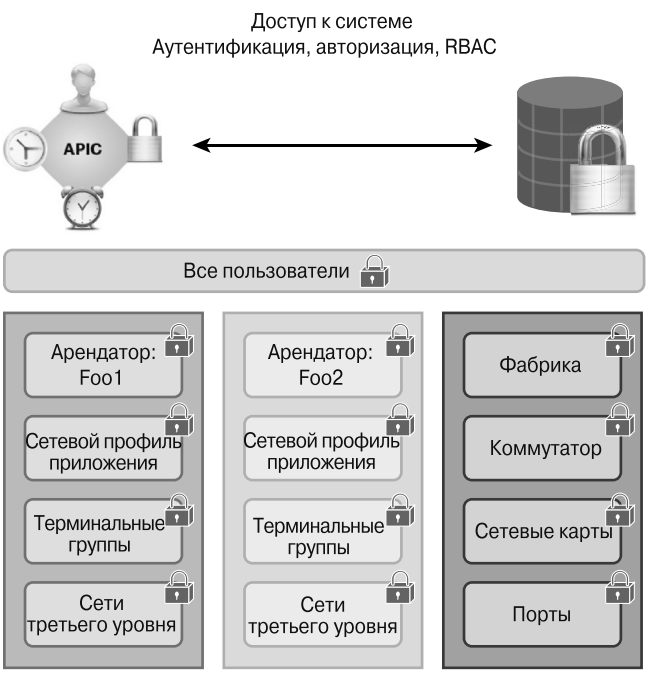

Рис. 3.35. Аутентификация, авторизация и RBAC

Резюме

Модель политики Cisco ACI включает конфигурацию и управление масштабируемой структурой сетевых и служебных объектов. Модель политики обеспечивает развитые и допускающие повторное использование средства управления, многопользовательский режим и содержит минимальные требования к знаниям деталей базовой сетевой инфраструктуры, поскольку эта информация известна контроллеру Cisco APIC. Модель разработана для многочисленных типов центров обработки данных, включая частные и публичные облака.

Инфраструктура ACI обеспечивает логическую модель для описания приложений, которая применяется к структуре контроллера Cisco APIC. Это помогает устранять разногласия между основными эксплуатационными требованиями и сетевыми конструкциями, которые их осуществляют. Модель Cisco APIC разработана для быстрой настройки приложений в сети, которая может быть связана с развитой системой применения политик при поддержании распределенной рабочей нагрузки.

Cisco APIC — это современная, хорошо масштабируемая система распределенного управления, которая управляет элементами коммутаторов Cisco ACI и предлагает автоматизированную сеть, управляемую политикой. Контроллер APIC разработан, чтобы выполнить эту работу, оставаясь при этом вне канала передачи данных, обеспечивая таким образом чрезвычайно высокую производительность инфраструктуры Cisco ACI.