Вісник Харківського національного університету
Серія «Математичне моделювання. Інформаційні технології. Автоматизовані системи управління»
№ 703, 2005, с.135-143

УДК 681.5:629.39.6

Разработка модели сети виртуальных каналов на основе многоагентного подхода

Т. В. Ивановская, К. М. Руккас
Харьковский национальный университет им. В.Н. Каразина, Украина

The paper deals with the multiagent approach to the creation of network control system. The aim of the work is to develop a network simulator system which is the base for future investigation in network resources management.

1. Введение

Последнее десятилетие характеризуется стремительным развитием Интернет технологий. Быстрый рост числа пользователей сетью сопровождался внедрением Интернет технологий во все области жизни. Также быстро развивались и средства связи. Таким образом, на сегодняшний момент структура сети значительно усложнилась, а, следовательно, возросли требования к средствам управления сетью. Количественный рост сети также приводит к необходимости качественно нового управления ею.

2. Основные алгоритмы маршрутизации потоков в сети.

Несмотря на кажущуюся сложность и многообразие, протоколы маршрутизации базируются всего на двух простых алгоритмах, известных уже несколько десятилетий. Для выполнения своей основной функции – управления сетевыми ресурсами - каждый маршрутизатор использует таблицу, в которой отражена топология сети на данный момент времени. В самом общем случае таблица маршрутизации содержит адрес сети назначения, адрес следующего узла на пути к этой сети и метрику (стоимость) пути. Создание и последующее обновление таблицы маршрутизации при изменении топологии сети осуществляется с помощью протоколов маршрутизации. Наибольшей популярностью пользуются протоколы динамической маршрутизации. Алгоритм Беллмана-Форда [1](также известный как алгоритм Форда-Фулкерсона) был положен в основу первого протокола маршрутизации, созданного для сети ARPANET. Так называемые протоколы вектора расстояния (distance vector protocols), такие, как RIP, IGRP, BGP, используют те же принципы. В 1979 году на смену протоколу вектора расстояний пришел протокол состояния канала (link state protocol), ставший основным в ARPANET. Современные протоколы состояния канала включают OSPF, IS-IS, NLSP и др. В настоящее время оба типа протоколов нашли себе применение, так как у каждого из них есть свои достоинства и недостатки.

[1]*Савельев А. Современные протоколы маршрутизации. http://unix.org.ua/routing/1/*

Первый из них, даже с учетом всех улучшений (метода расщепления горизонта, метода временного отказа от приема сообщений – hold-down и т.д.) не решает полностью так называемой проблемы «возрастания до бесконечности» или делает это за счет увеличения реактивности сети. Использование второго метода приводит, с одной стороны, к большой загрузке сети служебной информацией, а с другой – к большей нагрузке на маршрутизаторы, поскольку требует построения полной топологии сети на каждом из них.

3. Современные требования к управлению трафиком и качеству обслуживания (QoS)

В то же время значительно возрос интерес к технологиям, позволяющим предоставлять клиенту необходимый ему уровень услуг (QoS – Quality of Service). QoS представляет собой совокупность технологий, которые позволяют приложениям запрашивать и получать предсказуемый уровень услуг с точки зрения пропускной способности, временного разброса задержки отклика, а также общей задержки доставки данных. В частности, QoS подразумевает улучшение параметров или достижение большей предсказуемости предоставляемых услуг. На сегодняшний день существует два стандарта, представляющих собой два взаимодополняющих подхода управления трафиком: интегрированные службы (Integrated Services - IS) и дифференцированные службы (Differentiated Services - DS). IntServ для явного задания уровня услуги (QoS) использует протокол RSVP. Поставщик интегрированных услуг изучает суммарные требования трафика, ограничивает поддерживаемый трафик объемами, соответствующими текущим возможностям сети и резервируются ресурсы сети для предоставления определенного качества обслуживания в соответствии с конкретными требованиями. Это делается путем уведомления о требовании клиента всех узлов вдоль пути обмена. Если все сетевые устройства вдоль пути могут предоставить запрошенную полосу, резервирование завершается успешно. Дифференцированные службы не пытаются рассматривать суммарные запросы трафика в глобальном смысле и заранее резервировать сетевые ресурсы. Вместо этого в структуре дифференцированных служб трафик классифицируется по группам. На практике DS, вместо того чтобы уведомлять о требованиях приложения, использует в IP-заголовке DiffServ Code Point (DSCP), чтобы указать требуемые уровни QoS. Обе эти службы не всегда эффективно решают задачу качественного обслуживания потоков при ограниченном наборе сетевых ресурсов. IS резервирует потоки без учета требования других потоков, а при использовании DS необходимо заранее разбить трафик на классы, что также снижает эффективность использование сетевых ресурсов.

Из всего вышесказанного следует вывод о необходимости исследования новых схем управления сетевыми ресурсами. Один из перспективных путей - использование многоагентных систем (МАС) для этих целей.

4. Преимущества применения многоагентных систем (МАС) в управлении сетевыми ресурсами.

МАС предназначена для решения широкого ряда вариативных задач. При этом она характеризуется следующими свойствами: автономность, гибкость, ситуативность и социальность. Центральным понятием в теории многоагентных систем является агент. Техническое определение агента следующее: это компьютерная программа, обладающая такими свойствами, как мобильность, автономность, коммуникативность.[5]

МАС идеально подходит для решения задач, имеющих множественные решения, имеет преимущества распределенного и конкурентного решения задач. Поэтому использование многоагентных систем в области динамического управления сетевыми ресурсами является актуальным и перспективным.

5. Общая постановка задачи.

Для решения задач управления сетевыми ресурсами необходимо иметь средства, позволяющие моделировать реальные процессы, происходящие в сети, и оценивать эффективность исследуемых схем. В ходе исследования новых способов управления трафиком была поставлена следующая задача: на основе одной из многоагентных платформ разработать систему моделирования информационных потоков в сети с возможностью управления ими.

6. Выбор основных средств и алгоритмов.

Для построения нашей системы моделирования сети была выбрана многоагентная платформа JADE (Java Agent Development Framework) версии 3.3. После проведения анализа программных средств для разработки приложений, основанных на агентах, было принято решение использовать именно эту платформу, поскольку она совместима со стандартами FIPA [6][2], полностью написана на Java, что обеспечивает независимость от операционной системы, является распределенной (см. рис.1).

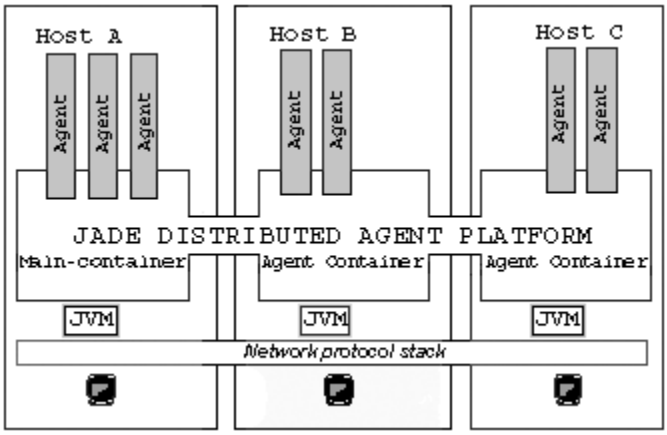

Рис.1.Платформа JADE

Дополнительным преимуществом являются различные средства, позволяющие упростить процесс отладки приложений и визуализировать обмен

[2] *FIPA, Foundation for Intelligent Physical Agents, http://www.fipa.org*

сообщений между агентами приложения, входящие в состав данной агентной платформы, например, Sniffer.

В качестве протокола, управляющего потоками данных в разработанной модели сети, был выбран протокол коммутации меток, сходный с протоколом MPLS (MultiProtocol Label Switching). Относительно новый протокол MPLS, появившийся в 1997 году является наиболее универсальным решением проблем качества обслуживания (QoS), стоящих перед сегодняшними пакетными сетями. Основной идеей протокола MPLS является идея коммутации по меткам. Она заключается в том, что в сети выделяются так называемые ***коммутируемые по меткам тракты*** LSP (Label-Switched Path). При поступлении пакетов в сеть граничный маршрутизатор назначает пакету метку, соответствующую его ***классу эквивалентности пересылки*** – FEC (Forwarding Equivalency Class). Каждому FEC ставится в соответствие LSP, что дает возможность последующим узлам сети просто переключать пакет по метке, не выполняя полного анализа заголовка пакета [3].

7. Описание функционирования системы.

Для эффективного управления сетевыми ресурсами над реальной физической сетью строится сеть логических путей. Управление ресурсами осуществляется с помощью изменения топологии логической сети, а также с помощью перераспределения в ней информационных потоков.

Каждый узел в системе представлен JADE агентом. Отдельным агентом представлен контроллер сети, который создает агентов – узлов, инициализирует их и запускает сеть.

В каждом агенте сети хранятся 3 таблицы:

1 – предназначена для хранения информации о физических связях данного узла с другими, такой, как узел, ширина канала, зарезервированная ширина канала (Physical Link Table).

2 – таблица меток (Label Switching Table), предназначенная для автоматической маршрутизации трафика

3 – таблица запросов (Request Table). Данная таблица предназначена, с одной стороны, для хранения начальных запросов по предоставлению соединения с заданной полосой пропускания, а с другой - для хранения запросов от других узлов и результатов обработки данного запроса (удалось или не удалось установить соединение с заданной шириной канала).

Общая схема работы модельной сети выглядит следующим образом: каждый агент, представляющий узел сети, после инициализации и получения сообщения о запуске сети начинает посылать запросы из Request Table соответствующему узлу и параллельно отвечает на запросы других узлов, выдавая информацию об установлении соединения. Конечным итогом работы будут заполненные таблицы запросов у всех агентов, что дает ответ на вопрос, удается ли при данной конфигурации физической и логической сети обеспечить клиентов требуемыми ресурсами.

Например, на рис. 2 и рис. 3 представлена физическая и логическая топология сети, соответственно, состоящей из 4 узлов (Node1, Node2, Node3, Node4).

На рисунке 3 изображены следующие логические пути:

LSP0: Node3 – Node1,
LSP1: Node4 – Node2- Node1,
LSP2: Node3 –Node1,
LSP3: Node1 – Node2 - Node4,
LSP4: Node4 – Node3

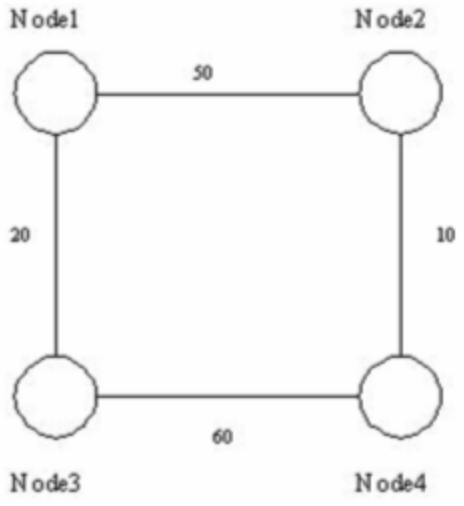

Рис.2.Физическая топология сети

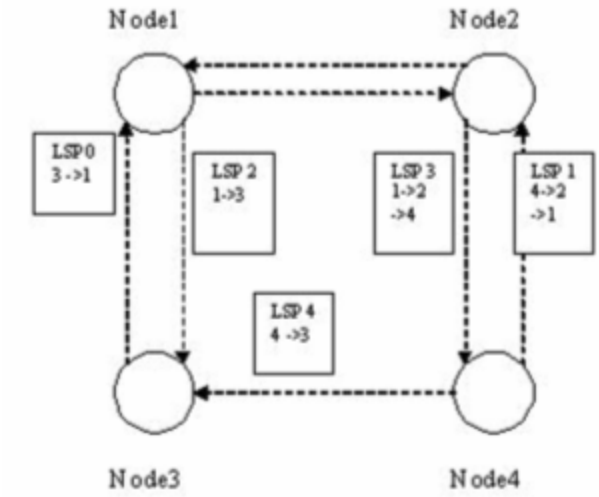

Рис.3.Логическая топология сети

Сплошной линией обозначены физические связи между узлами, цифрами обозначены ширины физических каналов. Штрих - пунктиром обозначена топология логической сети. Итак, класс – контроллер сети (назовем его NetController) считывает из конфигурационного файла nodes.xml имена агентов - узлов и значения ширины канала между ними, создает агентов и отсылает им таблицы физических связей. Например, для агента Node1 таблица физических связей будет выглядеть следующим образом:

```
<node name="node1">
```

```
    <physical_link>
      <to_node name="node2" bw="50"/>
      <to_node name="node3" bw="20"/>
    </physical_link>
</node>
```

это означает, что этот узел связан с узлами Node 2 и Node3 с шириной канала 50Mb/s и 20 Mb/s соответственно. Как только NetConroller получает уведомления от агентов о том, что инициализация таблицы физических связей прошла успешно, он начинает считывать из конфигурационного файла логических путей данные, формировать Label Switching Tables и последовательно рассылать их агентам. Рассмотрим это на конкретном примере:

```
<lsp id="0">
    <node name="node3"/>
    <node name="node1"/>
</lsp>
<lsp id="1">
    <node name="node4"/>
    <node name="node2"/>
    <node name="node1"/>
</lsp>
```

Контроллер сети считывает составляющие LSP0, формирует строчку таблицы для Node3 следующего вида: Label:-1; Node: Node1 – это означает, что Node1 – последний узел в данном LSP. Напомним, что метка означает смещение в таблице меток следующего в данном пути агента. Затем NetController считывает составляющие LSP1 посылает узлу Node2 запрос о № последней строке таблицы в его LST. Node2 возвращает 0 (так как его LST пока пуста), тогда NetController формирует строчку таблицы для Node4 следующего вида: Label:0; Node: Node2, далее он формирует строчку таблицы LS для агента Node2 и так далее.

Теперь NetController считывает из конфигурационного файла запросов данные для формирования Request Tables у агентов, например, зададим следующие параметры:

```
<requests>
  <request id="0" s_n_name="node4" d_n_name="node1"
label="0" bw="10"/>
  <request id="1" s_n_name="node1" d_n_name="node2"
label="1" bw="60"/>
  <request id="2" s_n_name="node1" d_n_name="node3"
label="0" bw="20"/>
  <request id="3" s_n_name="node3" d_n_name="node1"
label="0" bw="15"/>
</requests>
```

т.е. первый запрос – из Node4 в Node1 с шириной канала 10Mb/s и меткой 0 (по таблице меток узла Node 4 это означает, что мы выбираем логический путь LSP1). Далее NetController формирует строки таблицы запросов для каждого

агента - источника и отсылает их. Когда все необходимые данные отосланы, контроллер посылает всем агентам уведомление о начале работы. И агенты начинают общаться для того, чтобы получить результат своих запросов, занесенных в их Request Table.

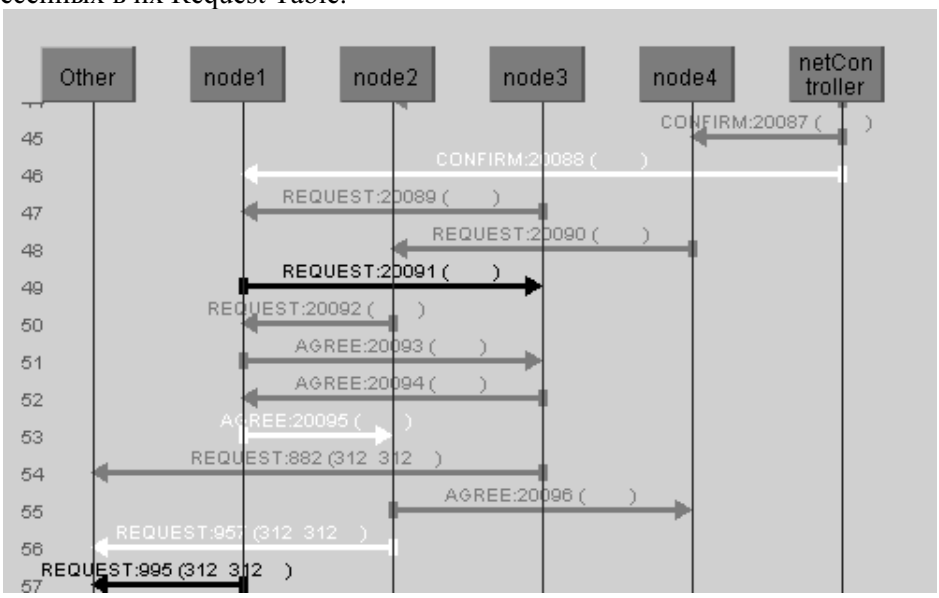

Рис.4.Последовательность запросов, перехваченная sniffer'ом Jade

Рассмотрим последовательность обработки запроса 0 (см. рис.4, 5). Исходная строка запроса записана в таблице агента Node4. Он посылает запрос (REQUEST)агенту Node2 (так как мы идем по пути LSP1):

```
 node4 sent request to node2
(REQUEST
 :sender  (node4 )
 :receiver  (node2)
 :content  "((action (agent-identifier :name node4)
 (RequestPackage :RequestID 0 :AskedBW 10 :RowLabel 0
:DestinationAID (agent-identifier :name node1))))"
:language  fipa-sl  :ontology  PhLinkDataOntology )
```
[3]

Агент Node2, получив данный запрос, проверяет, что он не является конечным пунктом назначения, получает по таблице меток следующий узел, проверяет по таблице, хватает ли полосы пропускания и пересылает запрос агенту Node1. Node1, получив запрос, дает на него положительный ответ Node2, который «по цепочке» возвращает положительный ответ Node4. Этот агент, получив положительный ответ, вносит данные в свою таблицу запросов и проверяет, есть ли еще не отосланные запросы. Если их больше нет, то отсылает данные контроллеру. Контроллер в это время ожидает таких сообщений от всех агентов. Как только все агенты пришлют их – контроллер «скажет» агентам-узлам «остановить общение» и выдаст результаты. В указанном примере результаты работы симулятора следующие:

[3] *Вид сообщения JADE сокращен для удобства чтения*

Source:node3 || Destination:node1 || BW:15 || Result:1
Source:node1 || Destination:node2 || BW:60 || Result:0
Source:node1 || Destination:node3 || BW:20 || Result:1
Source:node4 || Destination:node1 || BW:10 || Result:1

8. Выводы по результатам и направления дальнейших исследований.

В результате проделанной работы был разработан симулятор сети на основе многоагентной платформы JADE.. Также был разработан и внедрен протокол управления ресурсами сети, основанный на MPLS.

С помощью данного симулятора предполагается провести исследование в области управления нагрузкой сети сложной топологии. Также планируется осуществить поиск новых интеллектуальных алгоритмов управления сетевыми ресурсами.

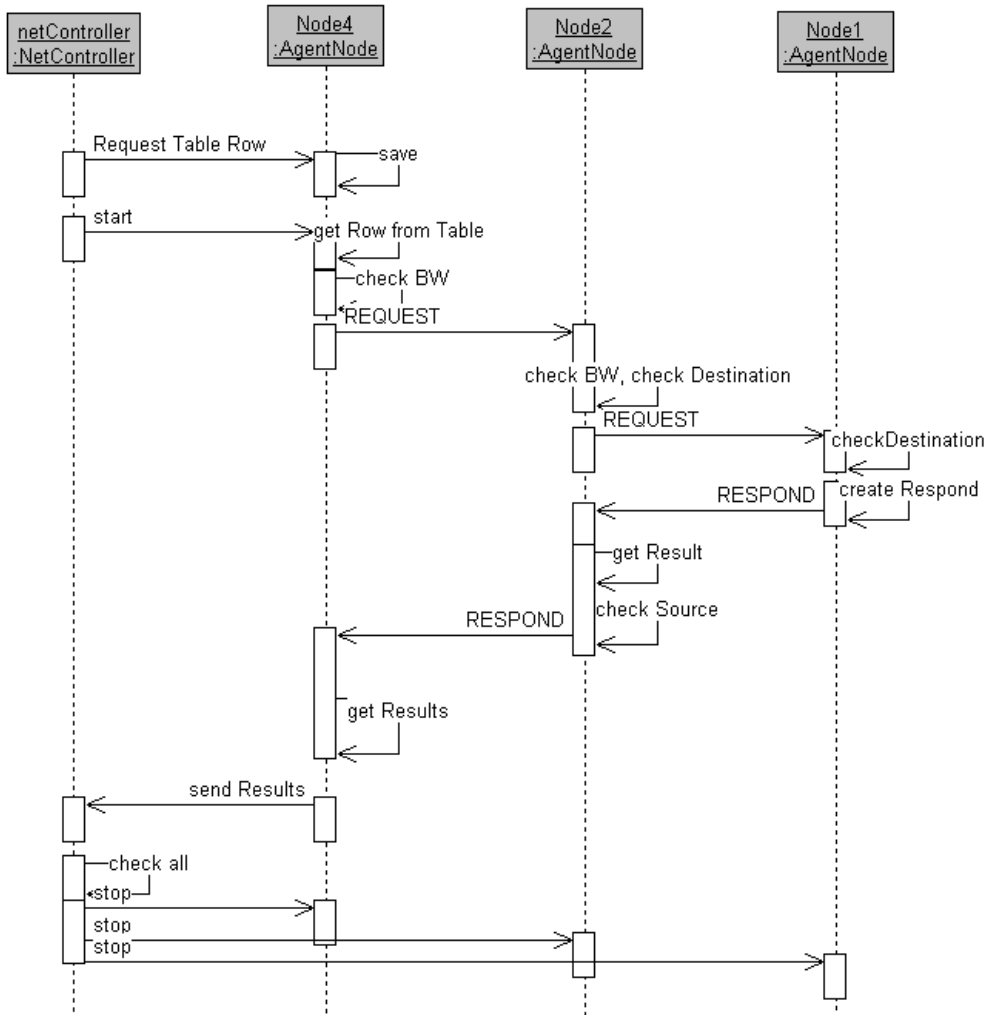

Рис.5.Диаграмма последовательности выполнения запроса на соединение

ЛИТЕРАТУРА

1. Santiago Cots, Teodor Jové, Pere Vilà. A Call-level Network Simulator Framework based on a Standard Agent Platform. - Broadband Communications and Distributed Systems Group Universitat de Girona, Girona, Spain.
2. Pere Vila, Josep L. Marzo, Antonio Bueno. Automated network management using a hybrid multiagent system. – Institut d'Informatica I Aplicacions, Universitat de Girona, Girona, Spain.
3. Гольдштейн А. Б., Гольдштейн Б. С. Технология и протоколы MPLS. – СПб.: БХВ – Санкт-Петербург, 2005. – 304 с.
4. Руккас К.М. Разработка модели базы знаний агентов многоагентной системы динамического управления компьютерными сетями. // Вестник Харьковского национального университета имени В.Н. Каразина – Выпуск 629. – Харьков: ХНУ, 2004. - С.165 – 170.
5. Тарасов В.Б. От многоагентных систем к интеллектуальным организациям: философия, психология, информатика. - М.: Эдиториал УРСС, 2002. – 352 с.