5. *Ghidini C., Giunchiglia F.* Local Models Semantics, or Contextual Reasoning = Locality + Compatibility // Artificial Intelligence. 2001. P. 221–259.
6. *Goh C. H.* Representing and Reasoning about Semantic Conflicts in Heterogeneous Information Sources // MIT. 1997. P. 2–37.
7. Peer-to-Peer Computing / D. S. Milojicic, V. Kalogeraki, R. Lukose et al. // HPL-2002-57, HP Laboratories, 2002. P. 2–54.

Н. А. ЗЕНЬКЕВИЧ

ПРИМЕНЕНИЕ МНОГОАГЕНТНЫХ ТЕХНОЛОГИЙ ДЛЯ МОНИТОРИНГА И УСТРАНЕНИЯ НЕИСПРАВНОСТЕЙ В ИНФОРМАЦИОННЫХ СЕТЯХ

С точки зрения обеспечения функций управления в числе ключевых особенностей архитектуры современных информационных сетей (ИС) можно выделить следующие: 1) большое количество компонентов (узлов, сервисов, источников данных); 2) распределенность – взаимозависимые компоненты одной корпоративной сети распределены в пространстве, и возможно, связаны через внешние компоненты; 3) гетерогенность – различные компоненты разработаны различными производителями, работают под управлением различными операционными системами и не имеют общих интерфейсов управления и контроля.

Перечисленные особенности обуславливают сложность организации мониторинга таких сетей. Во-первых, большое количество компонентов и их распределенность приводят к большому количеству управляющего трафика (в некоторых случаях он может достигать половины всего объема) [1]. Во-вторых, сложность и взаимосвязанность системы вызывает в случае отказа одного из компонентов большое число сообщений об отказах от других компонентов [2, 3]. В-третьих, сложность элементов не позволяет оценить работоспособность с помощью только пассивного мониторинга (анализа лог-файлов и контроля параметров, в отличие от активного мониторинга - эмуляции действий пользователя). В-четвертых, распределенность элементов усложняет проведение всеобъемлющего активного мониторинга из одного узла сети [4]. Данные проблемы можно разрешить путем применения для мониторинга ИС многоагентных систем (МАС) [5], однако при разработке таких систем неизбежно возникают новые проблемы: возросшая сложность агентов усугубляется гетерогенностью сети, а также дополнительная нагрузка ложится на узлы.

Целью данной работы является анализ особенностей мониторинга и поддержки работоспособности информационной системы с помощью МАС, а также разработка такой архитектуры МАС и отдельного агента, которые позволят реализовать весь спектр действий по мониторингу и

восстановлению работоспособности с возможностью гибкого подключения различных реализаций отдельных действий.

Архитектура конкретной МАС зависит не только от решаемой задачи, но и выбранной платформы. Для преодоления гетерогенности ИС, на наш взгляд, целесообразно воспользоваться платформой на основе Java [5]. Чтобы уменьшить нагрузку на узлы, желательно использовать специализированную платформу МАС для управления сетями, например, FIPAOS, или же собственную.

Задачу мониторинга ИС можно разбить на три подзадачи: построение модели сети, распределение агентов по узлам и задач между агентами и непосредственно сама работа системы (включая восстановление) [2, 3, 6].

Модель сети должна полным образом отображать функциональные зависимости элементов сети, а также ее топологию. Предлагается строить модель в виде фреймовой структуры (дерева объектов), в которой некоторые слоты будут представлять замеренные агентами параметры. Такую модель легко преобразовать во фреймовую структуру экспертной системы, байесовскую или нейронную сеть, диаграмму влияния. Также такая модель позволит с легкостью сохранять и применять накопленный опыт, а также построить онтологию для коммуникации между агентами. В объектно-ориентированном языке (таком как Java) такую модель легко хранить в виде иерархии объектов. Рассмотрим пример такой модели для трех узлов (рис. 1).

В модели стрелка лишь указывает на то, что сбой будет распространяться в ее направлении, движение по модели может осуществляться в любом направлении (фреймы содержат слоты-ссылки как нижестоящие, так и выше стоящие элементы). Полная модель сети есть у каждого аген-

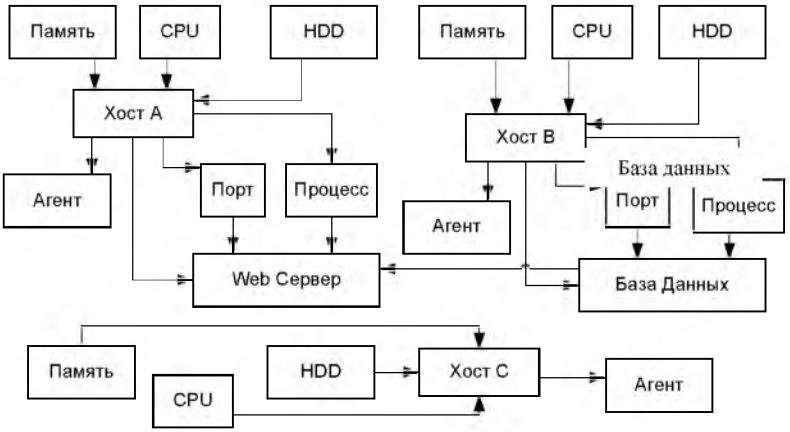

Рис. 1. Пример модели информационной сети

та, однако лишь ее часть будет наполнена данными мониторинга (теми, что он замеряет сам и запросил у других агентов). Весь обмен знаниями происходит на основе этой модели (передается часть модели с данными). Как видно модель хранит лишь текущие значения замеряемых параметров, историю предполагается накапливать отдельным механизмом, с возможностью ее анализа.

Построение модели администратором сети должно выполняться при помощи визуального компонента, входящего в состав системы. Эта модель должна поддерживаться и изменяться с изменением сети, для чего надо проводить периодическое сканирование сети (это только дополнительная возможность и должна выполняться в регламентное время, равно как и передача истории мониторинга в единый центр на хранение и анализ).

Различные подходы используются при мониторинге с помощью статичных или мобильных агентов. Статические агенты собирают информацию и затем обмениваются только необходимой для интеллектуального обнаружения сбоев. Мобильный агент в одиночку собирает всю необходимую информацию, перемещаясь по каркасу MAC и переносит ее в точку обработки информации. Таким образом, распределение мобильных агентов сводится к указанию времени и места их появления и алгоритма передвижения. На взгляд автора мониторинг лучше производить при помощи статичных агентов, т.к. будет сделан более быстрый сбор информации и ее локальная обработка, а также упростится активный мониторинг и снизится нагрузка на сеть и узлы.

Таким образом, на втором этапе предлагается разместить узлы каркаса по всем возможным узлам сети, содержащих серверные элементы, а также несколько узлов в подсетях пользователей для проверки доступности серверных узлов. В каждом узле предлагается разместить статичного агента, который будет выполнять мониторинг, обнаруживать сбои на основе этой информации, а затем производить более детальный анализ с помощью информации, запрошенной у других агентов. Первоначальные решения будут автоматически согласовываться при помощи обмена информацией мониторинга, т.к. предполагается, что все агенты одинаковы (одинаковая модель, одинаковые правила для обнаружения сбоев и выяснения их первопричины) и, обладая одинаковой информацией, они придут к единому мнению. Для восстановления будет использоваться мобильный агент, который, перемещаясь по нужным узлам, выполнит выработанную последовательность действий.

Итак, для каждого агента должен быть обозначен круг элементов, за которые он отвечает: производит мониторинг и может участвовать в восстановлении. В модели сети агенты, обслуживающие данный сервис, будут отмечены в качестве слотов фрейма этого агента и наоборот. Пред-

Таблица 1

Пример распределения задач мониторинга между агентами.

Агент A	Агент B	Агент C
A: память, CPU, HDD, Web Server, порт, процесс **B:** агент, хост, база данных, порт **C:** агент, хост	**A:** агент, хост, Web Server, порт **B:** память, CPU, HDD, база данных, порт, процесс **C:** агент, хост	**A:** агент, хост, Web Server, порт **B:** агент, хост, база данных, порт **C:** память, CPU, HDD

полагается распределять элементы между агентами автоматически (изначально и затем в случае отказа агента) с возможностью дальнейшего вмешательства администратора. Агент должен обслуживать все локальные для него элементы, плюс каждый элемент, который может тестироваться удаленно, должен находится в обслуживании еще у одного-двух агентов. В нашем случае распределение задач мониторинга может быть следующим (табл. 1).

Отметим, что, например, Web Server тестируется тремя способами: эмитируются пользовательские запросы (проверяется работоспособность всех технологий используемых на сервере, например, php, perl, ssi), проверяется открытый порт, и запущен ли процесс (служба). Причем, некоторые действия выполняют как удаленно, так и локально - это позволяет получить более детальную информацию.

Многие источники выделяют этапы решения задачи поддержки работоспособности системы: получение или сбор информации (мониторинг), фильтрация событий, корреляция событий, диагностика сбоя, выработка планов действия, приведение их в исполнение, проверка работоспособности восстановленных компонентов [1, 2].

В предлагаемой архитектуре МАС статичных интеллектуальных агентов для сбора информации и выработки решения, фильтрация информации не нужна, т.к. агент самостоятельно определяет какая информация ему необходима. Корреляция (в отличие от архитектур построенной на ловушках) также будет не сложна т.к. агент владеет событиями только интересующих его элементов и может осуществляться при помощи простых правил одновременно с определением первопричин.

Как уже отмечалось, предполагается использовать различные интеллектуальные технологии для обнаружения сбоев, выяснения первопричин и построения решения, построения планов с целью исследования эффективности различных подходов в такой архитектуре. Таким образом, предлагается тестировать свободно распространяемые: экспертную систему, основанную на правилах, например JESS; байесовскую сеть (например, JavaBayes); нейронную сеть (например, JOONE), собственную CBR систему на основе модели сети, т.е. в состав объектов модели

Рис. 2. Алгоритм поведения статичного агента

будут входить методы для поиска соответствующего случая в хранилище.

Обеспечение безопасной работы системы предполагается реализовать при помощи криптографической защиты всех передаваемых сообщений.

Приведем один из основных алгоритмов предлагаемой архитектуры алгоритм поведения статичного агента (рис. 2). Поведение агента усложняется тем, что он действует в кооперации с другими, но при этом асинхронно, а также тем, что новые поступающие данные могут препятствовать корректному решению (разные данные на разных этапах решения), поэтому пока цикл работы не завершается, данные находятся в буфере и только затем попадают в модель.

К сожалению, эта модель не отображает способа объединения двух проблемных областей в одну и способов различения двух проблемных областей. Однако она отображает механизм кооперации при обнаружении сбоев, что опускается во многих работах и системах [2, 3, 6]. Пренебрежения данной проблемой может привести к тому, что несколько агентов могут прийти к различным решениям и попытаться воплотить их. Как уже отмечалось, в данной работе согласование мнений происходит автоматически. Реализация согласования на основе диалогов или доски объявлений потребует более сложных алгоритмов [5].

Полученную архитектуру агента, реализующую данный алгоритм, следует отнести к типу вертикальных [5]. При этом проведение мониторинга будет являться проактивным поведением. Полученные алгоритмы поведения, архитектуры и модели могут быть обобщены на случай других объектов мониторинга.

ЛИТЕРАТУРА

1. Bieszczad A., Pagurek B., White T.. Mobile Agents for Network Management // IEEE Communication Surveys 1. 1998. Vol. 1, № 1. (http://www.comsoc.org/pubs/surveys/4q98issue/bies.html)
2. An Artificial Intelligence Approach to Network Fault management/ D. W. Gürer, I. Khan, R. Ogier, R. Keffer. // http://www.sce.carleton.ca/netmanage/docs/An_AI_Approach.pdf).
3. Ricciulli L., Shacham N., Park M. Modeling Correlated Alarms in Network Management Systems // http://www.metanetworks.org/pubs/ricciulli96modeling.ps.gz.
4. Welter P. Web Server Monitoring // http://www.freshtech.com.
5. Тарасов В. Б. От многоагентных систем к интеллектуальным организациям: философия, психология, информатика. Эдиториал УРСС, 2002. 352 с. (Науки об искусственном).
6. Gürer D. W., Lakshminarayan V., Sastry A. An Intelligent-Agent-Based Architecture for the Management of Heterogeneous Networks // http://www.erg.sri.com/publications/433-pa-98-120.pdf.