УДК 681.518:004:912

А. Л. Ерохин, А. П. Турута

ИДЕНТИФИКАЦИЯ НЕШТАТНЫХ СИТУАЦИЙ В ИНФОРМАЦИОННЫХ СЕТЯХ

1. Введение

Современный этап развития систем и средств искусственного интеллекта делает актуальной интеллектуализацию информационных сетей. В статье рассматривается разработка метода идентификации нештатных ситуаций в информационных сетях на примере *web*. При современной интенсивности работы сетей обеспечение бесперебойной доставки целевого продукта (информации) становится все более сложной задачей. Можно выделить два подхода в ее решении. Первый связан с аспектами защиты информации и основан на разработке методов и средств обеспечения целостности и конфиденциальности информации в сетях. Второй связан с тем, что в массовых информационных сетях (в частности, *web*), наряду с обеспечением конфиденциальности и целостности, актуальной становится задача обеспечения доступности ресурсов сети. Примером нарушения доступности ресурсов является нештатная ситуация типа «отказ в обслуживании».

Следует заметить, что решение проблем целостности и конфиденциальности информации в сетях находится не только в технической плоскости, но и в юридической. Так, Police and Justice Bill 2006, принятый в Великобритании, к разряду уголовных преступлений относит умышленный вывод из строя любой компьютерной системы [1]. В указанном проекте ответственность за атаку может быть возложена на заказчика атаки, непосредственного исполнителя, владельца бот-сетей, провайдера. Отсутствие в Украине норм права или законопроекта, регулирующего ответственность за совершение атак, направленных на отказ в обслуживании, скорее указывает на то, что такие атаки не фиксируются и не диагностируются, а не на отсутствие проблемы. Поэтому идентификация нештатных ситуаций, связанных с нарушением доступности ресурсов информационных сетей, является актуальной задачей.

2. Анализ предметной области и постановка задач исследования

К причинам нарушений доступности ресурсов относятся: 1) атака типа «отказ в обслуживании» (Denial of service — DoS); 2) распределенная атака типа «отказ в обслуживании» (Distribution denial of service — DDoS); 3) увеличение интенсивности использования ресурсов сети.

Причинами возникновения DoS могут быть: ошибки проектирования системы; большой объем сетевого трафика, заполняющего каналы связи (КС); большое количество пакетов, заполняющих очередь обслуживания и снижающих скорость передачи данных; повышение вычислительной сложности процесса, занимающего свободные ресурсы системы обслуживания; неправильный расчет нагрузки.

Наряду с умышленными атаками, угрозой для работы сети может стать ее расширение или изменение структуры. После обновления структуры сети может возникать ситуация, когда ресурсов требуется более, чем планировалось. Ущерб от DDoS определяется бесполезным простоем системы. Так, в работе [2] предлагается доход от работы порталов оценивать по формуле $P=((R_i-C_i)*L_i)-S_i$, где R_i — доход от i-го клиента; C_i — затраты на обслуживание i-го клиента; L_i — время жизни (работы) i-го клиента, выраженное в количестве взаимодействий; S_i — стоимость привлечения клиента.

DDoS существенно влияет на параметр L_i, чем уменьшает доход. Поэтому актуальным является уменьшение времени воздействия атаки, обнаружение и локализация источников атаки, использование алгоритмов восстановления нормального режима работы системы.

Для предотвращения атак типа «отказ в обслуживании» предлагаются различные подходы: «обнаружение сигнатур», анализ статистики, анализ заголовков сетевых пакетов [3]. Для ускорения выполнения запросов, для увеличения полосы пропускания каналов связи дублируются *web*-серверы, устанавливаются дополнительные *proxy*-серверы, обслуживающие запросы в направлении к *web*-серверам, а также дополнительные каналы связи.

До сих пор актуальной является разработка алгоритмов и методов выявления подготовки атак типа «отказ в обслуживании», определение источников атаки и их локализация. Необходимы способы организации систем обслуживания и хранения ресурсов, обеспечивающие эффективное обслуживание запросов. Сетевые атаки и системы защиты от них находятся в постоянном взаимодействии и развитии, поэтому любая система, которая не может вовремя адаптироваться и обучаться, рано или поздно станет уязвима. Для построения надежной системы необходимо разработать механизм самовосстановления и самообучения системы защиты.

Таким образом, для обеспечения бесперерывного предоставления информации в информационной сети необходимо решить следующие задачи:

1. Идентификация распределенных атак DoS на основе анализа информации о состоянии каналов связи, *web*-серверов, подконтрольных *proxy*-серверов, маршрутизаторов, результатов распознавания категорий сетевых агентов, а также накопленных

статистических данных. Решение этой задачи состоит из этапов:

1.1. Определение пути самовосстановления системы при идентификации отказа в обслуживании. Система может управлять следующими параметрами: блокировать каналы связи; отказывать в обслуживании отдельным категориям; менять приоритеты доступа и обслуживания.

1.2. Идентификация ситуации, когда система не способна самостоятельно перейти в нормальный режим работы, то есть необходимо внешнее вмешательство лица, принимающего решение (ЛПР). Определение порядка информирования ЛПР, выработка алгоритма внешнего воздействия, сбор и анализ информации, предшествующей отказу и сопровождающей его.

1.3. Построение временных портретов поведения каналов связи, *web*-серверов, промежуточных серверов для выявления аномалий в работе.

С первой задачей связана задача идентификации категории подключившихся сетевых агентов (СА).

2. Идентификация категории СА, запрашивающих файл, на основе: признаков запрашиваемого файла; анализа контента, передаваемого к серверу; анализа параметров, определяющих расположение категории СА; признаков, указывающих на систематичность работы СА; анализа параметров, свидетельствующих о суточных ритмах работы категорий СА. Решение этой задачи состоит из трех этапов:

2.1. При идентификации категории СА необходимо разработать методы выяснения признаков на основе: когнитивного вопроса; анализа области выбора; оценки поведения.

2.2. Для выявления поведения, систематичности алгоритмов необходимо решить задачу составления подключений: определение различных подключений одного СА; выявление признаков тематик или групп файлов среди контента серверов; определение точек входа и выхода при просмотре контента файла.

2.3. Построение временных портретов и портретов поведения категорий СА для обнаружения отклонений.

Полученные результаты при идентификации отказа в обслуживании и категорий СА можно использовать для улучшения работы *web*-сервера.

3. Улучшение характеристик обслуживания запросов *web*-серверами на основе: изменения приоритетов КС, обслуживания запросов, выбора ответов для категорий СА. Как функцию цели предполагается использовать: максимальный объем отданной информации; максимальное количество обслуженных СА в пределах Δ-изменений информации; максимальную скорость при обслуживании; минимальную нагрузку на сервер; минимальную нагрузку на канал связи.

4. Прогнозирование изменений нагрузки на *web*-серверы, подконтрольные *proxy*-серверы, маршрутизаторы, каналы связи на основе: анализа истории нагрузок; мгновенных оценок нагрузки; результатов идентификации категории СА и отказов в обслуживании; анализа соотношения ситуации с «портретами», полученными в предыдущих задачах; анализа обобщенной информации о категориях СА и запрашиваемых файлах.

Полученные результаты прогнозирования, улучшения и идентификации соотносятся с фактически полученными результатами и, в случае необходимости, корректировками «учителя». Для адаптации разработанной системы необходимо решить задачу обучения системы.

5. Задача обучения правилам идентификации категорий СА и отказа в обслуживании, прогнозирования нагрузки и улучшения характеристик работы систем на основе двух подходов: автоматического обучения на ошибках; корректировки «учителя».

Первый подход должен обеспечить обучение системы на основе расхождения прогнозных данных и реально наступивших событий. Автоматическое обучение сэкономит время человека при настройке системы.

Второй подход используется в таких случаях, когда полученные расхождения не позволяют системе произвести самонастройку. В таком случае система должна предоставить когнитивные ответы «учителю» для внесения необходимых корректировок.

3. Исследование модели работы информационной сети *web*

Общая модель работы глобальной информационной сети *web* рассматривается в [4–7]. Предлагается выделить функции, влияющие на процесс бесперерывного предоставления информации (рис. 1).

Представим функциональную модель сети следующими объектами:

1. Сетевые агенты, или клиенты, — объекты сети, которые имеют сетевой идентификатор, подключены линиями связи к компьютерной сети. Основной задачей сетевых агентов является взаимодействие (получение и отправка информации) с серверами.

2. Сервер — объект сети, основной задачей которого является обслуживание запросов клиентов (сетевых агентов). Сервер имеет сетевой идентификатор (IP-адрес), характеризуется системными ресурсами, контентом и способностью обслуживать запросы на статические и динамически формируемые файлы.

3. Доверитель — объект сети, имеющий сетевой идентификатор. Основная задача — обслуживать запросы СА, предоставлять статические файлы или обеспечивать доступ СА к *web*-серверам. Характеризуется системными ресурсами. Доверитель может содержать 0 файлов, тогда он называется маршрутизатором.

В таблице 1 представлены признаки, используемые для разделения функций *web*-серверов и подконтрольных *proxy*-серверов.

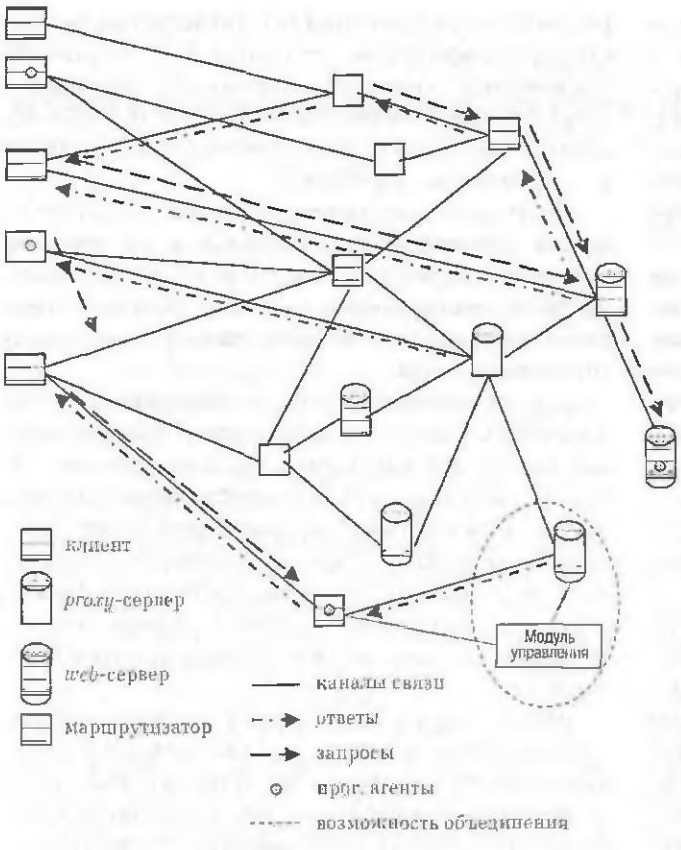

Рис. 1. Модель работы *web*

Таблица 1

Отличительные признаки *web*- и *proxy*-серверов

Признаки	*Web*-сервер	*Proxy*-сервер
Содержание ответов	Генерирует контент, динамически создает страницы, предоставляет статические файлы.	При первом запросе перезапрашивает файл на *web*-сервере, далее хранит статические файлы. Динамически формируемые файлы перезапрашивает всегда.
Интеллектуальная деятельность	Обрабатывает параметры запросов, работает с маршрутами, выделяет заданные признаки, генерирует страницы.	Обрабатывает параметры запросов относительно статических файлов, работает с маршрутами, принимает команды управления.
Стоимость простоя, требования к ресурсам	Высокая стоимость, длительна и нежелательна перезагрузка. Больше нуждается в вычислительных ресурсах.	Незначительная стоимость, быстрая перезагрузка. Больше нуждается в ресурсах для хранения и пересылки данных.
Приоритеты	Генерация динамических страниц, предоставление постоянных файлов, выделение заданных признаков.	Пересылка пакетов и данных, предоставление статических страниц, выделение заданных признаков.
Дополнительные функции	Функция управления.	Функция «предохранителя».
Контент	Динамические или статические файлы.	Статические файлы или ∅.

4. Маршрутизатор — объект, используемый в компьютерных сетях передачи данных, который на основании информации о топологии и определенных правил принимает решение о пересылке пакетов сетевого уровня их получателю. Характеризуется уникальным сетевым идентификатором, нагрузкой и производительностью.

5. Модуль управления — объект, собирающий информацию о других объектах сети, обрабатывающий ее, в случае необходимости может выполнить предписанный алгоритм действий. Работает в составе сервера или отдельно. Может выполнять функции «черного ящика» и реализовывать алгоритм восстановления в случае, если сервер не способен самостоятельно перейти в нормальный режим работы.

6. Каналы связи — объекты, соединяющие другие объекты сети. КС классифицируются по следующим признакам:

6.1. Подконтрольность — контролируемые линии связи (между серверами и маршрутизаторами); неконтролируемые (остальные, во внешней сети).

6.2. Назначение — служебные (связи серверов с управляющим модулем); информационные (используются для получения/отправки информации сетевыми агентами).

6.3. Длительность — маршруты; отдельные участки.

Под линиями связи (ЛС) будем понимать физическое соединение устройств, под каналами связи будем понимать логически объединенное соединение, основанное на ЛС между объектами сети, влияющими на маршрутизацию пакетов. На основе предложенной классификации предлагается выделить следующие виды линий связи: служебные; контролируемые; маршруты; внешние (неконтролируемые); информационные.

Контролируемые ЛС ограничены серверами или маршрутизаторами с обоих концов. Прохождение трафика по такой ЛС контролируется модулем управления.

7. Запросы клиентов — созданные сетевыми агентами специальные требования предоставить файл, передаются по сети по направлению к серверу-обработчику. Запросы характеризуются: *адресом отправителя, адресом назначения* (обработчиком), фактическим маршрутом, содержанием заголовка запроса, объемом.

8. Ответы — результаты обработки *proxy*- или *web*-серверами полученных запросов от клиентов. Ответы на *обработанные запросы* (ответ может быть и отрицательным, запрос может быть потерян) помещаются в сеть и отправляются сетевому агенту (клиенту). Ответ характеризуется: адресом отправителя (обработчика), адресом получателя, направлением отправки, содержанием, объемом, причиной-запросом, затратами ресурсов на генерацию.

9. Программные агенты — необязательные объекты, используются для логического (программного) улучшения характеристик работы серверов. Предлагается использовать программные агенты двух видов: *web*-ориентированные и локальные. Функции программных агентов: доставка файлов, их частей; нахождение «лучшего» места для обслуживания запроса файла или его части; слежение за перенаправлением СА; учет рекомендаций модуля управления; обеспечение механизма поиска частей файлов, проверки их подлинности и правильности «склеивания» частей в файл.

4. Идентификация сетевых агентов

Сетевые агенты характеризуются идентификатором, содержанием запроса, маршрутом, адресами (отправителя, автономной сети [8, с. 100], оценочным расположением), потоком запросов, текущими характеристиками (например, текущим временем). Дополнительно могут вычисляться соответствующие характеристики, например, соответствие географической зоны, другие признаки. Множество СА может быть разделено на подмножества по следующим критериям: по временному признаку; по географическому; по АС и по адресному пространству; узнанным категориям; по программному разбиению.

Пусть N — количество установленных уникальных сетевых агентов, A_i — информация об i-м сетевом агенте, $i = \overline{1, N}$. Каждый i-й сетевой агент характеризуется кортежем $A_i = \{A_{i1}, A_{i2}, ..., A_{ik}\}$, где k — количество элементов кортежа. Множество A может включать в себя подмножества A^j, где $j = \{1, 2, ..., 5\}$. а A^1 — множество элементов, объединенных по временному признаку; A^2 — множество элементов, объединенных по территориальному признаку; A^3 — множество элементов, объединенных по адресному признаку; A^4 — множество элементов, объединенных в результате узнавания; A^5 — множество элементов, объединенных как программное объединение. Причем $A^1 \subset A$, $A^2 \subset A$, $A^3 \subset A$, $A^4 \subset A$, $A^5 \subset A$.

Пусть NN — количество значений для обобщенной оценки. Тогда $K_k, k = \overline{1, NN}$ — k-е значение оценки, функция обобщенной оценки будет функцией отнесения объекта A_i к классу K_k, дополнительной оценкой будет подмножество A^j, $A_i \in K_k$, $A_i \in A^j$, где $i = \overline{1, N}$, $j = \{1, 2, ..., 5\}$, $k = \overline{1, NN}$.

5. Формализация характеристик *web*-сервера

Выделим следующие группы характеристик *web*-сервера:

1) характеристики системных ресурсов — память, процессор, жесткий диск, очередь обслуживания запросов, файл страничной подкачки и др.;

2) наполнение (содержание) контента — генерируемые файлы, статические файлы, части файлов;

3) характеристики сетевых параметров — количество сетевых интерфейсов, трафик, пакеты, очередь, коллизии и ошибки, параметры скорости и эффективности передачи, наличие прямых соединений с серверами и маршрутизаторами;

4) характеристики систем реакции — правила управления ответами, маршрутизация, правила доступа; приоритеты доступа и обработки запросов;

5) (опционально) наличие модуля управления и результаты его работы.

Каждый параметр предлагается учитывать по следующим признакам: «занято»; «свободно»; история; тенденция; расчетные признаки. Пусть M — количество серверов, тогда $B_i, i = \overline{1, M}$ — i-й сервер задан общим количеством признаков P, по группам: P_1, P_2, P_3, P_4, P_5 — границы номеров признаков по группам.

Тогда обозначим характеристики сервера кортежем:

$$B_i = \left\{ \begin{matrix} B_{i,1}, ..., B_{i,p1}, B_{i,p1+1}, ..., B_{i,p2}, B_{i,p2+1}, \cdots \\ ..., B_{i,p3}, B_{i,p3+1}, ..., B_{i,p4}, B_{i,p4+1}, ..., B_{i,p5} \end{matrix} \right\},$$

где $i = \overline{1, M}$; $\{B_{i,1}, ..., B_{i,p1}\}$ — характеристики системных ресурсов; $\{B_{i,p1+1}, ..., B_{i,p2}\}$ — содержание контента сервера, принадлежит множеству G, $\{B_{i,p1+1}, ..., B_{i,p2}\} \subset G$; $B_{i,p2+1}, ..., B_{i,p3}$ — характеристики сетевых параметров, содержат перечень доступных линий связи; $B_{i,p3+1}, ..., B_{i,p4}$ — описание правил системных реакций; $B_{i,p4+1}, ..., B_{i,p5}$ — результаты работы модуля управления.

Каждый параметр $B_{i,j}$, $i = \overline{1, M}$, $j = \overline{1, P}$ задается кортежем из 5 параметров $\{b1, b2, b3, b4, b5\}$, что позволяет более полно характеризовать состояние признака («занято», «свободно», история, тенденция, расчетный признак), значение –1 указывает на то, что такой параметр не используется.

Для получения обобщенной характеристики состояния *web*-сервера предлагается использовать K-значную P-местную функцию оценки состояния. В качестве исходных данных использованы P значений характеристик:

$$f_s(x_1, x_2, ..., x_p) \in \{0, 1, ..., K-1\}.$$

Например, состояние *web*-сервера может быть оценено как «простой», «работает», «перегружен», «не отвечает». В таком случае $K = 4$, а значения функции по порядку обозначают предложенные состояния.

6. Формализация объектов *proxy*-серверов

Рассмотрим характеристики объектов proxy-серверов:

1) характеристики сети — сетевые интерфейсы, трафик, пакеты, направление (принятые, отправленные, пересланные данные);

2) характеристика системы — процессор, память, дисковое пространство;

3) контент — статические файлы, части файлов. Если доверитель без файлов — то он только включился. Количество файлов увеличивается, если поступают запросы на новые файлы, уменьшается — если истекает время хранения старых файлов;

4) характеристики систем управления — функции защиты, распределение приоритетов, маршруты.

Пусть M_c — количество доверителей, тогда C_i — идентификатор i-го доверителя, $i = \overline{1, M_c}$. Каждый доверитель задается P_c параметров, тогда

$$C_i = \{C_{i,1}, ..., C_{i,Pc1}, ..., C_{i,Pc}\},$$

где $\{C_{i,1}, ..., C_{i,Pc1}\}$ — характеристики сети; $\{C_{i,Pc1+1}, ..., C_{i,Pc2}\}$ — характеристики системы; $\{C_{i,Pc2+1}, ..., C_{i,Pc3}\}$ — подмножество контента, $\{C_{i,Pc2+1}, ..., C_{i,Pc3}\} \subset G$; $\{C_{i,Pc3+1}, ..., C_{i,Pc}\}$ — характеристики систем управления.

Обобщенную оценку работы доверителя предлагается реализовать K-значной P-местной функцией:

$$f_s(x_1, x_2, ..., x_{Pc}) \in \{0, 1, ..., k-1\}.$$

7. Формализация характеристик маршрутизаторов

Пусть R^* — множество всех маршрутизаторов, а R — подмножество множества R^*; R задает множество подконтрольных маршрутизаторов. R_l — l-й маршрутизатор, характеризуется кортежем

$$R_l = \left\{R_l^1, R_l^2, R_l^3, R_l^4, R_l^5\right\},$$

где R_l^1 — множество интерфейсов и их сетевых идентификаторов l-го маршрутизатора, $R_l^1 \subset X$; R_l^2 — множество подключенных линий связи, $R_l^2 \subset U$; R_l^3 — правила маршрутизации, заданные в стандартном формате [8, с. 201]; R_l^4 — правила фильтрации; R_l^5 — данные о нагрузке и производительности (переданные пакеты, обработанный трафик, возникающие коллизии и их структура по направлениям).

Обобщенную оценку работы маршрутизатора предлагается реализовать k-значной 5-местной функцией:

$$f_s(x_1, x_2, ..., x_5) \in \{0, 1, ..., k-1\}.$$

8. Формализация каналов связи

Каналы связи характеризуются:

1) идентификатором КС;

2) идентификаторами оконечных устройств;

3) вхождениями в группы в зависимости от следующих критериев:

- подконтрольность — контролируемые (между *web*-серверами и доверителями) и внешние (неконтролируемые);
- назначение — информационные и служебные (между доверителями, к управляющему блоку);
- длительность — маршруты и отдельные сегменты;

4) оценками производительности — трафиком (структура, в обоих направлениях), пакетами (структура, в обоих направлениях), скоростью (в обоих направлениях, номинальная, реальная, расчетная), простоем, коллизиям и ошибкам;

5) соединениями — в процессе установки, установленным (в работе), закрытым, оценками производительности для каждого работающего соединения;

6) приоритетами и оценками эффективности;

7) уровнями и протоколами.

Рассмотрим уровни с точки зрения характеристик производительности. Это физический (характеризуется топологией, размером кадра, номинальной скоростью), канальный, транспортный (характеризуется заголовками, реальной скоростью данных), прикладной (характеризуется избыточностью). Для объективной оценки эффективности предлагается учитывать данные, полученные на физическом, транспортном и прикладном уровнях. В соответствии с моделью OSI [9, с. 66], каждый уровень добавляет служебную информацию. Необходимо учитывать, что увеличение такой информации ведет к уменьшению реальной скорости и снижению эффективности работы ЛС.

С учетом того, что исследуемая проблема НС возникла в глобальных и локальных сетях, основанных на стеке протоколов TCP/IP [10, с. 498], то использованы следующие характеристики и возможности данного протокола.

Для описания линий связи необходимо ввести множество известных (зарегистрированных или всех) хостов. Пусть X — множество хостов сети. Такое множество счетное и конечное, количество элементов 2^{32}, каждый из них задается IP-адресом. Отметим, что один хост может иметь два адреса, а два хоста не могут иметь один адрес. Множества *proxy*- и *web*-серверов, маршрутизаторов включаются в множество хостов ($R^* \subset X, B \subset X, C \subset X$). Элементы множества X будем считать вершинами. Обозначим множеством U множество линий связи. Элемент множества U соединяет два элемента множества X и является дугой графа $GL(U, X)$. Канал связи мы рассматриваем как единицу соединения, на всем промежутке которой не изменяются правила маршрутизации.

Тогда $GL^1(U, X)$ — подконтрольный граф — подграф графа $GL(U, X)$, который включает в себя вершины множеств B, C и R, а также соединяющие их дуги.

Аналогично выделим: $GL^2(U, X)$ — информационный граф, подграф содержит дуги информационного назначения; $GL^3(U, X)$ — подграф маршрутов.

Указанные подграфы в случае необходимости раскрашивают исходный в различные цвета.

Каждая дуга U_i характеризуется кортежем параметров $U_i = \{U_{i,1}, U_{i,2}, U_{i,3}, U_{i,4}, U_{i,5}\}$, где параметр $U_{i,j}$ указывает на вхождение i-й КС в подграфы GU_j, $j = \overline{1,3}$; $U_{i,2}$ содержит оценки производительности, в том числе вычисляемые:

1) фактическую скорость $V_r = D/t$, где D — объем полезной информации, переданной по КС за расчетное время t;

2) эффективную скорость $V_\Delta = V_n * \frac{Q+14}{Q+64}$, где V_n — номинальная и минимальная среди всех ЛС

скорость работы физического уровня КС, Q — размер кадра.

Соотношение расчетных параметров предлагается использовать для определения скважности, коллизии, простоя системы или неэффективности работы.

Параметр $U_{i,3}$ содержит информацию о протоколах и уровнях, может быть задан кортежем значений для каждого протокола: $U_{i,3}=\{P_1, P_2, ..., P_{proto1}, P_{proto1+1}, ..., P_{protoN}\}$.

Параметр $U_{i,4}$ характеризует данные о соединениях, а параметр $U_{i,5}$ — данные о приоритетах и оценках эффективности.

Учитывая разнородность данных, различные точки сбора и неодновременность возникновения таких данных, оценить состояние ЛС в каждый момент времени не всегда представляется возможным. Предлагается ввести обобщенную оценку на основе необходимых известных данных. Такая оценка должна содержать информацию о том, в каком направлении можно увеличивать нагрузку на ЛС, а в каком стоит сокращать. Будем использовать пятиместную k-значную функцию

$$f_{ls}(U_{i1}, U_{i2}, U_{i3}, U_{i4}, U_{i5}) \in \{0, 1, ..., k-1\}.$$

Модуль управления может быть реализован как программно, так и аппаратно. При программной реализации модуль управления располагается в *web*-сервере, а при аппаратной — отдельно. Модуль выполняет следующие функции:

1. Прием и обработку входных параметров от подконтрольных объектов сети: k-значные входы от объектов сети; корректировки человека-оператора, в случае необходимости.

2. Накопление данных: запись информации на случай выхода из строя сервера в результате атаки или перегрузки для последующего анализа причин выхода из строя; сбор и обработку статистической информации.

3. Адаптацию интеллектуальных систем поддержки принятия решения: обучение ИС; отбор эталонных объектов для обучения.

4. Непосредственно идентификацию и распознавание ситуаций, объектов, категорий.

5. Реализацию внешних воздействий: изменение (улучшение) характеристик работы серверов при обслуживании запросов СА; алгоритм восстановления системы — в случае, когда система самостоятельно не может перейти в нормальный режим работы; визуализацию заданных параметров.

В качестве исходных данных модуль управления получает оценки *web*- и *proxy*-серверов, оценку маршрутизатора, оценку ЛС.

Пусть D — множество значений $D=\{0, 1, ..., k-1\}$, тогда $x_1, x_2, x_3...$ — места входных параметров, где $x_1 ... x_m$ — оценки *web*-сервера; $x_{m+1} ... x_{m+mc}$ — оценки *proxy*-сервера: $x_{m+mc+1} ... x_{m+mc+mr+u}$ — оценки маршрутизатора; — оценка ЛС, где m, mc, mr, u — количество *web*-, *proxy*-серверов, routes, ЛС соответственно.

Зададим предикат узнавания ситуации, когда система не способна без внешнего воздействия перейти в нормальный режим работы:

$$P\left(x_1^{aj1}, ... x_m^{ajm}, ... x_{m+mc}^{aj_{m+mc}}, ... x_{m+mc+mr+u}^{aj_{m+mc+mr+u}}\right),$$

где $a_j \in D$.

Для возможности дальнейшего решения запишем предикатное уравнение узнавания ситуации в виде СДНФ [11, с. 24]:

$$P = \vee x_1^{aj1} \wedge ... \wedge x_m^{ajm} \wedge ... \wedge x_{m+mc}^{aj_{m+mc}} \wedge ... \wedge x_{m+mc+mr+u}^{aj_{m+mc+mr+u}}.$$

9. Идентификация контента сервера

Сервер характеризуется содержанием и протоколами доступа. Протоколы доступа зависят от подключенных КС. Содержание (контент) включает файлы и переходы (ссылки файлов друг на друга). Контент представлен иерархическим деревом или ориентированным графом (рис. 2).

Пусть F — множество файлов, тогда F_i — i-й файл содержания. Обозначим Url — множество переходов между файлами, тогда $Url_{i,j}$ — переход от F_i к файлу F_j.

Получаем $G(F, Url)$ — ориентированный граф, который содержит множество файлов и переходов между ними. Контент (содержание) i-го *web*-сервера опишем как $G^{Bi,j}(F, Url)$, где $i=\overline{1, M_s}$, $j=\overline{p1, p2}$, а k-го доверителя — как $G^{Ck,l}(F, Url), k=\overline{1, M_c}$, $j=\overline{p_e 1, p_c 2}$.

Рассмотрим характеристики файлов:

1) идентификатор;

2) контрольная сумма для проверки целостности;

3) размер;

4) делимость на части идентификаторами частей;

5) вхождение в контенты доверителей и *web*-серверов, а также их расположение:

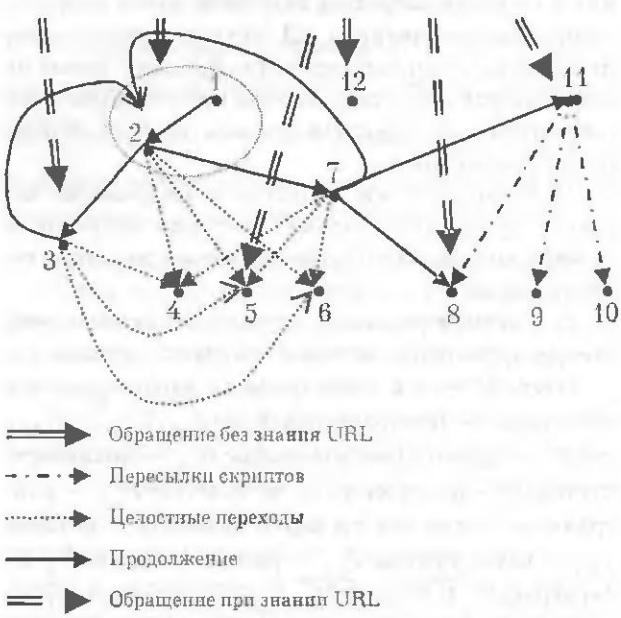

Рис. 2. Схема представления контента сайта:
1 — «/», 2 — index.html, 3 — news.html, 4 — style.css, 5 — images/*, 6 — function.js, 7 — download.php, 8 — file.zip, 9 — file.part1, 10 — file.part2, 11 — script.php, 12 — robot.txt

6) затрата ресурсов на обслуживание и обработку запросов;

7) частота вызова (популярность среди сетевых агентов).

Выделим группы файлов по критериям доступности и возвращаемого контента:

1) доступные по общим правилам («/», robot.txt, index.html);

2) доступные извне при знании идентификатора (обмен ссылками, баннерами; переходы из каталогов, поисковиков, избранного);

3) конечные служебные файлы общего пользования (элементы дизайна CSS, images/*; скрипты Js);

4) скрипты, не возвращающие данных, но обеспечивающие прохождение процедуры (когнитивный вопрос, авторизация, переход к загрузке файла);

5) файлы и части файлов для загрузки.

По информационному назначению выделим следующие группы файлов: страницы с содержанием (страницы с текстовой и графической полезной информацией); элементы дизайна (графические файлы, стили); скрипты (проверка форм, обработка взаимодействия, коммуникации); файлы с неоформленной информацией (RSS, XML); оформленные страницы.

По требованиям к обработке выделим: статические файлы (создаются до выдачи их клиенту); динамические файлы (создаются каждый раз при ответе клиенту); генерируемые файлы (меняются при изменении существенных параметров и, как правило, возвращаются клиенту как статические).

Такое разделение позволяет оценить требовательность обработки файлов к конкретным видам ресурсов сервера. Будем вести учет ресурсов, необходимых для генерации файлов: время формирования и отгрузки; затраты CPU, RAM; объем трафика, количество соединений с БД. Эти параметры следует использовать при распределении файлов и частей на *web*-серверах и доверителях; при принятии решения о предоставлении доступа сетевым агентам. Файлы будем группировать:

1) автоматически — путем исследования запроса, путем исследования истории запросов и ответов либо по атрибутам файлов и признакам их содержания;

2) в ручном режиме, в случае если автоматическое распределение не может выделить группы.

Пусть F_i — i-й файл графа G, характеризуется кортежем из M параметров $F_i = \{F_{i,1}, F_{i,2}, ..., F_{i,M}\}$, где $F_{i,1}$ — идентификатор файла; $F_{i,2}$ — количество частей, 0 — файл на части не делится; $F_{i,3}$ — контрольная сумма для проверки целостности файлов; $F_{i,4}$ — размер файла; $F_{i,5}$ — расположение на B_k, C_j серверах, $k = \overline{1, M}$, $j = \overline{1, M_c}$; $F_{i,6}$ — затраты ресурсов на обработку файла; $F_{i,7}$ — частота обращения к файлу; $F_{i,8}$ — идентификаторы частей файлов и контрольные суммы к ним.

С целью улучшения характеристик обслуживания запросов предлагается распределять статические файлы и части файлов в контенте (содержании) различных *proxy*- и *web*-серверов. При запросе файла оцениваются потенциально возможные точки его обслуживания, оценивается скорость передачи клиенту с зеркал и принимается решение о разбиении обслуживания целого файла на части, о выборе точки обслуживания.

При разбиении файлов на части предлагается разработать механизм поиска файла и его частей по контенту файла. Причем признаки поиска должны характеризовать как файл в целом, так и отдельные его части.

Также необходимо разработать механизм подтверждения целостности загруженных ресурсов (всего файла, каждой части файла). Предпосылки для разбиения файла на части:

1) размер превышает заданный порог ($F_{i,1} > \Delta$);

2) статический файл. В противном случае не определенным может быть процесс докачки частей и их склейки;

3) наличие частей на различных серверах $F_{i,5} \subset (B \cup C)$;

4) предварительная оценка скоростей указывает на целесообразность распараллеливания процесса;

5) наличие простоя отдельных ресурсов на определенных серверах.

Переходы между файлами характеризуются:

1) направленностью: переходы направлены; в некоторых случаях переходы одновременно могут в разных направлениях соединять два файла;

2) вероятностью (первым был вызван файл А, затем файл Б): вероятность того, что после файла А будет вызван файл Б; вероятность того, что файл Б был вызван по причине вызова файла А;

3) интервалом времени между вызовом А и вызовом Б (время чтения);

4) историей вызовов файлов;

5) возможностью параллельности и последовательности вызова.

Разделим переходы на пять типов:

1. Переход к сайту без знания конкретной страницы (по умолчанию).

2. Запрос страницы со знанием URL.

3. Последовательный/параллельный просмотр страниц.

4. Пересылки скриптами и переход к закачке файла.

5. Дозагрузка элементов целой страницы.

Для дальнейшего анализа будем выделять: точку входа; точку выхода; последовательность, цепочку; повторы и возвращения; перебор файлов и случайный просмотр.

Анализ переходов позволяет выделить следующие группы файлов: а) страница; б) тематический обход; в) файлы общего пользования.

Переход $Url_{i,j}$ характеризуется кортежем из N параметров: $Url_{i,j} = \{Url_{i,j,1}, Url_{i,j,2}, ..., Url_{i,j,n}\}$, где

$Url_{i,j,1}$ — вероятность перехода; $Url_{i,j,2}$ — условная вероятность перехода (байесова); $Url_{i,j,3}$ — частота использования в день; $Url_{i,j,4}$ — частота использования перехода на сессию; $Url_{i,j,5}$ — среднее время срабатывания перехода.

Процесс коммуникации сетевых агентов и серверов по средствам запросов и ответов разделим на три этапа (см. рис. 3).

1. **Запрос**. СА создает отдельный запрос и передает его серверу на обслуживание. Запрос характеризуется параметрами, указывающими на источник, передаваемый контент, и ресурсами, необходимыми для обработки запроса.

2. **Сессия**. В процессе коммуникации СА создает несколько запросов, и эти запросы идентифицированы от одного СА. Сессия характеризуется такими параметрами, как порядок просмотра, темп работы, группы ресурсов, время и интенсивность работы, время прочтения информации.

Если запрос — элементарная часть коммуникации, то объединенные в группу подряд идущие запросы образуют сессию. Сессия — элементарная часть построения образов и сбора информации.

Обозначим S — множество сессий, тогда S_l — l-я сессия, которая характеризуется N_C-местным вектором, $S_l = \{S_l^0, S_l^1, ..., S_l^{N_C^1}, S_l^{N_C^1+1}, ..., S_l^{N_C^2}, ..., S_l^{N_C}\}$, с тремя группами параметров, по N_C^1, N_C^2, N_C^3 в каждой:

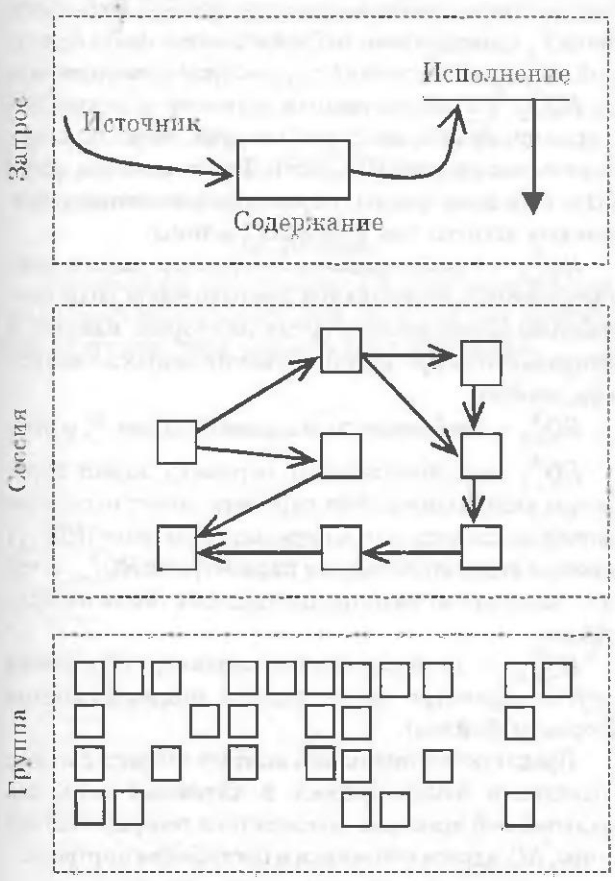

Рис. 3. Процесс коммуникации средствами запросов

1) S_l^0 — множество запросов, образовавших сессию, $S_l^0 \subset RQ$, $S_l^0 = \{RQ_1^{S_l}, RQ_2^{S_l}, ..., RQ_{Ms}^{S_l}\}$. $N_C^1 = 5$ — количество признаков, образующих сессию, RQ — множество всех запросов. S_l^1 задает *phpsession*, существует на протяжении сессии. $S_l^1 = RQ_{i,2}^2$ *xor* $RQ_{i,2}^3$; S_l^2 задает *cookie*, может существовать до, после и во время сессии. При необходимости может меняться сервером во время сессии. $S_l^2 = RQ_{i,2}^3$; S_l^3 — адрес отправителя, определен на протяжении всей сессии, стабильность зависит от способа подключения СА. $S_l^3 = RQ_{i,1}^1$; S_l^4 задает файл, с которого был сделан вызов, имеет смысл только для одного перехода (*refer*), $S_l^4 \subset RQ_{i,2}^4$; S_l^5 — данные формы, могут определять различную информацию — авторизацию, активные действия пользователя, признаки, загрузки файлов и сообщений. Данные формы постоянны для одного перехода, $S_l^5 \in RQ_{i,2}^2$; для всех $RQ_i \subset S_l^0$, $i = \overline{1, M_{si}}$.

2) $N_C^2 = 5$ — количество признаков, которые вычисляются при объединении запросов в сессию. S_l^6 — точка входа. $S_l^6 \approx Url_{0,RQ_{i,2}^1}$, $Url_{0,RQ_{i,2}^1} \in G(F, Url)$, $RQ_i \subset S_l^0$. S_l^7 — точка выхода, $S_l^7 \approx Url_{RQ_{i,2}^1,0}$, $Url_{RQ_{i,2}^1,0} \in G(F, Url)$, $\max(RQ_{j,5}) \forall RQ_j \in S_l^0$. Сессия характеризуется максимальной паузой между запросами (TR). Как только это время превышено, следующий запрос автоматически образовывает новую сессию. Последняя загруженная страница в старой сессии считается точкой выхода.

$$TR > (\textit{Реальное время}) - RQ_{j,5}.$$

S_l^8 — порядок обхода страниц, $S_l^8 \subset GL(F, Url)$, задан кортежем $S_l^8 = \{RQ_1, RQ_2, ..., RQ_p\}$, где RQ_1 — точка входа, RQ_p — точка выхода, $RQ_{p,1}^1 = RQ_p$. Этот параметр позволяет определить количество и объем просмотренных страниц. S_l^9 — темп просмотра, время чтения страниц и время срабатывания переходов.

Паузы при просмотре страниц, срабатывании переходов могут являться следствием прочтения предварительно открытых страниц. S_l^{10} — содержание ответов на творческие вопросы и авторизация.

3) $N_C^3 = 5$ — количество обобщенных признаков, определяющих сессию и вычисленных во время нее. S_l^{11} — множество целых страниц с подчиненными файлами, с оценкой статуса страницы. Выделение целой страницы позволяет оценивать статус документа и автоматически распределять вероятности вызовов и приоритеты предоставления следующих файлов в графе $GL(F, Url)$.

Статус страницы определяет следующие параметры: а) приложение активно/неактивно = читается/не

читается; б) документ закрыт; в) страница сохранена, добавлена в избранное; г) загружается страница/части; д) время прочтения.

S_t^{12} отражает предварительную группировку тематик, порядок основного просмотра. S_t^{13} — влияние темпа просмотра на загрузку файлов с сервера. S_t^{14} — определение нагрузки на сервера, обслуживающие запросы. S_t^{15} — идентификация предустановок СА. Например, загружаются ли изображения, работают ли скрипты, язык отображения, использование объектов, используются ли стили и др.

3. **Группы сессий.** Перед модулем управления ставится задача объединения различных сессий в группу для отдельного СА или производить группировку по иным критериям. Группы могут характеризовать одного сетевого агента, однотипных СА, временные (суточные, недельные и др.) ритмы сетевых агентов, территориальную, тематическую группу СА и др. Рассмотрим исходные данные, которые содержит запрос.

1. Параметры источника.

2. Параметры контевта, сопровождающего запрос.

3. Ресурсы, требуемые на обработку запроса.

4. Параметры, характеризующие работу на протяжении сессии.

5. Параметры, позволяющие группировать запросы.

Обозначим множеством RQ — множество запросов на получение файлов. Тогда RQ_i — i-й запрос, характеризуется вектором параметров:

- $RQ_i = \{A_j^{RQ_i}, RQ_{i,1}, RQ_{i,2}, RQ_{i,3}\}$, где $RQ_{i,1}$ — кортеж параметров источника;
- $RQ_{i,1} = \{RQ_{i,1}^1, RQ_{i,1}^2, RQ_{i,1}^3, RQ_{i,1}^4, RQ_{i,1}^5\}$, где $RQ_{i,1}^1$ определяет адрес источника соединения, $RQ_{i,1}^1 \in X$; по $RQ_{i,1}^1$ определяется вхождение СА в адресное подмножество, $A_j^{RQ_i} \in A^3$; $RQ_{i,1}^2$ — адреса *proxy*-серверов и скрытые за ними клиенты, $RQ_{i,1}^2 \in X$; $RQ_{i,1}^3$ — маршрут следования пакетов i-го запроса, $RQ_{i,1}^3 \in GL(U, X)$. Маршрут следования характеризуется адресом источника, адресом назначения, изменчивостью маршрута в пределах передачи пакетов для запроса, адресами промежуточных маршрутизаторов [12, с. 294]. Адреса источника соединения могут совпадать с адресом источника маршрута, а могут и отличаться, так как первый может быть скрыт за *proxy*-сервером. Маршрут — последовательность сетевых адресов (IP), через которые были переданы пакеты запросов от сетевого агента к серверу обслуживания; $RQ_{i,1}^4$ — номер автономной системы источника, получается путем запроса в специализированных базах данных [13]; $RQ_{i,1}^5$ — определяет предположительное географическое расположение источника (такую информацию предоставляет сервис Whois).

В дальнейшем этот параметр предлагается использовать: для оценки соответствия географической зоны и регистратора; для группировки и установление суточного времени.

Отдельные запросы предлагается группировать с использованием: адреса источников — для одной сессии; АС и географических признаков — для группировок.

Для группирования не подходят (могут произвольно меняться) маршруты и адреса источников между сессиями:

- $RQ_{i,2}$ — контент (содержание) запроса, задается вектором параметров $RQ_{i,2} = \{RQ_{i,2}^1, RQ_{i,2}^2, RQ_{i,2}^3, RQ_{i,2}^4, RQ_{i,2}^5, RQ_{i,2}^6\}$. Полное описание параметров запроса подробно изложено в спецификации. Выделим признаки, существенные для поставленной задачи.

$RQ_{i,2}^1$ обозначает URI [14, 15] запроса и в общем виде имеет следующий формат: <схема>:<идентификатор_источника/> <путь/> <имя_файла>.

<схема> — определяет протокол и *web*-сервис, который будет обрабатывать запрос;

<идентификатор_источника/> — адрес сервера, $\in (B \cup C)$;

<имя_файла> — идентификатор файла из контента сервера $\in (B \cup C)$, <имя_файла> $\in F$.

Файл, с которого был инициирован запрос, обозначим F_j, а запрашиваемый файл обозначим F_i, тогда переходом будем считать $Url_{i,j}$. Следовательно, запрос имеет отображение в графе $G(F, Url)$. Если файл F_j не определен, то будем считать файл F_i точкой входа и задавать как $Url_{i,0}$ =<путь/><имя_файла>.

$RQ_{i,2}^2$ — необязательный параметр и задает информационные поля форм, которые могут быть переданы методами GET, POST. Таким методом могут быть переданы файлы, параметры для авторизации сетевых агентов как клиентов системы.

$RQ_{i,2}^3$ — необязательный параметр, задает данные, которые сохраняются произвольное (или специально определенное) время на стороне клиента и направляются при инициировании запроса (например, *cookie*).

$RQ_{i,2}^4$ — необязательный параметр, задает F_j (*refer*).

$RQ_{i,2}^5$ — необязательный параметр, задает параметры авторизации. Этот параметр может использоваться после того, как на предыдущем шаге ($RQ_{i-\Delta}$) прошел этап авторизации с параметрами $RQ_{i-\Delta,2}^2$, где Δ — количество пакетов, прошедших после авторизации.

$RQ_{i,2}^6$ — необязательный параметр, обозначает другие параметры (язык, сжатие, поддерживаемые форматы файлов).

Предлагается оценивать контент запроса с целью выявления языка запроса и ключевых слов для дальнейшей проверки соответствия географической зоны, АС, адреса источника и построения портрета.

Предлагается оценивать параметры *cookie* ($RQ^3_{i,2}$), *phpsession* ($RQ^3_{i,2}$, $RQ^2_{i,2}$), полей в форме ($RQ^2_{i,2}$) перехода для оценки параметров работы сетевого агента.

Для дальнейшего использования предлагается применять:

- *cookie* ($RQ^3_{i,2}$) — для сессии и для группы;
- *phpsession* ($RQ^3_{i,2}$, $RQ^2_{i,2}$) — только для сессии;
- поля в форме ($RQ^2_{i,2}$) — для одного перехода;
- *refer* ($RQ^4_{i,2} \in F$) — для одного перехода;
- авторизацию ($RQ^2_{i,2}$, $RQ^5_{i,2}$) — часть сессии и для групп.

- $RQ_{i,3}$ — определяет предположительные необходимые ресурсы сервера и задается кортежем параметров $RQ_{i,3} = \{RQ^1_{i,3}, RQ^2_{i,3}, RQ^3_{i,3}, RQ^4_{i,3}, RQ^5_{i,3}\}$. Здесь $RQ^{1,k}_{i,3}$ — предположительные системные ресурсы, $k = \{cpu, ram, hdd, net\}$, $RQ^{1,k}_{i,3} \approx F_{i,6}$; $RQ^{2,p}_{i,3}$ — ресурсы обслуживания, $p = \{sess, connect, httpd\_proc\}$; $RQ^3_{i,3}$ — предположительный (среднестатистический) объем ответа $RQ^3_{i,3} \approx F_{i,4}$; $RQ^4_{i,3}$ — предположительное (среднестатистическое) время генерации ответа; $RQ^5_{i,3}$ — расчетная нагрузка на сеть в связи с необходимостью доставить ответ сетевому агенту.

Из характеристик файла $F_i \in G(F, Url)$ можно получить вероятности вызова следующих файлов, таким образом можно оценить последующую (цепную) нагрузку на сервер.

- $RQ_{i,4}$ — задает сетевого агента, который инициировал запрос, $RQ_{i,4} = A_j \in A$, где $j = \overline{1, N}$.
- $RQ_{i,5}$ — задает время поступления i-го запроса на обслуживание сервером.

10. Выводы

Разработаны и исследованы формальные модели компонент информационной сети *web* при нештатной ситуации типа «отказ в обслуживании».

Предложен метод идентификации нештатных ситуаций в информационных сетях *web*, предназначенный для обнаружения распределенных атак на *web*-серверы на основе распознавания категорий сетевых агентов.

Разработан метод выбора объектов для обучения во время функционирования системы. Предложены критерии оценивания относимости и чистоты объектов.

Список литературы: 1. *Кошкина Э.* В Великобритании будут сажать в тюрьму за организацию DoS-атак. (http://security.compulenta.ru/294763/). 2. *Черняк Л.* Порталы и жизненные циклы (Portal Lifecycle Management) // Открытые системы.– 2002.– № 2 (http://citforum.ru/consulting/portal/life). 3. *Алиев А. А., Последов А. В.* Обнаружение распределенных атак в сети Интернет // Научно-методич. конф. «Интернет и современное общество» (http://ims2002.nw.ru/02-r2f03.html). 4. *Вишневский В. М.* Теоретические основы проектирования компьютерных сетей.– М.: Техносфера, 2003.– 512 с. 5. *Олійник В. Ф.* Основи теорії систем зв'язку: Математичні моделі телекомунікаційних систем.– К.: Техніка, 2000.– 152 с. 6. *Кульгин М. В.* Компьютерные сети. Практика построения. Для профессионалов. 2-е изд.– СПб.: Питер, 2003.– 462 с. 7. *Хелеби С., Мак-Ферсон Д.* Принципы маршрутизации в Internet. 2-е изд.: Пер. с англ.– М.: Изд. дом «Вильямс», 2001.– 448 с. 8. *Сергеев А. П.* Офисные локальные сети: Самоучитель.– М.: Изд. дом «Вильямс», 2003.– 320 с. 9. *Олифер В. Г., Олифер Н. А.* Компьютерные сети. Принципы, технологии, протоколы.– СПб.: Питер, 2001.– 672 с. 10. *Таненбаум Э.* Компьютерные сети. 4-е изд.– СПб.: Питер, 2003.– 992 с. 11. *Шабанов-Кушнаренко Ю. П.* Теория интеллекта. Математические средства.– Харьков: Вища школа, 1984.– 144 с. 12. *Колисниченко Д. Н.* Сделай сам компьютерную сеть. Монтаж, настройка, обслуживание.– СПб.: Наука и техника, 2004.– 400 с. 13. RIPE Network coordination center, http://www.ripe.net (11.09.2006). 14. http://www.ietf.org/rfc/rfc3986.txt (27.5.2006). 15. http://ru.wikipedia.org/wiki/Uri (10.04.2006).

Поступила в редколлегию 01.09.2006