УДК 621.391.31

Е. Д. КУЗЬМИНЫХ

БОРЬБА С ОТКАЗАМИ И ПЕРЕРАСПРЕДЕЛЕНИЕ НАГРУЗКИ НА ОСНОВЕ ПРОТОКОЛА SIP В IP-ТЕЛЕФОНИИ

Введение

Во всем мире наблюдается увеличение количества пользователей сетью Интернет, а следовательно, увеличение нагрузки на сети, обслуживающие и предоставляющие потребителям услуги. Сегодня уже количество пользователей Интернет превысило количество стационарных абонентов. Основанные на IP телефонные услуги предлагаются как альтернатива сети общего пользования и предлагают ряд новых услуг поверх ТФОП. Конечно, ТФОП уже зарекомендовала себя как более надежная в плане предоставления качества услуг, нежели сеть Интернет и поэтому, чтобы составить достойную конкуренцию и получить широкое применение, IP-услуги должны демонстрировать высокие показатели надежности и иметь значительную масштабируемость. Например, сеть ТФОП предоставляет надежность системы в так называемые «5 девяток», т.е. 99,999 % устойчивой работы, что допускает только 5 *мин* простоя в год. Протокол инициирования сеансов связи SIP (Session Initiation Protocol) изначально ориентирован на использование в IP-сетях и является предпочтительным протоколом сигнализации для IP-телефонии. Основным элементом SIP-сети является SIP-сервер, от надежности и производительности которого зависит качество предоставления услуг и эффективность работы сети.

Если по какой-то причине на сервере происходят неполадки или сбой в работе, инициирование запроса и передача сообщений не могут быть направлены по нужному пути. Мы можем увеличить надежность в предоставлении услуги, добавив резервный сервер, который автоматически принимает на себя роль ведущего в случае отказа первого сервера. Если есть тысячи зарегистрированных пользователей и один сервер не может справиться с нагрузкой, то второй сервер может работать наряду с первым, распределяя между собой нагрузку. Наша задача состоит в том, чтобы обеспечить оптимальную пропускную способность канала для обслуживания и одного, и десяти миллионов вызовов в ЧНН (часы наибольшей нагрузки) для IP телефонии, используя аппаратные средства ЭВМ. Мы опишем некоторые из способов борьбы с отказами и распределения нагрузки для SIP-серверов в первой и второй части данной статьи. Эти методы также применяются не только в телефонии, а, например, при обмене текстовыми сообщениями или при презентациях, когда используются те же самые серверы SIP для регистрации и передачи сообщений. В части 3 количественно оценены преимущества применения предложенных методов по оптимизации распределения нагрузки.

Проблема борьбы с отказами и распределением нагрузки для web-серверов хорошо изучена [1]. Для решения этих проблем и повышения надежности используют протокол TCP (Transport Control Protocol), возвращение IP адреса и возвращение MAC– адреса [1]. Для web-серверов свойственно распределение нагрузки через диспетчеры связи и через HTTP-запросы или перенаправление вызовов во время сессии [2]. Хотя SIP и использует схему «запрос-ответ», как и HTTP, но имеет некоторые фундаментальные отличия, которые делают и саму проблему распределения нагрузки немного отличной. Например, серверы SIP могут использовать и TCP, и UDP протокол для передачи сообщений, сигнальные сообщения запросов и ответов не требуют специальной полосы пропускания, сохранение ответов не обязательно даже во время установления соединения, а процесс обновления данных (сообщение REGISTER) и процесс поиска (сообщение INVITE), проще и быстрее, чем обновления массивов баз данных и web-приложений.

В реализации услуг телефонии на основе протокола SIP возникают три критических параметра в обеспечении масштабируемости: передача сигнальной информации, передача данных в реальном времени и обработка в шлюзе.

В статье рассматриваются лишь проблемы, возникающие при передаче сигнальной информации.

Мультимедийная подсистема 3GPP (IMS) использует SIP для управления вызовами, чтобы позволять обслуживать миллионы пользователей. Управления вызовами обеспечивается расстановкой приоритетов на SIP-сервере, ролью самого сервера при предоставлении сервисов. Например, SIP-сервер может выполнять роль исходящего прокси-сервера при посещении другой сети, запрашивающего сервера для входящих запросов в локальной сети, сервера, предоставляющего определенные услуги в зависимости от приоритета пользователя.

В работе будут описаны и сравнены некоторые из этих методов при использовании SIP. А также представлена реализация метода борьбы с отказами в SIP-телефонии и предложены некоторые практические рекомендации.

Работоспособность системы: борьба с отказами

Высокая работоспособность системы может быть получена при внедрении резервного элемента такого, как SIP-сервер или пользовательская база данных.

В зависимости от того, где обнаружен отказ и какой способ решения проблемы выбирают, различают следующие методы борьбы с отказами: на основе клиента, на основе DNS, через базу данных и через IP.

При использовании метода борьбы с отказами на основе клиента все вопросы резервирования решаются на стороне клиента. Серверы работают независимо друг от друга. В IP-телефонах прописываются адреса двух SIP-серверов.

Метод борьбы с отказами на основе DNS-сервера, использующий SRV [3] и NAPTR [4] данные, наиболее качественный в обеспечении отказоустойчивости. DNS может использоваться, чтобы установить соединение с SIP-сервером или изменить IP-адрес основного сервера $P1$ на резервный $P2$.

Не все телефоны SIP могут зарегистрироваться на нескольких серверах, поэтому клиент может зарегистрироваться только на одном сервере, и потом уже размножить регистрацию к другим узлам. Основной сервер $P1$ хранит записи в базе данных $D1$. Резервный сервер $P2$ использует базу данных $D2$. Любое изменение в $D1$ дублируется в $D2$. Когда $P1$ перестает работать, $P2$ может взять на себя функции основного сервера и использовать базу $D2$.

Процесс перераспределения IP-адреса может быть выполнен без внешнего вмешательства. $P1$, и $P2$ имеют одинаковую конфигурацию, но обслуживают разные узлы в сети Ethernet. Оба сервера используют одну и ту же пользовательскую базу данных, $D1$. Резервная база D2 дублирует $D1$. Однако, если ведомый сервер дублирует ведущий, то в конфигурации первого необходимы изменения, потому что, если выходит из строя $P1$, то все обновления базы $D1$, которой в момент отказа будет пользоваться $P2$, будут невозможны. Если это случается, пользователь становится недоступен максимум на время $Tr + TR$, где Tr – интервал обновления регистрации (обычно равен одному часу), и TR – таймаут (время ожидания) клиента до следующей попытки вызова.

Метод объединения ресурсов, заключается в объединении клиентских терминалов в объединение пользователей (PU – Pool User), сервера $P1$ и $P2$ в элементы объединения (PE – Pool Elements) SIP-серверов, и баз данных $D1$ и $D2$ тоже в элементы объединения PE, но в группе баз данных. $P1$ и $P2$ регистрируются на сервере имен $NS2$, который контролирует их, и сообщает другим серверам имен (NS) об этих элементах объединения PE. Точно так же $D1$ и $D2$ регистрируются на сервере имен $NS2$. Элемент объединения удаляется из объединения, если выходит из строя.

Масштабируемость системы: распределение нагрузки

В случае отказа в работе резервный сервер принимает на себя функции основного, тогда как в распределении нагрузки участвуют все серверы. Некоторые из методов борьбы с отказами могут быть применены и при распределении нагрузки.

Распределение нагрузки на основе DNS-сервера. Механизмы DNS SRV [3] и NAPTR [4] могут использоваться для распределения нагрузки, учитывая приоритетность и весомость ресурсов на SIP-серверах, как показано ниже:

```
example.com
_sip._udp 0 40 a.example.com
          0 40 b.example.com
          0 20 c.example.com
          1 0 backup.somewhere.com
```

Эта запись DNS SRV указывает на то, что серверы **a**, **b**, **c** должны использоваться как основные (приоритет 0), в то время, как backup.somewhere.com как резервный сервер (приоритет 1). Основные серверы **a** и **b** вместе обрабатывают 80 % от общего количества запросов, а сервер **c**, возможно более медленный, должен обрабатывать оставшиеся 20 %. Клиенты могут также использовать случайный закон распределения.

Телефоны SIP обычно производят обновление REGISTER раз в час, в то время, как для оператора беспроводной сети с одним миллионом абонентов требуется приблизительно 280 обновлений в секунду.[5]

Эта архитектура также обеспечивает высокую надежность из-за избыточности. Предположим, что среднее время восстановления намного меньше, чем среднее время отказа, и надежность каждого прокси-сервера R_p, а сервера базы данных R_d, и предположим, что есть P прокси-серверов и D серверов баз данных, тогда надежность всей системы можно выразить как $(1-(1-R_p)^P)(1-(1-R_d)^D)$. Надежность возрастает с увеличением D и P.

При распределении нагрузки по значению идентификатора пользовательское пространство разделено на группы, которые не пересекаются между собой. Каждый пользователь имеет свой уникальный идентификатор, относящий его к определенной группе, которая определяется, например, по первой букве идентификатора. Например, $P1$ обслуживает начальные буквы идентификатора с a по h, $P2$ обслуживает с i по q, и $P3$ обслуживает с r по z. Высокоскоростной сервер первого уровня $P0$ распределяет запросы к серверам второго уровня $P1$, $P2$ и $P3$ по принципу соответствия идентификатора пользователя.

Если запрос получен для получателя *bob@home.com*, он идет к $P1$, тогда как *sam@home.com* идет к $P3$. Каждый сервер имеет его собственную базу данных и взаимодействует с другими. Чтобы гарантировать почти однородное распределение запросов вызова между серверами, можно использовать алгоритм хэширования типа SHA1 или группы могут определяться динамически в зависимости от нагрузки.

Однако из-за малой избыточности эта архитектура не улучшает надежность системы. Предположим, что среднее время восстановления намного меньше среднего времени отказа, а надежности прокси-сервера первого уровня $P0$, прокси-сервера второго уровня и сервера базы данных определены как R_0, R_p и R_d, и предположим, что есть D групп, тогда надежность всей системы можно выразить как $R_0 \cdot (R_p)^D \cdot (R_d)^D$. Надежность системы уменьшается с увеличением числа D.

Узким местом в работе такой схемы является прокси-сервер первого уровня.

При использовании серверов с одинаковыми IP-адресами все резервные серверы одной сети (например, Ethernet) используют один и тот же IP-адрес. Маршрутизатор подсети работает с MAC-адресами этих серверов и направляет поступающие пакеты на один из этих серверов. Маршрутизатор может использовать алгоритмы типа "round robin" или "время ответа сервера", чтобы выбрать наименее загруженный сервер.

Чтобы избавить маршрутизатор подсети от избыточной информации о состояниях транзакции, этот метод предлагают использовать только для SIP-серверов без сохранения состояний транзакций, которые используют транспортный протокол UDP и рассматривают каждый запрос независимо от предыдущих состояний транзакции.

При отсутствии DNS SRV и NAPTR, мы можем использовать этот метод на прокси-сервере первого уровня. Но тогда из-за полосы пропускания сети ограничивается число сер-

веров в группе и производительность подсети может упасть. Кроме того, этот метод не работает, если сама сеть недоступна.

Так как ни один из механизмов, описанных ранее, не обеспечивает достаточной надежности и расширяемости сети, попробуем объединить два метода в двухуровневую расширенную архитектуру (рис. 1), чтобы улучшить и надежность и масштабируемость. На первом уровне серверы, выбранные с помощью механизмов DNS NAPTR и SRV, направляют запрос к определенной группе серверов второго уровня на основе метода распределения нагрузки по значению идентификатора. Сервер в определенной группе также выбирается на основе механизмов DNS. Сервер второго уровня обрабатывает запрос. Использование DNS не требует от серверов быть расположенными рядом и они могут быть географически разнесены.

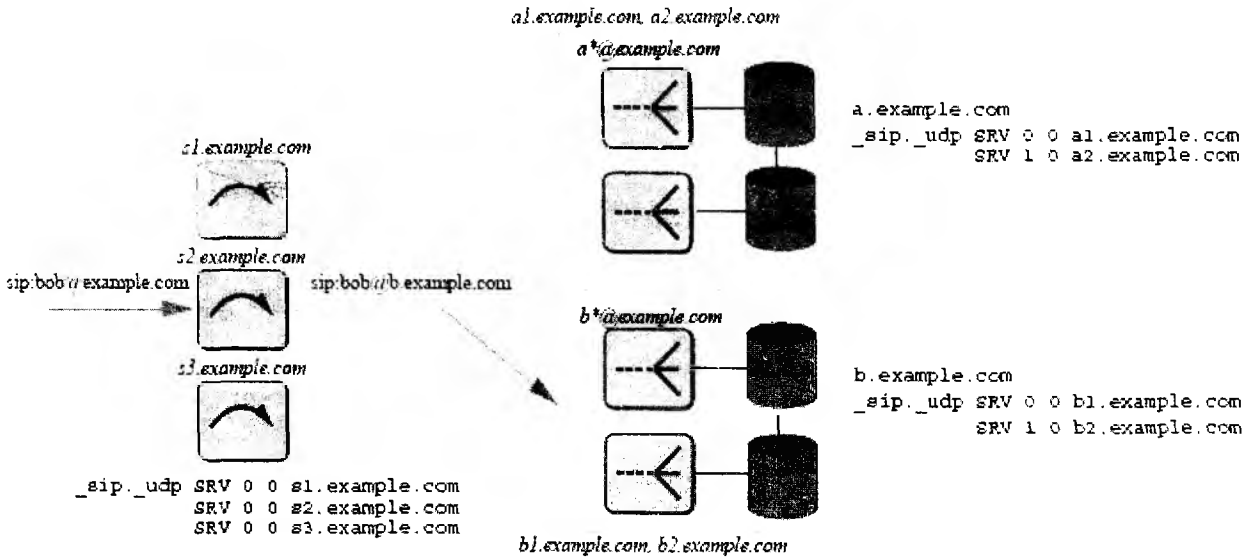

Рис. 1

Предположим, что есть S серверов первого уровня, P групп на втором уровне, и B прокси-серверов и баз данных в каждой группе. Каждая группа второго уровня имеет один главный сервер и $B-1$ резервных. Все базы данных в группе копируются, используя циклическое дублирование. Предположим, что появление сообщения REGISTER однородно распределено со значением нагрузки λ_R, а появление сообщения INVITE (или другие запросы, которые можно отнести к типу СООБЩЕНИЯ) распределено по пуассоновскому закону с нагрузкой λ_P, тогда общий запрос имеет нагрузку $\lambda = \lambda_R + \lambda_P$. Предположим, что интенсивность обслуживания вызовов на сервере первого уровня μ_s, а на сервере второго уровня μ_r и μ_p для регистрации и обработки соответственно. Выбираем функцию хэширования так, чтобы интенсивность обслуживания вызовов на входе кластера приблизительно была равна $\frac{\mu}{B}$.

Цель состоит в том, чтобы количественно получить отношения между различными параметрами обслуживания μ, нагрузкой системы λ и параметрами избыточности S, B, P. Необходимо получить ответы на такие вопросы как, (1) когда есть необходимость в сервере первого уровня и (2) какие оптимальные значения для параметров избыточности, чтобы достигнуть нужной масштабируемости и надежности сети. Наша цель состоит в том, чтобы обеспечить надежность доставки 99.999 % и расширяемость системы до 10 миллионов ВНСА, используя аппаратные средства ЭВМ. Результаты измерений представлены ниже.

Анализ результатов

На рис. 2 показана зависимость производительности сети от $S_n P_m$ конфигурации. Он показывает усредненный результат трех экспериментов для каждой конфигурации при различной интенсивности поступления запросов. Одиночный SIP-сервер обрабатывает приблизительно 900 *запросов/с* (см. $S_0 P_1$ на рис. 2), что соответствует приблизительно трем миллионам вызовам в ЧНН. Когда нагрузка больше, чем эффективность сервера, производительность остается почти постоянной (приблизительно 900 *запросов/с*). Когда производительность такая же, как нагрузка, то график прямолинеен. При нагрузке 1800 *запросов/с*, система успевает обслужить только 50 % (то есть, производительность равна половине нагрузки). На практике же процент обработанных вызовов должен быть приблизительно 100 %.

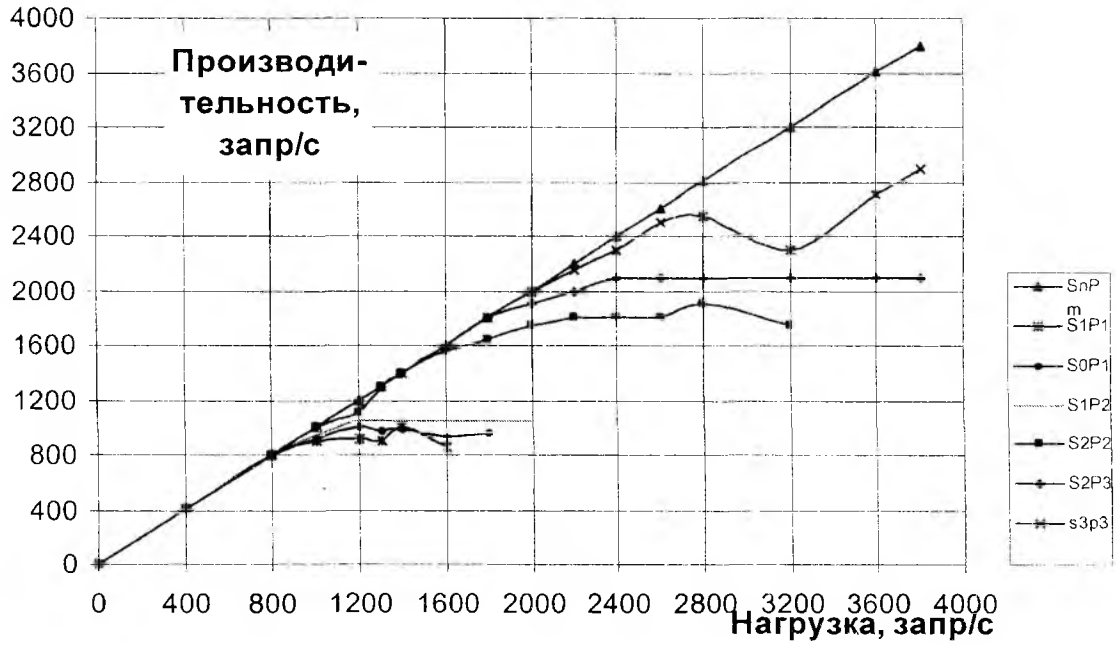

Рис. 2

Когда сервер перезагружен, использование центрального процессора близко к 100 %. Добавление кластеров второго уровня и использование метода распределения нагрузки на прокси-сервере первого уровня повышает производительность, но увеличивает зависимость от работы сервера первого уровня и поэтому производительность не превышает 1050 *запросов/с* (S_1P_2 на рис. 2). Добавление еще одного сервера первого уровня S_2P_2 увеличивает производительность приблизительно вдвое по сравнению с использованием одного сервера второго уровня. Уже S_3P_3 может обработать приблизительно 2800 *запросов/с*, что уже в три раза больше производительности одиночного сервера второго уровня, а S_2P_3 может обработать 2100 *запросов/с*, что вдвое больше производительности одиночного сервера первого уровня.

Из результатов следует, что мы можем достичь линейной зависимости, используя больше серверов первого и второго уровня в нашей архитектуре. Ниже представлен теоретический анализ двухуровневой архитектуры, обеспечивающей надежность и масштабируемость сети. Предположим, что серверы первого и второго уровней в S_nP_m имеют пропускную способность C_s и C_p соответственно (обычно, $C_s \geq C_p$). Серверы определены как S_i и P_j, где $1 \leq i \leq n$, $1 \leq j \leq m$ для первого и второго уровня соответственно. Предположим, что входящие запросы поступают с интенсивностью λ, распределенной по экспоненциальному закону. Предположим, что нагрузка распределена равномерно по всем n серверам первого уровня и каждый сервер первого уровня получает $\frac{\lambda}{n}$ *запросов*. Предположим, хеш-функция распределяет запросы к серверу второго уровня таким образом, что i-й сервер, P_i, получает долю, f_i,

из запросов $\left(\sum f_i = 1\right)$. Предположим, что все пользователи могут получить вызов с равной вероятностью, а хеш-функция однородно распределяет пользовательские идентификаторы среди серверов второго уровня, тогда все $f_i = 1$. Однако на практике разное число входящих запросов для различных пользователей имеет неоднородное распределение.

Производительность τ при заданной нагрузке λ является суммарной производительностью на двух уровнях. Производительность на первом уровне $\lambda' = \min(\lambda, nC_s)$ является входящей нагрузкой на второй уровень. Сервер, P_j, второго уровня имеет $\min(\lambda' f_j, C_p)$. Таким образом

$$\tau(\lambda) = \sum_{j=1}^{m} \min\left(f_j \min\left(\lambda, nC_s\right), C_p\right).$$

Предположим, что $f_i \geq f_j$ для $i > j$. Суммарная производительность определено как $m+1$ линейных сегментов, L_i: $(\lambda_i, \tau_i) \rightarrow (\lambda_{i+1}, \tau_{i+1})$ для i=0 до m, где (λ_k, τ_k) определено как (0, 0 при k=0,

$$\left(\frac{C_p}{f_k}, \tau_{k-1} + (\lambda_k - \lambda_{k-1})F_k\right) \text{ при } 1 \leq k \leq m \text{ и } f_k \geq \frac{C_p}{nC_s},$$

$$(nC_s, \tau_{k-1} + (\lambda_k - \lambda_{k-1})F_k) \text{ при } 1 \leq k \leq m \text{ и } f_k < \frac{C_p}{nC_s}, \ (\infty, \tau_m) \quad \text{при } k = m + 1,$$

где $F_k = (1 - \sum_{i=k}^{m} f_i)$.

Начальный линейный сегмент представляет 100% обработку запросов с наклоном 1. При нагрузке $\frac{C_p}{f_1}$ сервер P_1 имеет полную загруженность и не может обрабатывать дополнительные запросы. Таким образом, увеличение пропускной способности первого уменьшает нагрузку на другие серверы, P_k, k =2, 3, ..., m.

Это дает наклон $F_1 = (1 - (f_2 + f_3 + ... + f_m))$ для второго линейного сегмента. Точно так же P_2 достигает полной загруженности при нагрузке $\frac{C_p}{f_2}$, и так далее. Когда все серверы второго уровня перезагружены, производительность остается постоянной, равной последнему линейному сегменту. Если пропускная способность сервера второго уровня P_j равна C_p больше, чем поступающая нагрузка $f_j(nC_s)$, тогда производительность системы не ограничена P_j.

Для эксперимента был выбран набор из сотни пользовательских идентификаторов. Хеш-функция распределила их идентификаторы следующим образом: для $m = 2$ f определено {0.6, 0.4}, для $m = 3$ f приблизительно равно {0.4, 0.3, 0.3}. Из 1000 или 10 000 пользовательских идентификаторов, та же самая хеш-функция распределила набор более однородно, но наше искаженное распределение сотни идентификаторов помогает нам проверять результаты, принимающие неоднородное распределение запроса для различных пользователей.

Архитектура метода объединения ресурсов для обеспечения надежности обеспечивает высокую надежность из-за своей избыточности. Предположим, что надежность состояний основного и резервного серверов равна R, тогда полная надежность: $(1 - (1 - R)^2)$.

Отказ сервера влияет на задержку в установке вызова (так как клиент повторяет запрос, запрашивает ко второму серверу после тайм-аута) и на готовность пользователя (вероятность того, что пользователь достижим через сервер, с которым соединен его SIP-телефон). Если основной сервер в течение длительного времени остается неработоспособным, DNS-сервер может обновить свои записи и сделать резервный сервер главным. Если надежность отдельного сервера R ($0 \leq R \leq 1$), время ожидания повторного запроса клиента TR, а время жизни DNS TTL – TD, то средняя задержка установки вызова равна $TR(1 - R)P[t_M < T_D]$, где задержка сети равна нулю и $R \approx 1$), $P[t_M < T_D]$ – вероятность того,

что время, t_M на восстановление сервера меньше, чем DNS TTL. Например, если время ремонта экспоненциально распределено по значению T_M, тогда $P[t_M < T_D] = 1 - e^{-\frac{T_D}{T_M}}$, предполагая, что среднее время отказа намного больше, чем среднее время восстановления (то есть, $(1-R)T_M \approx 0$).

На готовность пользователя отказ основного сервера обычно не влияет, потому что большинство регистраций восстанавливается с помощью сообщений REGISTER. Однако, если основной сервер выходит из строя после того, как телефон зарегистрировал новый контакт, но прежде, чем регистрация была дублирована на резервный сервер, тогда местоположение телефона некоторое время будет недоступно, пока не повторится следующая регистрация. В этом случае, предположим, что продолжительность работы сервера распределена экспоненциально, и учитывая то, что запросы на сервере не хранятся, можно сделать вывод, что время отказа имеет то же самое распределение. Предположим, что среднее время отказа TF, а время ожидания ответа от базы данных Td, тогда вероятность, что сервер выйдет из строя до дублирования регистрации на резервный сервер, равна $P[\text{время жизни} < T_d] = 1 - e^{-\frac{T_d}{T_f}}$.

Например, если T_F одна неделя, а T_d – 10 *с*, тогда эта вероятность равна 0.0000165 ≈ 0. Если это случается, пользователь становится недоступен максимум на время T_r+T_R, где T_r – интервал обновления регистрации (обычно равен одному часу), и T_R – тайм-аут (время ожидания) клиента до следующей попытки вызова. По истечении этого времени клиент заново регистрируется и может быть обслужен резервным сервером.

Чтобы улучшить работу сервера, желательно кэшировать информацию о пользователе на SIP-сервере. Конечно, это является причиной больших задержек при обновлении пользовательской регистрации на серверах *P*1 и *P*2. Но в то же время, если отказ происходит до дублирования на резервный сервер *P*2, то можно использовать кэшированные записи. Однако на практике телефоны обновляют регистрацию прежде, чем истечет время на обновление кэшированных записей. Например, предположим, что регистрация происходит каждые два часа, а обновление информации – каждые 50 *мин*. Предположим, *P*1 фиксирует нового пользователя, но выходит из строя до того, как успел внести эту информацию в базу *D*1. В этом случае регистрация на *D*2 может занять 70 *мин*, а *P*2 может все еще посылать запросы на этот телефон. Следующее обновление информации произойдет через 50 *мин*, до истечения времени на регистрацию на *D*2. Если новый пользователь отправил вызов как раз перед отказом на сервере *P*1, то он будет недоступен до следующего обновления. Предположим, что T_d и T_F определены как и прежде, а *Tc* – интервал обновления базы данных, тогда вероятность того, что сервер выйдет из строя до дублирования регистрации на резервный сервер, равна $1 - e^{-\frac{T_d+T_c}{T_f}}$.

Телефон фирмы Cisco поддерживает двойную регистрацию и на основном, и на резервном прокси-сервере. И в *D*1, и в *D*2 фиксируются одни и те же изменения. Однако пока контакт определяется на пользовательском идентификаторе, вторая запись затирает первую. Как альтернатива, можно модифицировать сервер, чтобы обеспечить непосредственную синхронизацию между кэш-памятью и внешней базой данных в случае, когда сервер не загружен.

Двухстороннее копирование может быть распространено на большее количество серверов, используя циклическое дублирование типа *D*1-*D*2-*D*3-*D*1. Например, если каждый сервер имеет надежность 98 %, то при тройном дублировании надежность равна $1 - 0.02^3 = 0.999992$. Чтобы обеспечить борьбу с отказами индивидуальных серверов (например, *D*1 выходит из строя, но не *P*1), SIP сервер *P*1 должен пользоваться базой данных *D*2, если *D*1 становится недоступной.

Выводы

Рассмотрены некоторые из существующих методов борьбы с отказами и распределения нагрузки применимо к SIP-серверам и предложена двухуровневая архитектура распределе-

ния нагрузки с использованием значения идентификатора. Использование доменной системы имен DNS предпочтительней для избыточных систем, так как при этом серверы могут быть разнесены географически. Например, мы можем разместить SIP-серверы в разных сетях. При использовании метода возвращение IP адреса или NAT это довольно трудно. DNS сама себя дублирует, поэтому выход из строя одного блока преобразования имен не затрагивает весь процесс. Мы объединили DNS, резервирование серверов и двухуровневую архитектуру распределения нагрузки с использованием значения идентификатора, которая может теоретически расширяться для любой пропускной способности.

Общее количество пользователей разделено среди независимых серверов второго уровня так, что нагрузка каждого сервера остается ниже его пропускной способности.

Также была проведена оценка работы двухуровневой архитектуры распределения нагрузки. Была достигнута производительность сети в 2800 *запросов/с* при использовании конфигурации S_3P_3 (три сервера первого уровня и три сервера второго уровня), что означает, что 10 *млн* входящих запросов в час могут быть обработаны с использованием шести серверов.

В дальнейшем планируется расширить эксперимент с учетом состояний транзакций на SIP-сервере (с сохранением и без сохранения состояний транзакций), исследовать поведение системы при добавлении других видов нагрузки (голосовая почта, режим конференции, взаимодействие с ТфОП), расширить рамки эксперимента при большем количестве серверов и с учетом выхода в глобальную сеть, учитывать потери и перегрузки не только на узлах доступа, но и в самой сети, и использовать повторную передачу сообщений при потере. Другие виды нагрузки требуют большей работы по обеспечению отказоустойчивости и процесса перераспределения нагрузки в середине вызова, а не только при установке соединения.

При использовании двухуровневой архитектуры распределения нагрузки была достигнута надежность «пяти девяток», даже если каждый сервер имеет надежность только 99 % (3 дня простоя в год).

Обнаружение и восстановление глобальных выходов из строя является уже индивидуальной задачей на каждом сервере. Но вместо статического конфигурирования резервных серверов, было бы эффективнее, если бы серверы автоматически могли обнаружить и выбрать конфигурацию других доступных Internet-серверов, например, чтобы обработать временную перегрузку.

SIP-серверы в VoIP сети могли бы автоматически обнаруживать, самоорганизовывать и конфигурировать себя как серверы первого или второго уровней.

Список литературы: 1. *Гольдштейн В. С., Пинчук А. В., Суховицкий А. Л.* IP-Телефония. М.: Радио и связь, 2001. 336с. 2. *Cisco* Systems. Failover configuration for LocalDirector, http://www.cisco.com/warp/public/cc/pd/cxsr/400/tech/locdf wp.htm. 3. *A. Gulbrandsen, P. Vixi and L. Esibov.* A DNS RR for specifying the location of services (DNS SRV). RFC 2782, Internet Engineering Task Force, Feb. 2000. 4. *M. Mealling and R. W. Daniel.* The naming authority pointer (NAPTR) DNS resource record. RFC 2915, Internet Engineering Task Force, Sept. 2000. 5. *J. Rosenberg and H. Schulzrinne.* Session initiation protocol (SIP): locating SIP servers. RFC 3263, Internet Engineering Task Force, June 2002.

Харьковский национальный университет радиоэлектроники

Поступила в редколлегию 23.10.2007