УДК 004.724.4(045)

КУЛАКОВ Ю.А.,
ГАБИНЕТ А.В.

СПОСОБ ВЫБОРА КЛИЕНТОВ В PEER-TO-PEER СЕТЯХ С УЧЕТОМ ИХ МЕСТОПОЛОЖЕНИЯ И ЗАДЕРЖЕК МЕЖДУ НИМИ

В настоящей статье был предложен способ выбора клиентов, учитывающий физическое местоположение пиров и задержки между ними. Кратчайший путь между клиентами в соответствующих AS определяется на основе построения графа из ближайших автономных систем. Проведен анализ зависимости коэффициента нагрузки от кол-ва клиентов и размещения их в различных автономных системах.

In this article was proposed a method of selecting clients, taking into account the physical location of peers and the delay between them. The shortest path between clients in the relevant AS is based on constructing a graph of the closest autonomous systems. The analysis of the dependence of the load factor on the number of customers and placing them in different autonomous systems is adopted.

Введение

Функционирование любой peer-to-peer системы распределения контента опирается на узлы и соединения между ними. Эта сеть образуется поверх и независимо от базовой (в типичном случае IP) и поэтому часто называется оверлейной. Топология, структура, степень централизации оверлейной сети, механизмы локализации и маршрутизации, которые в ней используются для передачи сообщений и контента, являются решающими для работы системы, поскольку они воздействуют на ее отказоустойчивость, производительность, масштабируемость и безопасность.

В основном существуют три различные архитектуры p2p систем[1,2]:

1) Централизованные (Napster[3]),
2) Децентрализованные структурированные (Gnutella[4], Edutella[5]),
3) Децентрализованные неструктурированные (Pastry[6], Chord[7], CAN[8], Tapestry[9]).

Децентрализованные структурированные системы используют распределенные хеш-таблицы (DHT[10]) для поиска данных на узлах. Недостатком таких систем является то, что не учитывается физическое расстояние между узлами. Таким образом в статье предложен способ, который позволяет на основании задержек между автономными системами находить кратчайший маршрут и использовать существующие алгоритмы внутри самой автономной системы.

Модифицированный способ

Сервер, который транслирует потоковое видео при подключении клиента к трансляции определяет по IP адресу, в какой автономной системе он находится. Таким образом будет определяться физическое место клиента в сети Интернет.

Поиск нужных кусков информации осуществляется с помощью связи ключ-значение. Идентификаторы ключа и значения состоят из следующих компонентов: IP адрес транслирующего сервера (32 бита), номер автономной системы (32 бита), IP адрес клиента (32 бита) и случайно сгенерирована хэш функция (32 бита). Последнее значение необходимо для того, чтобы каждый ключ и кусочек мультимедийных данных имели уникальный идентификатор, размером 128 бит.

Во время того, как клиент подключается к серверу, последний строит граф, в вершину которого он помещает себя. Ветки этого графа – это номера автономных систем, которые существуют в сети Интернет. В результате клиент, который послал запрос серверу для получения трансляции, будет помещен в узел дерева с номером соответствующей автономной системы. Пример графа показан на рис. 1.

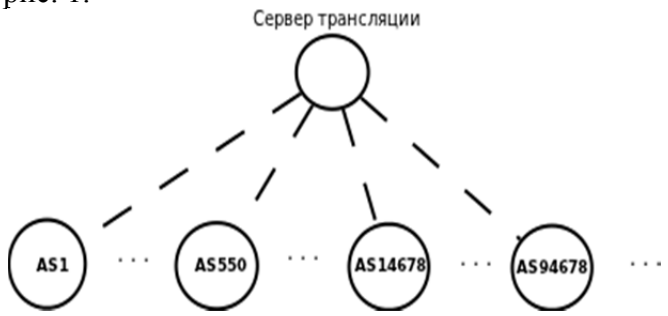

Рис. 1. Построения графа при подключении клиентов

Важный момент заключается в том, что при определении AS клиента, сервер также ищет

все соседние автономные системы по отношению к AS клиента (рис. 2)

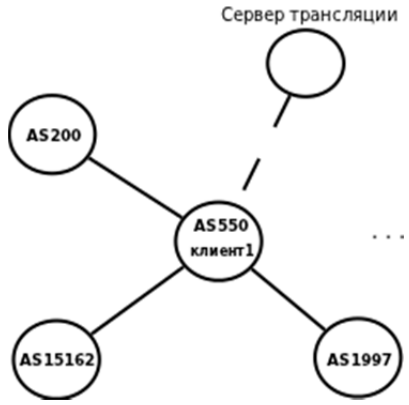

Рис. 2. Добавление соседних вершин узла

Таким образом, если соседняя автономная система окажется также соседней по отношению к AS другого клиента, то таким образом появится маршрут между двумя пирами, который будет является кратчайшим. При появлении новых клиентов и добавлении новых вершин к графу (рис.3) будет строиться остовное дерево с помощью алгоритма Прима[11]. Поиск кратчайшего маршрута между пирами будет осуществляться с помощью этого построенного дерева.

Поиск пиров для обмена мультимедийными данными происходим следующим образом. Внутри каждой автономной системе поиск нужной информации осуществляется средствами распределенных хеш-таблиц (DHT), которые используют существующий алгоритм Pastry. То есть здесь используется принцип децентрализации и чем больше активных узлов будет находиться в одной автономной системе, тем эффективнее он будет.

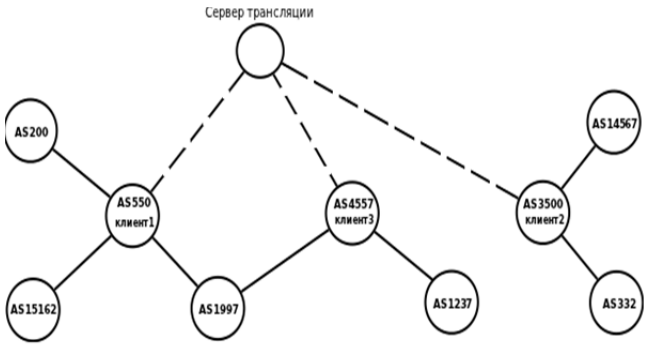

Рис. 3. Построение графа на сервере трансляции

Когда клиент найдет всех возможных пиров в локальной автономной системе, он отправляет запрос к серверу трансляции для получения информации о пирах из других автономных систем.

Сервер посылается на свой граф, который был построен в результате подключения новых клиентов. Таким образом он выдает адрес потенциально ближайшего пира к клиенту, который запрашивает информацию. Стоит отметить, что если в этой автономной системе находятся несколько пиров, то будет отдаваться адрес пира случайным образом. Далее по аналогии клиент с помощью алгоритмов DHT ищет пиров в ближайшей автономной системе.

По окончанию поиска клиент опять обращается с запросом о новых пирах. Если нету связи между вершинами автономных систем графа, в которых размещены клиенты, то сервер определяет случайным образом номер автономной системе для дальнейшего поиска.

Рис. 4. Физическая топология

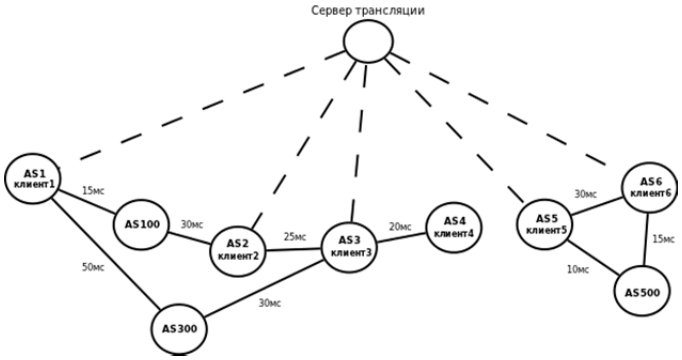

Рис. 5. Логическая топология для проведения экспериментов с предложенным алгоритмом

Моделирования передачи данных в p2p сети

На рис. 4 представлена полносвязная топология между узлами и сервером трансляции. Если рассматривать логические соединения, то для проведения экспериментов

использовались две разные топологии(рис. 5, рис. 6).

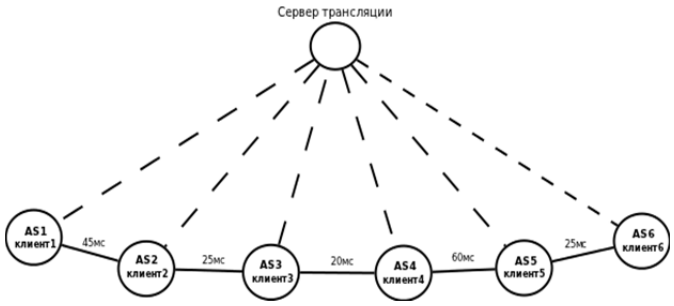

Рис. 6. Логическая топология для проведения экспериментов с базовым алгоритмом

Проведения экспериментов и анализ результатов

Для числового отображения качества алгоритма мы ввели переменную коэффициента нагрузки. Он показывает распределение потока входных и выходных данных на каждом из узлов и считается по формуле:

$$К_н = \frac{V_i}{V_o}, \quad (1)$$

где V_i – количество входящих данных, V_o – количество исходящих данных.

Для сбора статистики входящих и исходящих данных, а также вычисления $К_н$ на каждом узле был написан скрипт, который размещался на каджом из узлов.

Для того, чтобы статистика собиралась централизовано, использовался скрипт, который опрашивал каждый из узлов и получал соответствующие значения $К_н$.

Эксперименты были поделены на 2 этапа: использование готового алгоритма в идеальных условиях (нету задержек между пирами); использование готового и предложенного алгоритмов в реальной сети (в качестве узлов используются автономные системы, между которыми существуют различные задержки).

Результаты всех экспериментов приведены ниже на графиках (рис. 7, 8, 9).

На рис. 7 при проведении эксперимента на основе базового алгоритма Pastry виден экспоненциальный график. Это значит, что при увеличении количества клиентов при трансляции потокового видео коэффициент нагрузки стремится к единице. То есть количество входящего и исходящего трафика практически равно между собой. Получается, что клиент получает столько же данных для просмотра потокового видео, сколько он отдает (ретранслирует) другим пирам в сети.

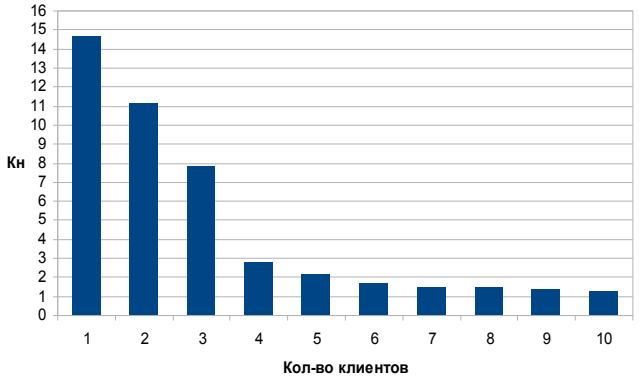

Рис. 7. Зависимость коэффициента нагрузки от количества клиентов

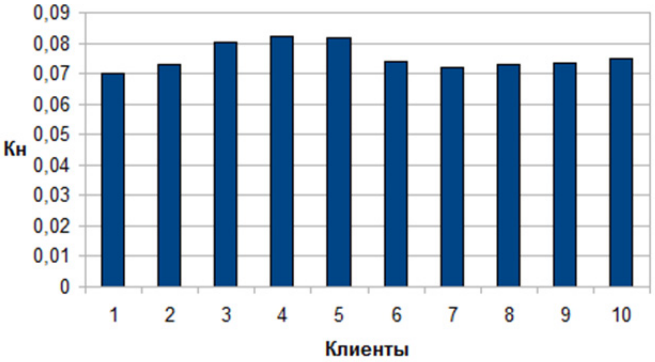

Рис. 8. Зависимость коэффициента нагрузки на сервере трансляции от количества подключенных клиентов

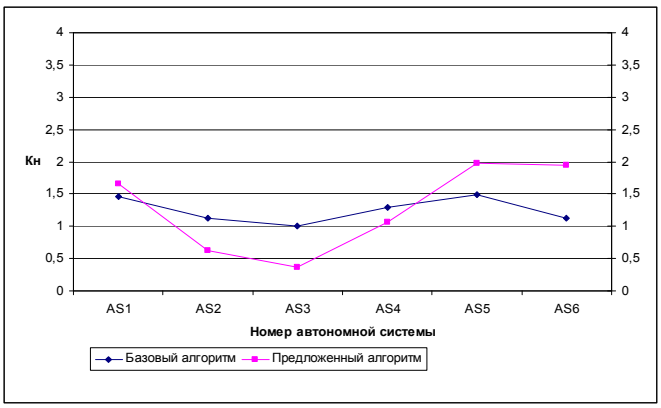

Рис. 9. Графики зависимости коэффициента нагрузки от автономной системы

Поэтому можно сделать вывод о том, что при большом количестве клиентов нагрузка на peer-to-peer сеть равномерно распределяется между ними. В этом и заключается основная задача DHT алгоритмов.

На рис. 8 был представлен график зависимости коэффициента нагрузки на сервере трансляции от количества подключенных клиентов. Он приближенный к прямой линии. Это объясняется тем, что коэффициент нагрузки на сервере совсем не зависит от

количества клиентов. При этом эти значения значительно меньше единицы, потому что на сервере трансляции в основном существует только исходящий трафик.

Последние два эксперимента заключались в сравнении базового алгоритма Pastry и предложенного нами алгоритма с учетом задержек между узлами. То есть клиенты находились в разных сетях, а точнее в разных автономных системах. Физическая топология в обоих случая была одинакова (рис. 4), но логические топологии отличались друг от друга (рис. 5, рис. 6). На рис. 9 изображено 2 графика, которые характеризуют зависимость коэффициента нагрузки от номера автономной системы, в которой размещался тот или иной клиент.

Проанализируем эти 2 графика отдельно:

– в базовом алгоритме совсем не учитываются задержки между узлами. Коэффициенты нагрузки на каждом из клиентов практически равны между собой и не зависят от расстояния между клиентами.

– в предложенном алгоритме идет неравномерное распределение коэффициентов нагрузки между автономными системами. Это объясняется тем, что в предложенном алгоритме учитывается расстояние между автономными системами. Поэтому в клиентах, расположенные в автономных системах AS2, AS3, которые имеют соседние вершины совсем близко друг к другу, больше исходящего трафика чем в клиентах из отдаленных автономных системах(AS1, AS4).

Также можно отметить, что клиенты, которые находятся в вершинах графа, который не связан с основным (граф, который возник при появлении первых клиентов, размещенных в соседних автономных системах) является наиболее отдаленным (автономные системы AS5, AS6). На графике видно, что отрезок между этими автономными системами имеет форму прямой линии еще и с довольно большим коэффициентом нагрузки. Это можно объяснить тем, что входящий трафик больше исходящего, потому что передача видео данных клиентам, которые находятся далеко от них, не столь приоритетна, как обмен видео потоком между соседними автономными системами. А вот клиент, который размещен в автономной системе AS3 взаимодействует с другими активными автономными системами больше всего. В этом случае исходящий трафик превосходит входящий.

При сравнении с базовым алгоритмом можно отметить, что коэффициент нагрузки при использовании предложенного алгоритма на клиентах, которые находятся в автономных системах AS2, AS3, AS4 меньше на 24,9%, 33,3%, 8,3% соответственно.

В результате сравнения двух графиков после проведения экспериментов базового и предложенного алгоритмов на основе данной топологии были обнаружены недостатки предложенного нами алгоритма. Они заключаются в том, что коэффициент нагрузки клиентов, которые находятся далеко от других (вершины графа, которые не соединены с основным, а также вершины, которые находятся далеко от основного скопления других активных вершин) достаточно больше по сравнению с базовым алгоритмом (около 10,99%, 19,7%, 32,1% соответственно для AS1, AS5, AS6).

Выводы

1. При увеличении количества клиентов коэффициент нагрузки на каждом из них стремится к единице, что означает равномерную нагрузку в peer-to-peer сети.

2. На сервере трансляции коэффициент нагрузки не зависит от количества включенных пиров.

3. Предложенный способ за счет учета задержек между клиентами позволяет повысить эффективность передачи видео данных через peer-to-peer сеть.

4. В качестве недостатка предложенного алгоритма можно отметить следующее, что коэффициент нагрузки удаленных клиентов от основного скопления активных пиров, больше по сравнению с базовым алгоритмом Pastry.

Список литературы

1. D. Chopra, H. Schulzrinne, E. Marocco, and E. Ivov. Peer-to-Peer Overlays for Real-time Communication: Security issues and Solutions. IEEE Communications Surveys Tutorials 11, no. 1, 2009, pages 4–12.
2. Philip A. Bernstein, Fausto Giunchiglia, Anastasios Kementsietsidis, John Mylopoulos, Luciano Serafini, and Ilya Zaihrayeu, “Data management for peer-to-peer computing : A vision,” in WebDB 2002, June 2002, pp. 89–94.

3. Napster: http://www.collegetermpapers.com/viewpaper/1304273730.html.
4. Gnutella: http://web.archive.org/web/20090331221153/http://wiki.limewire.org/index.php?title=GDF.
5. Edutella: http://edutella.jxta.org/
6. A. I. T. Rowstron, P. Druschel. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In: Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), 2001, pages 329–350.
7. I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications. IEEE/ACM Transactions on Networking 11, no. 1, 2003, pages 17–32.
8. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content-addressable Network. In: Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (ACM SIGCOMM), 2001, pages 161–172.
9. B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz. Tapestry: A Resilient Global-scale Overlay for Service Deployment. IEEE Journal on Selected Areas in Communications 22, no. 1, 2004, pages 41–53.
10. T. Balke, W. Siberski, DHT Algorithms, Springer LNCS 3485, 2007
11. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to Algorithms, Third Edition. MIT Press, ISBN 0-262-03384-4. Section 23.2: The algorithms of Kruskal and Prim, 2009, pp. 631–638.