$$Z_{11}^{(2)} = Z_{22}^{(2)} = \frac{R_2}{\Theta_2} * cth\Theta_2, \quad Z_{12}^{(2)} = Z_{21}^{(2)} = \frac{R_2}{\Theta_2} * \frac{1}{sh\Theta_2}, \quad (6)$$

где $\Theta_1 = \sqrt{jwR_1C_1}$, $\Theta_2 = \sqrt{jwR_2C_2}$.

Подставив последнее выражение в формулу (5) и произведя некоторые преобразования, получаем

$$\dot{K}(w) = \frac{1 + \frac{R_2}{R_1} * \frac{\Theta_1}{\Theta_2} * \frac{sh\Theta_2}{sh\Theta_1}}{ch\Theta_1 * \left(\frac{ch\Theta_2}{ch\Theta_1} + \frac{R_2}{R_1} * \frac{\Theta_1}{\Theta_2} * \frac{sh\Theta_2}{sh\Theta_1} \right)}. \quad (7)$$

Коэффициент передачи фильтра нижних частот (7) и аналогично параметры матрицы (4) зависят от соотношений $\frac{R_2}{R_1}$ и $\frac{\Theta_2}{\Theta_1}$. В случае одинаковых RC-структур, то есть $R_1=R_2$ и $\Theta_1 = \Theta_2$, параметры матрицы (4) и выражение коэффициента передачи (7) упрощаются:

$$\left\| Z^{(1,2)} \right\| = \frac{R_1}{2 * \Theta_2} \begin{Vmatrix} cih\Theta_1 & \operatorname{csc} h\Theta_1 \\ \operatorname{csc} h\Theta_1 & cth\Theta_1 \end{Vmatrix}, \quad (8)$$

$$\dot{K}(w) = \frac{1}{ch\Theta_1}. \quad (9)$$

Сравнение параметров матриц (8) и (6) показывает, что у фильтра нижних частот, полученного параллельным соединением двух одинаковых RC-структур сои распределенными параметрами, входное и выходное сопротивления в два раза меньше, чем у фильтра на основе одиночной RC-структуры с однородными распределенными параметрами.

Коэффициент передачи (9) рассматриваемого фильтра совпадает с выражением коэффициента передачи одиночной RC-структуры с однородными распределенными параметрами.

МАРШРУТИЗАЦИЯ В СЕТЯХ С КОММУТАЦИЕЙ ПАКЕТОВ

Кравчук А. В.

Задача маршрутизации – это определение оптимальных трактов маршрутизации для транспортировки информационных групп (обычно называемых пакетами) через объединенную сеть. Алгоритмы маршрутизации

решают задачу выбора оптимального маршрута к пункту назначения, основываясь на вычислении показателей качества маршрутов. Показатели качества вычисляются на основе различных характеристик (например, длина маршрута, пропускная способность канала, состояние канала и другие) или их комбинаций. Существующие в настоящее время варианты решения задачи маршрутизации можно разделить на два класса: алгоритмы, основанные на вычислении длины маршрута - DVA *(Distance Vector Algorithms)* и алгоритмы, основанные на вычислении показателей состояния каналов - LSA*(Link State Algorithms)*.[5] В больших сетях со сложной топологией и неоднородной кабельной системой LSA предпочтительнее, чем DVA. Это является следствием того, что показатели качества (длина маршрута), применяемые в DVA, не учитывают текущее состояние сети (загруженность каналов, ширина каналов, надежность и т.п.). Основным недостатком LSA являются повышенные требования к аппаратуре, используемой для маршрутизации (по памяти, производительности и, как следствие по цене). Решения на базе этого подхода гораздо более сложны в настройке и сопровождении, по сравнению с DVA. В сетях относительно небольшого размера с простой топологией и однородной кабельной системой DVA весьма успешно справляется с задачей маршрутизации, не требуя столь мощной аппаратуры для своей работы как LSA. В данном случае все преимущества LSA оказываются невостребованными, поэтому в таких сетях применение более простых DVA становится оправданным. Однако существующие реализации DVA-подхода имеют ряд серьезных недостатков, которые могут негативно отразиться на эффективности работы даже сети простой конфигурации. В частности, наиболее широко распространенная реализация DVA-подхода – протокол RIP страдает таким серьезным недостатком, как большой объем служебного трафика, занимающего существенную часть полосы пропускания каналов передачи данных и, как следствие, снижающего эффективность работы сети. В данной работе предлагается алгоритм, основанный на DVA-подходе, позволяющий устранить, указанный выше недостаток RIP[3,5]. Суть предлагаемого решения состоит в том, чтобы пересылать только обновления маршрутной таблицы. В пакеты обновления включается информация и об удаленных строках маршрутной таблицы. Такие строки помечаются специальным флагом удаления для того, чтобы принимающая сторона могла вычеркнуть их из своей маршрутной таблицы. Кроме того, вводится счетчик обновлений, поскольку при сбое сети может произойти потеря очередного обновления и синхронизация между маршрутизаторами будет нарушена. Маршрутизатор, посылающий обновление маршрутной таблицы, увеличивает значение счетчика обновлений на единицу и записывает его в пакет обновления. Принимающая сторона сравнивает это значение

со значением счетчика, полученного в предыдущем пакете обновления от данного соседа для обнаружения возможной потери пакета обновления. В случае несовпадения значений счетчиков, маршрутизатор запрашивает полную таблицу. При получении полной таблицы, счетчик сбрасывается. Кроме того, через некоторые фиксированные промежутки времени можно проводить рассылки полной маршрутной таблицы для синхронизации данных между различными маршрутизаторами. Современная технология построения сетей передачи данных позволяет передавать данные с ничтожно малой вероятностью появления ошибок. Кроме того, при использовании протокола с гарантированной доставкой (TCP) в качестве транспортного протокола, можно утверждать, что потери пакетов обновлений маршрутной информации будут весьма редким явлением.

Подход к решению задачи маршрутизации, предложенный в этой работе является модификацией алгоритма Беллмана-Форда[1,2]. Суть модификации состоит в том, что исходные значения матрицы пересчета могут быть произвольными, кроме диагональных элементов, которые должны быть нулевыми. Этот момент является очень важным для практической реализации, поскольку в данном случае гарантируется, что при выходе из строя какого-либо канала передачи данных, соответствующий маршрут будет удален из маршрутной таблицы. В классическом же алгоритме Беллмана-Форда предполагается, что все начальные значения матрицы пересчета, кроме диагональных, равны ∞, т.е., маршрутная таблица изначально пуста.

Математическое описание алгоритма

Дано: Связный граф G=(E,U)

$$d(i,j) = \begin{cases} вес\ ребра(i,j) > 0, & если\ (i,j) \in U \\ \infty & , иначе \end{cases}$$

Найти:

$$\forall i,j \in E, r(i,j): V(r(i,j)) = \min_{q(i,j) \in Q} V(q(i,j)) \quad ,$$

$$где\ V(r(i,j)) = \sum_{(m,l) \in r(i,j)} d(m,l)$$

Q – *множество цепей*

Т.е. r(i,j) — оптимальный путь от i до j.

Решение:

$D0(i,i)=0;\qquad D0(i,j)=$ *произвольное при* $i \neq j$

$p=0$

Выполнить

$$\begin{cases} p=p+1 \\ D_p(i,i)=0 \\ D_p(i,j)=\min\limits_{k\in E}[d(i,k)+D_{p-1}(k,j)], \quad i\neq j \qquad (1) \\ K_p(i,j)=k_{\min}, \quad \text{где } D_p(i,j)=d(i,k_{\min})+D_{p-1}(k_{\min},j) \end{cases}$$

Пока $D_p(i,j)\neq D_{p-1}(i,j)$

Заметим, что в уравнении (1) мы можем ограничиться только теми значениями k, которые соответствуют непосредственным соседям i (т.к. для других k d(i,k)=∞). Тогда $D_p(i,j)$ будет значением оптимального пути от i до j, $K_p(i,j)$ будет значением первой промежуточной вершины оптимального пути от i до j.

(Заметим, что относительно начальных значений $D_0(i,j)$ никаких предположений не делается, кроме $D_0(i,i)=0$).

Обоснование алгоритма

Пусть O(i,j) — значение оптимального пути.

1) Пусть $O(i,j)\le D_0(i,j)$, $\forall i,j$.

Покажем, что в этом случае алгоритм сойдется не более чем за n итераций. Пусть $O_p(i,j)$ — значение оптимального пути из вершины i в вершину j, включающего не более p промежуточных вершин. Покажем по индукции, что выполняется неравенство $D_p(i,j)\le O_{p-1}(i,j)$.

а) p=1. В этом случае $O_0(i,j)=d(i,j)$ (т.к. $O_0(i,j)\neq\infty$ только если существует дуга (i,j). Тогда $O_0(i,j)=d(i,j)$ — стоимость пути, состоящего из одной дуги).

б) Допустим, что для p неравенство выполняется. Докажем для $p+1$.

$$D_1(i,j)=\min_k[d(i,k)+D_0(k,j)]\le d(i,j)+D_0(j,j)=d(i,j)=O_0(i,j),$$

$D_{p+1}(i,j) = \min_k [d(i,k) + D_p(k,j)] \le \min_k [d(i,k) + O_{p-1}(k,j)] = O_p(i,j)$,

Можно доказать (по индукции), что $D_p(i,j) \ge O(i,j)$ т.к.

$O(i,j) = \min_k [d(i,k) + O(k,j)] \le \min_k [d(i,k) + D_0(k,j)] = D_1(i,j)$

(и т.д. по индукции).

В результате получаем: $O(i,j) \le D_p(i,j) \le O_{p-1}(i,j)$.

Но в тоже время $O_{n-1}(i,j) = O(i,j)$ (т.к. кратчайший путь не может включать в себя какую-либо вершину более одного раза, поскольку это означало бы, что путь содержит положительную петлю, выкинув которую мы можем получить более короткий путь). Тогда $O(i,j) = D_n(i,j)$, т.е. алгоритм в этом случае сходится не более, чем за *n* итераций.

Пусть $\exists\ D_0(i,j) < O(i,j)$.

Покажем, что за конечное число шагов мы получим матрицу $D_p(i,j)$, которая удовлетворяет условию п.1).

Пусть $C_p = \{(i,j) : D_p(i,j) < O(i,j)\}$

Обозначим $I_p = \min\{D_p(i,j) : (i,j) \in C_p\}$.

Пусть при $k = k_{\min}$ достигается минимум в выражении (1).

Рассмотрим $D_p(i,j) = d(i,k_{\min}) + D_{p-1}(k_{\min}, j)$.

а) Пусть $(k_{\min}, j) \in C_{p-1}$. Тогда $D_p(i,j) \ge d(i,k_{\min}) + I_{p-1} > I_{p-1}$.

б) Пусть $(k_{\min}, j) \notin C_{p-1}$. Тогда справедливо

$D_p(i,j) = d(i,k_{\min}) + D_{p-1}(k_{\min}, j) \ge d(i,k_{\min}) + O(k_{\min}, j) \ge O(i,j)$ Из этого неравенства следует, что $(i,j) \notin C_p$.

В результате получаем набор чисел $I_0 < I_1 < I_2 < \cdots < I_p$. Причем $I_p - I_{p-1} \ge d_{\min} > 0$, где $d_{\min} = \min\{d(i,j)\}$. Этот набор чисел не может быть бесконечным, т.к. $I_p < \max O(i,j)$ (по определению I_p) и элементы увеличиваются при каждой итерации как минимум на $d_{\min}$. Сле-

довательно, на некотором шаге $p_{\max}$ процесс должен остановиться. Это произойдет либо из-за того, что $C_{p_{\max}} = \varnothing$, либо из-за условия остановки алгоритма $D_{p_{\max}}(i,j) = D_{p_{\max}-1}(i,j)$. Допустим, что выполняется условие $D_{p_{\max}}(i,j) = D_{p_{\max}-1}(i,j)$. Тогда $C_{p_{\max}} = C_{p_{\max}-1}$ и, следовательно $I_{p_{\max}} = I_{p_{\max}-1}$. Это противоречит условию $I_{p_{\max}} > I_{p_{\max}-1}$. Следовательно, $C_{p_{\max}} = \varnothing$, т.е. матрица $D_{p_{\max}}(i,j)$ удовлетворяет условию п.1) и ее можно рассматривать в качестве начальной матрицы для п.1). Для п.1) сходимость уже доказана выше. Следовательно, алгоритм сходится за конечное число итераций.

Механизм работы алгоритма на маршрутизаторах

На маршрутизаторах алгоритм функционирует следующим образом. Маршрутизатор *i* получает от каждого своего соседа *k* значение D(k,j) (иными словами это представление маршрутизатора *k* о расстоянии от *k* до *j*). Затем прибавляет к каждому полученному значению d(i,k) (это стоимость использования сети, соединяющей *i* и *k*) и выбирает из этих чисел минимальное. Доказательство того, что алгоритм сойдется за конечное время приведено выше. Не делается никаких предположений относительно того, в какие моменты времени рассылается информация и вычисляется min. Поэтому отсутствует необходимость синхронизации рассылки информации (маршрутизаторы работают асинхронно, т.е. каждый из них осуществляет рассылку по собственному таймеру). Поскольку не делается практически никаких предположений относительно начальных значений, алгоритм может отслеживать изменения. Когда происходит изменение в системе, алгоритм начинает сходиться к новой точке равновесия, используя старое значение как начальное.

Реализация

Изложенный выше алгоритм был реализован и встроен в программу routed из поставки Unix (FreeBSD). Программа работает как фоновый Unix-процесс на 520 UDP-порту.[4,6] Полученные на этот порт сообщения обрабатываются в зависимости от значения, записанного в поле rip_cmd данного сообщения.

Поле rip_cmd может принимать следующие значения:

RIPCMD_REQUEST (1)

Запрос на получение частичной или полной маршрутной информации.

RIPCMD_RESPONSE (2)

Отклик, содержащий информацию о расстояниях из маршрутной таблицы отправителя.

RIPCMD_UPDATE

Сообщение, содержащее информацию об изменениях в маршрутной таблице отправителя. Маршруты, удаленные из таблицы, записываются с показателем DEL_ROUTE в сообщении этого типа.

Тестирование программы показало, что задачу поиска оптимальных маршрутов она решает успешно. Маршрутные таблицы соответствуют реальной топологии сети. Объем служебного трафика сокращается на 70-80 процентов в зависимости от частоты изменений топологии сети. Таким образом, для сетей небольшого размера, построенных на надежных каналах передачи данных, применение описанного выше алгоритма представляется недорогим (с точки зрения затрат на аппаратные средства) и эффективным решением. Наиболее существенное сокращение служебного трафика достигается на сетях с редко меняющейся топологией.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Алгоритмы оптимизации на сетях и графах. Э. Майника. –М.: Мир ,1981
2. Введение в прикладную комбинаторику. А.Кофман. –М.: Наука ,1975,
3. Лабиринт Internet (практическое руководство). П.Храмцов. – М.: ЭЛЕКТРОНИНФОРМ ,1996.
4. Системное программирование на C++ для Unix. Теренс Чан. – К.:Издательская группа BHV,1997.
5. Internetworking Technology Overview. (http://www.mark-itt.ru/Collection/CISCO/ITO/)
6. Unix: Руководство системного администратора. Э.Немет, Г.Снайдер, С.Сибасс, Т.Хейн. –К.: BHV,1996.