УДК 519.178 Время построения остовного дерева в сети с неизвестной топологией и синхронизированным временем.

М.Н. Вялый, И.М. Хузиев

15 октября 2015 г.

Введение

Распределённые вычисления — важная и активно развивающая область теоретической информатики. В разработке распределённых протоколов значительное место занимает решение проблемы передачи данных от одного узла сети ко всем другим узлам (broadcasting). Эта задача нетривиальна, если сеть состоит из узлов, которые могут обмениваться сообщениями лишь по предустановленным соединениям. Для решения такой задачи оповещения полезно иметь в сети структуру остовного дерева, так как это позволяет передавать данные без избыточных пересылок и необходимости фильтровать дублирующие сообщения.

В данной работе мы рассматриваем задачу построения остовного дерева в синхронизированной сети с априорно неизвестной топологией. Основная для этой работы постановка задачи предполагает, что узлы сети различимы, но различающая их информация (длины идентификаторов узлов) может быть сколь угодно велика. В качестве меры сложности можно рассматривать либо объём — общее количество переданных битов по всем связям сети —, либо время — общее время работы протокола в сети. Последнее и будет темой данного доклада.

Постановка задачи при минимизации объёма близка к задачам коммуникационной сложности, но с большим количеством участников и синхронизацией передачи. В [2] Импальяццо и Вильямс предложили модель коммуникационной сложности с синхронизированным временем и установили близкие верхние и нижние оценки, связывающие синхронизированную коммуникационную сложность и обычную детерминированную коммуникационную сложность в предположении, что время работы

протокола ограничено полиномом от числа булевых переменных у участников протокола. Заметим, что при двух участниках отсутствие ограничений на время делает задачу тривиальной: синхронизированная коммуникационная сложность любой функции становится $O(1)$ (участники протокола могут обменяться своими входными данными, отправив 2 бита информации в моменты времени, кодирующие эти данные).

В случае большого числа участников задача остаётся осмысленной и при отсутствии ограничений на время работы протокола. Именно этот случай мы изучаем в главе **??**.

Нижней оценкой для распределённой коммуникационной сложности протоколов построения остовного дерева является количество рёбер в сети (см. ниже раздел **??**). С точностью до мультипликативной константы эта оценка оптимальна, если узлам сети известна верхняя оценка размера сети (раздел **??**).

Построение такого протокола опирается на предположение о синхронизированности сети. В этом случае узлы сети могут передавать информацию, выбирая время отправки сообщения (которое является ресурсом, известным всем узлам сети). Эта идея не нова и использовалась, например, в работе [3] для решения задачи выбора лидера в кольцевой сети.

Существует субоптимальная оценка, которая не зависит от размера различающей узлы информации и не использует априорной информации о размерах компонент. Для любой неограниченной монотонной функции $g(x)$ мы приводим протокол сложности $Eg(V)$, где E — количество рёбер в сети, а V — количество вершин.

Заметим, что для асинхронных сетей такой объём недостижим. Это следует из нижних оценок для числа сообщений (неограниченного размера) в асинхронных круговых сетях, требуемых для решения некоторых родственных задач. Для кольцевых сетей в [4] доказана нижняя оценка $\Omega(V \log V)$ для числа сообщений в протоколе, решающем задачу выбора лидера (уникального узла в сети). В работе [5] получена такая же нижняя оценка для решения задачи достижения консенсуса (все узлы должны выдать одинаковый бит) в кольцевой сети. Нетрудно видеть, что эти задачи решаются с дополнительными $O(V)$ сообщениями, если в кольцевой сети с заданными ориентациями связей в каждой вершине (по и против часовой стрелки) построено остовное дерево (т.е. удалено одно ребро).

Подробнее про постановку задачи с минимизацией объёма можно прочесть в статье [1].

Что касается времени работы протокола построения остовного дерева, то известный и используемый на практике алгоритм Р. Перльман [6] работает за время $O(V \log N_{\min})$, где $N_{\min}$ — наименьший номер узла в сети. В разделе 1 будет доказана нижняя оценка на время работы протокола в кольцевой сети $\Theta(V + \log N_{\min})$. Первое слагаемое следствие того, что протокол в любой сети не может завершится раньше, чем через R тактов, R — радиус графа (теорема 1). Второго слагаемое является следствие теоремы 2, которая позволяет в графах имеющих симметрию использовать классическую коммуникационную сложность для оценки времени работы протокола. В разделе 2 приводится протокол, работающий на кольце $O(V \log N_{min} + \log N_{min})$. Точность оценки $O(V \log N_{\min})$ в худшем случае остаётся открытым вопросом.

Отметим, что известны нижние оценки времени распределённого построения кратчайшего остовного дерева [7], однако в другой модели: узлы сети пронумерованы без пропусков числами от 1 до V, за один такт разрешается передать $O(\log V)$ битов.

Определения

Нас интересует установление соединений в сетях очень большого размера, в которых узлы сети не имеют заранее заданной нумерации и различающая их информация может быть очень избыточной по сравнению с количеством узлов в сети. В таких предположениях оказывается удобной следующая формулировка.

Имеется неориентированный граф H со счётным количеством вершин (узлов), причём степени (количество соседей) всех узлов конечны и все компоненты связности конечны. Обозначим множество компонент связности через $F(H)$. Далее считаем, что вершины графа перенумерованы натуральными числами, которые мы будем называть идентификаторами узлов. Кроме того ребра (связи) в каждом узле также перенумерованы.

В такой модели сеть представляется компонентой связности бесконечного графа. Формальное удобство такого определения состоит в единообразном описании протоколов (см. ниже) и упрощении доказательств (см. доказательство теоремы **??** в следующем разделе **??**).

Каждый узел является участником интерактивного протокола. Предполагаем, что время дискретно и сеть синхронизирована. Это означает, что общение в сети происходит по тактам. В каждом такте по всем связям, причём в обе стороны, могут передаваться сообщения (пустые сообщения также допустимы).

Помимо идентификатора каждый узел может хранить неограниченное количе-

ство информации. Протокол предполагается однородным, т.е. действия каждого узла описываются одинаково.

Мы будем рассматривать несколько вариантов постановки задачи. Они будут отличаться начальными состояниями узлов сети, скоростью передачи по связи и возможностью использования случайных битов в протоколе.

Основным вариантом для нас является случай одинаковых начальных состояний, единичной скорости передачи (один бит за такт) и детерминированных протоколов.

Основной задачей, которую мы рассматриваем, является построение остовного дерева в каждой компоненте связности графа. Когда остовное дерево уже построено, передача сообщения от одного узла всем остальным в данной компоненте (broadcasting) выполняется оптимальным образом: объём передаваемых данных равен $L(V-1)$, где L — длина сообщения, а V — количество узлов в компоненте связности. Время передачи информации по дереву оценивается как $V+L$.

В первой главе нас прежде всего интересует объём передаваемых данных при построении остовного дерева. Эта величина характеризует распределённую коммуникационную сложность построения остовного дерева.

Во второй главе нас интересует время, за которое можно построить дерево. Эта величина также тесно связана с коммуникационной сложностью функций: полученные нижние оценки сводятся к коммуникационной сложности асимметричных двухместных функций.

Теперь дадим формальные определения.

В начальный момент $t=0$ каждый узел сети помимо идентификатора n имеет начальное состояние $s_n(0)$, которое задаётся натуральным числом. Считаем, что в момент времени t узлу доступна вся информация, которая проходила через этот узел к этому моменту. Она описывается последовательностью состояний $s = s_n(i)$, $0 \leqslant i \leqslant t$, и последовательностями сообщений, полученных по каждой связи $m = m_n^k(i)$, $0 \leqslant i < t$, $1 \leqslant k \leqslant d$, где d — количество связей данного узла. Сообщения будем считать двоичными словами.

Детерминированный протокол – это набор функций

$$\pi_\ell(n, s, m),$$

которые при $1 \leqslant \ell \leqslant d$ задают сообщение, пересылаемое узлом n по ℓ-й связи на основании истории s, m, а при $\ell = 0$ — новое состояние узла n.

Таким образом, состояние узла в момент $t+1$ задаётся как

$$s_n(t+1) = \pi_0(n, s_n(t), m_n(t)),$$

а полученные сообщения имеют вид

$$m_n^k(t) = \pi_\ell(u, s_u(t), m_u(t)),$$

где k-я связь соединяет узлы n и u, причём номер этой связи в узле u равен ℓ.

Завершающее состояние, которое будем без ограничения общности нумеровать нулём, удовлетворяет следующим свойствам:

$$\pi_0(n, (s_0, s_1, \ldots, 0), m) = 0,$$
$$\pi_\ell(n, (s_0, s_1, \ldots, 0), m) = \lambda,$$

где λ — пустое слово. Узел в завершающем состоянии отключается от коммуникации: не посылает никаких сообщений и не изменяет состояния при получении любых сообщений.

Ждущим состоянием будем называть такое состояние, что узел находящийся в нём

- посылает только пустые сообщения,
- может изменить состояние только при получении непустого сообщения.

Из определения сразу следует, что если все узлы компоненты связности графа перешли в ждущие состояния, то далее никаких изменений в состояниях узлов не происходит.

Вероятностный протокол – это набор функций

$$\pi_\ell(n, s, m, r_n),$$

аналогичных функциям детерминированного протокола. Дополнительный аргумент r_n считаем бесконечной последовательностью случайных битов. Будем считать, что при фиксированных n, s, m значение функции π_ℓ зависит только от конечного префикса r_n. Неформально это условие означает, что на каждом такте работы узел выполняет лишь конечное число действий, включая подбрасывания монеты.

Замечание 1. Состояние узла в данный момент времени полностью определяется начальным состоянием, историей полученных сообщений и, если протокол вероятностный, случайными битами, доступными узлу.

Определение 1. Протокол назовём корректным, если для любого графа H и любой компоненты связности $G \in F(H)$ существует такой момент времени T_G, начиная с которого состояния всех узлов компоненты завершающие: $s_n(t) = 0$ для $t \geqslant T_G$ и $n \in G$.

Определение 2. Протокол назовём слабо корректным, если для любого графа H и любой компоненты связности $G \in F(H)$ существует такой момент времени T_G, начиная с которого состояния всех узлов компоненты ждущие.

Определения корректного (processor terminating) и слабо корректного (message terminating) протоколов взяты из работы [8]. Ниже мы приводим определения для корректных протоколов. Их обобщение на случай слабо корректных протоколов производится очевидным образом.

Для корректного протокола в каждом узле определена конечная последовательность состояний $s_n(\infty)$, заканчивающаяся нулём, и конечные последовательности слов $m_n^k(\infty)$, задающие полученные по k-й связи сообщения (до полной истории полученных по этой связи сообщений она дополняется бесконечной последовательностью пустых слов). Через $m_n(\infty)$ обозначаем набор всех сообщений, полученных узлом n в корректном протоколе.

Для компоненты связности $G \in F(H)$ объёмом корректного протокола назовём общую длину сообщений, переданных внутри этой компоненты

$$S_G = \sum_{n \in G} \sum_{k=1}^{d(n)} \sum_{t=0}^{\infty} |m_n^k(t)|.$$

Здесь $d(n)$ — количество связей в узле n, а $|m|$ — длина слова m.

Определение 3. Корректный протокол строит остовное дерево в каждой компоненте, если существует такая функция $c\colon (n, s, m) \mapsto N \subset \mathbb{N}$, что следующие свойства выполняются для любого графа H

- $c(n, s_n(\infty), m_n(\infty)) \subseteq \{1, \ldots, d\}$, где d — количество связей узла n;
- функция c задаёт связи: если узлы n' и n'' связаны, причём номер этой связи в узле n' равен k', а в узле n'' равен k'', то $k' \in c(n', s_{n'}(\infty), m_{n'}(\infty))$ влечёт $k'' \in c(n'', s_{n''}(\infty), m_{n''}(\infty))$;
- в каждой компоненте связности множество заданных функцией c связей образует остовное дерево.

Определение 4. Корректный протокол строит остовное корневое дерево в каждой компоненте, если помимо функции c со свойствами из определения 3 для него также определена функция ρ: $(n, s, m) \mapsto \alpha \in \{0, 1\}$, которая обладает следующим свойством: для любого графа H и любой компоненты связности $G \in F(H)$ только одно из значений $\rho(n, s_n(\infty), m_n(\infty))$ равно 1 при $n \in G$. (В дереве выделен единственный корень.)

Для вероятностных протоколов мы используем те же определения. В этом случае указанные в определениях свойства могут не выполняться с некоторой вероятностью ошибки. Протокол работает с вероятностью ошибки не более ε, если для каждой компоненты связности вероятность ошибки не превосходит ε. Нас интересует построение протоколов со сколь угодно малой вероятностью ошибки.

Работу протокола будем оценивать двумя параметрами: объёмом S и временем работы T. Временем работы корректного протокола на компоненте связности G называем первый момент времени, когда все узлы компоненты перешли в завершающее состояние. Для слабо корректного протокола вместо завершающих состояний используем ждущие.

Объем и время работы протокола зависят от компоненты связности. Мы будем искать оценки на объём и время работы, зависящие от числа узлов V в компоненте, числа связей E в компоненте и значения минимального из номеров вершин компоненты $N_{min} = \min\limits_{G} n$.

Как уже говорилось выше, нас прежде всего интересует случай, когда начальные состояния узлов одинаковы (без ограничения общности полагаем $s_n(0) = 1$), а длина передаваемого за один такт сообщения $|m_n^k(t)|$ не превосходит 1 (ограниченные протоколы). Однако мы будем также рассматривать оценки и для протоколов, в которых длина передаваемого за такт сообщения не ограничена константой.

Заметим, что если для неограниченного протокола существует функция $M(t)$, которая ограничивает длину сообщений, передаваемых в момент времени t по всем узлам, то такой протокол может быть преобразован в ограниченный того же объёма. Действительно, заменим t-й такт работы неограниченного протокола на $M(t)$ тактов, в i-м из которых узлы передают значение i-го бита сообщения из неограниченного протокола (или передают пустое слово, если сообщение короче).

Для оценок времени удобно использовать равномерно ограниченные протоколы, то есть протоколы, в которых длина сообщения за один такт не превосходит некото-

рой константы C. Рассмотрение равномерно ограниченных протоколов равносильно использованию алфавита Σ, $|\Sigma| \leqslant 2^C$.

1 Нижние оценки времени

Докажем вначале две несложных теоремы. Первая даст нам оценку снизу на сложность протокола в терминах структуры графа, а вторая — в терминах номеров вершин, входящих в компоненту.

Теорема 1. *Для всякого ограниченного детерминированного корректного протокола построения остовного дерева выполняется* $T_G = \Omega(R_G)$, *где* R_G *— радиус графа.*

Как и в разделе **??** доказывается, что в графе H может найтись не более двух компонент, в которых не выполняется оценка.

Доказательство. Идея доказательства проста: за меньшее время сигнал просто не успеет распространиться.

Напомним, что радиусом графа называют

$$R_G = \min_{u \in G} \max_{v \in G} d(u, v),$$

где $d(u, v)$ — наикратчайшее в графе расстояние между двумя вершинами.

Пусть протокол π фиксирован. Предположим, что имеются две компоненты $G_1, G_2 \in F(H)$, в которых протокол завершается за время меньшее $R(G_1)$ и $R(G_2)$ соответственно. Пусть вершины $v_1 \in G_1, v_2 \in G_2$ являются корнями деревьев, которые строит π. Пусть также вершины $u_1 \in G_1, u_2 \in G_2$ отдалены от v_1 и v_2 не меньше чем на R_{G_1}, R_{G_2} соответственно.

Рассмотрим граф, состоящий из объединения графов G_1, G_2 (объединение дизъюнктно в силу того, что это подграфы H) и дополнительного ребра (u_1, u_2). Из предположения о времени работы π, протокол переведёт вершины v_1 и v_2 в завершающие состояния к моменту времени $\max\{R_{G_1} - 1, R_{G_2} - 1\}$, причём обе вершины будут считать себя корнем: на такте t состояния всех узлов, лежавших в G_1 и удалённых от v_1 на расстояние $< R_{G_1} - t$, совпадает с состоянием в графе G_1 (до добавления ребра), а значит в момент времени $R_{G_1} - 1$ узел v_1 должен находиться в завершающем состоянии и считать себя корнем. Аналогичное рассуждение верно и для v_2.

Таким образом, π не строит остовное дерево. □

Перед доказательством теоремы 2 докажем её упрощённый вариант.

Утверждение 1. *Для всякого ограниченного протокола π время его работы в худшем случае на графе из одного ребра оценивается как $T_G(\pi) = \theta(\log N_{\min})$.*

Отдельно поясним, какая именно оценка доказывается. Для всякого числа N_{min} существуют числа N_1, N_2 такие, что длина их битовой записи совпадает с длиной битовой записи N_{min} и протокол π на графе из одного ребра, вершины которого имеют номера N_1 и N_2, будет работать не менее $\lceil \log N_{\min} \rceil / 2$ тактов времени.

Доказательство. Для доказательства будем использовать коммуникационную сложность функции двух аргументов и метод сложных множеств (подробнее см [9]).

Рассмотрим произвольную булеву функцию $f(x, y)$, определённую на парах слов длины n. Пусть также она асимметрична, то есть $f(x, y) \neq f(y, x)$, если $y \neq x$.

Для всякой такой функции диагональ является трудным множеством (два диагональных элемента не могут входить в один комбинаторный прямоугольник). Следовательно, её коммуникационная сложность не меньше n.

Задачу построения дерева с корнем в нашем графе можно рассматривать как коммуникационную задачу: есть два узла A, B со входами x_A, x_B, узел A должен вычислить $f(x_B, x_A)$, а узел B — $f(x_A, x_B)$. Тот узел, результат которого оказался равным единице, считается корнем.

То есть, каждому протоколу π соответствует некоторая асимметричная функция f^n_π, которая представляет собой ограничение результата работы π на случай графа из одного ребра и входов длины n.

Учитывая то, что за такт времени передаётся 2 бита информации (канал двунаправленный), то время работы протокола не может быть меньше половины коммуникационной сложности функции f^n_π. Следовательно, время работы протокола π не меньше $n/2$ в случае, когда длины входов совпадают и равны n. □

Будем говорить, что граф G обладает симметрией между множествами $A \subset V(G)$ и $B = V \backslash A$, если существует автоморфизм графа g такой, что $g(A) = B$ и $g^2(v) = v, \forall v \in V$.

Через $E(A, B)$ будем обозначать число рёбер графа G, проходящих между этими множествами. Другими словами, $E(A, B)$ — это число таких рёбер e, что оба множества $e \cap A$ и $e \cap B$ не пусты.

Теорема 2. *Пусть дан ограниченный протокол π построения остовного дерева. Пусть граф G обладает симметрией между множествами A и $V(G)\backslash A$.*

Тогда время работы на графе оценивается как

$$T_G(\pi) = \Omega\Big(\frac{\log N_{\min}}{2E(A, V\backslash A)}\Big)$$

.

Доказательство. Наша цель, как и в предыдущем утверждении, свести задачу построения остовного дерева к вычислению асимметричной функции.

Пусть вершины $a_1, \ldots, a_{n/2}$ составляют множество A, вершины $a_{n/2+1}, \ldots, a_n$ составляют множество V A. Пусть также функция g задаёт автоморфизм из определения симметрии между множествами $A, V\backslash A$, причём вершины нумерованы так, что $g(a_i) = g(a_{i+n/2})$ для всех $1 \leqslant i \leqslant n/2$.

Будем обозначать через $y = \pi(v_1, \ldots, v_n)$ то, что в результате работы протокола π в графе G, вершины которого занумерованы числами $v_1, \ldots, v_n$ корнем построенного дерева оказывается вершина y.

Из однородности протокола и симметрии графа ясно, что

$$g(y) = \pi(v_{n/2+1}, \ldots, v_n, v_1, \ldots, v_{n/2}),$$

то есть, если переставить номера вершин в соответствии с отображением g, то корнем вершины станет вершина $g(y)$.

Определим вспомогательную функцию $h : \mathbb{N}^2 \to \mathbb{N}$ следующими правилами: $h(x, 0) = x$, $h(x, y)$ получается приписыванием битовой записи числа y к битовой записи числа x.

Построим функцию $f(x_1, x_2)$ по следующему правилу: вершины из множества A занумеруем числами $h(x_1, 0), h(x_1, 2), \ldots, h(x_1, n/2)$, вершины из множества B числами $h(x_2, 2), h(x_2, 2), \ldots, h(x_2, n/2)$. Затем запустим протокол π на полученном графе. Если корнем станет вершина из множества A, то определим $f(x_1, x_2) = 1$, иначе определим $f(x_1, x_2) = 0$.

Заметим, что от построенной нумерации нам нужно, чтобы одна из вершин оказалась с номером x_1 и одна с номером x_2, все другие вершины должны быть занумерованы уникально и иметь бóльшую длину битовой записи, у вершин из множества A не должно быть информации о x_2 в начальный момент времени, а у вершин из множества $V\backslash A$ не должно быть информации о x_1.

Из построения видно, что $f(x_2, x_1) = 1 - f(x_1, x_2)$ при $x_1 \neq x_2$. Значит коммуникационная сложность функции f на словах длины n хотя бы n.

Таким образом, суммарное число бит, которое должно быть переслано между множествами A и B должно быть не меньше n. Так как в единицу времени может передаваться не более $2E(A, V \backslash A)$ бит между ними, то получаем необходимую оценку времени. □

Теорема 3. *Для всякого ограниченного протокола π время работы в худшем случае* $T(\pi) = \Omega(\log N_{\min} + V)$.

Доказательство. В силу неравенства $\max(V, \log N_{min}) \geqslant 0.5(V + \log N_{min})$ достаточно предъявить семейство графов, на которых выполнены обе оценки.

В качестве примера подойдёт кольцо с чётным числом вершин C_{2n}. Такое кольцо имеет радиус $n = O(2n) = O(V)$. В качестве автоморфизма, задающего симметрию подойдёт поворот на π радиан, за множество A можно взять любые n подряд идущих вершин. Между A и $V \backslash A$ проходит всего два ребра. □

2 Верхняя оценка времени на кольце

В предыдущем разделе была показана оценка для сложности протоколов на кольце $T_{C_n}(\pi) = \Omega(\log N_{\min} + V)$.

Упомянутый ранее алгоритм Р. Перльман [6] даёт верхнюю оценку $T_G(\pi) = O(\operatorname{diam} G \cdot \log N_{\min})$. И в случае кольца имеется существенный зазор между оценками:

$$\Omega(V + \log N_{\min}) \leqslant T_{C_n}(\pi) \leqslant O(V \cdot \log N_{\min}).$$

В этом разделе мы предъявим протокол, который на кольце имеет сложность $O(V \log\log N_{\min} + \log N_{\min})$, что даёт существенный выигрыш, если номера вершин велики. Вопрос о возможности переноса результата на другие классы графов остаётся открытым.

Начнём с неформального сравнения протокола Перльман и улучшенной версии. Оба протокола основаны на следующей простой идее: каждый узел пытается присоединить с помощью распределённого поиска в ширину все вершины графа. В спорных ситуациях, когда узел может присоединиться к нескольким деревьям, узел выбирает дерево с наименьшим цветом. Различие между протоколами заключается в способе передачи ключа и моменте принятия решения о присоединении к дереву. Протокол

Перльман передаёт ключ полностью, решение о присоединении принимается после полного получения данных (рис 1). Улучшенная версия посылает перед ключом заголовок, который позволяет использовать предыдущую историю общения и информацию о длине ключа, решение о присоединении принимается на основании префикса передаваемых данных, то есть до полного получения ключа (рис 2).

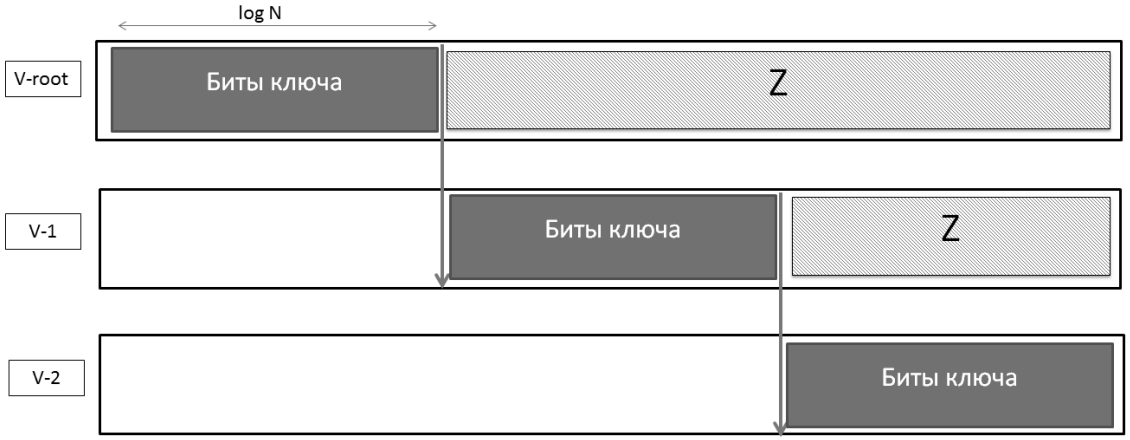

Рис. 1: Протокол Перльман.

Рис. 2: Улучшение протокола.

2.1 Описание протокола

Если говорить неформально, то протокол действует следующим образом: каждый узел пытается выполнить поиск в ширину и включить все вершины в своё дерево. При этом предложение на вхождение в дерево состоит из двух частей: длины ключа, за которыми следуют непосредственно биты ключа.

Каждый узел анализирует сообщения, которые приходят от его соседей. При известной длине ключей можно сравнить их, зная лишь достаточно длинные префиксы. Если оказывается, что один из соседей имеет ключ меньший, чем собственный ключ узла, то он соглашается на вхождение в дерево соседа. Действие такого рода мы будем называть перекраской узла. После того, как узел принял перекраску есть два принципиальных случая: новый ключ (о котором известна лишь длина и некоторый префикс) либо имеет длину равную длине предыдущего ключа, либо меньшую.

Во втором случае действия узла не отличаются от ранее описанных: узел посылает предложение о присоединении к дерево своим соседям, используя новый ключ (точнее его префикс) и новую длину, мы покажем, что отсутствие полной информации о ключе не будет проблемой. В случае же равенства длин ключей будет использоваться другой формат заголовка пересылаемого предложения на перекраску: в заголовке будет содержаться длина префикса нового ключа, которая совпадает с префиксом ключа, который передавался ранее. Этот метод позволит использовать историю общения между узлами, которая была до принятия перекраски в минимальный ключ. Более того, будет показано, что эффект будет накапливаться и в итоге суммарное время работы окажется существенно меньше, чем в алгоритме Перльман.

Заметим, что приводимый ниже протокол будет строить корректное дерево в любом графе. Но оценка времени работы будет сделана только для колец.

Перейдём к формальному описанию протокола.

Мы будем использовать конечный алфавит Σ, состоящий из сообщений, которые узлы могут послать друг другу за один такт. В Σ входят следующие элементы:

1. Z — "молчащее"сообщение, будет использоваться, если ни одно из правил выбора сообщений для отправки по узлу не сработало.

2. N — сообщение о начале передачи нового ключа.

3. *K0*, *K1* — пара сообщений, которые будут использоваться для передачи значащих битов ключа. Будем обозначать пару для краткости как $K*$.

4. *L0*, *L1* — пара сообщений, которые будут использоваться для передачи битов длины нового ключа. Будем обозначать пару для краткости как *L**. Будем также называть заголовок, состоящий из сообщений *L**, заголовком типа L.

5. *S0*, *S1* — пара сообщений, которые будут использоваться для передачи битов длины префикса нового ключа, совпадающего с префиксом ранее передаваемого ключа. Будем обозначать пару для краткости как *S**. Будем также называть заголовок, состоящий из сообщений *S**, заголовком типа S.

   Заметим, что заголовок типа S содержит лидирующие нули так, чтобы длина заголовка совпадала с длиной заголовка типа L для того же ключа. Такое соглашение сделает анализ протокола проще.

6. *E* — сообщение о завершении протокола в поддереве узла.

7. *R* — сообщение о принятии предложения вступить в дерево (принятие перекраски).

Для каждого узла v будем обозначать через $L(v)$ длину текущего ключа, через $K(v)$ — префикс текущего ключа. Через $K_v(u), L_v(u)$ будем обозначать префикс и длину ключа u, которые были получены узлом v.

Будем называть узел корневым, если его текущий ключ равен исходному ключу, т.е. если узел ни разу не принимал предложения о включении в компоненту. Будем говорить, что узел u является родителем узла v (и одновременно узел v является дочерним для узла u) к моменту времени t, если узел v принял последнее предложение о перекраске от узла u.

Опишем правила, в соответствии с которыми действует узел v в момент времени t:

1. Если от узла u было получено сообщение *K**, а на предыдущем раунде было сообщение *L**, значит узел u закончил передачу длины своего ключа. Если оказывается, что полученная от u длина $L_v(u)$ меньше $L(v)$, то v выполняет процедуру “перекраска от узла u принимается”, действия в этом состояние будут описаны далее. Если узлов, которые закончили передавать длину ключа несколько, то выбирается узел, передавший минимальную длину (а среди передавших минимальную длину - любой).

   Отдельно заметим, что в этом случае обязательно было получено сообщение *K1*, так как двоичная запись натуральных чисел всегда начинается с единицы.

2. Если от узла u было получено сообщение K^*, а на предыдущем раунде было либо сообщение S^* либо сообщение K^*, то узел v обновляет информацию о ключе узла u: если предыдущее сообщение было S^* (на этом этапе закончилась передача заголовка типа S), то обрезается предыдущий префикс. Полученный на этом этапе бит дописывается к $K_v(u)$.

   Если длина ключа u (которую v знает) равна $L(v)$, то выполняется лексикографическое сравнение $K(v)$ и $K_v(u)$. Если $K_v(u)$ оказывается меньше и не является префиксом $K(v)$, или $K(v)$ является префиксом $K_v(u)$ и родитель узла v начал передавать новый ключ, то v выполняет процедуру "перекраска от узла u принимается". Если сравнение выиграли несколько соседних узлов, то выбирается сосед с минимальным префиксом (среди минимальных выбирается любой).

3. Опишем процедуру "перекраска от узла u принимается", то

   - $K(v)$ и $L(v)$ устанавливаются равными $K_v(u)$ и $L_v(u)$.
   - Узлу u отправляется сообщение R.
   - Всем остальным соседям передаётся сообщение N.
   - Для всех узлов вычисляется заголовок, который будет необходимо отправить. Если длина была изменена, то новый заголовок также будет типа L. Если же длина не изменилась, то новый заголовок будет типа S, который будет содержать минимум из двух величин: длины совпадения префиксов предыдущего и нового ключей, числа бит предыдущего ключа, которые были отправлены.
   - Узел u считается родителем узла v.
   - Список дочерних узлов вершины v очищается.

4. Всем узлам, кроме родительского, которым был полностью отправлен заголовок, но не был отправлен полностью ключ, отправляется очередной бит ключа с помощью сообщений K^*. Если очередной бит ещё не получен от родителя, то отправляется сообщение Z.

5. Всем узлам, кроме родительского, которым не был полностью отправлен заголовок, отправляется очередной бит заголовка с помощью сообщений L^* или S^*.

6. Если от узла u было принято сообщение $R*$, то u добавляется в список дочерних узлов, $L_v(u)$ и $K_v(u)$ устанавливаются в соответствии с тем, в какой цвет была принята перекраска. Каждый раз когда дочернему узлу отправляется очередной бит ключа, переменная $K_v(u)$ обновляется соответствующим образом.

   Другими словами, в протоколе считается, что дочерний узел u прислал родительскому v ту информацию об общем ключе, что v успел отправить вершине u.

Завершение протокола контролируется следующими правилами:

1. Если от всех соседей, которые не являются ни детьми узла v, ни родительским узлом, был полностью получен их ключ и он совпадает с ключом узла v, от всех дочерних узлов было получено сообщение E, то родителю отправляется сообщение E. Если же v является корневым узлом (не имеет родителя), то E отправляется всем дочерним узлам и v переходит в завершающее состояние.

2. Если получено E от родительского узла, то оно пересылается всем дочерним узлами и v переходит в завершающее состояние.

В случае, если ни одно из правил не предписывает посылать сообщение по связи в некотором раунде, то посылается Z.

Утверждение 2. *Протокол корректен и строит остовное дерево с корнем. Причём корнем дерева окажется вершина, которая имеет в начальный момент времени минимальный ключ* N_{min}.

Доказательство. Из описания протокола ясно, что если узел вошёл в минимальное дерево, то он не выйдет из него.

Также протокол гарантирует, что пока все вершины не покрашены в один цвет, ни один узел не перейдёт в завершающее состояние.

Осталось заметить, что если узел v имеет соседа u из дерева с ключом N_{min}, то через конечное число шагов v тоже войдёт в дерево с ключом N_{min}. □

2.2 Время работы протокола

Для всякого числа A будем обозначать длину его двоичной записи как $l(A) = \lceil log_2 A \rceil$. Заметим, что в таких обозначениях длина заголовка для ключа A равна $l(l(A))$.

Все утверждения доказываются для кольцевой сети.

Лемма 1. *Рассмотрим ситуацию, когда узел v общается с не родительским узлом u и посылает информацию о ключе A. Пусть v передал в заголовке типа S число x (если заголовок был типа L, то положим $x = 0$), затем передал $y \geqslant 1$ сообщений K^*, после которых и принял предложение на перекраску (в момент времени t). Пусть узел u до принятия перекраски принадлежит дереву с ключом B, причём $l(A) = l(B) = l(N_{min})$.*

Тогда u отправит своему второму соседу w заголовок типа S, содержащий число $x + y - 1$.

Доказательство. Лемма доказывается по индукции. Рассматриваемые конфигурации сводятся к аналогичному утверждению только для меньшего числа вершин или в более ранний момент времени.

Заметим, что в условиях утверждения, число $x + y$ является первой позицией, в которой ключи A и B отличаются, иначе предложение о перекраске было бы принято раньше.

Другими словами, доказывается, что u передал узлу w не меньше данных о ключе B, чем требуется для сравнения ключей A и B.

Если u является корнем дерева с ключом B, то узлу w было передано $t - l(l(B)) - 1$ битов ключа B (либо весь ключ, если t достаточно велико), корень дерева A также передал своим соседям не более $t - l(l(A)) - 1$ битов ключа. Следовательно выполняется неравенство $x + y \leqslant t - l(l(A)) - 1$. Учитывая равенство из условия $l(A) = l(B)$, доказываем требуемое.

Если u является родителем узла w, и при этом не является корнем, то узел v до принятия перекраски в цвет A был родителем узла u. Значит, в момент принятия узлом v перекраски в цвет A, узел w знал о ключе B ровно на $l(B)$ бит меньше, чем узел u (это следует из того, что можно применять доказываемую лемму для цвета B, базой индукции является корневой узел и его сосед–ребёнок). За время, пока v передаёт заголовок, u сообщит недостающую информацию узлу w. В этом пункте мы воспользовались минимальностью длины ключей: пересылаемые заголовки должны быть типа S и все одинаковой длины $l(N_{min})$.

Если w был родителем узла u в дереве B, значит w знает о ключе B не меньше, чем u и заголовок $x + y - 1$ будет корректным. В противном случае, некорректным было бы выполненное по условию утверждения сравнение. □

Утверждение 3. *Пусть узел v принял перекраску от узла u в цвет A и начал*

передавать новую информацию о новом ключу второму соседу w. Пусть также, длина ключа A минимальна в сети. Покажем, что невозможна ситуация, когда узлу v требуется передать очередной бит ключа, но узел v ещё не получил эту информацию от родителя.

Доказательство. После принятия перекраски v отправляет заголовок размера $l(l(A))$ узлу w. Пока v отправляет заголовок, узел u присылает узлу v новые биты ключа. Таким образом создаётся и поддерживается буфер размера $l(l(A)) + 1$.

Уменьшаться этот буфер может либо если получен весь ключ, либо если u примет предложение присоединится к дереву с некоторым ключом B и начнёт передавать заголовок для перекраски в этот ключ.

Если заголовок был передан полностью, то как следует из леммы 1, узел v примет перекраску в цвет B и начнёт передавать новый заголовок. И мы свели утверждение к аналогичному, но с меньшим ключом.

Покажем, что для ключей минимальной длины невозможна ситуацию, когда заголовок не был передан полностью. Действительно, если заголовок для присоединения к дереву B не был передан полностью, значит началась передача заголовка для некоторого третьего ключа C. Но узел u не мог принять предложение о перекраске в цвет C до полного получения заголовка. Так как в протоколе передача заголовка начинаются сразу после принятия перекраски, то между моментами времени t_B и t_C, в которые узел u принял перекраску в цвета B и C соответственно должно было пройти не менее $l(l(C))$ тактов. Из условия известно, что размер заголовка для ключа C равен $l(l(C)) \geqslant l(l(B))$.

Заметим, что мы пользуемся тем, что u знает биты, которые ему требуется передавать дочернему узлу. Если не так, то мы можем повторить рассуждения относительно вершины u, базой такой индукции будет корневой узел. □

Теорема 4. *Построенный нами протокол завершается в кольце из n вершин за время $O(V \log\log N_{min} + \log N_{min})$.*

Доказательство. Рассмотрим распространение информации из ключа v_0 с минимальным ключом N_{min}.

Все вершины графа будут присоединены к дереву с корнем в v_0.

Рассмотрим три соседних узла u, v и w таких, что u является родителем в итоговом дереве N_{min} для вершины v, а v является родителем для w. Разницу в моментах времени t_u и t_v, в которые узлы u и v были включены в дерево обозначим через Δ_{uv}.

В силу утверждения 3 выполнено равенство $\Delta_{uv} = l(l(N_{min})) + b$, где b — число сообщений K^*, которые были отправлены узлом u после заголовка и до момента t_v. Пусть в этих сообщениях K^* передавалась информация о битах с номерами из отрезка $[a, a']$. Аналогично определим отрезок $[b, b']$ для пары v и w.

Из леммы 1 следует, что $[a, a']$ и $[b, b']$ не пересекаются.

Рассмотрим произвольный (в кольце их всего два) путь по рёбрам итогового дерева из вершины v_0 до листовой вершины v_c с промежуточными вершинами $v_1, v_2, \ldots, v_{c-1}$. Время t_{v_c}, в которое вершина v_c вошла в дерево N_{min} можно оценить следующим образом:

$$t_{v_c} = \sum_{i=0}^{c-1} \Delta_{i,i+1} = c \cdot l(l(N_{min})) + \sum_{i=0}^{c-1} b_i$$

Как было показано выше, $\sum\limits_{i=0}^{c-1} b_i$ равен сумме длин непересекающихся отрезков, которые вложены в отрезок $[0, l(N_{min})]$. Значит, верно неравенство

$$t_{v_c} \leqslant V \cdot \log\log N_{min} + \log N_{min}.$$

Через дополнительные $l(N_{min}) + O(V)$ тактов времени все вершины получат ключ полностью и распространится информация о том, что все вершины графа включены в дерево, после чего все узлы перейдут в завершающее состояние. □

Список литературы

[1] Распределенная коммуникационная сложность построения остовного дерева М. Н. Вялый, И. М. Хузиев Пробл. передачи информ., 51:1 (2015), 54–71

[2] Impagliazzo R., Williams R. Communication complexity with synchronized clocks // Proc. of the 2010 IEEE 25th Annual Conference on Computational Complexity, CCC'10, Washington, DC, USA, 2010. P. 259–269.

[3] Frederickson G. N., Lynch N. A. Electing a leader in a synchronous ring // Journal of the ACM, 1987. Vol. 34. P. 98–115.

[4] Burns J.E. A formal model for message passing systems. Tech. Rep. TR–91. Indiana Univ. 1980.

[5] Dinitz Y., Moran Sh., Rajsbaum S. Bit complexity of breaking and achieving symmetry in chains and rings // Journal of the ACM, 2008. Vol. 55, no 1. P. 1–28.

[6] Perlman R. An algorithm for distributed computation of a spanning tree in an extended LAN // ACM SIGCOMM Computer Communication Review 15 (4), 1985. P. 44–53.

[7] Peleg D., Rubinovich V. A near-tight lower bound on the time complexity of distributed MST construction // SIAM J. on Computing, 2000. Vol. 30. P. 1427–1442.

[8] Itai A., Rodeh M. Symmetry breaking in distributed networks. // Information and Computation, 1990. Vol. 88, no 1. P. 60–87.

[9] В.В. Подольский, А.Е. Ромащенко. Введение в коммуникационная сложность. Конспект лекций. // Мехмат МГУ, весенний семестр, 2012.