1 Глава 10: Модульный анализ: Вычисления с помощью кривых

Содержание

В предыдущей части были представлены концепции для вычисления границ производительности одного сервера. Цель данной части - использовать их для изучения более сложных коммуникационных сетей. Общая проблема вычисления границ производительности в сетях привлекает большое внимание, и было использовано множество методов, в зависимости от гипотезы о процессах прихода и ухода.

Данная часть состоит из трех глав. Первые две главы посвящены общим методам вычисления границ производительности в сетях с прямой передачей (т.е. при отсутствии циклической зависимости между потоками, циркулирующими в сети). Сети с прямой передачей Сети являются общим классом сетей, для которых выполняется условие устойчивости (т.е. условие, при котором объем данных в сети остается ограниченным). В результате единственной проблемой, которая здесь рассматривается, является проблема вычисления границ производительности, максимально приближенные к точному значению наихудшего случая. Третья глава будет посвящена циклическим сетям, где не известно ни одного общего результата для устойчивости больших классов сетей.

Применительно к ациклическим сетям существует два основных подхода: 1) первый использует кривые прибытия и обслуживания для вычисления наихудших границ задержки, ограничений на задержку. Как мы увидим, жесткость границ производительности, которую мы имели в случае с одним сервером, теряется, но разработанные методы очень эффективны с алгоритмической точки зрения. Ключевым инструментом является алгебраический перевод сериализации серверов в виде свертки (min,plus). Этому и посвящена данная глава; 2) второй непосредственно использует траектории потоков, удовлетворяющих кривым прибытия и обслуживания. Результаты получаются гораздо более точными. Это имеет два следствия: в некоторых случаях можно получить точные границы производительности в наихудшем случае. Это и будет являться целью Главы 11.

В этой главе мы сосредоточимся на границах производительности в сетях, которые могут быть вычислены с помощью (min, plus), определенной в предыдущих главах. Мы сосредоточимся на вычислении наихудших задержек для данного потока или наихудшего отставания на сервере, а также на ациклических сетях. Можно использовать несколько методов, но они опираются на одни и те же элементы (для вычисления задержки): 1) отсортировать серверы в топологическом порядке так, чтобы они анализировались в этом порядке; 2) для каждого сервера вычислить кривые остаточного обслуживания для каждого потока и вычислить кривые промежуточного прибытия для каждого потока. 3) вычислить глобальную кривую обслуживания и верхнюю границу сквозной задержки для каждого потока.

Шаг 1 может быть легко выполнен за линейное время, зависящее от размера сети, с помощью алгоритма например, алгоритмом глубинного поиска. Шаг 2 выполняется с использованием результаты глав 5 и 7, и вычисления зависят от политики обслуживания. Более того, если известна максимальная кривая обслуживания, то этот шаг может быть улучшен с помощью методов группировки. Шаг 3 выполняется с использованием алгебраиче-

ских свойств свертки. В данной главе подробно рассматривается шаг 3 и усовершенствования для шага 2. Глава организована следующим образом: сначала мы представляем модель сети и обозначения, которые мы будем использовать в этой части книги. Раздел 10.3 посвящен специальному случаю вложенных тандемов, для которого можно получить лучшие ограничения, используя явление мультиплексирования с оплатой только один раз В разделе 10.4 представлены несколько общих алгоритмов вычисления границ производительности в сетях с прямой передачей, начиная с наивных и заканчивая более совершенными. Наконец, в разделе 10.5 мы докажем NP-трудность вычисления точных границ задержки в наихудшем случае с учетом характеристик сети.

2 10.1. Модель сети

2.1 10.1.1. Описание сети и условные обозначения

В нашей модели сеть N, состоящая из n серверов и m потоков, описывается несколькими элементами.

Потоки. Для всех i в 1,...,m, поток i описывается кривой прибытия i и путем pi =<pi(1) ,. . . , pi(l)> , где pi(l)$\in 1, ..., n$.

Мы пишем hили i(h), если существует l$\in 1, ..., li, pi(l) = h.pi, .h = pi(l), predi(h) = pi(l-1), , l = 1, predi(h) = 0$.

Заметим также, что fst(i) = pi(1) - первый сервер, который посетил i, а end(i) - последняя система, которую посетил i, т.е. pi(li). Биты данных потока i последовательно пересекают каждый сервер pi в порядке, указанном в пути.

Кумулятивный процесс прибытия потока i до посещаемого им сервера обозначается Fi(0) и является i-ограниченным. Для h pi обозначим через Fi(h) совокупный процесс ухода потока i после сервера h.

Серверы. Для всех h в 1,...,n сервер S(h) - или просто сервер h - описывается кривой обслуживания (h) $\in C, T(h)(-)(, , FIFO, ,).h \in 1, ..., nS(h) = ST(h)((h)).,, h \in 1, ..., n,$

$((F_i^{pred_i(h)})_{i \in Fl(h)}$, $(F_i^h) \in S^{(h)}$

Для сервера h и t 0 обозначим через Starth(t) начало периода отставания сервера h в момент времени t. Графом, индуцированным этой сетью, является направленный граф GN = (V,E), где V = {1, ..., n) - множество серверов, а E = $\{(,l) \in V \times V | \exists i \in 1, ..., m, pi =< ..., h, l, ... > -.GN()$.

3 10.2. Некоторые специальные топологии

В этой части мы остановимся на некоторых топологиях, показанных на рис. 10.1.

Сети с прямой передачей. Сеть с прямой связью - это сеть N, индуцированный граф GN, который является ациклическим. В этом случае можно

пронумеровать серверы так, чтобы путь каждого потока был возрастающим:

$\forall i \in 1, \ldots, m, \forall l < l_i, p_i(l) < p_i(l+1)$.

Такую нумерацию можно получить, выполнив топологическую сортировку. Мы всегда будем использовать этот тип нумерации при работе с ациклическими сетями (в главах 10 и 11). Тандемные сети. Тандемная сеть является частным случаем сети с прямой связью с линейной топологией: ребрами индуцированной сети являются

$(h, h+1), h \in P\{1, ..., n\breve{}1)$. В результате путь каждого потока имеет вид <fst(i), fst(i)+1,..., end(i)>.

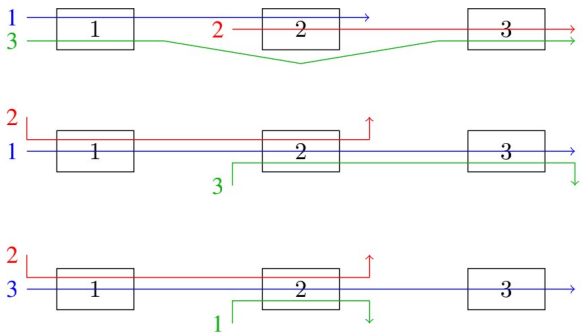

Рисунок 10.1. Примеры топологий сетей. Верхняя часть: сеть с прямой передачей, но не тандемная; средняя часть: тандемная сеть, но не вложенная; внизу - вложенная тандемная сеть. Вложенные тандемные сети. Вложенная тандемная сеть - это тандемная сеть такая, что путь каждого потока включен в путь другого. Тогда потоки могут быть пронумерованы так, что

(Nest)

$$\forall i, j \in \{1, \ldots, m\}, i < j \iff p_i \subseteq p_j$$

4 10.3. Явление мультиплексирования с оплатой только один раз

Рассмотрим сеть на рис. 10.2, в которой два потока пересекают два сервера.

Рисунок 10.2. Сеть с двумя потоками и двумя серверами для иллюстрации явления мультиплексирования с оплатой только один раз.

Предположим, что на первом сервере пакет данных потока 1 обслуживается раньше, чем данные потока 2, хотя данные потока 2 поступили на

сервер раньше этого всплеска. Тогда на втором сервере этот всплеск данных потока 1 не может снова обогнать данные потока 2, так как она уже поступила ранее на второй сервер. Однако, применяя теоремы 6.1 и 7.1, мы получаем остаточную кривую мин-плюс обслуживания.

$$\beta_1 = [\beta^{(1)} - \alpha_1]^+_\uparrow * [\beta^{(2)} - \alpha_1^{(1)}]^+_\uparrow$$

Где $\alpha^{(1)} = \alpha_1 \emptyset [\beta^{(1)} - \alpha_2]^+_\uparrow$ это кривая поступления потока 1 на сервер 2. В этой формуле, имеет место мультиплексирование двух потоков в двух серверах, то есть бит данных может пострадать от одного и того же всплеска потока 1 в каждом из серверов, что является слишком пессимистичным. Однако в некоторых случаях прямое вычисление дает лучшие результаты.

Действительно, предположим, что кривые обслуживания являются жесткими, а политика обслуживания - слепой. Пусть $t \in R+$. Мы можем определить s = Start2(t) и u = Start1(s), соответствующие началу периода отставания для t на сервере 2 и для s на сервере 1. Мы можем записать:

$$F_1^{(2)} + F_2^{(2)}(t) \geq F_1^{(1)}(s) + F_2^{(1)}(s) + \beta^2(t-s)$$

и

$$F_1^{(1)}(s) + F_2^{(1)}(s) \geq F_1^{(0)}(u) + F_2^{(0)}(u) + \beta^1(s-u),$$

Индуцирование

$$F_1^{(2)} + F_2^{(2)}(t) \geq F_1^{(0)}(u) + F_2^{(0)}(u) + \beta^1(s-u) + \beta^2(t-s)$$

как

$$F_1^{(2)}(t)b \leq F_1^{(0)}(t)$$

и $F_2^{(2)}(t) \geq F_2^{(1)}(s) \geq F_1^{(0)}(u)$, мы получаем $F_2^{(2)}(t) \geq F_1^{(0)}(u) + (\beta^{(1)} * \beta^{(2)} - \alpha_1)^+(t-u)$ [10.1]

и $\widetilde{\beta_2} = (\beta^{(1)} * \beta^{(2)} - \alpha_1)^+$ является кривой обслуживания для потока 2. Мы называем это явление мультиплексирование с оплатой только один раз.

Это понятие было введено Шмиттом и др., и можно отметить, что этот результат не может быть получен путем применения Теоремы 6.1, а затем Теоремы 7.1, так как кривая обслуживания, полученная по Теореме 6.1, не является строгой. Однако данный пример показывает, что прямое вычисление дает тот же результат.

4.1 10.3.1. Потеря герметичности

Явление PMOO может быть обобщено на произвольную сеть, но это не является решением, позволяющим всегда вычислять лучшие границы производительности, хотя интуитивно так и должно быть, поскольку всплеск

потока 1 учитывается только один раз. В данном разделе мы проиллюстрируем сложность получения хороших границ.

Мы по-прежнему рассматриваем небольшую сеть рис. 10.2 со следующими параметрами:

- для $h \in 1, 2$, сервер h предлагает строгую кривую обслуживания $\beta^{(h)} = \beta_{(R_h,T_h)}$, а для каждого сервера обслуживание является произвольным; - для $i \in 1, 2$, поток i ограничен кривой поступления $\alpha_i = \gamma_{(r_i,b_i)}$ и сравнить две полученные кривые обслуживания, $\widetilde{\beta_1}$ и $\widetilde{\beta_2}$

$$\widetilde{\beta_1}(t) = (min(R_1, R_2) - r_1)[t - \tfrac{b_1+R_1T_1}{R_1-r_1} - \tfrac{b_1+r_1T_1+R_2T_2}{R_2-r_1}]^+$$

и

$$\widetilde{\beta_2}(t) = (min(R_1, R_2) - r_1)[t - \tfrac{b_1+min(R_1,R_2)(T_1+T_2)}{min(R_1,R_2)-r_1}]^+$$

Если $\beta 1 = \beta 2$, тогда $\widetilde{\beta_2} \leq \widetilde{\beta_1}$. Но если b1 = 0, T1 = 0 и R2 > R1, тогда $\widetilde{\beta_1} \leq \widetilde{\beta_2}$.

Первый метод, используемый для вычисления 1, может быть обобщен на общие сети с прямой связью. Этому посвящен раздел 10.4. Второй метод, приводящий к 2, не может быть обобщен столь простым способом, когда вмешивается большее число потоков, и особенно когда потоки не вложены друг в друга. В следующем разделе показано, что формула может быть обобщена для тандемных сетей при произвольном мультиплексировании. Этот результат может быть трудно применить к конкретным политикам обслуживания. Одна из причин, показанная в в разделе 10.3.3, заключается в том, что политики обслуживания неустойчивы при композиции серверов. Однако в разделе 10.3.4 мы покажем, как можно воспользоваться феноменом PMOO в вложенном тандеме для статических приоритетов и политик FIFO. Другим способом обобщения данного подхода является использование линейного программирования. Действительно, здесь используется примерно такая же вычислительная схема: сначала работаем с траекториями и связываем их, используя кривые прибытия и обслуживания только на последнем шаге. Это будет рассмотрено в главе 11.

4.2 10.3.2. Многомерный оператор для PMOO

В этом разделе мы обобщим принцип PMOO на любую тандемную сеть.

ТЕОРЕМА 10.1 (многомерный оператор PMOO). Рассмотрим тандемную сеть N, в которой поток 1 пересекает все серверы. Если каждый сервер h предлагает строгий сервис кривая (h) При тех же допущениях и обозначениях, что и выше, кривая минимального обслуживания, предлагаемая для потока 1 имеет вид:

$$\beta(t) = \left[\inf_{\substack{u_1,\ldots,u_n \geqslant 0 \\ u_1+\cdots+u_n = t}} \sum_{j=1}^{n} \beta^{(j)}(u_j) - \sum_{i=2}^{k} \alpha_i(\sum_{j=\mathrm{fst}(i)}^{\mathrm{end}(i)} u_j) \right]^+ .$$

Сначала докажем лемму, касающуюся кумулятивных процессов. ЛЕММА 10.1.- С учетом предыдущих предположений и обозначений, $\forall t \in R+, \exists u1, ..., un \in R+$ такие, что

$$F_1^{(n)}(t) - F_1^{(0)}(t - \sum_{j=1}^{n} u_j) \geqslant \sum_{j=1}^{n} \beta_j(u_j)$$
$$- \sum_{i=2}^{k} \left(F_i^{(\mathrm{end}(i))}(t - \sum_{j=\mathrm{end}(i)+1}^{n} u_j) - F_i^{(0)}(t - \sum_{j=\mathrm{fst}(i)}^{n} u_j)\right)$$

Более того, $F_1^{(n)}(t) - F_1^{(0)}(t - \sum_{j=1}^{n}(u_j) \geq 0$

ДОКАЗАТЕЛЬСТВО.-Для упрощения обозначений будем отождествлять $F_i^{(0)}$ с $F^{(fst_i - 1)}$

Зададим $s_{n+1} = t$ и для всех h $\in 1, ..., ns_h = Start_h(s_h+1)$ и $u_h = s_{h+1} - s_h$

Поскольку серверы предлагают жесткие кривые обслуживания, имеем, что для всех серверов h,

$$\sum_{i \in \mathrm{Fl}(h)} F_i^{(h)}(s_{h+1}) \geqslant \beta^{(h)}(s_{h+1} - s_h) + \sum_{i \in \mathrm{Fl}(h)} F_i^{(h-1)}(s_h).$$

[10.2]

Суммируя все эти неравенства и собирая члены по индексам расхода, получаем

$$\sum_{i=1}^{m} \sum_{h=\mathrm{fst}(i)}^{\mathrm{end}(i)} F_i^{(h)}(s_{h+1}) \geqslant \sum_{h=1}^{n} \beta^{(h)}(u_h) + \sum_{i=1}^{m} \sum_{h=\mathrm{fst}(i)}^{\mathrm{end}(i)} F_i^{(h-1)}(s_h).$$

Устранив операции, присутствующие с обеих сторон, он упрощается до:

$$\sum_{i=1}^{m} F_i^{(\mathrm{end}(i))}(s_{\mathrm{end}(i)+1}) \geqslant \sum_{h=1}^{n} \beta^{(h)}(u_h) + \sum_{i=1}^{m} F_i^{(\mathrm{fst}(i)-1)}(s_{\mathrm{fst}(i)}).$$

что дает первое неравенство леммы $s_h = t - \sum_{j=h}^{n}(u_j$

Более того, для всех потоков имеем $i, F_i^{(h)}(s_h+1) \geq F_i^{(h)}(s_h) = F_i^{(h-1)}(s_h$, так что $F_i^{(end(i))}(s_{end(i)+1}) \geq F_i^{fst(i)-1)}(s_{fst(i)}$, что дает второе неравенство.

Из этой леммы можно вывести кривую остаточного обслуживания, предоставляемого потоку 1 на всем его пути.

Доказательство теоремы 10.1.- Рассмотрим неравенства, доказанные в лемме 10.1. Очевидно, что

$\forall \in 1..., m, \forall h \in fst(i), ...end(i), \forall t \in_+ , F^{(h)_i}(t) \leq F_i^{(0)}(t)$

Более того, для всех 2 <= i <= m,

$$F_i^{(end(i)))}(s_{end(i)-1)} - F_i^{(0)}(s_{fst(i)}) \leq F_i^{(0)}(s_{end(i)-1)} - F_i^{(0)}(s_{fst(i)}) \leq \alpha_i(\sum_{j=fst(i)}^{end(i)} u_j$$

Таким образом,

$$F_1^{(n)}(t) - F_1^{(0)}(t - \sum_{j=1}^{n} u_j \geq \sum_{h=1}^{n} \beta^{(h)}(u_h) - \sum_{i=1}^{k} \alpha_i(\sum_{j=fst(i)}^{end(i)} u_j)$$

Из леммы 10.1 также следует, что
$F_1^{(n)}(t) - F_{(0)1}(t - \sum_{j=1}^{n} u_j) \geq 0$

В результате этих двух неравенств имеем $F_1^{(n)} \geq F_1^{(0)} * \beta$, где - функция, определенная выше.

ПРИМЕР 10.1.-Для иллюстрации явления PMOO рассмотрим сеть на Рис. 10.1 (середина), которая представляет собой пример, подробно описанный Шмиттом и др. Теорема 10.1 позволяет получить следующую кривую обслуживания для этого пути:

$$\beta(t) = \Big[\inf_{\substack{u_1, u_2, u_3 \geqslant 0 \\ u_1 + u_2 + u_3 = t}} \beta^{(1)}(u_1) + \beta^{(2)}(u_2) + \beta^{(3)}(u_3) - \alpha_1(u_1 + u_2) - \alpha_2(u_2 + u_3) \Big]^+.$$

Этот результат вводит многомерный оператор для сетевого исчисления. Он обобщает Теорему 7.1, в которой рассматриваются пути с единственным узлом и уникальным перекрестным транспортным потоком. Если кривые обслуживания являются скоростно-латентными ($\beta^{(h)} = \beta_{R_h, T_h}$ а кривые прибытия являются кривыми утечки-ведра. Кривые прибытия - это кривые "ведро утечки"($\alpha_i = \gamma_{r_i, b_i}$) , то формула приобретает вид:

$$\beta(t) = \Big[\inf_{\substack{u_1, \ldots, u_n \geqslant 0 \\ u_1 + \cdots + u_n = t}} \sum_{h=1}^{n} \beta^{(h)}(u_h) - \sum_{i=1}^{m} \alpha_i(\sum_{j=\mathrm{fst}(i)}^{\mathrm{end}(i)} u_j) \Big]^+$$

$$= \Big[- \sum_{i=1}^{n} b_i + \inf_{\substack{u_1, \ldots, u_n \geqslant 0 \\ u_1 + \cdots + u_n = t}} \sum_{h=1}^{n} \big(R_h[u_h - T_h]^+ - (\sum_{i \in \mathrm{Fl}(h)} r_i) u_h \big) \Big]^+$$

$$= \Big[- \sum_{i=1}^{n} b_i + (\gamma_1 * \cdots * \gamma_n)(t) \Big]^+,$$

С $\gamma_h(t) = R_h[t - T_h]_+ - (\sum_{i(h) r_i) t}$, которая является выпуклой функцией, состоящей из двух отрезков: отрезка с наклоном $-\sum_{i \in Fl(h) r_i}$ между 0 и Tj и бесконечного отрезка участок с наклоном $R_h - \sum_{i \in Fl(h)} r_i$. Поэтому $(\gamma_1 * \ldots * \gamma_n)$ может быть легко вычислено, используя Теорему 4.1. Конечные отрезки имеют отрицательные наклоны, и нас интере-

сует только та часть, где свертка больше $\sum_{i=1}^{m} b_i$, которая имеет наклон $R = min_{h=1}^{n}(R_h - \sum_{iinFl(h)r_i}$. После вычислений получаем, что $\beta = \beta_{R,T}$ с $R = min_{h=1}^{n}(R_h - \sum_{i(h)} r_i$ и $T = \sum_{h=1}^{n} T_h(1 + \frac{\sum_{i \in Fl(h)} r_i}{R}) + \sum_{i=1}^{m} \frac{b_i}{R}$.

4.3 10.3.3. Состав и сервисные политики

ВНИМАНИЕ! Феномен PMOO следует использовать с осторожностью. Действительно, предыдущая теорема не означает, что система, состоящая из двух серверов с одинаковой политикой обслуживания, подчиняется этой сервисной политике. Это верно в нескольких частных случаях: слепое мультиплексирование (это очевидно, так как нет предположения о политике обслуживания) и FIFO мультиплексирование. Действительно, если бит данных a поступает раньше, чем бит данных b, то a уйдет с первого сервера раньше, чем b, и то же самое произойдет с сервером 2.

Однако для других сервисных политик это не так. Проиллюстрируем этот факт на примере двух политик обслуживания: статического приоритета (SP) и обобщенного разделения процессоров (GPS). В следующем примере мы рассмотрим простую сеть, изображенную на рис. 10.2.

- Статический приоритет: это следует из того факта, что кривая сквозного обслуживания двух серверов в тандеме не является строгой кривой обслуживания. Рассмотрим рисунок 6.5 и предположим, что нарисованный поток является потоком с наивысшим приоритетом. Теперь рассмотрим второй поток (с меньшим приоритетом). Поскольку кривые обслуживания являются чистыми задержками, то если данные от второго потока поступают в момент времени 0, то эти данные выйдут в моменты времени T1+T2. Но в это время поток 1 все еще находится в застое, что означает, что система, состоящая из двух серверов глобально не является статическим сервером приоритетов.

- Обобщенное разделение процессоров: возьмем $\beta^{(1)} = \lambda_4$, $\beta^{(2)} = \lambda_5$, $\alpha_1 = \alpha_2 = \gamma_{1,10}$ и 1 = 2 - доли обслуживания, гарантированные для потоков 1 и 2 на каждом сервере. Предположим, что первый поток поступает в соответствии с 1 и что половина потока обслуживается сразу, в момент времени 0+. Поток 2 поступает с момента времени 1, в соответствии с $\alpha_2(t-1)$. Между временами 0 и 1 обслуживается только поток 1. Начиная с момента времени 0+, серверы обслуживают данные в точном соответствии с кривой их поступления. Начиная с момента времени 1, сервер 1 отстает для обоих потоков. Затем каждый поток поступает со скоростью 2 на сервер 2.

На сервере 2 поток 2 не может отставать (ему гарантирована скорость обслуживания 2,5), поэтому он обслуживается со скоростью 2. Тогда поток 1, который отстает, обслуживается со скоростью 5 - 2 = 3. В результате потоку 2 не гарантируется половина обслуживания глобальной системы. Можно поменять местами роли потоков 1 и 2, и тогда можно сделать вывод, что ни одному из этих двух потоков не может быть гарантирована половина обслуживания глобальной системы. Следовательно, глобальная система не

является GPS.

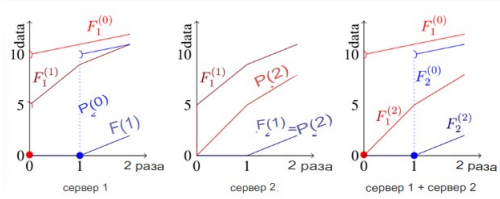

Рисунок 10.3. Два GPS-сервера в тандеме - это не GPS-сервер.

4.4 10.3.4. Вложенные тандемы

Вложенные тандемные системы представляют собой частный случай, когда PMOO может быть обобщен для нескольких политик обслуживания. Мы уже приводили общую формулу для слепого мультиплексирования (Теорема 10.1), а в данном разделе мы покажем, что это явление может быть учтено для систем с фиксированными приоритетами и для политик обслуживания с фиксированными приоритетами и FIFO. Интуитивно понятно, что это возможно когда потоки могут быть удалены по одному, начиная с самого вложенного и заканчивая самым длинным. Рассматриваются два случая: вложенные потоки имеют более высокий приоритет (уравнение [10.1] может быть применено непосредственно) или политика обслуживания FIFO, поскольку политика обслуживания сохраняется за счет состава серверов.

4.4.1 10.3.4.1. Статические приоритеты и PMOO

Предположим, что вложенный тандем с нумерацией (Nest) пересекают m потоков и работает статический приоритет, согласно которому потоки с наибольшим номером имеют наименьший приоритет. Другими словами, если pi $\subseteq pj, i, j.ii3.[10.1], 10.1, .$

Алгоритм 3: PMOO для статической политики приоритетного обслуживания. Данные: $(\alpha_i)_m^{i=1} \in F^m$, $(\beta^{(h)})_{h=1}^n \in F^n$ $(p_i)_{i=1}^m$ возрастающие и вложенные пути.

Результат: $\widetilde{\beta}_i$, кривая обслуживания для потока i

begin
  for $i = 1$ **to** n **do**
    $PF(i) \leftarrow \{j < i \mid \mathbf{p}_j \subseteq \mathbf{p}_i \text{ and } \nexists k,\ i > k > j,\ \mathbf{p}_j \subseteq \mathbf{p}_k\}$;
    $N(i) \leftarrow \mathbf{p}_i \backslash \left(\bigcup_{j<i} \mathbf{p}_j\right)$;
    $\tilde{\beta}_i \leftarrow \circledast_{h \in N(i)} \beta^{(h)} * \circledast_{j \in PF(i)} \left[\tilde{\beta}_j - \alpha_j\right]_{\uparrow}^{+}$.

ПРИМЕР 10.2.- Рассмотрим вложенную сеть в нижней части рис. 10.1. Мы Вычислим

$$\begin{cases} \tilde{\beta}_1 = \beta^{(2)}, \\ \tilde{\beta}_2 = \beta^{(1)} * \left[\tilde{\beta}_1 - \alpha_1\right]^+_\uparrow = \beta^{(1)} * \left[\beta^{(2)} - \alpha_1\right]^+_\uparrow \\ \tilde{\beta}_3 = \left[\tilde{\beta}_2 - \alpha_2\right]^+_\uparrow * \beta^{(3)} = \left[\beta^{(1)} * \left[\beta^{(2)} - \alpha_1\right]^+_\uparrow - \alpha_2\right]^+_\uparrow * \beta^{(3)}. \end{cases}$$

4.4.2 10.3.4.2. FIFO и PMOO

То же самое можно сделать и для мультиплексирования FIFO, используя кривые остаточного обслуживания благодаря следующим трем фактам: 1) рассматриваются только кривые обслуживания "мин-плюс"; 2) объединение FIFO-серверов в тандем приводит к FIFO-системе; 3) если рассматривать только подмножество потоков, то политика обслуживания только этих потоков также является FIFO.

Следующее предложение является прямым следствием теорем 7.5 и 6.1 и рекурсивно используется в алгоритме 4. Принцип работы этого алгоритма такой же, как и выше: мы удаляем потоки по одному, начиная с самого вложенного. Для каждого потока вычисляется новая кривая остаточного обслуживания для остальных потоков. При этом мы используем следующее предложение.

ПРЕДЛОЖЕНИЕ 10.1.- Рассмотрим последовательность из n FIFO-серверов, предлагающих соответствующие кривые обслуживания (h), пересекаемые m потоками. Предположим, что поток 1 ограничен кривой прибытия . Тогда для всех > 0,

$[*_{h=1}^{n} \beta^{(h)} - \alpha * \delta_\theta] \wedge_\theta$

является кривой минимального обслуживания для m - 1 других потоков.

Алгоритм 4: PMOO для политики обслуживания FIFO во вложенных тандемах.

Данные: $(\alpha_i)_m^{i=1} \in F^m$, $(\beta^{(h)})_{h=1}^n \in F^n$ $(p_i)_{i=1}^m$ возрастающие и вложенные пути, $(\theta_i)_{i=1}^m$, параметры кривых остаточного обслуживания FIFO. Результат:$\widetilde{\beta_n}$ - кривая остаточного обслуживания для потока, пересекающего тандемную сеть с потоками 1,..., m в качестве перекрестного трафика.

begin
  for $i = 1$ **to** n **do**
    $PF(i) \leftarrow \{j < i \mid \mathbf{p}_j \subseteq \mathbf{p}_i \text{ and } \nexists k,\ i > k > j,\ \mathbf{p}_j \subseteq \mathbf{p}_k\}$;
    $N(i) \leftarrow \mathbf{p}_i \backslash \left(\bigcap_{j<i} \mathbf{p}_j\right)$;
    $\tilde{\beta}_i \leftarrow \left[\left(*_{h \in N(i)} \beta^{(h)} * *_{j \in PF(i)} \tilde{\beta}_j\right) - \alpha_i * \delta_{\theta_i}\right]^+ \wedge \delta_{\theta_i}$.

Существенным недостатком этого алгоритма является то, что параметры i должны быть зафиксированы заранее. Если получены аналитические формулы, то можно также выполнить шаг оптимизации, вычислив значения (i), которые, например, дадут оптимизационный шаг, вычисляя значения (i), которые, например, дадут наименьшую границу задержки. В общем случае это очень сложная задача, которая подробно изучалась Лензини, Стеа и другие. В работе Лензии доказано, что в случае тандемных топологий типа sink-tree (все потоки останавливаются на последнем сервере) оптимальные параметры могут быть вычислены рекурсивно и эти параметры приводят к жесткой кривой остаточного обслуживания, в том смысле, что из кривой остаточного обслуживания можно вычислить жесткую границу задержки кривой обслуживания. В работе [LEN 08] этот подход распространен на любую тандемную топологию путем разрезания потоков с целью получения вложенного тандема. Наконец, в работе [BIS 08] на простом примере показано, что герметичность достигается только для топологий типа sink-tree.

Рисунок 10.4. Тандемная сеть с синк-деревом.

ПРИМЕР 10.3.- В качестве примера оптимизации параметров 1, вдохновленного работой [LEN 06, Теорема 4.6], рассмотрим очень простой пример на рис. 10.4. и вычислим кривую остаточного обслуживания, которая будет получена для потока, пересекающего два сервера. Мы предполагаем, что кривая прибытия для потока i ограничена ri,bi и что сервер h предлагает кривую обслуживания с минимальным плюсом Rh,Th . Сначала мы вычисляем

$$\widetilde{\beta_1} = [\beta^{(2)} - \alpha_1 * \delta_{\theta_1}]^+ \wedge \delta_{\theta_1}$$

Из примера 7.1 следует, что мы можем рассматривать только те значения 1, которые больше C1 = T2 + b1/R2 . Полученная кривая остатков имеет вид

$$\widetilde{\beta_1} = \delta_{\theta_1} * \gamma_{R_2 - r_1, R_2(\theta_1 - C_1)}.$$

В качестве второго шага алгоритма,

$\widetilde{\beta_2} = [(\beta^{(1)} * \widetilde{\beta_1}) - \alpha_2 * \delta_{\theta_2}]^+ \wedge \delta_{\theta_2}$. Первый, $\beta^{(1)} * \widetilde{\beta_1} = \delta_{\theta_1 + T_1} * (\gamma_{R_2 - r_1, R_2(\theta_1 - C_1)} \wedge \gamma_{R_1, 0})$.

Здесь мы снова можем отбросить значения 2, которые не удовлетворяют $\alpha_2 * \delta_{\theta_2}(\theta_2 +) \leq \beta^{(1)} * \widetilde{{}_{\beta_1}(\theta_2)}$, так как они не будут оптимальными. Тогда 2 имеет нижнее ограничение

$C_2 = \theta_1 + T_1 + max(\frac{b_2}{R_1}, \frac{b_2}{R_2 - r_1} - \frac{(\theta_1 - C_1)R_2}{R_2 - r_1}$.
Тогда $\widetilde{\beta_2}$ имеет вид $\delta_{\theta_2} * (\gamma_{r_1', b_2'} \wedge \gamma_{r_2', b_2'})$ с

$r_1' = R_1 - r_2 \qquad b_1' = R_1(\theta_2 - \theta_1 - T_1) - b_2$
$r_2' = R_2 - r_1 - r_2 \qquad b_2' = (R_2 - r_1)(\theta_2 - \theta_1 - T_1) + (\theta_1 - C_1)R_2 - b_2.$

Теперь предположим, что через эту сеть проходит поток, ограниченный кривой прибытия r3,b3 тандемную сеть. Тогда кривая остаточного обслуживания для потока 3 имеет вид $\widetilde{\beta_2}$. В условиях устойчивости $R_1 > r_2 + r_3 R_2 > r_1 + r_2 + r_3)$, наихудшая задержка определяется как $\widetilde{\beta_2^{-1}(b_3)}$, т.е.

$d = \theta_2 + max(\frac{[b_3 - b_1']^+}{r_1'}, \frac{[b_3 - b_2']^+}{r_2'})$.

Эта задержка зависит как от 1, так и от 2, но мы можем сначала зафиксировать 1, вычислить оптимальное значение 2 для этого фиксированного 1 и, наконец, оптимизировать 1. После вычислений находим, что оптимальное значение 2 равно $\theta_2 = \theta_1 + T_1 + max(\frac{b_2 + b_3}{R_1}, \frac{b_2 + b_3 - (\theta_1 - C_1)R_2}{R_2 - r_1}$. В этом случае d = 2 = 2. Таким образом, легко видеть, что для минимизации максимальной задержки необходимо минимизировать 1, а 1 = C1 = T2 + b1/R2 . Наконец, находим:

$d = T_1 + T_2 + \frac{b_1}{R_2} + max(\frac{b_2 + b_3}{R_1}, \frac{b_2 + b_3}{R_2 - r_1})$.

5 10.4. Потоковый анализ сетей

задержки от самого наивного метода до более эффективных.

5.1 10.4.1. Анализ суммарного выхода

Если использовать только результаты главы 5, то граница задержки может быть вычислена с помощью Алгоритма 5. При этом требуется, чтобы кривые обслуживания были строгими.

Алгоритм 5: Общий анализ выхода (TOA)

Данные: Ациклическое описание сети: pi, i, (h) строгие кривые обслуживания. Результат: Верхняя граница наихудшей задержки для каждого сервера

```
1 begin
2   for h = 1 to n do
3     α^(h) ← Σ_{(ℓ,h)∈E} α^(ℓ) ⊘ β^(ℓ) + Σ_{i | fst(i)=ℓ} α_i;
4     d^(h) ← inf{t > 0 | α^(h)(t) ⩽ β^(h)(t)}
5   for i = 1 to m do
6     d_i ← Σ_{h∈p_i} d^(h).
```

В строке 3 (h) - это кривая прибытия для кумулятивной функции прибытия на сервер h. После агрегирования потока вычисление кривых для отдельных потоков не производится. Потоки не могут быть вычислены по отдельным кривым, отсюда и пессимизм. Тогда d(h) - это максимальная задержка, которой подвергается бит данных на сервере, предлагающем строгую кривую обслуживания (h) и имеющем кривую прибытия (h). Поскольку поток представляет собой совокупность нескольких потоков, не имеющих априорной политики обслуживания, этот поток не может рассматриваться как FIFO, и задержка не может быть вычислена как горизонтальное расстояние между кривыми прибытия и обслуживания. Вместо этого необходимо учитывать максимальную длительность периода задержки, что соответствует формуле строки 4 (Теорема 5.5). Данный алгоритм является весьма пессимистичным, так как не учитывает политику обслуживания. Тем не менее, он может быть использован при очень общих предположениях: если FIFO-per-flow не требуется, что невозможно для других методов, представленных далее (отметим, что если потоки не FIFO, то существуют более эффективные методы; см. например, [SCH 11]).

ПРИМЕР 10.4.- Рассмотрим не тандемную сеть на рис. 10.1 (вверху). Мы находим

```
1 begin
2   for h = 1 to n do
3     α^(h) ← Σ_{(ℓ,h)∈E} α^(ℓ) ⊘ β^(ℓ) + Σ_{i | fst(i)=ℓ} α_i;
4     d^(h) ← inf{t > 0 | α^(h)(t) ⩽ β^(h)(t)}
5   for i = 1 to m do
6     d_i ← Σ_{h∈p_i} d^(h).
```

и задержки могут быть непосредственно получены из этих формул.

В таком виде этот метод весьма пессимистичен, особенно потому, что он рассчитывает производительность так, как если бы все данные, обслуживаемые данным сервером, передавались всем его преемникам. Это можно обойти, используя Теорему 5.4 и заменив строку 3 на

$$\alpha^{(h)} \leftarrow \sum_{i \in Fl(l)} \alpha_i + \sum_{(l,h)\in E} \alpha^{(l)}\beta^{(l)}(0),$$

или

$$\alpha^{(h)} \leftarrow \sum_{i \in Fl(l)} \alpha_i + \Big(\sum_{(l,h)\in E} \alpha^{(l)}\beta^{(l)}(0) - \sum_{i \in Fl(h), fs(i)\neq l} \alpha_i(0+)\Big) \wedge \delta_0 \ ,$$

в зависимости от формы кривых прихода. Очевидно, что данный метод может быть усовершенствован. Это можно сделать несколькими способами:

1) С учетом предположения FIFO-per-flow и политики обслуживания. Например, если серверы работают по принципу FIFO, то строку 4 можно заменить на $d(h) = hDev(\alpha^{(h)}, \beta^{(h)})$ и можно рассматривать кривые обслуживания по принципу "мин-плюс";

2) учет явления плати только один раз , соответствующего составу серверов, что является предметом рассмотрения в остальной части раздела; 3) учет, когда это возможно, явления мультиплексирования с оплатой только один раз, что было рассмотрено в разделе 10.3.

5.2 10.4.2. Анализ разделенных потоков

Алгоритм 6 дает общий способ вычисления границ производительности, учитывающий явление "плати только один раз". Сначала вычисляется кривая прибытия для каждого потока в каждой промежуточной системе: i(h) - кривая прибытия для Fj(h). Затем вычисляется кривая остаточного обслуживания для каждого потока в каждой системе: i(h) - кривая обслуживания для потока i в системе h. Наконец, вычисляется глобальная кривая обслуживания для каждого потока путем свертки, и верхние границы наихудшей производительности могут быть вычислены с использованием этой кривой и Теоремы 5.2.

Алгоритм 6 справедлив для слепого мультиплексирования (а значит, и для любой другой политики обслуживания) если системы предлагают строгие кривые обслуживания. Границы производительности, вычисленные с помощью этого алгоритма, могут быть улучшены, если известно больше информации о системах и кривых.

1) Если максимальная кривая обслуживания (h) и формирователь (h) известны для каждого сервера h, а также минимальная кривая прибытия i для каждого потока i, то строки (4-5) можно заменить следующими тремя строками (с очевидными обозначениями), следуя Теореме 5.3:

$$\beta_i^{(h)} \leftarrow \left[\beta^{(j)} - \sum_{j \in \mathrm{Fl}(h) \setminus \{i\}} \alpha_j^{(\mathrm{pred}_i(h))}\right]^+;$$
$$\alpha_i^{(h)} \leftarrow \min(\alpha_i^{(\mathrm{pred}_i(h))} * \overline{\beta}^{(\mathrm{pred}_i(h))} \oslash \beta_i^{(h)}, \sigma^{(\mathrm{pred}_i(h))});$$
$$\underline{\alpha}_i^{(h)} \leftarrow \underline{\alpha}_i^{(\mathrm{pred}_i(h))} * \beta_i^{(h)}.$$

2) Если политика обслуживания - FIFO (сервер может предложить кривую обслуживания "минимум плюс"), SP или GPS, то достаточно заменить строки (4-5) формулами, соответствующими конкретной политике, т.е. использовать теоремы 7.5, 7.6 или 7.7. Естественно, можно комбинировать различные политики обслуживания с кривыми максимального обслуживания и минимального прибытия.

Алгоритм 6: Общий SFA (анализ разделенных потоков) Данные: $(\alpha_i)_m^{i=1} \in F^m$, $(\beta^{(h)})_{h=1}^n \in F^n$ $(p_i)_{i=1}^m$ возрастающие пути. Результат: $\widetilde{\beta_i}$ кривая мини-

мального обслуживания для каждого потока i для глобальной сети.

begin
  for $i = 1$ **to** m **do**
    $\alpha_i^{(0)} \leftarrow \alpha_i$;
  for $h = 1$ **to** n **do**
    foreach i **such that** $h \in \mathbf{p}_i$ **do**
      $\beta_i^{(h)} \leftarrow \left[\beta^{(h)} - \sum_{j \in \mathrm{Fl}(h)\setminus\{i\}} \alpha_j^{(\mathrm{pred}_i(h))}\right]^+$;
      $\alpha_i^{(h)} \leftarrow \alpha_i^{(\mathrm{pred}_i(h))} \oslash \beta_i^{(h)}$;
  for $i = 1$ *to* m **do**
    $\tilde{\beta}_i \leftarrow \circledast_{h \in \mathbf{p}_i} \beta_i^{(h)}$.

Представленный здесь алгоритм претендует не на оптимальность, а на алгоритмическую эффективность. Действительно, он требует O(nm) базовых операций (свертка, деконволюция и т.д.). ПРИМЕР 10.5.- Снова рассмотрим пример рис. 10.1 (вверху). Получаем:

$$\begin{cases}
\alpha_1^{(0)} = \alpha_1, & \alpha_2^{(0)} = \alpha_2, \\
\alpha_3^{(0)} = \alpha_3, & \\
\beta_1^{(1)} = \left[\beta^{(1)} - \alpha_3\right]^+, & \alpha_1^{(1)} = \alpha_1 \oslash \left[\beta^{(1)} - \alpha_3\right]^+, \\
\beta_3^{(1)} = \left[\beta^{(1)} - \alpha_1\right]^+, & \alpha_3^{(1)} = \alpha_3 \oslash \left[\beta^{(1)} - \alpha_1\right]^+, \\
\beta_1^{(2)} = \left[\beta^{(2)} - \alpha_2\right]^+, & \alpha_1^{(2)} = \alpha_1^{(1)} \oslash \left[\beta^{(2)} - \alpha_2\right]^+, \\
\beta_2^{(2)} = \left[\beta^{(2)} - \alpha_1^{(1)}\right]^+, & \alpha_2^{(2)} = \alpha_2 \oslash \left[\beta^{(2)} - \alpha_1^{(1)}\right]^+, \\
\beta_2^{(3)} = \left[\beta^{(3)} - \alpha_3^{(1)}\right]^+, & \alpha_2^{(3)} = \alpha_2^{(2)} \oslash \left[\beta^{(3)} - \alpha_3^{(1)}\right]^+, \\
\beta_3^{(3)} = \left[\beta^{(3)} - \alpha_2^{(2)}\right]^+, & \alpha_3^{(3)} = \alpha_3^{(1)} \oslash \left[\beta^{(3)} - \alpha_2^{(2)}\right]^+.
\end{cases}$$

Если политика обслуживания статическая приоритетная, при этом поток 1 имеет наивысший приоритет, а поток 3 - наименьший, то вычисленные кривые прибытия и обслуживания имеют вид:

$$\begin{cases}
\alpha_1^{(0)} = \alpha_1, & \alpha_2^{(0)} = \alpha_2, \\
\alpha_3^{(0)} = \alpha_3, & \\
\beta_1^{(1)} = \beta^{(1)}, & \alpha_1^{(1)} = \alpha_1 \oslash \beta^{(1)}, \\
\beta_3^{(1)} = \left[\beta^{(1)} - \alpha_1\right]^+, & \alpha_3^{(1)} = \alpha_3 \oslash \left[\beta^{(1)} - \alpha_1\right]^+, \\
\beta_1^{(2)} = \beta^{(2)}, & \alpha_1^{(2)} = \alpha_1^{(1)} \oslash \beta^{(2)}, \\
\beta_2^{(2)} = \left[\beta^{(2)} - \alpha_1^{(1)}\right]^+, & \alpha_2^{(2)} = \alpha_2 \oslash \left[\beta^{(2)} - \alpha_1^{(1)}\right]^+, \\
\beta_2^{(3)} = \beta^{(3)}, & \alpha_2^{(3)} = \alpha_2^{(2)} \oslash \beta^{(3)}, \\
\beta_3^{(3)} = \left[\beta^{(3)} - \alpha_2^{(2)}\right]^+, & \alpha_3^{(3)} = \alpha_3^{(1)} \oslash \left[\beta^{(3)} - \alpha_2^{(2)}\right]^+.
\end{cases}$$

5.3 10.4.3. Анализ группового потока

Анализ общего выхода и анализ разделенных потоков рассматривают либо весь набор потоков, пересекающих сервер, либо каждый поток в отдельности. Это два экстремальных решения, а группировка потоков может быть выполнена более продуманно. Это особенно актуально, когда для каждого сервера h известен формирователь (h). В TOA, когда поток агрегируется на сервере, для следующего сервера используется вся наихудшая граница отставания. Кроме того, вычисляется наихудшая задержка для каждого сервера.

В SFA кривая остаточного обслуживания для каждого сервера и потока получается по формулам $\beta_i^{(h)} = [\beta^{(j)} - \sum_{j \in Fl(h)\{ß}\alpha_j^{(l)}]^+$ и $\alpha_i^{(h)} = min(\alpha_i^{(l)}\beta_i^{(h)}, \sigma^{(l)})$, С $l = pred_i(h)$. Этот метод может стать очень пессимистичным, когда несколько потоков следуют по одной и той же дуге в сети. Действительно, рассмотрим сеть на рис. 10.1 (середина) и сервер 2. Кривые поступления $\alpha_1^{(1)} = [\beta^{(1)} - \alpha_2]^+ \wedge \sigma(1)$ и $\alpha_2^{(1)} = [\beta^{(1)} - \alpha_1]^+ \wedge \sigma(1)$ были вычислены. При вычислении услуги предлагаемой сервером 2 потоку 3, мы вычислим $\beta_3^{(2)} = [\beta^{(2)} - \alpha_1^{(1)} - \alpha_2^{(1)}]^+$. Другим решением будет вычисление глобальной кривой прихода на сервер 2 с сервера 1, то есть с сервера 1, которая, согласно теореме 5.3, равна $\eta = (\alpha_1 + \alpha_2)\beta^{(1)} \wedge \sigma^{(1)}$, а остаточная кривая обслуживания для потока 3, таким образом, равна $[\beta^{(2)} - \eta]^+ \geq \beta_3^{(2)}$. Принцип анализа групповых потоков заключается в вычислении кривых прибытия потоков по дуги сети. Тот же самый аргумент можно использовать для кривых максимального обслуживания вместо кривых формирования, но для простоты здесь будут рассмотрены только кривые формирования. Аналогичным образом этот подход может быть применен к любому способу группировки потоков. Здесь мы ограничимся группировкой по дугам сети. Это естественный способ группировки, использующий преимущества формирователя. Такой способ группировки потоков может привести к алгоритму с экспоненциальной сложностью.

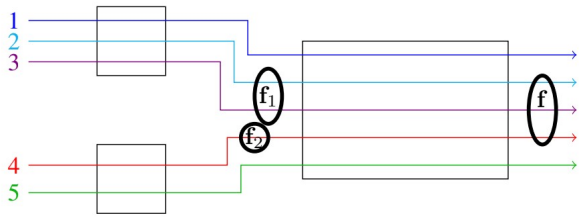

Рисунок 10.5. Группировка потоков в соответствии с дугами сети: группировка потоков f = t2, 3, 4 после сервера h требует их разбиения на два подмножества f1 и f2 в зависимости от их предыдущего посещения сервера.

Для обобщения приведенного примера рассмотрим сервер h, изображенный на рис. 10.5. Для вычисления кривой остаточного обслуживания потоков в f и их кривой прибытия после сервера h, нам необходимо знать кривую прибытия этой группы потоков и кривую прибытия других потоков 1, чтобы воспользоваться формулами
$\beta_f^{(h)} = [\beta^{(h)} - \eta^{'}]^+$ и $\alpha_f^{(h)} = \eta\beta_f^{(h)} \wedge \sigma^{(h)}$.

Для расчета кривых прибытия и 1 можно сгруппировать потоки в соответствии с Дугой , которую они пересекают. Для этого введем дополнительные обозначения: Fl(l, k) – множество потоков, последовательно пересекающих серверы и k (эквивалентно, это множество потоков, которые следуют по дуге (l, k)). Таким образом, теперь мы можем записать формулу:

$$\eta = \sum_{l|(l,h)\in E} +\alpha^{(l)}_{f\cap Fl(l,h)} + \sum_{i\in f|fst(i)=h} \alpha_i^{(0)},$$

и аналогично для ', заменив f на $\bar{f} = Fl(h)$.

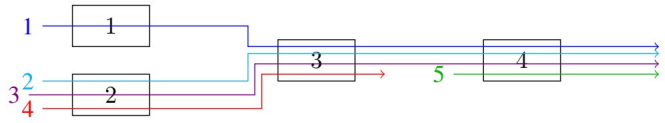

Рисунок 10.6. Пример сети для анализа группового потока.

Анализ группового потока выполняется в следующие два этапа, представленные в виде Алгоритм 7: 1) Вычисление группы потоков, необходимых для анализа. Это делается путем обратного поиска на серверах. Если производительность вычисляется для каждого потока i, то необходимо вычислить кривую остаточного обслуживания для каждого потока i для каждого сервера, через который он проходит. Поэтому строка 2 является инициализирующей. Затем в строке 6 вычисляются группы потоков, которые будут оцениваться на каждом сервере вычисляются группы потоков, подлежащих оценке на каждом сервере, по примеру рис. 10.5 для каждого сервера h. 2) Вычисление кривых остаточного обслуживания и кривых прибытия, необходимых для анализа потока i. 3) Вычисление кривых остаточного обслуживания и кривых прибытия, необходимых для анализа потока i. Это выполняется на каждом сервере в соответствии с топологическим порядком. Первый шаг гарантирует, что все необходимые группировки были вычислены. Заметим, что пустое множество может быть элементом Fh. Мы используем соглашение о том, что $\alpha(h) = 0$.

Поскольку число запрашиваемых групп потоков может быть экспоненциальным, целесообразно ограничиться вычислением кривой остаточного обслуживания интересующего подмножества потоков (например, одного потока i). В этом случае изменяется только строка 2 шага инициализации, а именно заменой $F_h \leftarrow \cup_{i\in Fl(h)}\{\{i\}\}$ на $F_h \leftarrow \{\{i\}\}$ если $i \in Fl(h)$ и в противном случае.

ПРИМЕР 10.6.- Рассмотрим пример рис. 10.6, где интересующим нас

потоком является поток 2. На шаге инициализации в строке 2 получаем F2 = F3 = F4 = 2 и F1 = . Тогда обратное вычисление групп дает (мы не учитываем пустые множества, которые могут появиться) пустые множества, которые могут появиться): F4 = 2 (оно никогда не изменяется), F3 = 2, 1,3, F2=3,2,4,2, 3,4 и F1= 1.

Алгоритм 7: Общий GFA (анализ групповых потоков)

Данные: $(\alpha_i)_m^{i=1} \in F^m$, $(\beta^{(h)})_{h=1}^n \in F^n$ $(p_i)_{i=1}^m$ возрастающие пути, E множество дуг GN . Результат: $\widetilde{\beta_i}$ - кривая минимального обслуживания для каждого потока i в глобальной сети.

begin
  for $h = 1$ **to** n **do** $\mathbf{F}_h \leftarrow \cup_{i \in \mathrm{Fl}(h)} \{\{i\}\}$;
  for $h = n$ **to** 1 **do**
    foreach $\mathbf{f} \in \mathbf{F}_h$ **do**
      foreach ℓ **such that** $(\ell, h) \in E$ **do**
        $\mathbf{F}_\ell \leftarrow \mathbf{F}_\ell \cup \{\mathbf{f} \cap \mathrm{Fl}(\ell, h), \bar{\mathbf{f}} \cap \mathrm{Fl}(\ell, h)\}$;
  for $h = 1$ **to** n **do**
    foreach $\mathbf{f} \in \mathbf{F}_h$ **do**
      $\eta_{\mathbf{f}}^{(h)} \leftarrow \sum_{\ell \mid (\ell,h) \in E} \alpha^{(\ell)}_{\mathbf{f} \cap \mathrm{Fl}(\ell,h)} + \sum_{i \in \mathbf{f} \mid \mathrm{fst}(i) = h} \alpha_i$;
      $\eta_{\bar{\mathbf{f}}}^{(h)} \leftarrow \sum_{\ell \mid (\ell,h) \in E} \alpha^{(\ell)}_{\bar{\mathbf{f}} \cap \mathrm{Fl}(\ell,h)} + \sum_{i \notin \mathbf{f} \mid \mathrm{fst}(i) = h} \alpha_i$;
    foreach $\mathbf{f} \in \mathbf{F}_h$ **do**
      $\beta_{\mathbf{f}}^{(h)} \leftarrow [\beta^{(h)} - \eta_{\bar{\mathbf{f}}}^{(h)}]^+$;
      $\alpha_{\mathbf{f}}^{(h)} \leftarrow (\eta_{\mathbf{f}}^{(h)} \oslash \beta_{\mathbf{f}}^{(h)}) \wedge \sigma^{(h)}$;
  for $i = 1$ *to* m **do**
    $\tilde{\beta}_i \leftarrow *_{h \in \mathbf{p}_i} \beta_{\{i\}}^{(h)}$.

Теперь мы можем вычислить кривые поступления и кривые обслуживания:

В сервере 1, $\eta_{\{1\}}^{(1)} = \alpha_1$, $\beta_{\{1\}}^{(1)} = \beta^{(1)}$ и $\alpha_{\{1\}}^{(1)} = \alpha_1 \beta^{(1)} \wedge \sigma^{(1)}$.

В сервере 2, $\eta_{\{3\}}^{(2)} = \alpha_3$, $\eta_{\{2,4\}}^{(2)} = \alpha_2 + \alpha_4$, $\eta_{\{2\}}^{(2)} = \alpha_2$ и $eta_{\{3,4\}}^{(2)} = \alpha_3 + \alpha_4$, Тогда

$$\begin{cases} \beta_{\{3\}}^{(2)} = \beta^{(2)} - \eta_{\{2,4\}}^{(2)} & \alpha_{\{3\}}^{(2)} = \alpha_2 \oslash \beta_{\{3\}}^{(2)} \wedge \sigma^{(2)} \\ \beta_{\{2,4\}}^{(2)} = \beta^{(2)} - \eta_{\{3\}}^{(2)} & \alpha_{\{2,4\}}^{(2)} = \alpha_2 \oslash \beta_{\{2,4\}}^{(2)} \wedge \sigma^{(2)} \\ \beta_{\{2\}}^{(2)} = \beta^{(2)} - \eta_{\{3,4\}}^{(2)} & \alpha_{\{2\}}^{(2)} = \alpha_2 \oslash \beta_{\{2\}}^{(2)} \wedge \sigma^{(2)} \\ \beta_{\{3,4\}}^{(2)} = \beta^{(2)} - \eta_{\{2\}}^{(2)} & \alpha_{\{3,4\}}^{(2)} = \alpha_2 \oslash \beta_{\{3,4\}}^{(2)} \wedge \sigma^{(2)}. \end{cases}$$

В 3 сервере, $\eta_{\{1,3\}}^{(3)} = \alpha_{\{}^1 1\} + \alpha_{\{3\}}^{(2)}$, $\eta_{\{2,4\}}^{(3)} = \alpha_{\{}^{(2)} 2, 4\}$, $\eta_{\{2\}}^{(3)} = \alpha_{\{}^{(2)} 2\}$ и

$\eta^{(3)}_{\{1,3,4\}} = \alpha^{(1)}_{\{1\}} + \alpha^{(2)}_{\{3,4\}}$, тогда $\begin{cases} \beta^{(3)}_{\{1,3\}} = \beta^{(3)} - \eta^{(3)}_{\{2,4\}} & \alpha^{(3)}_{\{1,3\}} = \eta^{(3)}_{\{1,3\}} \oslash \beta^{(2)}_{\{1,3\}} \wedge \sigma^{(3)} \\ \beta^{(3)}_{\{2\}} = \beta^{(3)} - \eta^{(3)}_{\{1,3,4\}} & \alpha^{(3)}_{\{2\}} = \eta^{(3)}_{\{2\}} \oslash \beta^{(3)}_{\{2\}} \wedge \sigma^{(3)}. \end{cases}$

Наконец, в сервере 4 мы имеем $\eta^{(4)}_2 = \alpha^{(3)}_{\{2\}}$ и $\eta_{\{1,3,5\}} = \alpha^{(3)}_{\{1,3\}} + \alpha_5$ тогда $\beta^{(4)}_{\{2\}} = \beta^{(4)} - \eta^{(3)}_{\{1,3,5\}}$ и $\alpha^{(4)}_{\{2\}} = \eta^{(4)}_{\{2\}} \beta^{(4)}_{\{2\}} \wedge \sigma^{(4)}$.

6 10.5. NP-трудность вычисления узких границ

В предыдущих разделах было описано несколько способов вычисления границ наихудшей производительности. Все алгоритмы имеют полиномиальную сложность по размеру сети и количеству выполняемых базовых операций (причем эти операции для обычных классов функций имеют полиномиальную сложность). Однако в разделе 10.3.1 было также показано, что получить жесткие границы производительности не представляется возможным. Здесь под жесткими понимается возможность нахождения кумулятивных процессов отправления и прибытия, которые подчиняются всем ограничениям (кривые прибытия, кривые обслуживания), такие, что максимальная задержка потока или максимальное отставание от графика в точности соответствует ограничению, вычисляется с помощью остаточной кривой обслуживания, полученной в результате работы алгоритма. В этом разделе мы покажем, что проблема нахождения точной наихудшей границы производительности является NP-трудной задачей, т.е. если P = NP, то не существует полиномиального алгоритма для решения точной задачи о наихудшей производительности в рамках сетевого исчисления.

ТЕОРЕМА 10.2.- Рассмотрим сеть с прямой передачей данных, в которой каждый сервер имеет произвольную политику обслуживания мультиплексирования. - Вычисление максимального отставания на данном сервере является NP-трудной задачей; - вычисление максимальной задержки потока является NP-трудным.

ДОКАЗАТЕЛЬСТВО. - Мы сводим задачу точного трехслойного покрытия (X3C) к нашей задаче. Экземпляр X3C - это коллекция C = c1,...,c3 из 3q элементов и коллекция U = u1,..., us из s наборов по три элемента из C. Задача состоит в том, чтобы решить,. существует ли покрытие C q элементами U. Мы сведем эту задачу к решению вопроса о том, можно ли достичь заданного отставания или задержки на сервере сети. Из экземпляра X3C построим сеть, как показано на рис. 10.7. Более точнее, верхняя ступень состоит из 3q серверов U1,...,U3q, каждый из которых имеет строгое обслуживание кривой 1. Средний этап состоит из s серверов V1,...,Vs, каждый из которых имеет строгую кривую обслуживания 2. Наконец, на нижней ступени имеется только один сервер V , с кривой обслуживания R с R >

3s. Имеется 3s потоков, каждый из которых пересекает три сервера сверху вниз. Поток i, j пересекает серверы Vj , Vi и W тогда и только тогда, когда uj $\in vi. : t \longrightarrow min(t, 1)(,)$.

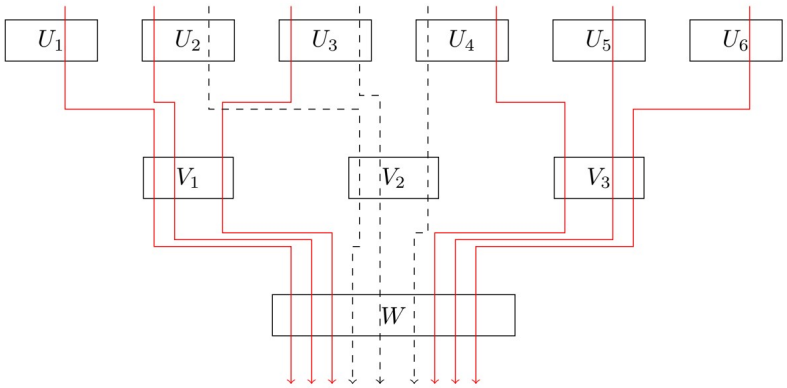

Рисунок 10.7. Преобразование экземпляра X3C в сеть. На рис. 10.7 приведен пример с U = 1, 2, 3, 4, 5, 6 и V 1, 2, 3, 2, 3, 4, 4, 5, 6, поэтому 1, 2, 3, 4, 5, 6 является 3-покрытием C. NP-трудность максимального отставания. Мы хотим решить, может ли отставание в W быть не менее 3s - 2q.

Сначала мы докажем, что наихудшее отставание сервера W будет получено в момент времени 1 -, если с момента времени 0 потоки поступают в соответствии с , а серверы Uj и Vi являются точными серверами на [0, 1) и бесконечны в момент времени 1. Тогда отставание сервера W в момент времени 1 равно отставанию всей системы в момент времени 1 - . Зафиксируем 0 как дату поступления первого бита данных в систему.

Наихудший бэклог получен в момент времени s >= 1. Если бы оно было получено в момент времени s > 1, то каждый бит данных еще не был бы отправлен в сеть, и если в момент времени s успело образовать некоторое отставание, то до момента времени 1 может образоваться еще некоторое отставание. В наихудшем случае отставание получается в момент времени s = 1. Если бы оно было получено в момент времени s > 1, рассмотрим следующее преобразование: задержим прибытие каждого потока так, чтобы каждый поток начал поступать в момент времени s -1. Тогда количество данных, обслуживаемых в течение [s - 1, s) после преобразования меньше, чем количество данных, обслуживаемых за время [0, s) в начальной конфигурации, следовательно отставание увеличилось, и первый бит данных поступает в момент времени s - 1, а не 0, и существует наихудшая траектория с s = 1.

Соответствие с X3C. Рассмотрим бесконечно малый интервал времени, в течение которого каждый поток (i, j) на верхней ступени обслуживается со скоростью ri,j, причем $\sum_{i:u_j \in u_i} r_{i,j} = 1$. Скорость обслуживания Ui равна $min(2, \sum_{j:u_j \in u_i} r_{i,j}$,а затем темп роста отставания за этот промежу-

ток времени на среднем этапе составляет $\sum_{i\in\{1,...s\}}(\sum_{j:u_j\in u_i} r_{i,j} - 2)_+$. На верхнем этапе создается отставание со скоростью $3s - \sum_{i,j} r_{ij} = 3(s-q)$, которая не зависит от обслуживаемых потоков. На сервере V отставание не создается, так как R > 3s. Максимизация отставания эквивалентна максимизации следующей функции: $\sum\limits_{i\in\{1,...s\}}[\sum\limits_{j:u_j\in u_i} r_{i,j} - 2]^+$,с учетом ограничений: $j \in \{1,...,3q\}, \sum_{i:u_j\in u_i} r_{i,j} = 1$ и $\forall i,j, r_{i,j} \geq 0$.
Наша задача сводится к максимизации выпуклой функции на выпуклом множестве. При этом максимальные значения получаются в некоторых экстремальных вершинах выпуклого множества. Экстремальными вершинами выпуклого множества являются такие, что $\sum_{i:u_j\in u_i} r_{i,j} = 1, r_{i,j} \in \{0,1\}$ и тогда $\sum_{i:u_j\in u_i} r_{i,j} \in \{0,1,2,3\}$. Максимизация нашей функции эквивалентна максимизации числа i таких, что $\sum_{i:u_j\in u_i} r_{i,j} = 3$.

Это число ограничено сверху q, и этот максимум достигается тогда и только тогда, когда существует покрытие X3C (фактически, существует соответствие "точка-точка"между множеством экстремальных вершин, достигающих q, и множеством покрытий X3C).

Если существует покрытие X3C, то отставание на среднем этапе увеличивается со скоростью q (со скоростью 1 для каждого сервера, получающего данные со скоростью 3). Если же покрытие X3C отсутствует, то отставание на среднем этапе увеличивается не более чем на q - 1. Наконец, приведенные выше рассуждения справедливы для интервала времени [0, 1]. Действительно, отставание является субаддитивным: если кумулятивная функция прибытий F1, определенная на интервале [0, s] в сервере создается отставание b1 в момент времени s, и если кумулятивная функция прибытия F2 такая, что F2(s) = 0, создает отставание b2 в момент времени t s (когда сервер пуст в момент времени s), то процесс F1 + F2 создает отставание b <= b1 + b2 в момент времени t. В результате нет никакого преимущества в изменении тарифов обслуживания ri,j в течение интервала времени времени [0, 1]. В момент времени 1 серверы Uj и Vi обслужили весь свой бэклог, и бэклог в сервере W составляет не более 3s – 2q. Этот максимум достигается тогда и только тогда, когда существует покрытие X3C. Если покрытия X3C нет, то отставание составляет не более 3s - 2q - 1.

NP-трудность задержки. Мы сохраняем ту же схему редукции. Имеется дополнительный поток, пересекающий только сервер W, и теперь нас интересует вычисление наихудшей задержки для бита данных данного потока (можно взять r при r < R – 3s и максимальная задержка может быть получена для первого бита данных, поступающего в систему).

В наихудшем сценарии для этой сети сначала создается отставание на сервере W, а затем все остальные серверы рассматриваются как серверы с бесконечной пропускной способностью. Для создания отставания мы используем предыдущую конструкцию.

Запас на сервере верхней ступени может быть создан, если данные поступают со скоростью больше, чем 1.

Запас на сервере средней ступени может быть создан, если на сервер средней ступени поступают данные со скоростью r $>$ 2 на сервер средней ступени. Запас растет со скоростью r - 2. Если скорость поступления 3, то потоки поступают со скоростью 1, блокируя потоки на верхнем уровне, и, следовательно, именно в этом сценарии образуется наибольшее отставание.

Предположим, что ровно k серверов средней стадии имеют скорость поступления 3 между временем 0 и tk. Без потери общности, серверами, создающими отставание, являются V1,...,Vk. Созданное отставание равно (Nk - 2k)tk, где $Nk = |(l,j)|\exists i <= k, u_j \in v_i \cap v|$, а для остальных потоков - либо они простаивают, либо передаются. Для этих других потоков на среднем этапе не создается запаса, но отставание может быть создано на верхней стадии со скоростью $[\sum_{l:u_j \in u_l} r_{l,j} - 1]^+$ для каждого сервера Cj . Верхнее ограничение на создаваемое отставание мы можем получить, используя неравенство Nk <= 3s – 3(q – k), а оставшиеся потоки ограничить снизу на 3(q – k), и считая их незанятыми (они начнут передавать данные в момент времени tk). Отметим, что если k = q, то это не является приближением.

В результате в момент времени t $\geq tk, Wtk()t, tk()t,$
$Q_k(t) = (3s - 3q + k)t_k + 2s(min(t,1) - t_k) + 3(q-k)(max(t,1) - 1)$. Таким образом, задержка меньше t - tk такая, что Формула!!

Если t <= 1, то $Q_k(k) = R(t - t_k)$ и t - tk возрастает с tk. Если t >= 1, то $t - t_k = \frac{3(s-q-k)-2kt_k}{R-3(q-k)}$, и t - tk убывает с tk.Поэтому, чтобы для получения максимальной задержки необходимо выбрать tk таким образом, чтобы t = 1. Тогда $t_k = \frac{R-3s}{R-3q+k}$ и $d_k = \frac{3(s-q)+k}{R-3q+k}$ является верхней границей задержки. Поскольку R $>$ 3s, dk увеличивается с ростом k. dq может быть эффективно достигнуто для k = q, поскольку в этом конкретном случае аппроксимация отсутствует. Если существует покрытие X3C, то максимальная задержка составляет $\frac{3s-2q}{R-2q}$, а если нет X3C, то максимальная задержка, которую можно получить, составляет $\frac{3s-2q-1}{R-2q-1}$. Таким образом, вопрос о том, может ли быть достигнута задержка, большая или равная $\frac{3s-2q}{R-2q}$, является NP-трудным.

7 10.6. Заключение

Несмотря на то, что существует несколько результатов, позволяющих точно рассчитать наихудшую производительность при рассмотрении одного сервера, о чем говорится в Теореме 5.2 и в параграфе о герметичности после теорем 7.1 и 7.4 (разделы 7.2.1 и 7.3.2.1), при рассмотрении целой сети проблема оказывается более сложной. сложнее, если рассматривать всю сеть. Эта глава началась с иллюстрации основных источников потери герметичности при рассмотрении последовательности серверов. Затем были представлены три алгоритма, работающие с общими топологиями с прямой передачей данных. Наконец, был получен общий результат, показывающий, что вычисление точной наихудшей границы в общем случае является NP-трудным. Отметим, что определение алгоритмов, основанных на итерационном применении локальных результатов, является активной областью

исследований [BON 16c, BON 17b], а три алгоритма, представленные в работе, носят скорее поучительный характер, чем актуальный. Вычисление точных границ при использование кривых остаточного обслуживания требует наличия различных точек зрения на сеть: разрезание одного потока на последовательность из двух потоков [LEN 08], расширение потока [BON 17a], максимизация агрегации потоков при перекрестном движении [BON 16a] и т.д. Для оценки пессимистичности сетевого исчисления нам, возможно, придется обратиться к различные исследования, как промышленные [BOY 12b], так и академические [BON 16c, BON 17b].