Распределенные алгоритмы

ЛЕКТОР: В.А. Захаров

Лекция 5.

Задача маршрутизации.
Маршрутизация на основе узлов-адресатов.
Задача построения кратчайших путей для всех пар вершин.
Алгоритм Флойда–Уоршалла.
Алгоритм Туэга.
Алгоритм Мерлина–Сигалла.
Алгоритм Чанди–Мисры.

Задача маршрутизации

В общем случае процесс (узел в компьютерной сети) непосредственно не соединен каналами связи с каждым другим процессом.

Из каждого узла пакеты информации могут непосредственно передаваться только некоторому подмножеству других узлов, которые называются соседями этого узла.

Маршрутизация — это процедура принятия решения о том, какому соседу (иногда не единственному) следует передать пакет, чтобы он в конце концов был доставлен по назначению.

Цель, которая ставится при проектировании алгоритма маршрутизации, состоит в том, чтобы снабдить каждый узел процедурой, которая сможет выполнять эту функцию и гарантировать доставку каждого пакета по назначению.

Задача маршрутизации

В каждом узел должна храниться некоторая информация о топологии сети, и локальная процедура должна принимать решение на основе этой информации.

Эта информация представлена в таблицей маршрутизации .
Задачу маршрутизации можно разбить на две алгоритмические составляющие:

Задача маршрутизации

В каждом узел должна храниться некоторая информация о топологии сети, и локальная процедура должна принимать решение на основе этой информации.

Эта информация представлена в таблицей маршрутизации . Задачу маршрутизации можно разбить на две алгоритмические составляющие:

1. Вычисление таблиц. Таблицы маршрутизации должны быть вычислены при инициализации сети и должны обновляться при изменении топологии сети.

Задача маршрутизации

В каждом узел должна храниться некоторая информация о топологии сети, и локальная процедура должна принимать решение на основе этой информации.

Эта информация представлена в таблицей маршрутизации . Задачу маршрутизации можно разбить на две алгоритмические составляющие:

1. Вычисление таблиц. Таблицы маршрутизации должны быть вычислены при инициализации сети и должны обновляться при изменении топологии сети.
2. Продвижение пакета. Когда пакет пересылается по сети, то его продвижение осуществляется на основе таблиц маршрутизации.

Задача маршрутизации

Критерии оценки качества методов маршрутизации учитывают следующие показатели.

Задача маршрутизации

Критерии оценки качества методов маршрутизации учитывают следующие показатели.

1. Корректность.
2. Эффективность.
3. Сложность.
4. Устойчивость.
5. Адаптивность.
6. Справедливость.

Задача маршрутизации

Критерии оценки качества методов маршрутизации учитывают следующие показатели.

1. Корректность. Алгоритм должен доставлять каждый пакет, поступивший в сеть, в точности по назначению.
2. Эффективность.
3. Сложность.
4. Устойчивость.
5. Адаптивность.
6. Справедливость.

Задача маршрутизации

Критерии оценки качества методов маршрутизации учитывают следующие показатели.

1. Корректность.
2. Эффективность. Алгоритм должен отправлять пакеты по «наилучшим» путям; например, ожидается, что выбранные пути приводят к небольшой задержке и обеспечивают высокую пропускную способность всей сети. Алгоритм называется оптимальным , если он позволяет задействовать «наилучшие» пути.
3. Сложность.
4. Устойчивость.
5. Адаптивность.
6. Справедливость.

Задача маршрутизации

Критерии оценки качества методов маршрутизации учитывают следующие показатели.

1. Корректность.
2. Эффективность.
3. Сложность. Алгоритм вычисления таблиц должен использовать как можно меньше сообщений, расходовать как можно более экономно время и память.
4. Устойчивость.
5. Адаптивность.
6. Справедливость.

Задача маршрутизации

Критерии оценки качества методов маршрутизации учитывают следующие показатели.

1. Корректность.
2. Эффективность.
3. Сложность.
4. Устойчивость. В случае изменения топологии (добавлении или удалении канала или вершины) алгоритм вносит изменения в таблицы маршрутизации, для того чтобы задачу маршрутизации можно было решить в модифицированной сети.
5. Адаптивность.
6. Справедливость.

Задача маршрутизации

Критерии оценки качества методов маршрутизации учитывают следующие показатели.

1. Корректность.
2. Эффективность.
3. Сложность.
4. Устойчивость.
5. Адаптивность. Для того чтобы выровнять загруженность каналов и вершин, алгоритм приспосабливает к этому таблицы, избегая прокладывать пути через те каналы и вершины, которые чрезмерно загружены, и отдавая предпочтение тем каналам и вершинам, на которые в это время возложена наименьшая нагрузка.
6. Справедливость.

Задача маршрутизации

Критерии оценки качества методов маршрутизации учитывают следующие показатели.

1. Корректность.
2. Эффективность.
3. Сложность.
4. Устойчивость.
5. Адаптивность.
6. Справедливость. Алгоритм должен обслуживать всех пользователей в равной мере.

Задача маршрутизации

Сеть представляется графом, вершины которого соответствуют вершинам сети; соседние вершины соединены ребром, если между ними есть канал связи. Существует несколько вариантов понятия «наилучший путь».

Задача маршрутизации

Сеть представляется графом, вершины которого соответствуют вершинам сети; соседние вершины соединены ребром, если между ними есть канал связи. Существует несколько вариантов понятия «наилучший путь».

1. Минимальное количество звеньев.
2. Минимальная стоимость.
3. Минимальная задержка.

Задача маршрутизации

Сеть представляется графом, вершины которого соответствуют вершинам сети; соседние вершины соединены ребром, если между ними есть канал связи. Существует несколько вариантов понятия «наилучший путь».

1. Минимальное количество звеньев. Стоимость использования пути определяется количеством звеньев (пройденных каналов или шагов от одной вершины к другой) в этом пути. Алгоритм маршрутизации с минимальным числом звеньев выбирает пути, имеющих наименьшее возможное число звеньев.
2. Минимальная стоимость.
3. Минимальная задержка.

Задача маршрутизации

Сеть представляется графом, вершины которого соответствуют вершинам сети; соседние вершины соединены ребром, если между ними есть канал связи. Существует несколько вариантов понятия «наилучший путь».

1. Минимальное количество звеньев.
2. Минимальная стоимость. Каждому каналу связи придается неизменная (обычно неотрицательная) стоимость (или вес), и стоимость пути полагается равной суммарному весу всех каналов пути. Алгоритм кратчайшего пути выбирает путь наименьшей стоимости.
3. Минимальная задержка.

Задача маршрутизации

Сеть представляется графом, вершины которого соответствуют вершинам сети; соседние вершины соединены ребром, если между ними есть канал связи. Существует несколько вариантов понятия «наилучший путь».

1. Минимальное количество звеньев.
2. Минимальная стоимость.
3. Минимальная задержка. Каждому каналу связи придается динамически изменяемая характеристика, зависящая от трафика через этот канал. Алгоритм минимальной задержки постоянно пересматривает таблицы так, чтобы всегда выбирались пути с (почти) минимальной общей задержкой.

Маршрутизация на основе узлов-адресатов

Выбор маршрута для продвижения пакета обычно проводится только с учетом узла-адресата пакета независимо от исходного отправителя (источника) пакета.

Маршрутизация на основе узлов-адресатов

Выбор маршрута для продвижения пакета обычно проводится только с учетом узла-адресата пакета независимо от исходного отправителя (источника) пакета.

Основные допущения о свойствах маршрутов.

Маршрутизация на основе узлов-адресатов

Выбор маршрута для продвижения пакета обычно проводится только с учетом узла-адресата пакета независимо от исходного отправителя (источника) пакета.

Основные допущения о свойствах маршрутов.

1. Стоимость отправления пакета по пути P не зависит от того, используются ли ребра пути P для продвижения других сообщений. Поэтому стоимость $C(P)$ пути P — это функция, зависящая только от самого пути.

Маршрутизация на основе узлов-адресатов

Выбор маршрута для продвижения пакета обычно проводится только с учетом узла-адресата пакета независимо от исходного отправителя (источника) пакета.

Основные допущения о свойствах маршрутов.

1. Стоимость отправления пакета по пути P не зависит от того, используются ли ребра пути P для продвижения других сообщений. Поэтому стоимость $C(P)$ пути P — это функция, зависящая только от самого пути.
2. При последовательном соединении двух путей образуется новый путь, стоимость которого равна сумме стоимостей двух исходных путей, т.е.
   $C(\langle u_0, u_1, \ldots, u_k \rangle) = C(\langle u_0, \ldots, u_i \rangle) + C(\langle u_i, \ldots, u_k \rangle)$.
   Стоимость пустого пути $\langle u_0 \rangle$ равна 0.

Маршрутизация на основе узлов-адресатов

Выбор маршрута для продвижения пакета обычно проводится только с учетом узла-адресата пакета независимо от исходного отправителя (источника) пакета.

Основные допущения о свойствах маршрутов.

1. Стоимость отправления пакета по пути P не зависит от того, используются ли ребра пути P для продвижения других сообщений. Поэтому стоимость $C(P)$ пути P — это функция, зависящая только от самого пути.
2. При последовательном соединении двух путей образуется новый путь, стоимость которого равна сумме стоимостей двух исходных путей, т.е.
   $C(\langle u_0, u_1, \ldots, u_k \rangle) = C(\langle u_0, \ldots, u_i \rangle) + C(\langle u_i, \ldots, u_k \rangle)$.
   Стоимость пустого пути $\langle u_0 \rangle$ равна 0.
3. В графе нет циклов отрицательной стоимости.

Маршрутизация на основе узлов-адресатов

Выбор маршрута для продвижения пакета обычно проводится только с учетом узла-адресата пакета независимо от исходного отправителя (источника) пакета.

Основные допущения о свойствах маршрутов.

1. Стоимость отправления пакета по пути P не зависит от того, используются ли ребра пути P для продвижения других сообщений. Поэтому стоимость $C(P)$ пути P — это функция, зависящая только от самого пути.
2. При последовательном соединении двух путей образуется новый путь, стоимость которого равна сумме стоимостей двух исходных путей, т.е.
   $C(\langle u_0, u_1, \ldots, u_k \rangle) = C(\langle u_0, \ldots, u_i \rangle) + C(\langle u_i, \ldots, u_k \rangle)$.
   Стоимость пустого пути $\langle u_0 \rangle$ равна 0.
3. В графе нет циклов отрицательной стоимости.

Путь из вершины u в вершину v называется оптимальным , если нет пути меньшей стоимости из u в v

Маршрутизация на основе узлов-адресатов

Лемма 5.1. (о простых путях).

Пусть u и v — вершины графа G . Если в графе G есть путь P из u в v , то имеется также и простой путь из u в v , который является при этом оптимальным.

Маршрутизация на основе узлов-адресатов

Лемма 5.1. (о простых путях).

Пусть u и v — вершины графа G . Если в графе G есть путь P из u в v , то имеется также и простой путь из u в v , который является при этом оптимальным.

Задача.

1. Верно ли, что каждый оптимальный путь обязательно должен быть простым путем?
2. Насколько существенным здесь является требование отсутствия циклов отрицательной стоимости?

Маршрутизация на основе узлов-адресатов

Теорема 5.1. (о дереве оптимальных путей).

Для каждой вершины $u \in V$ связного графа G существует такое дерево $T_u = (V, E_u)$, $E_u \subseteq E$, в котором для любой вершины $v \in V$, единственный путь из v в u в дереве T_u является оптимальным путем из v в u в графе G .

Маршрутизация на основе узлов-адресатов

Теорема 5.1. (о дереве оптимальных путей).

Для каждой вершины $u \in V$ связного графа G существует такое дерево $T_u = (V, E_u)$, $E_u \subseteq E$, в котором для любой вершины $v \in V$, единственный путь из v в u в дереве T_u является оптимальным путем из v в u в графе G .

Доказательство.

Пусть $V = \{v_1, \ldots, v_N\}$. Построим индуктивно такие деревья $T_i = (V_i, E_i)$, $i = 0, \ldots, N$, что

Маршрутизация на основе узлов-адресатов

Теорема 5.1. (о дереве оптимальных путей).

Для каждой вершины $u \in V$ связного графа G существует такое дерево $T_u = (V, E_u)$, $E_u \subseteq E$, в котором для любой вершины $v \in V$, единственный путь из v в u в дереве T_u является оптимальным путем из v в u в графе G .

Доказательство.

Пусть $V = \{v_1, \ldots, v_N\}$. Построим индуктивно такие деревья $T_i = (V_i, E_i)$, $i = 0, \ldots, N$, что

1. Каждое дерево T_i является подграфом графа G .

Маршрутизация на основе узлов-адресатов

Теорема 5.1. (о дереве оптимальных путей).

Для каждой вершины $u \in V$ связного графа G существует такое дерево $T_u = (V, E_u)$, $E_u \subseteq E$, в котором для любой вершины $v \in V$, единственный путь из v в u в дереве T_u является оптимальным путем из v в u в графе G .

Доказательство.

Пусть $V = \{v_1, \ldots, v_N\}$. Построим индуктивно такие деревья $T_i = (V_i, E_i)$, $i = 0, \ldots, N$, что

1. Каждое дерево T_i является подграфом графа G .
2. Каждое дерево T_i — поддерево дерева T_{i+1} .

Маршрутизация на основе узлов-адресатов

Теорема 5.1. (о дереве оптимальных путей).

Для каждой вершины $u \in V$ связного графа G существует такое дерево $T_u = (V, E_u)$, $E_u \subseteq E$, в котором для любой вершины $v \in V$, единственный путь из v в u в дереве T_u является оптимальным путем из v в u в графе G .

Доказательство.

Пусть $V = \{v_1, \ldots, v_N\}$. Построим индуктивно такие деревья $T_i = (V_i, E_i)$, $i = 0, \ldots, N$, что

1. Каждое дерево T_i является подграфом графа G .
2. Каждое дерево T_i — поддерево дерева T_{i+1} .
3. Для каждого $i > 0$, выполняются $v_i \in V_i$ и $u \in V_i$.

Маршрутизация на основе узлов-адресатов

Теорема 5.1. (о дереве оптимальных путей).

Для каждой вершины $u \in V$ связного графа G существует такое дерево $T_u = (V, E_u)$, $E_u \subseteq E$, в котором для любой вершины $v \in V$, единственный путь из v в u в дереве T_u является оптимальным путем из v в u в графе G .

Доказательство.

Пусть $V = \{v_1, \ldots, v_N\}$. Построим индуктивно такие деревья $T_i = (V_i, E_i)$, $i = 0, \ldots, N$, что

1. Каждое дерево T_i является подграфом графа G .
2. Каждое дерево T_i — поддерево дерева T_{i+1} .
3. Для каждого $i > 0$, выполняются $v_i \in V_i$ и $u \in V_i$.
4. Для каждой вершины $w \in V_i$ путь из w в u в дереве T_i является оптимальным путем из w в u в графе G .

Маршрутизация на основе узлов-адресатов

Теорема 5.1. (о дереве оптимальных путей).

Для каждой вершины $u \in V$ связного графа G существует такое дерево $T_u = (V, E_u)$, $E_u \subseteq E$, в котором для любой вершины $v \in V$, единственный путь из v в u в дереве T_u является оптимальным путем из v в u в графе G .

Доказательство.

Пусть $V = \{v_1, \ldots, v_N\}$. Построим индуктивно такие деревья $T_i = (V_i, E_i)$, $i = 0, \ldots, N$, что

1. Каждое дерево T_i является подграфом графа G .
2. Каждое дерево T_i — поддерево дерева T_{i+1} .
3. Для каждого $i > 0$, выполняются $v_i \in V_i$ и $u \in V_i$.
4. Для каждой вершины $w \in V_i$ путь из w в u в дереве T_i является оптимальным путем из w в u в графе G .

Ясно, что дерево T_N будет искомым.

Маршрутизация на основе узлов-адресатов

Доказательство.

Положим $V_0 = \{u\}$ и $E_0 = \emptyset$.

Маршрутизация на основе узлов-адресатов

Доказательство.

Положим $V_0 = \{u\}$ и $E_0 = \emptyset$.

Дерево T_{i+1} строится так. Выбирается оптимальный простой путь $P = \langle u_0, \ldots, u_k \rangle$ из v_{i+1} в u . Пусть ℓ — наименьший индекс, для которого $u_\ell \in T_i$. Положим

$$V_{i+1} = V_i \cup \{u_j : j < \ell\} \quad \text{и} \quad E_{i+1} = E_i \cup \{(u_j, u_{j+1}) : j < \ell\}.$$

Маршрутизация на основе узлов-адресатов

Доказательство.

Положим $V_0 = \{u\}$ и $E_0 = \emptyset$.

Дерево T_{i+1} строится так. Выбирается оптимальный простой путь $P = \langle u_0, \ldots, u_k \rangle$ из v_{i+1} в u . Пусть ℓ — наименьший индекс, для которого $u_\ell \in T_i$. Положим

$$V_{i+1} = V_i \cup \{u_j : j < \ell\} \quad \text{и} \quad E_{i+1} = E_i \cup \{(u_j, u_{j+1}) : j < \ell\}.$$

Ясно, что T_i является поддеревом графа T_{i+1} , и $v_{i+1} \in V_{i+1}$. Граф T_{i+1} является деревом, поскольку по построению T_{i+1} является связным графом, и число вершин в нем превосходит число ребер в точности на единицу.

Маршрутизация на основе узлов-адресатов

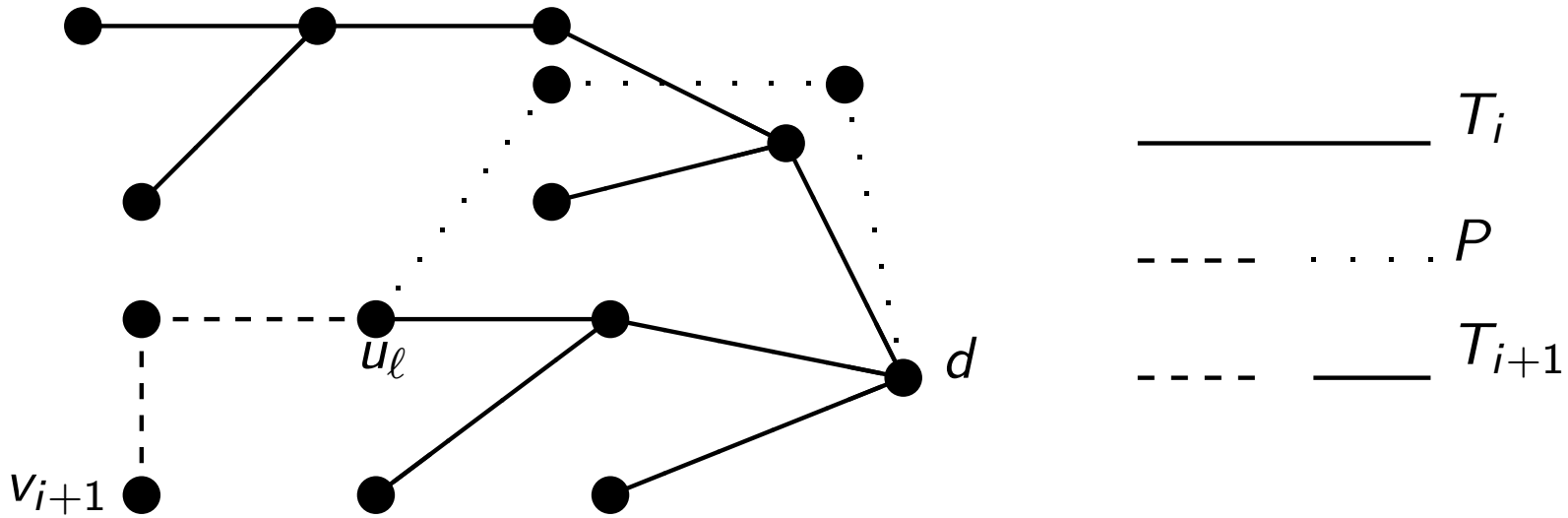

Рис.: Построение дерева T_{i+1}.

Маршрутизация на основе узлов-адресатов

Доказательство.

Покажем, что для всех вершин $w \in V_{i+1}$ путь из w в u в дереве T_{i+1} является оптимальным путем из w в u в графе G .

Маршрутизация на основе узлов-адресатов

Доказательство.

Покажем, что для всех вершин $w \in V_{i+1}$ путь из w в u в дереве T_{i+1} является оптимальным путем из w в u в графе G .
Для вершин $w \in V_i \subset V_{i+1}$ это верно, т.к. T_i является поддеревом дерева T_{i+1} .

Маршрутизация на основе узлов-адресатов

Доказательство.

Покажем, что для всех вершин $w \in V_{i+1}$ путь из w в u в дереве T_{i+1} является оптимальным путем из w в u в графе G .
Для вершин $w \in V_i \subset V_{i+1}$ это верно, т.к. T_i является поддеревом дерева T_{i+1} .
Пусть $w = u_j$, $j < \ell$, принадлежит $V_{i+1} \setminus V_i$, и пусть Q'' — путь из u_ℓ в u в дереве T_i . Тогда в дереве T_{i+1} из вершины u_j в вершину u ведет путь, полученный в результате соединения пути $Q' = \langle u_j, \ldots, u_\ell \rangle$ и пути Q'' .

Маршрутизация на основе узлов-адресатов

Доказательство.

Покажем, что для всех вершин $w \in V_{i+1}$ путь из w в u в дереве T_{i+1} является оптимальным путем из w в u в графе G .
Для вершин $w \in V_i \subset V_{i+1}$ это верно, т.к. T_i является поддеревом дерева T_{i+1} .
Пусть $w = u_j$, $j < \ell$, принадлежит $V_{i+1} \setminus V_i$, и пусть Q'' — путь из u_ℓ в u в дереве T_i . Тогда в дереве T_{i+1} из вершины u_j в вершину u ведет путь, полученный в результате соединения пути $Q' = \langle u_j, \ldots, u_\ell \rangle$ и пути Q'' .
Остается показать, что путь $Q = Q'Q''$ является оптимальным в графе G .

Маршрутизация на основе узлов-адресатов

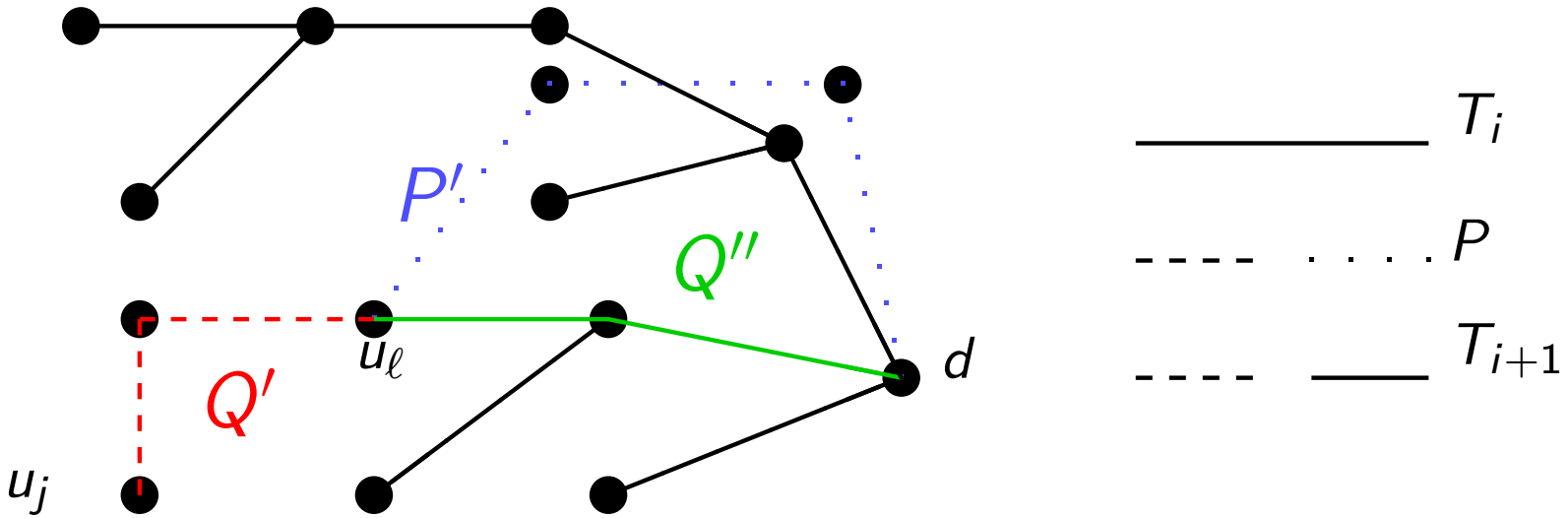

Рис.: Построение дерева T_{i+1}.

Маршрутизация на основе узлов-адресатов

Доказательство.

Заметим, что суффикс $P' = \langle u_\ell, \ldots, u_k \rangle$ пути P является оптимальным путем из u_ℓ в u . Поэтому $C(P') = C(Q)$. Это следует из оптимальности путей Q и P (учитывая свойство аддитивности стоимости путей).

Маршрутизация на основе узлов-адресатов

Доказательство.

Заметим, что суффикс $P' = \langle u_\ell, \ldots, u_k \rangle$ пути P является оптимальным путем из u_ℓ в u . Поэтому $C(P') = C(Q)$. Это следует из оптимальности путей Q и P (учитывая свойство аддитивности стоимости путей).

Допустим, что есть путь R из u_j в u меньшей стоимости, чем путь $Q'Q''$. Тогда стоимость R меньше, чем стоимость пути $Q'P'$, а это означает (учитывая свойство аддитивности стоимости путей), что путь, образованный присоединением к пути $\langle u_0, \ldots, u_j \rangle$ пути R , имеет меньшую стоимость, чем оптимальный путь P . □

Маршрутизация на основе узлов-адресатов

Остовное дерево, корнем которого служит вершина u , называется входным деревом для u , а дерево, удовлетворяющее теореме о дереве оптимальных путей называется оптимальным входным деревом .
Существование оптимального входного дерева означает, что алгоритм оптимальной маршрутизации может работать с использованием следующей локальной процедуры $table\_lookup_v$.

Маршрутизация на основе узлов-адресатов

Остовное дерево, корнем которого служит вершина u , называется входным деревом для u , а дерево, удовлетворяющее теореме о дереве оптимальных путей называется оптимальным входным деревом .
Существование оптимального входного дерева означает, что алгоритм оптимальной маршрутизации может работать с использованием следующей локальной процедуры $table\_lookup_v$.

Процедура $table\_lookup_v$ имеет один аргумент и выбирает одного из соседей вершины v . Т.к. все пакеты, адресованные узлу u оптимально направить по остовному дереву T_u , продвижение пакета будет оптимальным, если для каждой вершины $v \neq u$ процедура $table\_lookup_v(u)$ будет вычислять родительскую вершину узла v в дереве T_u .

Маршрутизация на основе узлов-адресатов

(* Пакет, адресованный вершине u, был получен или сформирован в вершине u *)

if $v = u$
  then доставить пакет локальными средствами
  else отправить пакет вершине $table\_lookup_v(u)$

Алгоритм оптимального продвижения пакетов.

Маршрутизация на основе узлов-адресатов

Но вполне возможна «маленькая неприятность»: разные узлы могут иметь в виду разные входные деревья. Не случится ли так, что пакет будет «передаваться по кругу»?

Маршрутизация на основе узлов-адресатов

Но вполне возможна «маленькая неприятность»: разные узлы могут иметь в виду разные входные деревья. Не случится ли так, что пакет будет «передаваться по кругу»?

Для доказательства корректности таблиц маршрутизации полезен следующий результат. Будем говорить, что таблицы маршрутизации (для узла-адресата u) содержат цикл , если существуют такие вершины $u_1, \ldots, u_k$, что

- $u_i \neq u$ для всех i ,
- $table\_lookup_{u_i}(u) = u_{i+1}$ для всех $i < k$ и
- $table\_lookup_{u_k}(u) = u_1$.

Таблицы называются ациклическими , если они не содержат цикла ни для одной вершины u .

Маршрутизация на основе узлов-адресатов

Но вполне возможна «маленькая неприятность»: разные узлы могут иметь в виду разные входные деревья. Не случится ли так, что пакет будет «передаваться по кругу»?

Для доказательства корректности таблиц маршрутизации полезен следующий результат. Будем говорить, что таблицы маршрутизации (для узла-адресата u) содержат цикл , если существуют такие вершины $u_1, \ldots, u_k$, что

- $u_i \neq u$ для всех i ,
- $table\_lookup_{u_i}(u) = u_{i+1}$ для всех $i < k$ и
- $table\_lookup_{u_k}(u) = u_1$.

Таблицы называются ациклическими , если они не содержат цикла ни для одной вершины u .

Лемма 5.2. (об ациклических таблицах)

Механизм продвижения доставляет каждый пакет по назначению в том и только том случае, когда таблицы маршрутизации являются ациклическими.

Маршрутизация на основе узлов-адресатов

Доказательство

Если таблицы маршрутизации содержат цикл для некоторого узла-адресата u , то пакет, адресованный процессу u , никогда не будет доставлен по назначению из узла, входящего в цикл.
Предположим, что таблицы являются ациклическими, и пакет, предназначенный узлу-адресату u и отправленный из узла-источника u_0 , продвинулся через вершины $u_0, u_1, u_2, \ldots$.
Если одна и та же вершина встречается в этой последовательности дважды, то таблицы содержат цикл вопреки предположению об ацикличности таблиц.
Значит каждая вершина встречается в последовательности не более одного раза. Поэтому эта последовательность является конечной, и оканчивается некоторой вершиной u_k $(k < N)$.
В соответствии с описанием процедуры продвижения пакета указанная последовательность может оканчиваться только вершиной u .

Маршрутизация на основе узлов-адресатов

Задача.

Допустим, что таблицы маршрутизации так обновляются после каждого изменения топологической структуры сети, что они остаются ациклическими по ходу обновления. Может ли это служить гарантией того, что пакеты всегда доставляются по адресу даже в том случае, когда сеть претерпевает бесконечно большое количество топологических изменений?

Докажите, что ни один алгоритм маршрутизации не способен обеспечить доставку пакетов по адресу, если сеть испытывает непрерывные изменения топологии.

Задача построения кратчайших путей для всех пар вершин

Для каждой пары вершин (u, v) алгоритм должен вычислить длину кратчайшего пути из вершины u в вершину v и занести в память процесса u первый канал этого пути.

Задача построения кратчайших путей для всех пар вершин

Для каждой пары вершин (u, v) алгоритм должен вычислить длину кратчайшего пути из вершины u в вершину v и занести в память процесса u первый канал этого пути.

Мы рассмотрим алгоритм Туэга (Toueg) совместного распределенного вычисления таблиц маршрутизации для всех узлов сети. В основу этого алгоритма положен централизованный алгоритм Флойда–Уоршалла.

Алгоритм Флойда–Уоршалла

Предположим, что имеется взвешенный граф $G = (V, E)$, в котором каждому ребру uv приписан вес ω_{uv} .

Мы будем считать, что в графе отсутствуют циклы отрицательного веса.

Ве пути $\langle u_0, \ldots, u_k \rangle$ — это число, равное $\sum_{i=0}^{k-1} \omega_{u_i u_{i+1}}$.

Расстояние между вершинами u и v — наименьший вес $d(u, v)$ пути, соединяющего вершины u и v (если таких путей нет, то $d(u, v) = \infty$).

Задача построения кратчайших путей для всех пар вершин состоит в том, чтобы вычислить расстояние $d(u, v)$ для каждой пары вершин u и v .

Алгоритм Флойда–Уоршалла

Для вычисления всех расстояний в алгоритме Флойда–Уоршалла используется понятие S-путей , в которых все промежуточные вершины принадлежат подмножеству S множества вершин V .

Определение 5.1. (S -расстояния)

Пусть S — некоторое подмножество множества вершин V . Путь $\langle u_0, \ldots, u_k \rangle$ называется S -**путем**, если $u_i \in S$ для каждого $i, 0 < i < k$. S-**расстоянием** между вершинами u и v , которое обозначается $d^S(u, v)$, называется наименьший вес S -пути между u и v (если таких путей нет, то $d^S(u, v) = \infty$).

Алгоритм Флойда–Уоршалла

Для вычисления всех расстояний в алгоритме Флойда–Уоршалла используется понятие S-путей , в которых все промежуточные вершины принадлежат подмножеству S множества вершин V .

Определение 5.1. (S -расстояния)

Пусть S — некоторое подмножество множества вершин V . Путь $\langle u_0, \ldots, u_k \rangle$ называется S -**путем**, если $u_i \in S$ для каждого $i, 0 < i < k$. S-**расстоянием** между вершинами u и v , которое обозначается $d^S(u, v)$, называется наименьший вес S -пути между u и v (если таких путей нет, то $d^S(u, v) = \infty$).

Работа алгоритма начинается с построения всех $\varnothing$ -путей, а затем множество S -путей наращивается для все более обширных подмножеств S , до тех пор пока не будут рассмотрены все V -пути.

Алгоритм Флойда–Уоршалла

Теорема 5.2. (об S-путях)

Для всех вершин u и подмножеств S выполняется равенство $d^S(u, u) = 0$. Если $u \neq v$, то S -пути подчиняются следующим правилам.

Алгоритм Флойда–Уоршалла

Теорема 5.2. (об S-путях)

Для всех вершин u и подмножеств S выполняется равенство $d^S(u, u) = 0$. Если $u \neq v$, то S -пути подчиняются следующим правилам.

1. $\varnothing$-путь из вершины u в вершину v существует в том и только том случае, когда $uv \in E$.

Алгоритм Флойда–Уоршалла

Теорема 5.2. (об S-путях)

Для всех вершин u и подмножеств S выполняется равенство $d^S(u, u) = 0$. Если $u \neq v$, то S -пути подчиняются следующим правилам.

1. $\varnothing$-путь из вершины u в вершину v существует в том и только том случае, когда $uv \in E$.
2. Если $uv \in E$, то $d^{\varnothing}(u, v) = \omega_{uv}$; иначе $d^{\varnothing}(u, v) = \infty$.

Алгоритм Флойда–Уоршалла

Теорема 5.2. (об S-путях)

Для всех вершин u и подмножеств S выполняется равенство $d^S(u, u) = 0$. Если $u \neq v$, то S -пути подчиняются следующим правилам.

1. $\varnothing$-путь из вершины u в вершину v существует в том и только том случае, когда $uv \in E$.
2. Если $uv \in E$, то $d^{\varnothing}(u, v) = \omega_{uv}$; иначе $d^{\varnothing}(u, v) = \infty$.
3. Если $S' = S \cup \{w\}$, то простой S' -путь из u в v — это либо S -путь из u в v , либо соединение двух S -путей, один из которых ведет из u в w , а другой — из w в v .

Алгоритм Флойда–Уоршалла

Теорема 5.2. (об S-путях)

Для всех вершин u и подмножеств S выполняется равенство $d^S(u, u) = 0$. Если $u \neq v$, то S -пути подчиняются следующим правилам.

1. $\varnothing$-путь из вершины u в вершину v существует в том и только том случае, когда $uv \in E$.
2. Если $uv \in E$, то $d^{\varnothing}(u, v) = \omega_{uv}$; иначе $d^{\varnothing}(u, v) = \infty$.
3. Если $S' = S \cup \{w\}$, то простой S' -путь из u в v — это либо S -путь из u в v , либо соединение двух S -путей, один из которых ведет из u в w , а другой — из w в v .
4. Если $S' = S \cup \{w\}$, то $d^{S'}(u, v) = \min\left(d^S(u, v), d^S(u, w) + d^S(w, v) \right)$.

Алгоритм Флойда–Уоршалла

Теорема 5.2. (об S-путях)

Для всех вершин u и подмножеств S выполняется равенство $d^S(u, u) = 0$. Если $u \neq v$, то S -пути подчиняются следующим правилам.

1. $\varnothing$-путь из вершины u в вершину v существует в том и только том случае, когда $uv \in E$.
2. Если $uv \in E$, то $d^{\varnothing}(u, v) = \omega_{uv}$; иначе $d^{\varnothing}(u, v) = \infty$.
3. Если $S' = S \cup \{w\}$, то простой S' -путь из u в v — это либо S -путь из u в v , либо соединение двух S -путей, один из которых ведет из u в w , а другой — из w в v .
4. Если $S' = S \cup \{w\}$, то $d^{S'}(u, v) = \min\left(d^S(u, v), d^S(u, w) + d^S(w, v) \right)$.
5. Вершины u и v соединены путем в том и только том случае, когда между вершинами u и v существует V -путь.

Алгоритм Флойда–Уоршалла

Теорема 5.2. (об S-путях)

Для всех вершин u и подмножеств S выполняется равенство $d^S(u, u) = 0$. Если $u \neq v$, то S -пути подчиняются следующим правилам.

1. $\varnothing$-путь из вершины u в вершину v существует в том и только том случае, когда $uv \in E$.
2. Если $uv \in E$, то $d^{\varnothing}(u, v) = \omega_{uv}$; иначе $d^{\varnothing}(u, v) = \infty$.
3. Если $S' = S \cup \{w\}$, то простой S' -путь из u в v — это либо S -путь из u в v , либо соединение двух S -путей, один из которых ведет из u в w , а другой — из w в v .
4. Если $S' = S \cup \{w\}$, то $d^{S'}(u, v) = \min\left(d^S(u, v), d^S(u, w) + d^S(w, v) \right)$.
5. Вершины u и v соединены путем в том и только том случае, когда между вершинами u и v существует V -путь.
6. $d(u, v) = d^V(u, v)$,

Алгоритм Флойда–Уоршалла

Доказательство

Правила (1)–(3) и (5)–(6) — самостоятельно .

Алгоритм Флойда–Уоршалла

Доказательство

Правила (1)–(3) и (5)–(6) — самостоятельно .

Правило (4). Применяя лемму 5.1. (о простых путях), можно показать, что в том случае, если в графе есть S' -путь из u в v , существует и *простой* S' -путь длины $d^{S'}(u, v)$ из u в v .
Согласно правилу (3) отсюда следует равенство
$d^{S'}(u, v) = \min\left(d^{S}(u, v), d^{S}(u, w) + d^{S}(w, v) \right)$.

Алгоритм Флойда–Уоршалла

begin(* Вначале S равно $\varnothing$, а в D заданы $\varnothing$-расстояния *)
$S := \varnothing$;
forall u, v **do**
if $u = v$ **then** $D[u, v] := 0$
else if $uv \in E$ **then** $D[u, v] := \omega_{uv}$ **else** $D[u, v] := \infty$;
(* Добавить к S опорную вершину *)
while $S \neq V$ **do**
(* Инвариант цикла: $\forall u, v : D[u, v] = d^S(u, v)$ *)
begin pick w from $V \setminus S$;
(* Глобальная обработка опорной вершины w *)
forall $u \in V$ **do**
(* Локальная обработку опорной вершины w для u *)
forall $v \in V$ **do**
$D[u, v] := \min(D[u, v], D[u, w] + D[w, v])$;
$S := S \cup \{w\}$
end (* $\forall u, v : D[u, v] = d(u, v)$ *)
end

Алгоритм Флойда–Уоршалла

Теорема 5.3. (об алгоритм Флойда–Уоршалла)

Алгоритм Флойда–Уоршалла вычисляет расстояние между всеми парами вершин за $\Theta(N^3)$ шагов.

Алгоритм Флойда–Уоршалла

Теорема 5.3. (об алгоритм Флойда–Уоршалла)

Алгоритм Флойда–Уоршалла вычисляет расстояние между всеми парами вершин за $\Theta(N^3)$ шагов.

Доказательство.

В начале работы алгоритма $D[u, v] = 0$, если $u = v$, $D[u, v] = \omega_{uv}$, если $uv \in E$, и $D[u, v] = \infty$ в остальных случаях. При этом $S = \varnothing$. Согласно правилам (1) и (2) Теоремы 5.2. о S -путях, $\forall u, v : D[u, v] = d^S(u, v)$. Если к S добавляется вершина w , то согласно правилам (3) и (4) оператор присваивания, уточняющий значение $D[u, v]$, обеспечивает сохранение выполнимости инварианта цикла $\forall u, v : D[u, v] = d^S(u, v)$. Программа завершает выполнение, когда $S = V$. Принимая во внимание правила (5) и (6), а также инвариант цикла, заключаем, что S-расстояния между всеми вершинами равны расстояниям.

Основной цикл выполняется N раз, и в каждой его итерации выполняется N^2 операций.

Алгоритм Туэга построения кратчайших путей

Основные допущения.

A1. Каждый цикл в сети имеет положительный вес.

Алгоритм Туэга построения кратчайших путей

Основные допущения.

A1. Каждый цикл в сети имеет положительный вес.

A2. Каждый процесс сети располагает информацией об отличительных признаках всех узлов системы (множества вершин V).

Алгоритм Туэга построения кратчайших путей

Основные допущения.

A1. Каждый цикл в сети имеет положительный вес.

A2. Каждый процесс сети располагает информацией об отличительных признаках всех узлов системы (множества вершин V).

A3. Каждый процесс располагает информацией о том, какие узлы являются его соседями (для каждой вершины u эти сведения содержатся в массиве $neigh_u$) и какой вес имеют каналы, соединяющие процесс с его соседями.

Алгоритм Туэга построения кратчайших путей

Основные допущения.

A1. Каждый цикл в сети имеет положительный вес.

A2. Каждый процесс сети располагает информацией об отличительных признаках всех узлов системы (множества вершин V).

A3. Каждый процесс располагает информацией о том, какие узлы являются его соседями (для каждой вершины u эти сведения содержатся в массиве $neigh_u$) и какой вес имеют каналы, соединяющие процесс с его соседями.

Нам будет проще разобраться с корректностью алгоритма Туэга, если вначале мы рассмотрим его упрощенный вариант.

Алгоритм Туэга (упрощенный вариант)

- Переменные и операции алгоритма Флойда-Уоршалла разносятся по разным узлам сети. Переменная $D[u, v]$ отдается процессу u ; для удобства обозначения мы будем записывать u на месте индекса следующим образом $D_u[v]$.

Алгоритм Туэга (упрощенный вариант)

- Переменные и операции алгоритма Флойда-Уоршалла разносятся по разным узлам сети. Переменная $D[u, v]$ отдается процессу u ; для удобства обозначения мы будем записывать u на месте индекса следующим образом $D_u[v]$.
- Операции, которые присваивают значения переменной $D_u[v]$ должны выполняться в узле u , и когда значение некоторой переменной, относящейся к узлу w , необходимо для этой операции, это значение должно быть послано процессу u .

Алгоритм Туэга (упрощенный вариант)

- Переменные и операции алгоритма Флойда-Уоршалла разносятся по разным узлам сети. Переменная $D[u, v]$ отдается процессу u ; для удобства обозначения мы будем записывать u на месте индекса следующим образом $D_u[v]$.
- Операции, которые присваивают значения переменной $D_u[v]$ должны выполняться в узле u , и когда значение некоторой переменной, относящейся к узлу w , необходимо для этой операции, это значение должно быть послано процессу u .
- В алгоритме Флойда-Уоршалла все вершины должны использовать информацию от опорной вершины, которая отправляет эту информацию одновременно всем вершинам посредством «широковещательной» рассылки .

Алгоритм Туэга (упрощенный вариант)

- Переменные и операции алгоритма Флойда-Уоршалла разносятся по разным узлам сети. Переменная $D[u, v]$ отдается процессу u ; для удобства обозначения мы будем записывать u на месте индекса следующим образом $D_u[v]$.
- Операции, которые присваивают значения переменной $D_u[v]$ должны выполняться в узле u , и когда значение некоторой переменной, относящейся к узлу w , необходимо для этой операции, это значение должно быть послано процессу u .
- В алгоритме Флойда-Уоршалла все вершины должны использовать информацию от опорной вершины, которая отправляет эту информацию одновременно всем вершинам посредством «широковещательной» рассылки .
- Нужно ввести специальную операцию, для того чтобы не только вычислять длины кратчайших S -путей, но также и первый канал в каждом таком пути (для этого мы будем использовать переменные $Nb_u[v]$).

Алгоритм Туэга (упрощенный вариант)

Лемма 5.3. (об отсутствии циклов)

Пусть заданы множество вершин S и вершина w .

Предположим также, что

1. для всех вершин u верно равенство $D_u[w] = d^S(u, w)$,
2. если $d^S(u, w) < \infty$ и $u \neq w$, то значением переменной $Nb_u[w]$ является имя соседа вершины u на кратчайшем S -пути, ведущем к вершине w .

Тогда ориентированный граф $T_w = (V_w, E_w)$, где

$$V_w = \{u \ : \ D_u[w] < \infty\} \quad \text{и} \quad E_w = \{ux \ : \ u \neq w \wedge Nb_u[w] = x\},$$

является деревом, в корне которого расположена вершина w .

Алгоритм Туэга (упрощенный вариант)

Доказательство.

Заметим, что если для вершины $u \neq w$ выполняется неравенство $D_u[w] < \infty$, то $Nb_u[w] \neq udef$ и $D_{Nb_u[w]}[w] < \infty$. Значит для каждой вершины $u \in V_w$, $u \neq w$, существует вершина $x \in V_w$, для которой верно равенство $Nb_u[w] = x$.

Алгоритм Туэга (упрощенный вариант)

Доказательство.

Заметим, что если для вершины $u \neq w$ выполняется неравенство $D_u[w] < \infty$, то $Nb_u[w] \neq udef$ и $D_{Nb_u[w]}[w] < \infty$. Значит для каждой вершины $u \in V_w$, $u \neq w$, существует вершина $x \in V_w$, для которой верно равенство $Nb_u[w] = x$. Для каждой вершины $u \in V_w$, $u \neq w$, в множестве E_w есть одно ребро, и поэтому число вершин в T_w больше числа ребер на 1. Осталось показать, что граф T_w не содержит циклов.

Алгоритм Туэга (упрощенный вариант)

Доказательство.

Заметим, что если для вершины $u \neq w$ выполняется неравенство $D_u[w] < \infty$, то $Nb_u[w] \neq udef$ и $D_{Nb_u[w]}[w] < \infty$. Значит для каждой вершины $u \in V_w$, $u \neq w$, существует вершина $x \in V_w$, для которой верно равенство $Nb_u[w] = x$. Для каждой вершины $u \in V_w$, $u \neq w$, в множестве E_w есть одно ребро, и поэтому число вершин в T_w больше числа ребер на 1. Осталось показать, что граф T_w не содержит циклов. Коль скоро для ребра $ux \in E_w$ имеет место равенство $d^S(u, w) = \omega_{ux} + d^S(x, w)$, наличие цикла $\langle u_0, u_1, \ldots, u_k \rangle$ в графе T_w повлекло бы за собой следующее соотношение

$$d^S(u_0, w) = \omega_{u_0 u_1} + \omega_{u_1 u_2} + \cdots + \omega_{u_{k-1} u_0} + d^S(u_0, w),$$

из которого вытекает $0 = \omega_{u_0 u_1} + \omega_{u_1 u_2} + \cdots + \omega_{u_{k-1} u_0}$, что противоречит допущению о том, что каждый цикл имеет положительный вес.

Алгоритм Туэга (упрощенный вариант)

var S_u : set of nodes ;
D_u : array of weights ; Nb_u : array of nodes ;
begin $S_u := \varnothing$;
forall $v \in V$ **do**
if $v = u$ **then begin** $D_u[v] := 0$; $Nb_u[v] := udef$ **end**
else if $v \in Neigh_u$
then begin $D_u[v] := \omega_{uv}$; $Nb_u[v] := v$ **end**
else begin $D_u[v] := \infty$; $Nb_u[v] := udef$ **end**;
while $S_u \neq V$ **do**
begin pick w from $V \setminus S_u$;
(* Все процессы выбирают одну и ту же вершину w *)
if $u = w$ **then** "broadcast D_w" **else** "receive D_w" ;
forall $v \in V$ **do**
if $D_u[w] + D_w[v] < D_u[v]$ **then**
begin $D_u[v] := D_u[w] + D_w[v]$; $Nb_u[v] := Nb_u[w]$ **end**;
$S_u := S_u \cup \{w\}$
end
end

Алгоритм Туэга (упрощенный вариант)

Теорема 5.4. (о корректности упрощенного алгоритма Туэга)

Упрощенный алгоритм Туэга завершает свое выполнение в каждом узле сети после N итераций основного цикла. Когда алгоритм завершается в узле u , для каждой вершины v выполняется равенство $D_u[v] = d(u, v)$, и в том случае, если в сети есть путь из u в v , то значением переменной $Nb_u[v]$ является наименование первого канала в кратчайшем пути из u в v ; в противном случае $Nb_u[v] = udef$.

Алгоритм Туэга (упрощенный вариант)

Теорема 5.4. (о корректности упрощенного алгоритма Туэга)

Упрощенный алгоритм Туэга завершает свое выполнение в каждом узле сети после N итераций основного цикла. Когда алгоритм завершается в узле u , для каждой вершины v выполняется равенство $D_u[v] = d(u, v)$, и в том случае, если в сети есть путь из u в v , то значением переменной $Nb_u[v]$ является наименование первого канала в кратчайшем пути из u в v ; в противном случае $Nb_u[v] = udef$.

Доказательство.

Завершаемость и частичная корректность нашего алгоритма следуют из корректности алгоритма Флойда-Уоршалла (Теорема 5.3.). Отсюда следует и справедливость утверждения о значении переменной $Nb_u[v]$, поскольку значение $Nb_u[v]$ изменяется всякий раз, когда переменной $D_u[v]$ присваивается новое значение присваивается. □

Алгоритм Туэга (упрощенный вариант)

Необходимо иметь средства для широковещательного распространения информации, в то время как каждый процесс имеет возможность общаться только со своими соседями.

Алгоритм Туэга (упрощенный вариант)

Необходимо иметь средства для широковещательного распространения информации, в то время как каждый процесс имеет возможность общаться только со своими соседями.
Для этого пригодны волновые алгоритмы .

Алгоритм Туэга (упрощенный вариант)

Необходимо иметь средства для широковещательного распространения информации, в то время как каждый процесс имеет возможность общаться только со своими соседями.
Для этого пригодны волновые алгоритмы .
Но Туэг заметил, что если $D_u[w] = \infty$ в начале этапа обработки опорной вершины, то по окончании этого этапа таблица маршрутизации узла u не изменяется, т.к. неравенство $D_u[w] + D_w[v] < D_u[v]$ неверно для каждой вершины v .

Алгоритм Туэга (упрощенный вариант)

Необходимо иметь средства для широковещательного распространения информации, в то время как каждый процесс имеет возможность общаться только со своими соседями.
Для этого пригодны волновые алгоритмы .
Но Туэг заметил, что если $D_u[w] = \infty$ в начале этапа обработки опорной вершины, то по окончании этого этапа таблица маршрутизации узла u не изменяется, т.к. неравенство $D_u[w] + D_w[v] < D_u[v]$ неверно для каждой вершины v .
Поэтому таблицу маршрутизации D_w нужно доставить только в те узлы, которые принадлежат дереву T_w (в том виде, в котором оно построено к началу этапа обработки опорной вершины), и широковещательную рассылку можно провести эффективно, отправляя таблицу D_w только по тем каналам, которые входят в состав дерева T_w .

Алгоритм Туэга (упрощенный вариант)

Необходимо иметь средства для широковещательного распространения информации, в то время как каждый процесс имеет возможность общаться только со своими соседями.
Для этого пригодны волновые алгоритмы .
Но Туэг заметил, что если $D_u[w] = \infty$ в начале этапа обработки опорной вершины, то по окончании этого этапа таблица маршрутизации узла u не изменяется, т.к. неравенство $D_u[w] + D_w[v] < D_u[v]$ неверно для каждой вершины v .
Поэтому таблицу маршрутизации D_w нужно доставить только в те узлы, которые принадлежат дереву T_w (в том виде, в котором оно построено к началу этапа обработки опорной вершины), и широковещательную рассылку можно провести эффективно, отправляя таблицу D_w только по тем каналам, которые входят в состав дерева T_w . Узел w отправляет таблицу D_w своим сыновним вершинам в дереве T_w , и каждый узел в дереве T_w , получив эту таблицу от родительской вершины в дереве T_w , передает ее своим сыновним вершинам в дереве T_w .

Алгоритм Туэга (полный вариант)

В начале этапа обработки опорной вершины w каждый узел u, для которого $D_u[w] < \infty$, знает, какая вершина является его родителем в дереве T_w, но не знает, какие вершины являются его сыновьями.

Поэтому каждый узел v должен отправить сообщение каждому своему соседу u, сообщив процессу u, является ли v сыновней вершиной для u в дереве T_w.

Алгоритм Туэга (полный вариант)

В начале этапа обработки опорной вершины w каждый узел u, для которого $D_u[w] < \infty$, знает, какая вершина является его родителем в дереве T_w, но не знает, какие вершины являются его сыновьями.

Поэтому каждый узел v должен отправить сообщение каждому своему соседу u, сообщив процессу u, является ли v сыновней вершиной для u в дереве T_w.

Любой узел может принять участие в распространении таблицы опорной вершины w, как только он получит известия о том, какие из соседей являются его сыновними вершинами в дереве T_w.

Алгоритм Туэга (полный вариант)

Инициализация

var S_u : set of nodes ;
D_u : array of weights ;
Nb_u : array of nodes ;

begin $S_u := \varnothing$;
forall $v \in V$ **do**
if $v = u$
then begin $D_u[v] := 0$; $Nb_u[v] := udef$ **end**
else if $v \in Neigh_u$
then begin $D_u[v] := \omega_{uv}$; $Nb_u[v] := v$ **end**
else begin $D_u[v] := \infty$; $Nb_u[v] := udef$ **end**;

Алгоритм Туэга (полный вариант)

Выявление сыновних узлов

while $S_u \neq V$ **do**
begin pick w from $V \setminus S_u$;
(* Построить дерево T_w *)
forall $x \in Neigh_u$ **do**
if $Nb_u[w] = x$ **then** send $\langle \mathbf{ys}, w \rangle$ to x
else send $\langle \mathbf{nys}, w \rangle$ to x ;
$rec_u := 0$; (* u должен получить $|Neigh_u|$ сообщений *)
while $rec_u < |Neigh_u|$ **do**
begin receive $\langle \mathbf{ys}, w \rangle$ or $\langle \mathbf{nys}, w \rangle$; $rec_u := rec_u + 1$ **end**;

Алгоритм Туэга (полный вариант)

Уточнение таблиц маршрутизации

if $D_u[w] < \infty$ **then** (*принять участие в обработке опорной вершины*
begin if $u \neq w$ **then** receive $\langle$**dtab**, $w, D\rangle$ from $Nb_u[w]$;
forall $x \in Neigh_u$ **do**
if $\langle$**ys**, $w\rangle$ was received from x
then send $\langle$**dtab**, $w, D\rangle$ to x ;
forall $v \in V$ **do** (*локальная обработка опорной вершины w*)
if $D_u[w] + D[v] < D_u[v]$ **then**
begin $D_u[v] := D_u[w] + D[v]$; $Nb_u[v] := Nb_u[w]$ **end**
end;
$S_u := S_u \cup \{w\}$
end
end

Алгоритм Туэга (полный вариант)

В алгоритме используются сообщения трех типов:

Алгоритм Туэга (полный вариант)

В алгоритме используются сообщения трех типов:

1. Сообщения $\langle \mathbf{ys}, w \rangle$ (**ys** от «**y**our **s**on») отправляется от узла u к узлу x в начале этапа обработки опорной вершины w , если x является родительской вершиной для u в дереве T_w .

Алгоритм Туэга (полный вариант)

В алгоритме используются сообщения трех типов:

1. Сообщения $\langle \mathbf{ys}, w \rangle$ (**ys** от «**y**our **s**on») отправляется от узла u к узлу x в начале этапа обработки опорной вершины w , если x является родительской вершиной для u в дереве T_w .
2. Сообщение $\langle \mathbf{nys}, w \rangle$ (**nys** от «**n**ot **y**our **s**on») отправляется от узла u к узлу x в начале этапа обработки опорной вершины w , если x не является родительской вершиной для u в дереве T_w .

Алгоритм Туэга (полный вариант)

В алгоритме используются сообщения трех типов:

1. Сообщения $\langle \mathbf{ys}, w \rangle$ (**ys** от «**y**our **s**on») отправляется от узла u к узлу x в начале этапа обработки опорной вершины w , если x является родительской вершиной для u в дереве T_w .
2. Сообщение $\langle \mathbf{nys}, w \rangle$ (**nys** от «**n**ot **y**our **s**on») отправляется от узла u к узлу x в начале этапа обработки опорной вершины w , если x не является родительской вершиной для u в дереве T_w .
3. Сообщение $\langle \mathbf{dtab}, w, D \rangle$ отправляется по ходу обработки опорной вершины w по каждому ребру дерева T_w , чтобы доставить таблицу D_w в каждую вершину, которая должна будет воспользоваться этим значением.

Алгоритм Туэга (полный вариант)

Пусть W — число битов для записи веса пути.

Теорема 5.5. (корректности и сложности алгоритма Туэга)

Для каждой пары вершин u и v алгоритм Туэга вычисляет расстояние между u и v . Если это расстояние конечно, то он также определяет первый канал в кратчайшем пути.

По ходу работы алгоритма по каждому каналу проходит $O(N)$ сообщений, $O(N^2 W)$ битов информации. Таким образом, суммарно по ходу работы алгоритма передается $O(N \cdot |E|)$ сообщений и $O(N^3 \cdot W)$ битов информации. Кроме того в каждом узле используется память, объем которой составляет $O(N \cdot W)$ битов.

Алгоритм Туэга (полный вариант)

Доказательство

Полная версия алгоритма Туэга построен на основе упрощенной версии, и поэтому корректна.

Алгоритм Туэга (полный вариант)

Доказательство

Полная версия алгоритма Туэга построен на основе упрощенной версии, и поэтому корректна. На каждом этапе обработки опорной вершины w по каждому каналу связи проходят два сообщения вида $\langle \mathbf{ys}, w \rangle$ или $\langle \mathbf{nys}, w \rangle$ (по одному сообщению в каждом направлении) и не более одного сообщения вида $\langle \mathbf{dtab}, w, D \rangle$, и значит по каждому каналу проходит не более $3N$ сообщений. Сообщения вида $\langle \mathbf{ys}, w \rangle$ или $\langle \mathbf{nys}, w \rangle$ содержат $O(W)$ битов, а сообщение вида $\langle \mathbf{dtab}, w, D \rangle$ содержит $O(NW)$ битов, и отсюда следует верхняя оценка битовой сложности обмена информацией по каждому каналу связи.

Алгоритм Туэга (полный вариант)

Доказательство

Полная версия алгоритма Туэга построен на основе упрощенной версии, и поэтому корректна. На каждом этапе обработки опорной вершины w по каждому каналу связи проходят два сообщения вида $\langle \mathbf{ys}, w \rangle$ или $\langle \mathbf{nys}, w \rangle$ (по одному сообщению в каждом направлении) и не более одного сообщения вида $\langle \mathbf{dtab}, w, D \rangle$, и значит по каждому каналу проходит не более $3N$ сообщений. Сообщения вида $\langle \mathbf{ys}, w \rangle$ или $\langle \mathbf{nys}, w \rangle$ содержат $O(W)$ битов, а сообщение вида $\langle \mathbf{dtab}, w, D \rangle$ содержит $O(NW)$ битов, и отсюда следует верхняя оценка битовой сложности обмена информацией по каждому каналу связи. За время работы алгоритма передается не более N^2 сообщений вида $\langle \mathbf{dtab}, w, D \rangle$ и $2N \cdot |E|$ сообщений вида $\langle \mathbf{ys}, w \rangle$ и $\langle \mathbf{nys}, w \rangle$; таким образом всего по сети передается $O(N^2 \cdot NW + 2N \cdot |E| \cdot W) = O(N^3 W)$ битов информации.

Алгоритм Туэга (полный вариант)

Доказательство

Полная версия алгоритма Туэга построен на основе упрощенной версии, и поэтому корректна. На каждом этапе обработки опорной вершины w по каждому каналу связи проходят два сообщения вида $\langle \mathbf{ys}, w \rangle$ или $\langle \mathbf{nys}, w \rangle$ (по одному сообщению в каждом направлении) и не более одного сообщения вида $\langle \mathbf{dtab}, w, D \rangle$, и значит по каждому каналу проходит не более $3N$ сообщений. Сообщения вида $\langle \mathbf{ys}, w \rangle$ или $\langle \mathbf{nys}, w \rangle$ содержат $O(W)$ битов, а сообщение вида $\langle \mathbf{dtab}, w, D \rangle$ содержит $O(NW)$ битов, и отсюда следует верхняя оценка битовой сложности обмена информацией по каждому каналу связи. За время работы алгоритма передается не более N^2 сообщений вида $\langle \mathbf{dtab}, w, D \rangle$ и $2N \cdot |E|$ сообщений вида $\langle \mathbf{ys}, w \rangle$ и $\langle \mathbf{nys}, w \rangle$; таким образом всего по сети передается $O(N^2 \cdot NW + 2N \cdot |E| \cdot W) = O(N^3 W)$ битов информации. Для хранения таблиц D_u и Nb_u каждому процессу u требуется $O(NW)$ битов памяти.

Алгоритм Туэга (полный вариант)

Задачи.

1. Зачем нужно передавать в каждом сообщении имя текущей опорной вершины w?. Что произойдет, если этого не делать? Можно ли так модифицировать алгоритм, чтобы не передавать имя опорной вершины в некоторых сообщениях?
2. Студент предложил исключить из алгоритма Туэга отправление сообщений $\langle \mathbf{nys}, w \rangle$. Он привел следующий аргумент: узел и так узнает о том, что его сосед не является сыновней вершиной, если от этого соседа не поступает сообщений $\langle \mathbf{ys}, w \rangle$.
   Можно ли таким образом модифицировать алгоритм? Будет ли он корректным? Какова будет сложность такого алгоритма?

Алгоритм Туэга (обсуждение)

Достоинства. Прост, имеет небольшую сложность, и строит оптимальные маршруты.

Алгоритм Туэга (обсуждение)

Достоинства. Прост, имеет небольшую сложность, и строит оптимальные маршруты.

Недостатки.

- Плохая устойчивость («робастость»): при изменении топологии сети все вычисления нужно проводить заново.
- Согласованный выбор очередной опорной вершины (w) всеми узлами сети предполагает, что множество участвующих в алгоритме процессов заранее известно.

Алгоритм Туэга (обсуждение)

Достоинства. Прост, имеет небольшую сложность, и строит оптимальные маршруты.

Недостатки.

- Плохая устойчивость («робастость»): при изменении топологии сети все вычисления нужно проводить заново.
- Согласованный выбор очередной опорной вершины (w) всеми узлами сети предполагает, что множество участвующих в алгоритме процессов заранее известно.
- В алгоритме Туэга часто применяется неравенства треугольника $d(u, v) \leq d(u, w) + d(w, v)$. Для вычисления правой части этого неравенства (в узле u) требуется «глобальная» информация о $d(w, v)$, которой не обладает ни процесс u, ни его соседи. Зависимость от удаленных данных вынуждает нас организовать доставку этой информации удаленным вершинам.

Алгоритм Туэга (обсуждение)

Альтернативное уравнение для построения кратчайших путей:

$$d(u, v) = \begin{cases} 0 & \text{если } u = v \\ \min\limits_{w \in Neigh_u} (\omega_{uw} + d(w, v)) & \text{иначе} \end{cases} \quad (1)$$

Алгоритм Туэга (обсуждение)

Альтернативное уравнение для построения кратчайших путей:

$$d(u, v) = \begin{cases} 0 & \text{если } u = v \\ \min\limits_{w \in Neigh_u} (\omega_{uw} + d(w, v)) & \text{иначе} \end{cases} \quad (1)$$

Полезны два свойства этого уравнения.

Алгоритм Туэга (обсуждение)

Альтернативное уравнение для построения кратчайших путей:

$$d(u, v) = \begin{cases} 0 & \text{если } u = v \\ \min\limits_{w \in Neigh_u} (\omega_{uw} + d(w, v)) & \text{иначе} \end{cases} \quad (1)$$

Полезны два свойства этого уравнения.

1. Локальность данных. Для вычисления правой части уравнения (1) в узле u требуется только локально доступная информация (а именно, ω_{uw}) или информация, которая имеется у соседей (а именно, $d(w, v)$). Необходимость в обмене данными между удаленными вершинами отпадает.

Алгоритм Туэга (обсуждение)

Альтернативное уравнение для построения кратчайших путей:

$$d(u, v) = \begin{cases} 0 & \text{если } u = v \\ \min\limits_{w \in Neigh_u} (\omega_{uw} + d(w, v)) & \text{иначе} \end{cases} \quad (1)$$

Полезны два свойства этого уравнения.

1. Локальность данных. Для вычисления правой части уравнения (1) в узле u требуется только локально доступная информация (а именно, ω_{uw}) или информация, которая имеется у соседей (а именно, $d(w, v)$). Необходимость в обмене данными между удаленными вершинами отпадает.
2. Независимость от вершин-адресатов. Для вычисления расстояния между вершинами u и v требуется знать только расстояние $d(w, v)$ между v и всеми соседями w вершины u . Поэтому вычисление и анализ всех расстояний до заданной вершины-адресата v_0 можно проводить независимо от вычислений расстояний до других вершин сети.

Алгоритм Мерлина-Сигалла.

Алгоритм, предложенный Мерлином и Сигаллом, вычисляет таблицы маршрутизации по отдельности для каждой вершины-адресата.

Вычисления таблиц для разных вершин-адресатов не оказывают совершенно никакого влияния друг на друга.

Для каждой вершины-адресата v , алгоритм строит дерево T_v с корнем в вершине v , и проводит итеративное преобразование этого дерева с тем чтобы в конце концов получить оптимальное входное дерево для вершины-адресата v .

Алгоритм Мерлина-Сигалла

В каждом узле u хранится оценка расстояния $D_u[v]$ до вершины v , а также имя родителя в дереве T_v , которому нужно передать пакет, адресованный узлу v .

При каждой итерации узел u отправляет сообщение $\langle \textbf{mydist}, v, D_u[v] \rangle$ с известной ему оценку расстояния $D_u[v]$ всем своим соседям кроме вершины $Nb_u[v]$.

Если вершина u получает от своего соседа w сообщение $\langle \textbf{mydist}, v, d \rangle$ и обнаруживает, что $d + \omega_{uw} < D_u[v]$, то она плагает $Nb_u[v]$ равным w , а $D_u[v]$ полагает равным $d + \omega_{uw}$.

Алгоритм Мерлина-Сигалла

В каждом узле u хранится оценка расстояния $D_u[v]$ до вершины v , а также имя родителя в дереве T_v , которому нужно передать пакет, адресованный узлу v .

При каждой итерации узел u отправляет сообщение $\langle \textbf{mydist}, v, D_u[v] \rangle$ с известной ему оценку расстояния $D_u[v]$ всем своим соседям кроме вершины $Nb_u[v]$.

Если вершина u получает от своего соседа w сообщение $\langle \textbf{mydist}, v, d \rangle$ и обнаруживает, что $d + \omega_{uw} < D_u[v]$, то она плагает $Nb_u[v]$ равным w , а $D_u[v]$ полагает равным $d + \omega_{uw}$.

Управляющий параметр преобразования — вершина v . Для его проведения требуется совершить обмен двумя сообщениями, состоящими из W битов, по каждому каналу связи.

Алгоритм Мерлина-Сигалла

В каждом узле u хранится оценка расстояния $D_u[v]$ до вершины v , а также имя родителя в дереве T_v , которому нужно передать пакет, адресованный узлу v .

При каждой итерации узел u отправляет сообщение $\langle \textbf{mydist}, v, D_u[v] \rangle$ с известной ему оценку расстояния $D_u[v]$ всем своим соседям кроме вершины $Nb_u[v]$.

Если вершина u получает от своего соседа w сообщение $\langle \textbf{mydist}, v, d \rangle$ и обнаруживает, что $d + \omega_{uw} < D_u[v]$, то она плагает $Nb_u[v]$ равным w , а $D_u[v]$ полагает равным $d + \omega_{uw}$.

Управляющий параметр преобразования — вершина v . Для его проведения требуется совершить обмен двумя сообщениями, состоящими из W битов, по каждому каналу связи.

После проведения i итераций все кратчайшие пути, состоящие не более чем из i звеньев, будут правильно вычислены. Для вычисления всех кратчайших путей нужно совершить не более N итераций. Кратчайшие пути ко всем вершинам-адресатам вычисляются за счет независимого применения предложенного алгоритма ко всем вершинам-адресатам.

Алгоритм Мерлина-Сигалла

Этот алгоритм можно приспособить к изменениям топологии сети и весовых характеристик каналов.

Важная особенность этого алгоритма состоит в том, что по ходу проведения итераций таблицы маршрутизации остаются ациклическими.

Теорема 5.6. (корректности и сложности алгоритма Мерлина-Сигалла)

Алгоритм Мерлина—Сигалла вычисляет таблицы маршрутизации по кратчайшим путям, совершая при этом обмен $O(N^2)$ сообщениями и передавая $O(N^2 \cdot W)$ битов информации по каждому каналу связи, имея, таким образом, коммуникационную сложность $O(N^2 \cdot |E|)$ и битовую сложность $O(N^2 \cdot |E|W)$.

Алгоритм Мерлина-Сигалла

Задачи.

1. Напишите псевдокод для алгоритма Мерлина–Сигалла. Докажите его корректность. Определите оценку сложности алгоритма Мерлина–Сигалла по числу сообщений и по числу пересылаемых битов.
2. Покажите, что (промежуточные) таблицы маршрутизации, вычисляемые алгоритмом Мерлина-Сигалла, всегда остаются ациклическими на протяжении работы алгоритма.
3. Докажите Теорему 5.6.

Алгоритм Чанди–Мисры

Алгоритм Чанди и Мисры вычисляет все кратчайшие пути к заданной вершине-адресату на основе принципа диффузных вычислений . Суть его такова: распределенное вычисление начинается в одной-единственной вершине, и другие узлы сети поочередно присоединяются к вычислению только после получения сообщения.

Алгоритм Чанди–Мисры

Алгоритм Чанди и Мисры вычисляет все кратчайшие пути к заданной вершине-адресату на основе принципа диффузных вычислений . Суть его такова: распределенное вычисление начинается в одной-единственной вершине, и другие узлы сети поочередно присоединяются к вычислению только после получения сообщения.

Чтобы вычислить расстояние от каждой вершины до заданного узла v_0 (а также предпочтительный исходящий канал связи), каждый узел u устанавливает начальное значение $D_u[v_0] = \infty$ и ожидает получения сообщений.

Алгоритм Чанди–Мисры

Алгоритм Чанди и Мисры вычисляет все кратчайшие пути к заданной вершине-адресату на основе принципа диффузных вычислений . Суть его такова: распределенное вычисление начинается в одной-единственной вершине, и другие узлы сети поочередно присоединяются к вычислению только после получения сообщения.

Чтобы вычислить расстояние от каждой вершины до заданного узла v_0 (а также предпочтительный исходящий канал связи), каждый узел u устанавливает начальное значение $D_u[v_0] = \infty$ и ожидает получения сообщений.

Узел v_0 отправляет сообщение $\langle \textbf{mydist}, v_0, 0 \rangle$ всем своим соседям. Как только вершина u получает сообщение $\langle \textbf{mydist}, v_0, d \rangle$ от соседа w и убеждается в том, что выполняется неравенство $d + \omega_{uw} < D_u[v_0]$, значением переменной $D_u[v_0]$ становится сумма $d + \omega_{uw}$, и всем соседям вершины u отправляется сообщение $\langle \textbf{mydist}, v_0, D_u[v_0] \rangle$.

Алгоритм Чанди–Мисры

var $D_u[v_0]$: вес **init** ∞ ;
$Nb_u[v_0]$: вершина **init** *udef* ;

Только для вершины v_0:

begin $D_{v_0}[v_0] := 0$;
forall $w \in Neigh_{v_0}$ **do** send $\langle$**mydist**, $v_0, 0\rangle$ to w
end

Обработка сообщения $\langle$**mydist**, $v_0, d\rangle$,
полученного вершиной u от соседа w:

$\{ \langle$**mydist**, $v_0, d\rangle \in M_{wu}$ $\}$
begin receive $\langle$**mydist**, $v_0, d\rangle$ from w ;
if $d + \omega_{uw} < D_u[v_0]$ **then**
begin $D_u[v_0] := d + \omega_{uw}$; $Nb_u[v_0] := w$;
forall $x \in Neigh_u$ **do** send $\langle$**mydist**, $v_0, D_u[v_0]\rangle$ to x
end
end

Алгоритм Чанди–Мисры

Задачи.

1. Докажите, что приведенная ниже формула задает инвариант алгоритма Чанди-Мисры вычисления путей, ведущих в вершину v_0.

$$\begin{aligned} & \forall u,\, w : \langle \mathbf{mydist}, v_0, d \rangle \in M_{wu} \quad \Longrightarrow \quad d(w,\, v_0) \leq d \\ \wedge \quad & \forall u : d(u,\, v_0) \leq D_u[v_0] \end{aligned}$$

   Приведите пример выполнения, в котором количество обменов сообщениям экспоненциально зависит от числа каналов в сети.

2. Используя инвариант, докажите, что значение $D_u[v_0]$ всегда является верхней оценкой величины $d(u,\, v_0)$, т.е. $d(u,\, v_0) \leq D_u[v_0]$.

Алгоритм Чанди–Мисры

Задачи.

1. Докажите, что в рамках допущения слабой справедливости о том, что в любом выполнении каждое отправленное сообщение рано или поздно будет получено, в любом вычислении алгоритма Чанди–Мисры достигается такая конфигурация, что в каждом узле u выполняется неравенство $D_u[v_0] \leq d(u, v_0)$. Отсюда следует корректность этого алгоритма
2. Докажите, что Алгоритм Чанди–Мисры вычисляет таблицы маршрутизации с наименьшим числом звеньев совершая при этом обмен $O(N^2)$ сообщениями и передавая $O(N^2 \cdot W)$ битов информации по каждому каналу связи .

Алгоритм Чанди–Мисры

Задачи.

1. В описании алгоритма Чанди–Мисры не указывается, до каких пор должно проводиться вычисление маршрутов в каждом процессе. Докажите, в любой конфигурации любого выполнения алгоритма Чанди–Мисры промежуточные таблицы маршрутизации являются ациклическими. Что нужно добавить к алгоритму Чанди–Мисры, для того чтобы каждый процесс мог узнать, что построение таблиц маршрутизации в сети завершено.

КОНЕЦ ЛЕКЦИИ 5.