

Распределенные алгоритмы

ЛЕКТОР: В.А. Захаров

Лекция 5 (Часть 2).

Алгоритм маршрутизации Netchange.

Описание алгоритма.

Инварианты алгоритма Netchange.

Корректность алгоритма Netchange.

Схрдимость алгоритма Netchange.

Особенности реализации алгоритма.

Другие виды маршрутизации:

интервальная, префиксная, иерархическая.

Алгоритм маршрутизации Netchange

Алгоритм Netchange был предложен Таджибнаписом в 1977 г. Он вычисляет оптимальные по числу звеньев таблицы маршрутизации.

В нем поддерживается дополнительная информация, чтобы в случае возникновения неисправности в канале или восстановления работоспособности канала уточнять таблицы путем их **частичного перевычисления**.

Алгоритм маршрутизации Netchange

Алгоритм Netchange был предложен Таджибнаписом в 1977 г. Он вычисляет оптимальные по числу звеньев таблицы маршрутизации.

В нем поддерживается дополнительная информация, чтобы в случае возникновения неисправности в канале или восстановления работоспособности канала уточнять таблицы путем их **частичного перевычисления**.

При построении алгоритма будем опираться на следующие допущения.

Алгоритм маршрутизации Netchange

Алгоритм Netchange был предложен Таджибнаписом в 1977 г. Он вычисляет оптимальные по числу звеньев таблицы маршрутизации.

В нем поддерживается дополнительная информация, чтобы в случае возникновения неисправности в канале или восстановления работоспособности канала уточнять таблицы путем их **частичного перевычисления**.

При построении алгоритма будем опираться на следующие допущения.

N1. Каждый узел осведомлен о размере всей сети (**N**).

Алгоритм маршрутизации Netchange

Алгоритм Netchange был предложен Таджибнаписом в 1977 г. Он вычисляет оптимальные по числу звеньев таблицы маршрутизации.

В нем поддерживается дополнительная информация, чтобы в случае возникновения неисправности в канале или восстановления работоспособности канала уточнять таблицы путем их **частичного перевычисления**.

При построении алгоритма будем опираться на следующие допущения.

N1. Каждый узел осведомлен о размере всей сети (**N**).

N2. В каналах поддерживается очередность сообщений.

Алгоритм маршрутизации Netchange

Алгоритм Netchange был предложен Таджибнаписом в 1977 г. Он вычисляет оптимальные по числу звеньев таблицы маршрутизации.

В нем поддерживается дополнительная информация, чтобы в случае возникновения неисправности в канале или восстановления работоспособности канала уточнять таблицы путем их **частичного перевычисления**.

При построении алгоритма будем опираться на следующие допущения.

- N1. Каждый узел осведомлен о размере всей сети (N).
- N2. В каналах поддерживается очередность сообщений.
- N3. Узлы получают уведомления о возникновении неисправностей и восстановлении работоспособности примыкающих к ним каналов.

Алгоритм маршрутизации Netchange

Алгоритм Netchange был предложен Таджибнаписом в 1977 г. Он вычисляет оптимальные по числу звеньев таблицы маршрутизации.

В нем поддерживается дополнительная информация, чтобы в случае возникновения неисправности в канале или восстановления работоспособности канала уточнять таблицы путем их **частичного перевычисления**.

При построении алгоритма будем опираться на следующие допущения.

- N1. Каждый узел осведомлен о размере всей сети (N).
- N2. В каналах поддерживается очередность сообщений.
- N3. Узлы получают уведомления о возникновении неисправностей и восстановлении работоспособности примыкающих к ним каналов.
- N4. Стоимость пути равна количеству каналов в этом пути.

Алгоритм маршрутизации Netchange



Алгоритм может справиться с возникновением неисправностей и восстановлением работоспособности каналов, а также с добавлением новых каналов связи, при условии, что всякий узел немедленно узнает о повреждении или восстановлении примыкающих к нему каналов связи.

В каждом узле u алгоритм вычисляет и поддерживает таблицу $Nb_u[v]$, в которой для каждой вершины-адресата v указывается тот сосед u , которому должен быть отправлен пакет, адресованный вершине v .

Алгоритм маршрутизации Netchange

Требования

К алгоритму предъявляются следующие требования.

- R1. Если топология сети после конечного числа изменений остается далее неизменной, то алгоритм завершает работу после конечного числа шагов.
- R2. Когда алгоритм завершает работу, таблицы $Nb_u[v]$ удовлетворяют следующим условиям:
- a) если $v = u$, то $Nb_u[v] = local$;
 - b) если существует путь из вершины u в вершину $v \neq u$, то $Nb_u[v] = w$, где w — это первая вершина-сосед узла u , который следует за u в кратчайшем пути из u в v ;
 - c) если пути из вершины u в вершину v нет, то $Nb_u[v] = undef$

Описание алгоритма Netchange

Структуры данных

- ▶ $Neigh_u$ — множество вершин, соседних с узлом u ;

Описание алгоритма Netchange

Структуры данных

- ▶ $Neigh_u$ — множество вершин, соседних с узлом u ;
- ▶ D_u — целочисленный массив размера $1 \dots N$;
элементы массива $D_u[v]$ — оценки расстояния $d(u, v)$
между узлами u и v ;

Описание алгоритма Netchange

Структуры данных

- ▶ $Neigh_u$ — множество вершин, соседних с узлом u ;
- ▶ D_u — целочисленный массив размера $1 \dots N$;
элементы массива $D_u[v]$ — оценки расстояния $d(u, v)$
между узлами u и v ;
- ▶ Nb_u — массив вершин размера $1 \dots N$;
элементы массива $Nb_u[v]$ — имена соседей узла u ,
которым предпочтительнее отправить пакеты,
адресованные узлу v ;

Описание алгоритма Nchange

Структуры данных

- ▶ $Neigh_u$ — множество вершин, соседних с узлом u ;
- ▶ D_u — целочисленный массив размера $1 \dots N$;
элементы массива $D_u[v]$ — оценки расстояния $d(u, v)$
между узлами u и v ;
- ▶ Nb_u — массив вершин размера $1 \dots N$;
элементы массива $Nb_u[v]$ — имена соседей узла u ,
которым предпочтительнее отправить пакеты,
адресованные узлу v ;
- ▶ $ndis_u$ — двумерный целочисленный массив размера $1 \dots N$;
элементы массива $D_u[w, v]$ — оценки расстояния $d(w, v)$
между узлами w и v .

Описание алгоритма Nchange

Инициализация узла u .

```
begin forall  $w \in Neigh_u, v \in V$  do  $ndis_u[w, v] := N$  ;  
  forall  $v \in V$  do  
    begin  $D_u[v] := N$  ;  $Nb_u[v] := undef$  end;  
     $D_u[u] := 0$  ;  $Nb_u[u] := local$  ;  
    forall  $w \in Neigh_u$  do send  $\langle \mathbf{mydist}, u, 0 \rangle$  to  $w$   
  end
```

Начиная работу, каждый узел u знает, что

Описание алгоритма Nchange

Инициализация узла u .

```
begin forall  $w \in Neigh_u, v \in V$  do  $ndis_u[w, v] := N$  ;  
  forall  $v \in V$  do  
    begin  $D_u[v] := N$  ;  $Nb_u[v] := undef$  end;  
     $D_u[u] := 0$  ;  $Nb_u[u] := local$  ;  
    forall  $w \in Neigh_u$  do send  $\langle mydist, u, 0 \rangle$  to  $w$   
  end
```

Начиная работу, каждый узел u знает, что

1) кратчайший маршрут из u в u имеет длину 0 ;

Описание алгоритма Nchange



Инициализация узла u .

```
begin forall  $w \in Neigh_u, v \in V$  do  $ndis_u[w, v] := N$  ;  
  forall  $v \in V$  do  
    begin  $D_u[v] := N$  ;  $Nb_u[v] := undef$  end;  
     $D_u[u] := 0$  ;  $Nb_u[u] := local$  ;  
    forall  $w \in Neigh_u$  do send  $\langle mydist, u, 0 \rangle$  to  $w$   
  end
```

Начиная работу, каждый узел u знает, что

- 1) кратчайший маршрут из u в u имеет длину 0 ;
- 2) размер сети N является верхней оценкой длины кратчайшего маршрута из u в любой другой узел сети.

Описание алгоритма Nchange

Процедура $Update(v)$

```
begin if  $v = u$ 
    then begin  $D_u[v] := 0$  ;  $Nb_u[v] := local$  end
    else begin (* Оценить расстояние до вершины  $v$  *)
         $d := 1 + \min\{ndis_u[w, v] : w \in Neigh_u\}$  ;
        choose  $w$ :  $d = 1 + ndis_u[u, v]$ ;
        if  $d < N$  then begin  $D_u[v] := d$ ;  $Nb_u[v] := w$  end
        else begin  $D_u[v] := N$  ;  $Nb_u[v] := undef$  end
    end;
    if переменная  $D_u[v]$  изменила значение then
        forall  $x \in Neigh_u$  do send  $\langle mydist, v, D_u[v] \rangle$  to  $x$ 
end
```

Вычисление оценок $D_u[v]$ проводится следующим образом.

Если $u = v$, то $d(u, v) = 0$, и в этом случае значение $D_u[v]$ полагается равным 0 .

Описание алгоритма Nchange

Процедура *Update(v)*

begin if $v = u$

then begin $D_u[v] := 0$; $Nb_u[v] := local$ **end**

else begin (* Оценить расстояние до вершины v *)

$d := 1 + \min\{ndis_u[w, v] : w \in Neigh_u\}$;

choose w : $d = 1 + ndis_u[u, v]$;

if $d < N$ **then begin** $D_u[v] := d$; $Nb_u[v] := w$ **end**

else begin $D_u[v] := N$; $Nb_u[v] := undef$ **end**

end;

if переменная $D_u[v]$ изменила значение **then**

forall $x \in Neigh_u$ **do** send $\langle mydist, v, D_u[v] \rangle$ to x

end

Если $u \neq v$, то кратчайший путь из u в v состоит из канала, соединяющего u с одним из соседей w , и кратчайшего пути из w в v . Поэтому $d(u, v) = 1 + \min_{w \in Neigh_u} d(w, v)$ — это оценка величины $d(u, v)$ в узле $u \neq v$ за счет применения данной формулы к оценкам величин $d(w, v)$, содержащимся в

Описание алгоритма Nchange

Процедура *Update(v)*

begin if $v = u$

then begin $D_u[v] := 0$; $Nb_u[v] := local$ end

else begin (* Оценить расстояние до вершины v *)

$d := 1 + \min\{ndis_u[w, v] : w \in Neigh_u\}$;

choose w : $d = 1 + ndis_u[u, v]$;

if $d < N$ then begin $D_u[v] := d$; $Nb_u[v] := w$ end

else begin $D_u[v] := N$; $Nb_u[v] := undef$ end

end;

if переменная $D_u[v]$ изменила значение then

forall $x \in Neigh_u$ do send $\langle mydist, v, D_u[v] \rangle$ to x

end

Коль скоро в сети имеется N узлов, длина пути с наименьшим числом звеньев не превосходит $N - 1$. Если же вычисленная оценка длины не меньше чем N , то можно предполагать, что такого пути вообще не существует; именно для этой цели в таблице используется значение N .

Описание алгоритма Nchange

Процедура *Update(v)*

```
begin if  $v = u$ 
    then begin  $D_u[v] := 0$  ;  $Nb_u[v] := local$  end
    else begin (* Оценить расстояние до вершины  $v$  *)
         $d := 1 + \min\{ndis_u[w, v] : w \in Neigh_u\}$  ;
        choose  $w$ :  $d = 1 + ndis_u[u, v]$ ;
        if  $d < N$  then begin  $D_u[v] := d$  ;  $Nb_u[v] := w$  end
        else begin  $D_u[v] := N$  ;  $Nb_u[v] := undef$  end
    end;
    if переменная  $D_u[v]$  изменила значение then
        forall  $x \in Neigh_u$  do send  $\langle mydist, v, D_u[v] \rangle$  to  $x$ 
end
```

Если оценка расстояния от узла u до узла v изменилась (т.е. уменьшилась), то узел u оповещает об этом всех своих соседей, чтобы те могли уточнить свои оценки.

Описание алгоритма Nchange

Proc-1 : Обработка сообщения $\langle \mathbf{mydist}, v, d \rangle$ от соседа w :

```
{ Сообщение  $\langle \mathbf{mydist}, v, d \rangle$  в начале очереди  $Q_{wu}$  }  
begin receive  $\langle \mathbf{mydist}, v, d \rangle$  from  $w$  ;  
          $ndis_u[w, v] := d$  ; Update ( $v$ )  
end
```

После получения сообщения $\langle \mathbf{mydist}, v, d \rangle$ от соседа w в узле u переменной $ndis_u[w, v]$ присваивается значение d . В результате изменения значения переменной $ndis_u[w, v]$ оценка $d(u, v)$ в узле u может измениться, и поэтому эта оценка перевычисляется всякий раз, когда происходят изменения в таблице $ndis_u$. Если оценка расстояния и в самом деле изменяется, например становится равной d' , то об этом оповещаются все соседи при помощи сообщений $\langle \mathbf{mydist}, v, d' \rangle$.

Описание алгоритма Netchange

В ответ на выход из строя или восстановление работоспособности канала связи алгоритм вносит изменения в локальные таблицы и отправляет сообщение типа **mydist**, если при этом изменяются оценки расстояний.

Предполагается, что уведомления о поломках и починках каналов связи (допущение N3) поступают в виде сообщений типа **fail** и **repair**. Канал связи между узлами u_1 и u_2 моделируется парой очередей $Q_{u_1 u_2}$ и $Q_{u_2 u_1}$.

Описание алгоритма Netchange

В ответ на выход из строя или восстановление работоспособности канала связи алгоритм вносит изменения в локальные таблицы и отправляет сообщение типа **mydist**, если при этом изменяются оценки расстояний.

Предполагается, что уведомления о поломках и починках каналов связи (допущение N3) поступают в виде сообщений типа **fail** и **repair**. Канал связи между узлами u_1 и u_2 моделируется парой очередей $Q_{u_1 u_2}$ и $Q_{u_2 u_1}$.

Когда канал выходит из строя, эти очереди удаляются из конфигурации (что немедленно влечет за собой потерю всех сообщений в обеих очередях), а узлы по обе стороны канала связи получают сообщение типа **fail**.

Описание алгоритма Netchange

В ответ на выход из строя или восстановление работоспособности канала связи алгоритм вносит изменения в локальные таблицы и отправляет сообщение типа **mydist**, если при этом изменяются оценки расстояний.

Предполагается, что уведомления о поломках и починках каналов связи (допущение N3) поступают в виде сообщений типа **fail** и **repair**. Канал связи между узлами u_1 и u_2 моделируется парой очередей $Q_{u_1 u_2}$ и $Q_{u_2 u_1}$.

Когда канал выходит из строя, эти очереди удаляются из конфигурации (что немедленно влечет за собой потерю всех сообщений в обеих очередях), а узлы по обе стороны канала связи получают сообщение типа **fail**.

Когда связь в канале восстанавливается (или в сети появляется новый канал связи), в конфигурации возникают две новые пустые очереди, а узлы по обе стороны этого канала связи получают сообщение типа **repair**.

Описание алгоритма Nchange

Proc-2 : В случае выхода из строя канала uw :
begin receive $\langle \text{fail}, w \rangle$; $Neigh_u := Neigh_u \setminus \{w\}$;
 forall $v \in V$ **do** $Update(v)$
end

Когда выходит из строя канал связи между узлами u и w ,
вершина w удаляется из списка $Neigh_u$.

Проводится перевычисление оценок расстояний до каждой
вершины, и в случае изменения оценки, об этом узнают все
оставшиеся соседи.

Это происходит, если неисправный канал ранее был одним из
звеньев наилучшего маршрута и не осталось ни одного соседа
 w' , для которого верно равенство $ndis_u[w', v] = ndis_u[w, v]$.

Описание алгоритма Nchange

Proc-3 : В случае восстановления канала uw :

```
begin receive  $\langle \text{repair}, w \rangle$  ;  $Neigh_u := Neigh_u \cup \{w\}$  ;  
    forall  $v \in V$  do  
        begin  $ndis_u[w, v] := N$ ; send  $\langle \text{mydist}, v, D_u[v] \rangle$  to  $w$  end  
end
```

Когда канал связи восстанавливается, вершина w добавляется к списку вершин $Neigh_u$, но в памяти узла u еще нет никаких оценок расстояния $d(w, v)$.

Новому соседу w немедленно сообщаются оценки $D_u[v]$ расстояний от узла u до всех возможных вершин-адресатов v (путем отправления сообщений $\langle \text{mydist}, v, D_u[v] \rangle$).

До тех пор пока узел u не получит такого сообщения от узла w , величина N будет использоваться в нем для оценки расстояния $d(w, v)$, т. е. значение переменной $ndis_u[w, v]$ полагается равным N .

Описание алгоритма Nchange

Вычисление алгоритма завершается, когда в каналах связи не остается ни одного сообщения, находящегося на этапе пересылки.

Конфигурации такого вида не являются заключительными конфигурациями для всей системы в целом, потому что вычисление системы может быть продолжено, после того как какой-нибудь канал выйдет из строя или восстановит свою работоспособность (что потребует реакции со стороны алгоритма).

Конфигурацию, не имеющую в каналах связи ни одного сообщения, мы будем называть **стабильной**. Для описания этого явления введем два предиката:

$up(u, w) \equiv$ канал связи между узлами u, w исправен

$stable \equiv \forall u, w : up(u, w) \Rightarrow Q_{wu}$ не содержит сообщений **mydist**.

Инварианты алгоритма Nchange

Чтобы обосновать корректность алгоритма Nchange относительно требований R1 и R2 нам понадобятся (как вы думаете что?)

Инварианты алгоритма Nchange

Чтобы обосновать корректность алгоритма Nchange относительно требований R1 и R2 нам понадобятся (как вы думаете что?) **ИНВАРИАНТЫ** !

$$P(u, w, v) \equiv up(u, w) \iff w \in Neigh_u \quad (1)$$

$$\begin{aligned} \wedge \\ up(u, w) \wedge Q_{wu} \text{ содержит сообщение } \langle \mathbf{mydist}, v, d \rangle \\ \implies \text{последнее сообщение в очереди} \\ \text{удовлетворяет равенству } d = D_w[v] \end{aligned} \quad (2)$$

$$\begin{aligned} \wedge \\ up(u, w) \wedge Q_{wu} \text{ не содержит сообщений } \langle \mathbf{mydist}, v, d \rangle \\ \implies ndis_u[w, v] = D_w[v] \end{aligned} \quad (3)$$

Формула $P(u, w, v)$ утверждает: если процесс u завершил обработку сообщений типа **mydist**, полученных от процесса w , то оценка расстояния $d(w, v)$ в узле u совпадает с оценкой расстояния $d(w, v)$ в узле w .

Инварианты алгоритма Netchange

Лемма 1.

Для любой тройки вершин u_0, w_0, v_0 формула $P(u_0, w_0, v_0)$ является инвариантом.

Инварианты алгоритма Netchange

Лемма 1.

Для любой тройки вершин u_0, w_0, v_0 формула $P(u_0, w_0, v_0)$ является инвариантом.

Доказательство.

Будем считать, что вначале все списки $Neigh_u$ правильно отражают работоспособность каналов связи между узлами, т. е. соотношение (1) считается верным.

Инварианты алгоритма Netchange

Лемма 1.

Для любой тройки вершин u_0, w_0, v_0 формула $P(u_0, w_0, v_0)$ является инвариантом.

Доказательство.

Будем считать, что вначале все списки $Neigh_u$ правильно отражают работоспособность каналов связи между узлами, т. е. соотношение (1) считается верным.

Нужно рассмотреть инициализацию и три типа процедур:

1. **Proc-1** : прием сообщения типа **mydist**. Происходит прием одного сообщения и (возможно) отправлен несколько сообщений.
2. **Proc-2** : обрыв канала и обработка сообщения типа **fail** в узлах по обе стороны канала.
3. **Proc-3** : восстановление канала и обработка сообщения типа **repair** в обоих узлах, соединенных этим каналом.

```

begin forall  $w \in Neigh_u, v \in V$  do  $ndis_u[w, v] := N$  ;
    forall  $v \in V$  do
        begin  $D_u[v] := N$  ;  $Nb_u[v] := undef$  end;
         $D_u[u] := 0$  ;  $Nb_u[u] := local$  ;
        forall  $w \in Neigh_u$  do send  $\langle \mathbf{mydist}, u, 0 \rangle$  to  $w$ 
    end
end

```

$$up(u_0, w_0) \iff w_0 \in Neigh_{u_0} \quad (1)$$

$$\begin{aligned}
 & up(u_0, w_0) \wedge Q_{w_0 u_0} \text{ содержит сообщение } \langle \mathbf{mydist}, v_0, d \rangle \\
 & \implies \text{последнее сообщение в очереди} \\
 & \quad \text{удовлетворяет равенству } d = D_{w_0}[v_0] \quad (2)
 \end{aligned}$$

$$\begin{aligned}
 & up(u_0, w_0) \wedge Q_{w_0 u_0} \text{ не содержит сообщений } \langle \mathbf{mydist}, v_0, d \rangle \\
 & \implies ndis_{u_0}[w_0, v_0] = D_{w_0}[v_0] \quad (3)
 \end{aligned}$$

В самом начале после выполнения процедуры инициализации в каждом узле соотношение (1) выполняется согласно сделанному допущению о списках $Neigh_u$.

```

begin forall  $w \in Neigh_u, v \in V$  do  $ndis_u[w, v] := N$  ;
    forall  $v \in V$  do
        begin  $D_u[v] := N$  ;  $Nb_u[v] := undef$  end;
         $D_u[u] := 0$  ;  $Nb_u[u] := local$  ;
        forall  $w \in Neigh_u$  do send  $\langle \mathbf{mydist}, u, 0 \rangle$  to  $w$ 
    end
end

```

$$up(u_0, w_0) \iff w_0 \in Neigh_{u_0} \quad (1)$$

$$\begin{aligned}
 & up(u_0, w_0) \wedge Q_{w_0 u_0} \text{ содержит сообщение } \langle \mathbf{mydist}, v_0, d \rangle \\
 & \implies \text{последнее сообщение в очереди} \\
 & \quad \text{удовлетворяет равенству } d = D_{w_0}[v_0] \quad (2)
 \end{aligned}$$

$$\begin{aligned}
 & up(u_0, w_0) \wedge Q_{w_0 u_0} \text{ не содержит сообщений } \langle \mathbf{mydist}, v_0, d \rangle \\
 & \implies ndis_{u_0}[w_0, v_0] = D_{w_0}[v_0] \quad (3)
 \end{aligned}$$

Если первоначально выполняется условие $\neg up(u_0, w_0)$, то соотношения (2) и (3), очевидно, выполняются.

```

begin forall  $w \in Neigh_u$ ,  $v \in V$  do  $ndis_u[w, v] := N$  ;
    forall  $v \in V$  do
        begin  $D_u[v] := N$  ;  $Nb_u[v] := undef$  end;
         $D_u[u] := 0$  ;  $Nb_u[u] := local$  ;
        forall  $w \in Neigh_u$  do send  $\langle \mathbf{mydist}, u, 0 \rangle$  to  $w$ 
    end
end

```

$$up(u_0, w_0) \iff w_0 \in Neigh_{u_0} \quad (1)$$

$$\begin{aligned}
 & up(u_0, w_0) \wedge Q_{w_0 u_0} \text{ содержит сообщение } \langle \mathbf{mydist}, v_0, d \rangle \\
 & \implies \text{последнее сообщение в очереди} \\
 & \quad \text{удовлетворяет равенству } d = D_{w_0}[v_0] \quad (2)
 \end{aligned}$$

$$\begin{aligned}
 & up(u_0, w_0) \wedge Q_{w_0 u_0} \text{ не содержит сообщений } \langle \mathbf{mydist}, v_0, d \rangle \\
 & \implies ndis_{u_0}[w_0, v_0] = D_{w_0}[v_0] \quad (3)
 \end{aligned}$$

Если же вначале выполняется условие $up(u_0, w_0)$, то $w_0 \in Neigh_{u_0}$ согласно (1), и поэтому после инициализации $ndis_{u_0}[w_0, v_0] = N$.

```

begin forall  $w \in Neigh_u$ ,  $v \in V$  do  $ndis_u[w, v] := N$  ;
  forall  $v \in V$  do
    begin  $D_u[v] := N$  ;  $Nb_u[v] := undef$  end;
     $D_u[u] := 0$  ;  $Nb_u[u] := local$  ;
    forall  $w \in Neigh_u$  do send  $\langle \mathbf{mydist}, u, 0 \rangle$  to  $w$ 
  end
end

```

$$up(u_0, w_0) \iff w_0 \in Neigh_{u_0} \quad (1)$$

$$\begin{aligned}
& up(u_0, w_0) \wedge Q_{w_0 u_0} \text{ содержит сообщение } \langle \mathbf{mydist}, v_0, d \rangle \\
& \implies \text{последнее сообщение в очереди} \\
& \quad \text{удовлетворяет равенству } d = D_{w_0}[v_0] \quad (2)
\end{aligned}$$

$$\begin{aligned}
& up(u_0, w_0) \wedge Q_{w_0 u_0} \text{ не содержит сообщений } \langle \mathbf{mydist}, v_0, d \rangle \\
& \implies ndis_{u_0}[w_0, v_0] = D_{w_0}[v_0] \quad (3)
\end{aligned}$$

Если $w_0 = v_0$, то $D_{w_0}[w_0] = 0$, но при этом в очереди $Q_{w_0 u_0}$ содержится сообщение $\langle \mathbf{mydist}, v_0, 0 \rangle$, и поэтому верны соотношения (2) и (3).

```

begin forall  $w \in Neigh_u, v \in V$  do  $ndis_u[w, v] := N$  ;
    forall  $v \in V$  do
        begin  $D_u[v] := N$  ;  $Nb_u[v] := undef$  end;
         $D_u[u] := 0$  ;  $Nb_u[u] := local$  ;
        forall  $w \in Neigh_u$  do send  $\langle mydist, u, 0 \rangle$  to  $w$ 
    end
end

```

$$up(u_0, w_0) \iff w_0 \in Neigh_{u_0} \quad (1)$$

$$\begin{aligned}
 & up(u_0, w_0) \wedge Q_{w_0 u_0} \text{ содержит сообщение } \langle mydist, v_0, d \rangle \\
 & \implies \text{последнее сообщение в очереди} \\
 & \text{удовлетворяет равенству } d = D_{w_0}[v_0] \quad (2)
 \end{aligned}$$

$$\begin{aligned}
 & up(u_0, w_0) \wedge Q_{w_0 u_0} \text{ не содержит сообщений } \langle mydist, v_0, d \rangle \\
 & \implies ndis_{u_0}[w_0, v_0] = D_{w_0}[v_0] \quad (3)
 \end{aligned}$$

Если $w_0 \neq v_0$, то $D_{w_0}[v_0] = N$ и в очереди канала $(w_0 u_0)$ нет сообщений об узле v_0 , поэтому соотношения (2) и (3) также будут верны.

Proc-1 : Обработка сообщения $\langle \mathbf{mydist}, v, d \rangle$ от соседа w :

{ Сообщение $\langle \mathbf{mydist}, v, d \rangle$ в начале очереди Q_{wu} }

begin receive $\langle \mathbf{mydist}, v, d \rangle$ from w ;

$ndis_u[w, v] := d$; $Update(v)$

end

$$up(u_0, w_0) \iff w_0 \in Neigh_{u_0} \quad (1)$$

$$\begin{aligned} up(u_0, w_0) \wedge Q_{w_0 u_0} \text{ содержит сообщение } \langle \mathbf{mydist}, v_0, d \rangle \\ \implies \text{ последнее сообщение в очереди} \\ \text{удовлетворяет равенству } d = D_{w_0}[v_0] \end{aligned} \quad (2)$$

$$\begin{aligned} up(u_0, w_0) \wedge Q_{w_0 u_0} \text{ не содержит сообщений } \langle \mathbf{mydist}, v_0, d \rangle \\ \implies ndis_u[w_0, v_0] = D_{w_0}[v_0] \end{aligned} \quad (3)$$

Предположим, что узел u получает сообщение $\langle \mathbf{mydist}, v, d \rangle$ от узла w .

Это не влечет за собой никаких изменений топологии сети, и поэтому списки вершин-соседей $Neigh$ не изменяются, и соотношение (1) остается верным.

Если $v \neq v_0$, то с получением этого сообщения в формуле $P(u_0, w_0, v_0)$ ничего не изменяется.

Proc-1 : Обработка сообщения $\langle \mathbf{mydist}, v, d \rangle$ от соседа w :
 $\{ \text{Сообщение } \langle \mathbf{mydist}, v, d \rangle \text{ в начале очереди } Q_{wu} \}$
begin receive $\langle \mathbf{mydist}, v, d \rangle$ from w ;
 $ndis_u[w, v] := d$; $Update(v)$
end

$$up(u_0, w_0) \iff w_0 \in Neigh_{u_0} \quad (1)$$

$$\begin{aligned} up(u_0, w_0) \wedge Q_{w_0 u_0} \text{ содержит сообщение } \langle \mathbf{mydist}, v_0, d \rangle \\ \implies \text{последнее сообщение в очереди} \\ \text{удовлетворяет равенству } d = D_{w_0}[v_0] \end{aligned} \quad (2)$$

$$\begin{aligned} up(u_0, w_0) \wedge Q_{w_0 u_0} \text{ не содержит сообщений } \langle \mathbf{mydist}, v_0, d \rangle \\ \implies ndis_{u_0}[w_0, v_0] = D_{w_0}[v_0] \end{aligned} \quad (3)$$

Если $v = v_0, u = u_0, w = w_0$, то значение $ndis_{u_0}[w_0, v_0]$ может измениться.

Если $Q_{w_0 u_0}$ содержит еще одно сообщение об узле v_0 , то значения, содержащиеся в этом сообщении, удовлетворяют (2).

Соотношения (3) сохраняется, т.к. его предпосылка ложна.

Если в $Q_{w_0 u_0}$ больше нет сообщений об узле v_0 , то согласно (2) верно $d = D_{w_0}[v_0]$, и (3) становится верным. Т.к. предпосылка утверждения (2) становится ложной, (2) остается верным.

Proc-1 : Обработка сообщения $\langle \mathbf{mydist}, v, d \rangle$ от соседа w :

{ Сообщение $\langle \mathbf{mydist}, v, d \rangle$ в начале очереди Q_{wu} }

begin receive $\langle \mathbf{mydist}, v, d \rangle$ from w ;

$ndis_u[w, v] := d$; **Update** (v)

end

$$up(u_0, w_0) \iff w_0 \in Neigh_{u_0} \quad (1)$$

$$\begin{aligned} up(u_0, w_0) \wedge Q_{w_0 u_0} \text{ содержит сообщение } \langle \mathbf{mydist}, v_0, d \rangle \\ \implies \text{последнее сообщение в очереди} \\ \text{удовлетворяет равенству } d = D_{w_0}[v_0] \end{aligned} \quad (2)$$

$$\begin{aligned} up(u_0, w_0) \wedge Q_{w_0 u_0} \text{ не содержит сообщений } \langle \mathbf{mydist}, v_0, d \rangle \\ \implies ndis_{u_0}[w_0, v_0] = D_{w_0}[v_0] \end{aligned} \quad (3)$$

Если $v = v_0, u = w_0$ (и u_0 — сосед u), то заключения утверждений (2) или (3) могут стать ложными в том случае, когда значение $D_{w_0}[v_0]$ изменяется в результате выполнения процедуры $Update(v)$ в узле w_0 .

Процедура $Update(v)$

begin if $v = u$

...

if переменная $D_u[v]$ изменила значение then

forall $x \in Neigh_u$ do send $\langle \mathbf{mydist}, v, D_u[v] \rangle$ to x

end

$$up(u_0, w_0) \iff w_0 \in Neigh_{u_0} \quad (1)$$

$$\begin{aligned} up(u_0, w_0) \wedge Q_{w_0 u_0} \text{ содержит сообщение } \langle \mathbf{mydist}, v_0, d \rangle \\ \implies \text{последнее сообщение в очереди} \\ \text{удовлетворяет равенству } d = D_{w_0}[v_0] \end{aligned} \quad (2)$$

$$\begin{aligned} up(u_0, w_0) \wedge Q_{w_0 u_0} \text{ не содержит сообщений } \langle \mathbf{mydist}, v_0, d \rangle \\ \implies ndis_{u_0}[w_0, v_0] = D_{w_0}[v_0] \end{aligned} \quad (3)$$

Но изменение значения $D_{w_0}[v_0]$ процедурой $Update(v_0)$ в узле w_0 сопровождается отправкой сообщения $\langle \mathbf{mydist}, v_0, .. \rangle$ с новым значением узлу u_0 . После этого предпосылка (3) становится ложной, а заключение (2) — истинным.

Это единственный случай, когда сообщение $\langle \mathbf{mydist}, v_0, .. \rangle$ становится в очередь $Q_{w_0 u_0}$. Тогда $d = D_{w_0}[v_0]$.

Процедура $Update(v)$

begin if $v = u$

...

if переменная $D_u[v]$ изменила значение **then**

forall $x \in Neigh_u$ **do** send $\langle \mathbf{mydist}, v, D_u[v] \rangle$ to x

end

$$up(u_0, w_0) \iff w_0 \in Neigh_{u_0} \quad (1)$$

$$\begin{aligned} up(u_0, w_0) \wedge Q_{w_0 u_0} \text{ содержит сообщение } \langle \mathbf{mydist}, v_0, d \rangle \\ \implies \text{последнее сообщение в очереди} \\ \text{удовлетворяет равенству } d = D_{w_0}[v_0] \end{aligned} \quad (2)$$

$$\begin{aligned} up(u_0, w_0) \wedge Q_{w_0 u_0} \text{ не содержит сообщений } \langle \mathbf{mydist}, v_0, d \rangle \\ \implies ndis_{u_0}[w_0, v_0] = D_{w_0}[v_0] \end{aligned} \quad (3)$$

Если $v = v_0$ и $u \neq u_0, w_0$, то в формуле $P(u_0, w_0, v_0)$ ничего не изменяется.

Proc-2 : В случае выхода из строя канала uw :
begin receive $\langle \text{fail}, w \rangle$; $Neigh_u := Neigh_u \setminus \{w\}$;
 forall $v \in V$ **do** $Update(v)$
end

$$up(u_0, w_0) \iff w_0 \in Neigh_{u_0} \quad (1)$$

$$up(u_0, w_0) \wedge Q_{w_0 u_0} \text{ содержит сообщение } \langle \text{mydist}, v_0, d \rangle \\ \implies \text{последнее сообщение в очереди} \\ \text{удовлетворяет равенству } d = D_{w_0}[v_0] \quad (2)$$

$$up(u_0, w_0) \wedge Q_{w_0 u_0} \text{ не содержит сообщений } \langle \text{mydist}, v_0, d \rangle \\ \implies ndis_{u_0}[w_0, v_0] = D_{w_0}[v_0] \quad (3)$$

Предположим, что канал (uw) выходит из строя.

Если $u = u_0, w = w_0$, то возникшая неисправность канала приводит к тому, что предпосылки (2) и (3) становятся ложными, и поэтому выполнимость этих утверждений сохраняется. Выполнимость (1) сохраняется, ввиду того что w_0 удаляется из списка $Neigh_{u_0}$.

То же самое имеет место и в случае $u = w_0$ и $w = u_0$.

Proc-2 : В случае выхода из строя канала uw :

```

begin receive  $\langle \text{fail}, w \rangle$  ;  $Neigh_u := Neigh_u \setminus \{w\}$  ;
      forall  $v \in V$  do Update( $v$ )
end

```

$$up(u_0, w_0) \iff w_0 \in Neigh_{u_0} \quad (1)$$

$$\begin{aligned}
 & up(u_0, w_0) \wedge Q_{w_0 u_0} \text{ содержит сообщение } \langle \text{mydist}, v_0, d \rangle \\
 & \implies \text{последнее сообщение в очереди} \\
 & \quad \text{удовлетворяет равенству } d = D_{w_0}[v_0] \quad (2)
 \end{aligned}$$

$$\begin{aligned}
 & up(u_0, w_0) \wedge Q_{w_0 u_0} \text{ не содержит сообщений } \langle \text{mydist}, v_0, d \rangle \\
 & \implies ndis_{u_0}[w_0, v_0] = D_{w_0}[v_0] \quad (3)
 \end{aligned}$$

Если $u = w_0$, но $w \neq u_0$, то заключения (2) и (3) могут стать ложными, поскольку *Update*(v_0) в узле w_0 может изменить значение $D_{w_0}[v_0]$.

Процедура *Update(v)*

begin if $v = u$

...

if переменная $D_u[v]$ изменила значение then

forall $x \in Neigh_u$ do send $\langle \mathbf{mydist}, v, D_u[v] \rangle$ to x

end

$$up(u_0, w_0) \iff w_0 \in Neigh_{u_0} \quad (1)$$

$$\begin{aligned} up(u_0, w_0) \wedge Q_{w_0 u_0} \text{ содержит сообщение } \langle \mathbf{mydist}, v_0, d \rangle \\ \implies \text{последнее сообщение в очереди} \\ \text{удовлетворяет равенству } d = D_{w_0}[v_0] \end{aligned} \quad (2)$$

$$\begin{aligned} up(u_0, w_0) \wedge Q_{w_0 u_0} \text{ не содержит сообщений } \langle \mathbf{mydist}, v_0, d \rangle \\ \implies ndis_{u_0}[w_0, v_0] = D_{w_0}[v_0] \end{aligned} \quad (3)$$

Но w_0 отправляет сообщения $\langle \mathbf{mydist}, v_0, .. \rangle$, и предпосылка (3) становится ложной, а заключение (2) становится истинным.

Поэтому (2) и (3) сохраняются.

Proc-2 : В случае выхода из строя канала uw :

```

begin receive  $\langle \text{fail}, w \rangle$  ;  $Neigh_u := Neigh_u \setminus \{w\}$  ;
      forall  $v \in V$  do Update( $v$ )
end

```

$$up(u_0, w_0) \iff w_0 \in Neigh_{u_0} \quad (1)$$

$$\begin{aligned}
 & up(u_0, w_0) \wedge Q_{w_0 u_0} \text{ содержит сообщение } \langle \text{mydist}, v_0, d \rangle \\
 & \implies \text{последнее сообщение в очереди} \\
 & \quad \text{удовлетворяет равенству } d = D_{w_0}[v_0] \quad (2)
 \end{aligned}$$

$$\begin{aligned}
 & up(u_0, w_0) \wedge Q_{w_0 u_0} \text{ не содержит сообщений } \langle \text{mydist}, v_0, d \rangle \\
 & \implies ndis_{u_0}[w_0, v_0] = D_{w_0}[v_0] \quad (3)
 \end{aligned}$$

Во всех остальных случаях в $P(u_0, w_0, v_0)$ не происходит изменений.

Proc-3 : В случае восстановления канала uw :

```
begin receive  $\langle \text{repair}, w \rangle$  ;  $Neigh_u := Neigh_u \cup \{w\}$  ;  
  forall  $v \in V$  do  
    begin  $ndis_u[w, v] := N$ ; send  $\langle \text{mydist}, v, D_u[v] \rangle$  to  $w$  end  
end
```

$$up(u_0, w_0) \iff w_0 \in Neigh_{u_0} \quad (1)$$

$$\begin{aligned} up(u_0, w_0) \wedge Q_{w_0 u_0} \text{ содержит сообщение } \langle \text{mydist}, v_0, d \rangle \\ \implies \text{последнее сообщение в очереди} \\ \text{удовлетворяет равенству } d = D_{w_0}[v_0] \end{aligned} \quad (2)$$

$$\begin{aligned} up(u_0, w_0) \wedge Q_{w_0 u_0} \text{ не содержит сообщений } \langle \text{mydist}, v_0, d \rangle \\ \implies ndis_{u_0}[w_0, v_0] = D_{w_0}[v_0] \end{aligned} \quad (3)$$

Предположим, что в сети появился канал (uw) .

Если $u = u_0, w = w_0$, то предикат $up(u_0, w_0)$ принимает значение true, но ввиду того что вершина w_0 добавляется в список $Neigh_{u_0}$ (а вершина u_0 — в список $Neigh_{w_0}$), выполнимость утверждения (1) сохраняется.

Proc-3 : В случае восстановления канала uw :

```
begin receive  $\langle \text{repair}, w \rangle$  ;  $Neigh_u := Neigh_u \cup \{w\}$  ;  
  forall  $v \in V$  do  
    begin  $ndis_u[w, v] := N$ ; send  $\langle \text{mydist}, v, D_u[v] \rangle$  to  $w$  end  
end
```

$$up(u_0, w_0) \iff w_0 \in Neigh_{u_0} \quad (1)$$

$$\begin{aligned} up(u_0, w_0) \wedge Q_{w_0 u_0} \text{ содержит сообщение } \langle \text{mydist}, v_0, d \rangle \\ \implies \text{последнее сообщение в очереди} \\ \text{удовлетворяет равенству } d = D_{w_0}[v_0] \end{aligned} \quad (2)$$

$$\begin{aligned} up(u_0, w_0) \wedge Q_{w_0 u_0} \text{ не содержит сообщений } \langle \text{mydist}, v_0, d \rangle \\ \implies ndis_u[w_0, v_0] = D_{w_0}[v_0] \end{aligned} \quad (3)$$

После отправления узлом w_0 сообщения $\langle \text{mydist}, v_0, D_{w_0}[v_0] \rangle$ заключение утверждения (2) становится истинным, а предпосылка утверждения (3) становится ложной, и поэтому выполнимость формулы $P(u_0, w_0, v_0)$ сохраняется. Во всех остальных случаях в формуле $P(u_0, w_0, v_0)$ не происходит никаких изменений.



Инварианты алгоритма Netchange

Инвариант $P(u_0, w_0, v_0)$ подтверждает согласованность данных в разных узлах сети по ходу вычисления алгоритма.

Нам понадобится еще один инвариант, который подтверждает согласованность данных в каждом отдельном узле сети.

Инварианты алгоритма Nchange

Инвариант $P(u_0, w_0, v_0)$ подтверждает согласованность данных в разных узлах сети по ходу вычисления алгоритма.

Нам понадобится еще один инвариант, который подтверждает согласованность данных в каждом отдельном узле сети.

$$L(u, v) \equiv u = v \implies (D_u[v] = 0 \wedge Nb_u[v] = local) \quad (4)$$

$\wedge$

$$(u \neq v \wedge \exists w \in Neigh_u : ndis_u[w, v] < N - 1) \\ \implies (D_u[v] = 1 + \min_{w \in Neigh_u} ndis_u[w, v] = 1 + ndis_u[Nb_u[v], v]) \quad (5)$$

$\wedge$

$$(u \neq v \wedge \forall w \in Neigh_u : ndis_u[w, v] \geq N - 1) \\ \implies (D_u[v] = N \wedge Nb_u[v] = undef) \quad (6)$$

Формула $L(u, v)$ гласит, что оценки расстояний $d(u, v)$ и содержимое списка $Nb_u[v]$ в узле u всегда согласовано с теми данными, которые хранятся в локальной памяти узла u .

Инварианты алгоритма Nchange

Лемма 2.

Для любой пары вершин u_0, v_0 формула $L(u_0, v_0)$ является инвариантом.

Инварианты алгоритма Netchange

Лемма 2.

Для любой пары вершин u_0, v_0 формула $L(u_0, v_0)$ является инвариантом.

Доказательство.

Самостоятельно.

Корректность алгоритма Nchange

Теорема 1.

Как только достигается стабильная конфигурация, таблицы $Nb_u[v]$ удовлетворяют следующим условиям

- 1) если $u = v$, то $Nb_u[v] = local$;
- 2) если существует путь из вершины u в вершину $v \neq u$, то $Nb_u[v] = w$, где w — первый сосед вершины u , который встречается на кратчайшем пути из u в v ;
- 3) если пути из вершины u в вершину v не существует, то $Nb_u[v] = undef$.

Корректность алгоритма Nchange

Теорема 1.

Как только достигается стабильная конфигурация, таблицы $Nb_u[v]$ удовлетворяют следующим условиям

- 1) если $u = v$, то $Nb_u[v] = local$;
- 2) если существует путь из вершины u в вершину $v \neq u$, то $Nb_u[v] = w$, где w — первый сосед вершины u , который встречается на кратчайшем пути из u в v ;
- 3) если пути из вершины u в вершину v не существует, то $Nb_u[v] = undef$.

Доказательство.

Когда алгоритм завершает свое выполнение, наряду с предикатом **stable** выполняется и формула $P(u, w, v)$ для всех троек вершин u, v, w , и отсюда следует, что для всех троек u, v, w справедливо соотношение

$$up(u, w) \implies ndis_u[w, v] = D_w[v]. \quad (1)$$

Корректность алгоритма Nchange

Принимая во внимание утверждение $L(u, v)$ для всех пар вершин u и v , мы получаем следующее соотношение:

$$D_u[v] = \begin{cases} 0, & \text{если } u = v, \\ 1 + \min_{w \in \text{Neigh}_u} D_w[v], & \text{если } u \neq v \wedge \exists w \in \text{Neigh}_u : D_w[v] < N - 1 \\ N, & \text{если } u \neq v \wedge \forall w \in \text{Neigh}_u : D_w[v] \geq N - 1. \end{cases} \quad (2)$$

которого достаточно, чтобы доказать, что $D_u[v] = d(u, v)$ всякий раз, когда вершины u и v находятся в одной и той же компоненте связности сети, и $D_u[v] = N$, если u и v находятся в разных компонентах связности.

Корректность алгоритма Nchange

Вначале мы покажем, воспользовавшись индукцией по $d(u, v)$, что всякий раз, когда вершины u и v находятся в одной и той же компоненте связности сети, верно неравенство $D_u[v] \leq d(u, v)$.

Корректность алгоритма Nchange

Вначале мы покажем, воспользовавшись индукцией по $d(u, v)$, что всякий раз, когда вершины u и v находятся в одной и той же компоненте связности сети, верно неравенство $D_u[v] \leq d(u, v)$.

Случай $d(u, v) = 0$.

В этом случае $u = v$, и поэтому $D_u[v] = 0$.

Корректность алгоритма Nchange

Вначале мы покажем, воспользовавшись индукцией по $d(u, v)$, что всякий раз, когда вершины u и v находятся в одной и той же компоненте связности сети, верно неравенство $D_u[v] \leq d(u, v)$.

Случай $d(u, v) = 0$.

В этом случае $u = v$, и поэтому $D_u[v] = 0$.

Случай $d(u, v) = k + 1$.

В этом случае существует такая вершина $w \in Neigh_u$, для которой $d(w, v) = k$. По индуктивному предположению $D_w[v] \leq k$, и отсюда вследствие соотношения

$$D_u[v] = 1 + \min_{w \in Neigh_u} D_w[v]$$

мы получаем $D_u[v] \leq k + 1$.

Корректность алгоритма Nchange

Теперь покажем индукцией по $D_u[v]$, что всякий раз, когда $D_u[v] < N$, из вершины u в вершину v существует путь, и при этом $d(u, v) \leq D_u[v]$.

Корректность алгоритма Nchange

Теперь покажем индукцией по $D_u[v]$, что всякий раз, когда $D_u[v] < N$, из вершины u в вершину v существует путь, и при этом $d(u, v) \leq D_u[v]$.

Случай $D_u[v] = 0$.

Из формулы (2) следует, что $D_u[v] = 0$ только тогда, когда $u = v$. Это означает, что между вершинами u и v есть пустой путь и $d(u, v) = 0$.

Корректность алгоритма Nchange

Теперь покажем индукцией по $D_u[v]$, что всякий раз, когда $D_u[v] < N$, из вершины u в вершину v существует путь, и при этом $d(u, v) \leq D_u[v]$.

Случай $D_u[v] = 0$.

Из формулы (2) следует, что $D_u[v] = 0$ только тогда, когда $u = v$. Это означает, что между вершинами u и v есть пустой путь и $d(u, v) = 0$.

$D_u[v] = k + 1 < N$.

Из формулы (2) следует, что существует такая вершина $w \in Neigh_u$, для которой $D_w[v] = k$. По индуктивному предположению между вершинами w и v существует путь и $d(w, v) \leq k$. Отсюда следует, что между вершинами u и v также есть путь и $d(u, v) \leq k + 1$.

Корректность алгоритма Nchange

$$u = v \implies (D_u[v] = 0 \wedge Nb_u[v] = local) \quad (4)$$

$$(u \neq v \wedge \exists w \in Neigh_u : ndis_u[w, v] < N - 1) \\ \implies (D_u[v] = 1 + \min_{w \in Neigh_u} ndis_u[w, v] = 1 + ndis_u[Nb_u[v], v]) \quad (5)$$

$$(u \neq v \wedge \forall w \in Neigh_u : ndis_u[w, v] \geq N - 1) \\ \implies (D_u[v] = N \wedge Nb_u[v] = udef) \quad (6)$$

$$D_u[v] = \begin{cases} 0, & \text{если } u = v, \\ 1 + \min_{w \in Neigh_u} D_w[v], & \text{если } u \neq v \wedge \exists w \in Neigh_u : D_w[v] < N - 1 \\ N, & \text{если } u \neq v \wedge \forall w \in Neigh_u : D_w[v] \geq N - 1. \end{cases}$$

Таким образом, если u и v находятся в одной и той же компоненте связности, то $D_u[v] = d(u, v)$, а в противном случае $D_u[v] = N$. Отсюда, принимая во внимание формулу (2) и предложение $\forall u, v : L(u, v)$, получаем заявленный в утверждении теоремы результат о таблицах $Nb_u[v]$. □

Завершаемость алгоритма Netchange

Чтобы убедиться в том, что стабильная ситуация рано или поздно наступит, после того как завершатся изменения топологии, мы введем нормирующую функцию по отношению к предикату **stable**. Для конфигурации γ нашего алгоритма будем полагать

$$t_i = \begin{aligned} & \text{(число сообщений типа } \langle \mathbf{mydist}, \dots, i \rangle) + \\ & + \text{(число упорядоченных пар } u, v, \text{ для которых } D_u[v] = i), \end{aligned}$$

и значением функции f будем считать $(N + 1)$ -местный набор

$$f(\gamma) = (t_0, t_1, \dots, t_N).$$

На множестве таких наборов введем лексикографический порядок $\leq_I$. Здесь нужно вспомнить о том, что $(\mathbb{N}^{N+1}, \leq_I)$ является вполне упорядоченным множеством.

Завершаемость алгоритма Netchange

Лемма 3.

Обработка сообщений типа **mydist** приводит к уменьшению значения f .

Завершаемость алгоритма Netchange

Лемма 3.

Обработка сообщений типа **mydist** приводит к уменьшению значения f .

Доказательство.

Предположим, что в узел u , у которого $D_u[v] = d_1$, поступило сообщение $\langle \mathbf{mydist}, v, d_2 \rangle$, и после выполнения перерасчета новое значение переменной $D_u[v]$ становится равным d . Из описания алгоритма вытекает, что $d \leq d_2 + 1$.

Завершаемость алгоритма Netchange

Случай $d < d_1$.

Тогда $d = d_2 + 1$, и отсюда следует, что значение t_{d_2} (равно как и t_{d_1}) уменьшается на единицу, и только значения t_d , у которых $d > d_2$, увеличиваются. Отсюда следует, что значение функции f уменьшается.

Завершаемость алгоритма Netchange

Случай $d < d_1$.

Тогда $d = d_2 + 1$, и отсюда следует, что значение t_{d_2} (равно как и t_{d_1}) уменьшается на единицу, и только значения t_d , у которых $d > d_2$, увеличиваются. Отсюда следует, что значение функции f уменьшается.

Случай $d = d_1$.

Узел u не отправляет никаких новых сообщений типа **mydist**, и на функции f это сказывается только в том, что значение t_{d_2} уменьшается на единицу. Поэтому значение f уменьшается.

Завершаемость алгоритма Netchange

Случай $d < d_1$.

Тогда $d = d_2 + 1$, и отсюда следует, что значение t_{d_2} (равно как и t_{d_1}) уменьшается на единицу, и только значения t_d , у которых $d > d_2$, увеличиваются. Отсюда следует, что значение функции f уменьшается.

Случай $d = d_1$.

Узел u не отправляет никаких новых сообщений типа **mydist**, и на функции f это сказывается только в том, что значение t_{d_2} уменьшается на единицу. Поэтому значение f уменьшается.

Случай $d > d_1$.

Тогда значение t_{d_1} (равно как и t_{d_2}) уменьшается на единицу, и только значения t_d , у которых $d > d_1$, увеличиваются. Отсюда следует, что значение функции f уменьшается.



Завершаемость алгоритма Netchange

Теорема 2.

Если начиная с какого-то момента топология сети остается неизменной, то спустя конечное число шагов алгоритм достигает стабильной конфигурации.

Завершаемость алгоритма Nchange

Теорема 2.

Если начиная с какого-то момента топология сети остается неизменной, то спустя конечное число шагов алгоритм достигает стабильной конфигурации.

Доказательство.

Если топология сети не претерпевает никаких изменений, то в дальнейшем происходит только обработка сообщений типа **mydist**, и согласно предыдущей лемме с каждым таким переходом значение функции f уменьшается.

В таком случае вследствие фундированности множества значений функции f может произойти только конечное число переходов. Значит, после конечного числа шагов алгоритм достигнет конфигурации, в которой предикат **stable** обращается в истину.



Особенности реализации алгоритма Netchange

Корректность алгоритма, гарантирующая построение правильных таблиц за конечное число шагов, после последнего топологического изменения, мало что говорит нам о настоящем поведении алгоритма.

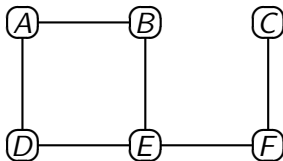
Пока предикат **stable** ложен, ничего определенного о таблицах маршрутизации не известно. Они могут содержать циклы или даже вообще давать неверную информацию о достижимости вершин-адресатов. Поэтому предложенный алгоритм можно использовать только в таких приложениях, где топологические изменения редки, а время сходимости алгоритма невелико по сравнению со средним периодом времени между возникновением двух изменений в топологии сети.

Еще более осложняет ситуацию то обстоятельство, что предикат **stable** задает глобальное свойство, и поэтому с точки зрения отдельного узла сети стабильная конфигурация алгоритма неотличима от нестабильной конфигурации.

Особенности реализации алгоритма Nchange

Задача.

Выясните, какие значения будут иметь все переменные в заключительной конфигурации алгоритма Nchange в том случае, когда этот алгоритм применяется к сети, имеющей следующую топологическую структуру:



После того как была достигнута заключительная конфигурация, в сети возник новый канал связи между узлами A и F. Какое сообщение узел F отправит узлу A при обработке уведомления $\langle \text{repair}, A \rangle$? Какое послание узел A отправит узлу F в ответ на это сообщение?

Особенности реализации алгоритма Netchange

Асинхронная обработка уведомлений

Мы считали, что уведомления о топологических изменениях обрабатываются синхронно по обе стороны канала связи.

Можно учитывать задержку обработки таких уведомлений.

Канал связи wu моделируется посредством трех очередей:

- 1) OQ_{wu} — выходная очередь узла w ;
- 2) TQ_{wu} — очередь сообщений (или пакетов данных), которые уже были переправлены;
- 3) IQ_{wu} — входная очередь узла u .

Когда в канале возникает неисправность, из очередей TQ_{wu} и OQ_{wu} выбрасываются все сообщения, а в конец очереди IQ_{wu} добавляется сообщение $\langle \text{fail}, w \rangle$. Когда возобновляется нормальное функционирование канала связи, в конец очереди IQ_{wu} добавляется сообщение $\langle \text{repair}, w \rangle$. В этом случае предикаты $P(u, w, v)$ будут иметь чуть более сложную структуру, но сам алгоритм останется без изменения.

Особенности реализации алгоритма Netchange

Маршрутизация по кратчайшим путям

Каждому каналу связи можно приписать весовой коэффициент; после этого рассмотренный нами алгоритм можно модифицировать так, чтобы он вместо путей с наименьшим числом звеньев вычислял кратчайшие пути. Процедура *Update* в алгоритме Netchange будет учитывать вес канала связи uw при оценке длины кратчайшего пути через узел w , если заменить константу 1 на весовой коэффициент ω_{uw} . При этом константа N должна быть заменена в алгоритме на какую-нибудь верхнюю оценку диаметра сети.

Можно даже провести такое обобщение этого алгоритма, чтобы он мог работать с каналами связи, имеющими переменный вес; реакцией узла u на изменение весового коэффициента канала должно быть переычисление значений переменных $D_u[v]$ для всех вершин v .

Другие виды маршрутизации

До сих пор мы имели дело с алгоритмами маршрутизации, которые в каждом узле сети создают и поддерживают таблицы маршрутизации с отдельным входом для каждой вершины-адресата. Можно сократить объем памяти и издержки, связанные с просмотром таблиц.

Общая стратегия получения таблиц меньшего размера такова. Для каждого канала связи, примыкающего к узлу, строится список вершин-адресатов, маршруты к которым начинаются с прохождения этого канала. Экономия памяти будет зависеть от того, насколько компактно можно представить множество всех вершин-адресатов, соответствующих каждому каналу связи. Чтобы эффективно проводить поиск в таблицах, они должны быть устроены так, чтобы для заданной вершины-адресата на основе таблиц можно было быстро выбрать соответствующий ей исходящий канал связи.

Другие виды маршрутизации

Схема древесной разметки

Этот метод помечает вершины целыми числами от 0 до $N - 1$ так, чтобы для каждого канала связи множество вершин-адресатов, сообщение с которыми осуществляется через этот канал, представляло собой интервал.

Циклическим интервалом $[a, b)$ на множестве $\mathbb{Z}$ будем называть всякое множество целых чисел, удовлетворяющее следующему соотношению:

$$[a, b) = \begin{cases} \{a, a + 1, \dots, b - 1\}, & \text{если } a < b, \\ \{0, \dots, b - 1, a, \dots, N - 1\}, & \text{если } a \geq b. \end{cases}$$

Другие виды маршрутизации

Схема древесной разметки

```
(* Пакет с адресом  $d$  был получен или создан в узле  $u$  *)  
if  $d = l_u$   
  then пакет доставлен по адресу  
  else begin выбрать такое число  $\alpha_i$ , что  $d \in [\alpha_i, \alpha_{i+1})$  ;  
              отправить пакет по каналу, помеченному  $\alpha_i$   
  end
```

Другие виды маршрутизации

Схема древесной разметки

(* Пакет с адресом d был получен или создан в узле u *)
if $d = l_u$
 then пакет доставлен по адресу
 else begin выбрать такое число α_i , что $d \in [\alpha_i, \alpha_{i+1})$;
 отправить пакет по каналу, помеченному α_i
 end

Теорема 3

Вершины дерева T можно занумеровать таким образом, чтобы для всякой вершины и каждого исходящего из этой вершины канала множество узлов, путь в которые проходит через этот канал, образовывало циклический интервал.

Другие виды маршрутизации

Схема древесной разметки

Достоинства

Для задания одного циклического интервала можно обойтись $2 \log N$ битами, указывая только границы интервала.

Другие виды маршрутизации

Схема древесной разметки

Достоинства

Для задания одного циклического интервала можно обойтись $2 \log N$ битами, указывая только границы интервала.

Недостатки

1. Каналы, не принадлежащие дереву T , не используются, а это означает, что ресурсы сети расходуются неэкономно.
2. Весь трафик сосредоточен в дереве, а это приводит к перегрузкам в сети.
3. Всякая неисправность в одном канале дерева T приводит к разрыву сети.

Другие виды маршрутизации

Схема интервальной разметки

Обобщение схемы древесной разметки, которое позволило применять эту схему к произвольным сетям таким образом, что почти каждый канал оказывается задействованным в продвижении пакетов.

Другие виды маршрутизации

Схема интервальной разметки

Обобщение схемы древесной разметки, которое позволило применять эту схему к произвольным сетям таким образом, что почти каждый канал оказывается задействованным в продвижении пакетов.

Префиксная маршрутизация

В префиксной маршрутизации пометками вершин и каналов служат строки. Чтобы выбрать канал, по которому нужно продвигать пакет, алгоритм рассматривает все пометки каналов, являющиеся префиксом адреса вершины назначения. Выбирается наиболее длинная из таких пометок, и продвижение сообщения проводится по соответствующему каналу.

Другие виды маршрутизации

Схема интервальной разметки

Обобщение схемы древесной разметки, которое позволило применять эту схему к произвольным сетям таким образом, что почти каждый канал оказывается задействованным в продвижении пакетов.

Префиксная маршрутизация

В префиксной маршрутизации пометками вершин и каналов служат строки. Чтобы выбрать канал, по которому нужно продвигать пакет, алгоритм рассматривает все пометки каналов, являющиеся префиксом адреса вершины назначения. Выбирается наиболее длинная из таких пометок, и продвижение сообщения проводится по соответствующему каналу.

Иерархическая маршрутизация

Иерархический метод маршрутизации предполагает разбиение сети на кластеры.

КОНЕЦ ЛЕКЦИИ 5-2.