

Theorem 4.13 *The algorithm of Chandy and Misra computes minimum-hop routing tables by exchanging $O(N^2)$ messages and $O(N^2W)$ bits per channel, and $O(N^2 \cdot |E|)$ messages and $O(N^2 \cdot |E| \cdot W)$ bits in total.*

An advantage of the algorithm of Chandy and Misra over that of Merlin and Segall is its simplicity, its smaller space complexity, and its lower time complexity.

4.3 The Netchange Algorithm

Tajibnapis' Netchange algorithm [Taj77] computes routing tables that are optimal according to the "minimum-hop" measure. The algorithm can be compared to the Chandy–Misra algorithm, but maintains additional information that allows the tables to be updated with only a *partial* recomputation after the failure or repair of a channel. The presentation of the algorithm in this section follows Lamport [Lam82]. The algorithm relies on the following assumptions.

- N1. The nodes know the size of the network (N).
- N2. The channels satisfy the fifo assumption.
- N3. Nodes are notified of failures and repairs of their adjacent channels.
- N4. The cost of a path equals the number of channels in the path.

The algorithm can handle the failure and repair or addition of channels, but it is assumed that a node is notified when an adjacent channel fails or recovers. The failure and recovery of nodes is not considered; instead it is assumed that the failure of a node is observed by its neighbors as the failure of the connecting channel. The algorithm maintains in each node u a table $Nb_u[v]$, giving for each destination v a neighbor of u to which packets for v will be forwarded. It cannot be required that the computation of these tables terminates within a finite number of steps in all cases because the repeated failure or repair of channels may ask for recomputation indefinitely. The requirements of the algorithm are as follows.

- R1. If the topology of the network remains constant after a finite number of topological changes, then the algorithm terminates after a finite number of steps.
- R2. When the algorithm terminates the tables $Nb_u[v]$ satisfy
 - (a) if $v = u$ then $Nb_u[v] = local$;
 - (b) if a path from u to $v \neq u$ exists then $Nb_u[v] = w$, where w is the first neighbor of u on a shortest path from u to v ;
 - (c) if no path from u to v exists then $Nb_u[v] = undef$.

Use outside Utrecht University and photocopying are prohibited.

```

var  $Neigh_u$       : set of nodes ;      (* The neighbors of  $u$  *)
       $D_u$           : array of 0..  $N$  ;    (*  $D_u[v]$  estimates  $d(u, v)$  *)
       $Nb_u$          : array of nodes ;    (*  $Nb_u[v]$  is preferred neighbor for  $v$  *)
       $ndis_u$       : array of 0..  $N$  ;    (*  $ndis_u[w, v]$  estimates  $d(w, v)$  *)

```

Initialization:

```

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

```

Procedure *Recompute* (v):

```

begin if  $v = u$ 
  then begin  $D_u[v] := 0$  ;  $Nb_u[v] := local$  end
  else begin (* Estimate distance to  $v$  *)
     $d := 1 + \min\{ndis_u[w, v] : w \in Neigh_u\}$  ;
    if  $d < N$  then
      begin  $D_u[v] := d$  ;
         $Nb_u[v] := w$  with  $1 + ndis_u[w, v] = d$ 
      end
    else begin  $D_u[v] := N$  ;  $Nb_u[v] := undef$  end
  end ;
  if  $D_u[v]$  has changed then
    forall  $x \in Neigh_u$  do send  $\langle mydist, v, D_u[v] \rangle$  to  $x$ 
  end

```

Algorithm 4.8 THE NETCHANGE ALGORITHM (PART 1, FOR NODE u).

4.3.1 Description of the Algorithm

Tajibnapis' Netchange algorithm is given as Algorithms 4.8 and 4.9. The steps of the algorithm will first be motivated by an informal description of the operation of the algorithm, and subsequently the correctness of the algorithm will be proved formally. For sake of clear exposition the modeling of topological changes is simplified as compared to [Lam82] by assuming that the notification of the change is processed simultaneously in the two nodes affected by the change. It is indicated in Subsection 4.3.3 how asynchronous processing of these notifications is treated.

The selection of a neighbor to which packets for destination v will be forwarded is based on estimates of the distance of each node to v . The preferred neighbor is always the neighbor with the lowest estimate of this distance. Node u maintains an estimate $D_u[v]$ of $d(u, v)$ and estimates $ndis_u[w, v]$ of $d(w, v)$ for each neighbor w of u . The estimate $D_u[v]$ is

```

Processing a  $\langle \mathbf{mydist}, v, d \rangle$  message from neighbor  $w$ :
  { A  $\langle \mathbf{mydist}, v, d \rangle$  is at the head of  $Q_{wv}$  }
  begin receive  $\langle \mathbf{mydist}, v, d \rangle$  from  $w$  ;
         $ndis_u[w, v] := d$  ; Recompute ( $v$ )
  end

Upon failure of channel  $uw$ :
  begin receive  $\langle \mathbf{fail}, w \rangle$  ;  $Neigh_u := Neigh_u \setminus \{w\}$  ;
        forall  $v \in V$  do Recompute ( $v$ )
        (* It suffices actually to do this for  $v$  s.t.  $Nb_u[v] = w$  *)
  end

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

```

Algorithm 4.9 THE NETCHANGE ALGORITHM (PART 2, FOR NODE u).

computed from the estimates $ndis_u[w, v]$, and the estimates $ndis_u[w, v]$ are obtained via communication with the neighbors.

The computation of the estimates $D_u[v]$ proceeds as follows. If $u = v$ then $d(u, v) = 0$ so $D_u[v]$ is set to 0 in this case. If $u \neq v$, a shortest path from u to v (if such a path exists) consists of a channel from u to a neighbor, concatenated with a shortest path from this neighbor to v , and consequently

$$d(u, v) = 1 + \min_{w \in Neigh_u} d(w, v).$$

Following this equation, node $u \neq v$ estimates $d(u, v)$ by applying this formula to the *estimated* values of $d(w, v)$, found in the tables as $ndis_u[w, v]$. As there are N nodes, a minimum-hop path has length at most $N - 1$. A node may suspect that no path exists if it computes an estimated distance of N or more; the value N is used in the table to represent this.

The algorithm requires a node to have an estimate of its neighbors' distances to v . These are obtained from these nodes because they communicate them in $\langle \mathbf{mydist}, ., . \rangle$ messages as follows. If node u computes the value d as an estimate of its distance to v ($D_u[v] = d$), this information is sent to all neighbors in a message $\langle \mathbf{mydist}, v, d \rangle$. Upon receipt of a message $\langle \mathbf{mydist}, v, d \rangle$ from neighbor w , u assigns $ndis_u[w, v]$ the value d . As a result of a change in $ndis_u[w, v]$ u 's estimate of $d(u, v)$ can change and

Use outside Utrecht University and photocopying are prohibited.

therefore the estimate is recomputed every time the $ndis_u$ table changes. If the estimate indeed changes, to d' say, this is of course communicated to the neighbors using $\langle \mathbf{mydist}, v, d' \rangle$ messages.

The algorithm reacts to failures and repairs of channels by modifying the local tables, and sending a $\langle \mathbf{mydist}, ., . \rangle$ message if distance-estimates change. We assume that the notification that nodes receive about channel ups and downs (assumption N3) is in the form of $\langle \mathbf{fail}, . \rangle$ and $\langle \mathbf{repair}, . \rangle$ messages. The channel between nodes u_1 and u_2 is modeled by two queues, $Q_{u_1 u_2}$ for the messages from u_1 to u_2 and $Q_{u_2 u_1}$ for the messages from u_2 to u_1 . When a channel fails these queues are removed from the configuration (effectively causing all messages in both queues to be lost) and the nodes at both ends of the channel receive a $\langle \mathbf{fail}, . \rangle$ message. If the channel between u_1 and u_2 fails, u_1 receives a $\langle \mathbf{fail}, u_2 \rangle$ message and u_2 receives a $\langle \mathbf{fail}, u_1 \rangle$ message. When a channel is repaired (or a new channel is added to the network) two empty queues are added to the configuration and the two nodes connected by the channel receive a $\langle \mathbf{repair}, . \rangle$ message. If the channel between u_1 and u_2 comes up u_1 receives a $\langle \mathbf{repair}, u_2 \rangle$ message and u_2 receives a $\langle \mathbf{repair}, u_1 \rangle$ message.

The reaction of the algorithm to the failures and repairs is as follows. When the channel between u and w fails, w is removed from $Neigh_u$ and vice versa. The distance estimate for each destination is recomputed and, of course, sent to all remaining neighbors if it has changed. This is the case if the best route previously was via the failed channel and there is no other neighbor w' with $ndis_u[w', v] = ndis_u[w, v]$. When the channel is repaired (or a new channel is added) w is added to $Neigh_u$, but u has as yet no estimate of the distance $d(w, v)$ (and vice versa). The new neighbor w is immediately informed about $D_u[v]$ for all destinations v (by sending $\langle \mathbf{mydist}, v, D_u[v] \rangle$ messages. Until u receives similar messages from w , u uses N as an estimate for $d(w, v)$, i.e., it sets $ndis_u[w, v]$ to N .

Invariants of the Netchange algorithm. We shall prove a number of assertions to be invariants; the assertions are given in Figure 4.10. The assertion $P(u, w, v)$ states that if u has finished processing $\langle \mathbf{mydist}, v, . \rangle$ messages from w then u 's estimate of $d(w, v)$ equals w 's estimate of $d(w, v)$. Let the predicate $up(u, w)$ be true if and only if a (bidirectional) channel between u and w exists and is operating. The assertion $L(u, v)$ states that u 's estimate of $d(u, v)$ is always in agreement with u 's local knowledge, and $Nb_u[v]$ is set accordingly.

The computation of the algorithm terminates when there are no more messages of the algorithm in transit in any channel. These configurations

$$P(u, w, v) \equiv \begin{aligned} & up(u, w) \iff w \in Neigh_u \\ & \wedge up(u, w) \wedge Q_{wu} \text{ contains a } \langle \mathbf{mydist}, v, d \rangle \text{ message} \end{aligned} \quad (1)$$

$$\Rightarrow \text{the last such message satisfies } d = D_w[v] \quad (2)$$

$$\wedge up(u, w) \wedge Q_{wu} \text{ contains no } \langle \mathbf{mydist}, v, d \rangle \text{ message} \Rightarrow ndis_u[w, v] = D_w[v] \quad (3)$$

$$L(u, v) \equiv \begin{aligned} & u = v \Rightarrow (D_u[v] = 0 \wedge Nb_u[v] = local) \end{aligned} \quad (4)$$

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

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

Figure 4.10 THE INVARIANTS $P(u, w, v)$ AND $L(u, v)$.

are not terminal for the whole system, because the system's computation may later continue, starting with a channel failure or repair (to which the algorithm must react). We shall call message-less configurations *stable*, and define the predicate **stable** by

$$\mathbf{stable} \equiv \forall u, w : up(u, w) \Rightarrow Q_{wu} \text{ contains no } \langle \mathbf{mydist}, \dots \rangle \text{ message.}$$

It must be assumed that initially the variables $Neigh_u$ correctly reflect the existence of working communication channels, i.e., that (1) holds initially. To prove the invariance of the assertions three types of transition must be considered.

- (1) The receipt of a $\langle \mathbf{mydist}, \dots \rangle$ message. The entire execution of the resulting code fragment is assumed to occur atomically and is considered a single transition. Note that in this transition a message is received and possibly a number of messages is sent.
- (2) The failure of a channel and the processing of a $\langle \mathbf{fail}, \dots \rangle$ message by the nodes at both ends of the channel.
- (3) The repair of a channel and the processing of a $\langle \mathbf{repair}, \dots \rangle$ message by the two connected nodes.

Lemma 4.14 *For all u_0, w_0 , and v_0 , $P(u_0, w_0, v_0)$ is an invariant.*

Proof. Initially, i.e., after the execution of the initialization procedure by each node, (1) holds by assumption. If initially we have $\neg up(u_0, w_0)$, (2) and (3) trivially hold. If initially we have $up(u_0, w_0)$, then $ndis_{u_0}[w_0, v_0] = N$. If $w_0 = v_0$ then $D_{w_0}[w_0] = 0$ but a message $\langle \mathbf{mydist}, v_0, 0 \rangle$ is in $Q_{w_0 u_0}$, so

Use outside Utrecht University and photocopying are prohibited.

(2) and (3) are true. If $w_0 \neq v_0$ then $D_{w_0}[v_0] = N$ and no message is in the queue, which also implies that (2) and (3) hold. We consider the three types of state transition mentioned above in turn.

Type (1). Assume that u receives a $\langle \mathbf{mydist}, v, d \rangle$ message from w .

This causes no topological change and no change in the *Neigh* sets, hence (1) remains true. If $v \neq v_0$ this receipt does not change anything in $P(u_0, w_0, v_0)$.

If $v = v_0$, $u = u_0$, and $w = w_0$ the value of $ndis_{u_0}[w_0, v_0]$ may change. However, if another $\langle \mathbf{mydist}, v_0, . \rangle$ message is still in the channel then the value of this message continues to satisfy (2), so (2) is preserved and (3) also because its premise is false. If the received message was the last one in the channel of this type then $d = D_{w_0}[v_0]$ by (2), which implies that the conclusion of (3) becomes true and (3) is preserved. The premise of (2) becomes false, so (2) is preserved.

If $v = v_0$, $u = w_0$ (and u_0 is a neighbor of u) the conclusion of (2) or (3) may be falsified if the value $D_{w_0}[v_0]$ changes as a result of the execution of *Recompute*(v) in w_0 . In this case, however, a message $\langle \mathbf{mydist}, v_0, . \rangle$ with the new value is sent to u_0 , which implies that the premise of (3) is falsified, and the conclusion of (2) becomes true, so both (2) and (3) are preserved. This is also the only case in which a $\langle \mathbf{mydist}, v_0, . \rangle$ message is added to $Q_{w_0 u_0}$, and it always satisfies $d = D_{w_0}[v_0]$.

If $v = v_0$ and $u \neq u_0$, w_0 nothing changes in $P(u_0, w_0, v_0)$.

Type (2). Assume that channel uw fails.

If $u = u_0$ and $w = w_0$ this failure falsifies the premise of (2) and (3) so these clauses are preserved. (1) is preserved because w_0 is removed from $Neigh_{u_0}$ and vice versa. The same happens if $u = w_0$ and $w = u_0$.

If $u = w_0$ but $w \neq u_0$ the conclusion of (2) or (3) may be falsified because the value $D_{w_0}[v_0]$ changes. In this case the sending of a $\langle \mathbf{mydist}, v_0, . \rangle$ message by w_0 again falsifies the premise of (3) and makes the conclusion of (2) true, hence (2) and (3) are preserved.

In all other cases nothing changes in $P(u_0, w_0, v_0)$.

Type (3). Assume that channel uw is added.

If $u = u_0$ and $w = w_0$ this makes $up(u_0, w_0)$ true, but by the addition of w_0 to $Neigh_{u_0}$ (and vice versa) this preserves (1).

The sending of $\langle \mathbf{mydist}, v_0, D_{w_0}[v_0] \rangle$ by w_0 makes the conclusion of (2) true and the premise of (3) false, so $P(u_0, w_0, v_0)$ is preserved.

In all other cases nothing changes in $P(u_0, w_0, v_0)$.

□

Lemma 4.15 *For each u_0 and v_0 , $L(u_0, v_0)$ is an invariant.*

Proof. Initially $D_{u_0}[u_0] = 0$ and $Nb_{u_0}[u_0] = \text{local}$. For $v_0 \neq u_0$, initially $ndis_{u_0}[w, v_0] = N$ for all $w \in \text{Neigh}_{u_0}$, and $D_{u_0}[v_0] = N$ and $Nb_{u_0}[v_0] = \text{undef}$.

Type (1). Assume that u receives a $\langle \text{mydist}, v, d \rangle$ message from w .

If $u \neq u_0$ or $v \neq v_0$ no variable mentioned in $L(u_0, v_0)$ changes.

If $u = u_0$ and $v = v_0$ the value of $ndis_{u_0}[w, v_0]$ changes, but $D_{u_0}[v_0]$ and $Nb_{u_0}[v_0]$ are recomputed exactly so as to satisfy $L(u_0, v_0)$.

Type (2). Assume that channel uw fails.

If $u = u_0$ or $w = u_0$ then Neigh_{u_0} changes, but again $D_{u_0}[v_0]$ and $Nb_{u_0}[v_0]$ are recomputed exactly so as to satisfy $L(u_0, v_0)$.

Type (3). Assume that channel uw is added.

If $u = u_0$ then Neigh_{u_0} changes by the addition of w , but as u sets $ndis_{u_0}[w, v_0]$ to N this preserves $L(u_0, v_0)$.

□

4.3.2 Correctness of the Netchange Algorithm

The two correctness requirements for the algorithm will now be proved.

Theorem 4.16 *When a stable configuration is reached, the tables $Nb_u[v]$ satisfy*

- (1) *if $u = v$ then $Nb_u[v] = \text{local}$;*
- (2) *if a path from u to $v \neq u$ exists then $Nb_u[v] = w$, where w is the first neighbor of u on a shortest path from u to v ;*
- (3) *if no path from u to v exists then $Nb_u[v] = \text{undef}$.*

Proof. When the algorithm terminates, the predicate **stable** holds in addition to $P(u, w, v)$ for all u, v , and w , and this implies that for all u, v , and w

$$up(u, w) \Rightarrow ndis_u[w, v] = D_w[v]. \quad (4.2)$$

Applying also $L(u, v)$ for all u and v we obtain

$$D_u[v] = \begin{cases} 0 & \text{if } u = v \\ 1 + \min_{w \in \text{Neigh}_u} D_w[v] & \text{if } u \neq v \wedge \exists w \in \text{Neigh}_u : D_w[v] < N - 1 \\ N & \text{if } u \neq v \wedge \forall w \in \text{Neigh}_u : D_w[v] \geq N - 1 \end{cases} \quad (4.3)$$

which is sufficient to prove that $D_u[v] = d(u, v)$ if u and v are in the same connected component of the network, and $D_u[v] = N$ if u and v are in different connected components.

Use outside Utrecht University and photocopying are prohibited.

First it is shown by induction on $d(u, v)$ that if u and v are in the same connected component then $D_u[v] \leq d(u, v)$.

Case $d(u, v) = 0$: this implies $u = v$ and hence $D_u[v] = 0$.

Case $d(u, v) = k + 1$: this implies that there exists a node $w \in \text{Neigh}_u$ with $d(w, v) = k$. By induction $D_w[v] \leq k$, which by (4.3) implies $D_u[v] \leq k + 1$.

Now it will be shown by induction on $D_u[v]$ that if $D_u[v] < N$ then there is a path between u and v and $d(u, v) \leq D_u[v]$.

Case $D_u[v] = 0$: Formula (4.3) implies that $D_u[v] = 0$ only for $u = v$, which gives the empty path between u and v , and $d(u, v) = 0$.

Case $D_u[v] = k + 1 < N$: Formula (4.3) implies that there is a node $w \in \text{Neigh}_u$ with $D_w[v] = k$. By induction there is a path between w and v and $d(w, v) \leq k$, which implies there is a path between u and v and $d(u, v) \leq k + 1$.

It follows that if u and v are in the same connected component then $D_u[v] = d(u, v)$, otherwise $D_u[v] = N$. This, Formula (4.2), and $\forall u, v : L(u, v)$ imply the stated result about $Nb_u[v]$. \square

To prove that a stable situation is eventually reached if topological changes cease, a norm function with respect to **stable** will be defined. Define, for a configuration γ of the algorithm,

$$t_i = \begin{aligned} & \text{(the number of } \langle \mathbf{mydist}, \cdot, i \rangle \text{ messages)} \\ & + \text{(the number of ordered pairs } u, v \text{ s.t. } D_u[v] = i) \end{aligned}$$

and the function f by

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

$f(\gamma)$ is an $(N + 1)$ -tuple of natural numbers, on which a lexicographic order (denoted \leq_l) is assumed. Recall that $(\mathbb{N}^{N+1}, \leq_l)$ is a well-founded set (Exercise 2.5).

Lemma 4.17 *The processing of a $\langle \mathbf{mydist}, \cdot, \cdot \rangle$ message decreases f .*

Proof. Assume node u with $D_u[v] = d_1$ receives a $\langle \mathbf{mydist}, v, d_2 \rangle$ message, and after recomputation the new value of $D_u[v]$ is d . The algorithm implies that $d \leq d_2 + 1$.

Case $d < d_1$: Now $d = d_2 + 1$ which implies that t_{d_2} is decreased by one (and t_{d_1} as well), and only t_d with $d > d_2$ is increased. This implies that the value of f is decreased.

Case $d = d_1$: No new $\langle \mathbf{mydist}, \dots \rangle$ messages are sent by u , and the only effect on f is that t_{d_2} is decreased by one, so the value of f is decreased.

Case $d > d_1$: Now t_{d_1} is decreased by one (and t_{d_2} as well), and only t_d with $d > d_1$ is increased. This implies that the value of f is decreased.

□

Theorem 4.18 *If the topology of the network remains constant after a finite number of topological changes, then the algorithm reaches a stable configuration after a finite number of steps.*

Proof. If the network topology remains constant only further processing of $\langle \mathbf{mydist}, \dots \rangle$ messages takes place, and, by the previous lemma, the value of f decreases with every such transition. It follows from the well-foundedness of the domain of f that only a finite number of these transitions can take place; hence the algorithm reaches a configuration satisfying **stable** after a finite number of steps. □

4.3.3 Discussion of the Algorithm

The formal correctness results of the algorithm, guaranteeing the convergence to correct tables within finite time after the last topological change, are not very indicative about the actual behavior of the algorithm. The predicate **stable** may in practice be false most of the time (namely, if topological changes are frequent) and when **stable** is false nothing is known about the routing tables. They may contain cycles or even give erroneous information about the reachability of a destination node. The algorithm can therefore only be used in applications where topological changes are so infrequent that the convergence time of the algorithm is small compared with the average time between (bursts of) topological changes. This is all the more the case because **stable** is a global property, and stable configurations of the algorithm are indistinguishable from non-stable ones for the nodes. This means that a node never knows whether its routing table correctly reflects the network topology, and cannot defer forwarding data packets until a stable configuration is reached.

Asynchronous processing of notifications. It has been assumed in this section that the notifications of topological changes are processed atomically together with the change in a single transition. The processing takes place at both sides of the removed or added channel simultaneously. Lamport [Lam82] has carried out the analysis in a little more detail to allow a delay

Use outside Utrecht University and photocopying are prohibited.

in processing these notifications. The communication channel from w to u is modeled as the concatenation of three queues.

- (1) OQ_{wu} , the output queue of w ;
- (2) TQ_{wu} , the queue of messages (and data packets) currently being transmitted;
- (3) IQ_{wu} , the input queue of u .

Under the normal operation of a channel w sends a message to u by appending it to OQ_{wu} , messages move from OQ_{wu} to TQ_{wu} and from TQ_{wu} to IQ_{wu} , and u receives them by deleting them from IQ_{wu} . When the channel fails the messages in TQ_{wu} are thrown away and messages in OQ_{wu} are thereafter also thrown away rather than appended to TQ_{wu} . The $\langle \text{fail}, w \rangle$ message is placed at the end of IQ_{wu} , and when normal operation is resumed the $\langle \text{repair}, w \rangle$ message is also placed at the end of IQ_{wu} . The predicates $P(u, w, v)$ take a slightly more complicated form, but the algorithm remains the same.

Shortest-path routing. It is possible to assign a weight to each channel and modify the algorithm so as to compute shortest paths rather than minimum-hop paths. The procedure *Recompute* of the Netchange algorithm takes the weight of channel uw into account when estimating the length of the shortest path via w if the constant 1 is replaced by ω_{uw} . The constant N in the algorithm must be replaced by an upper bound on the diameter of the network.

It is fairly easy to show that when the modified algorithm reaches a stable configuration the routing tables are indeed correct and give optimal paths (all cycles in the network must have positive weight). The proof that the algorithm eventually reaches such a configuration requires a more complicated norm function.

It is even possible to extend the algorithm to deal with varying channel weights; the reaction of node u to a change in a channel weight is the recomputation of $D_u[v]$ for all v . The algorithm would be practical, however, only in situations where the average time between channel-cost changes is large compared to the convergence time, which is a quite unrealistic assumption. In these situations an algorithm should be preferred that guarantees cycle-freedom also during convergence, for example the Merlin–Segall algorithm.