

Algorithm 18.3

Given:

A local forwarding table with a distance for each entry, a distance to reach each neighbor, and an incoming DV message from a neighbor

Compute:

An updated forwarding table

Method:

Maintain a *distance* field in each forwarding table entry;
Initialize forwarding table with a single entry that has the *destination* equal to the local packet switch, the *next-hop* unused, and the *distance* set to zero;

Repeat forever {

Wait for a routing message to arrive over the network from a neighbor; let the sender be switch *N*;

for each entry in the message {

Let *V* be the destination in the entry and let *D* be the distance;

Compute *C* as *D* plus the weight assigned to the link over which the message arrived;

Examine and update the local routing table:

if (no route exists to *V*) {

add an entry to the local routing table for destination *V* with next-hop *N* and distance *C*;

} else if (a route exists that has next-hop *N*) {
replace the distance in existing route with *C*;

} else if (a route exists with distance greater than *C*) {
change the next-hop to *N* and distance to *C*;

}

}

}

Algorithm 18.3 Distance-vector algorithm for route computation.