IP Layer
The layers in the 7-layer model:

- physical – the cables
- data link – communication between directly connected hosts (in Internet terminology, on a local area network – a LAN)
  - Ethernet (802.3)
  - Wireless (802.11)
- network – communication between indirectly connected hosts (different LANs)
- IP
- transport – communication between processes (there can be multiple processes on a host)
  - UDP (IP between processes)
  - TCP (reliable, in-order, exactly once)
- session -
- presentation -
- application – whatever

The above are the minimal services at the layers, other services can be offered as well. Also, a different layering may be used (e.g., ATM combines data link layer functionality with network functionality and some have investigated IP directly over fiber).

Shortest Paths
Assume now that we have directed weighted links. Each node knows the weights of all incident edges and the number of nodes in the graph.
Motivation is min-cost communication

Problem: Determine the shortest path from \( i_0 \) to each other node.

This algorithm is the basis of Bellman-Ford, aka RIP (Route Information Protocol). In the presence of failures this is an unstable routing protocol and isn’t much used.

We use a tree again: every node should output:
- its parent in the tree
- its distance from \( i_0 \) in the (spanning) tree (sum of weights on edges)

We assume each node knows the weights of all incident edges and the number of nodes in the digraph.
Bellman-Ford

Each node keeps track of the shortest known distance from \( i_0 \) and updates it as new information arrives; also communicates changed distances.

```plaintext
automaton rip(r:Int, n:Set[Int])

signature
    send (v:Message, const r, j:Int)
    receive(v:Message, i:Int, const r)

states
    distance:Array[Int, Int] := constant(-1),
    neighbors:Set[Int] := n,
    sent:Set[Int] := {}

transitions
    send(v, i, j)
    pre
        v=distances \land j \in nbrs \land \neg(j \in sent)
    eff
        sent := insert(j, sent)

    receive(v, i, j)
    eff
        for v[k] \neq -1 \land k \neq r do
            if v[k]+1 < distance[k] then
                distance[k] := v[k]+1;
                change := true
            fi
        od
        if change then sent := {}
```

Terminates after \( n-1 \) rounds

**Key properties:**
1) No cycles (if there are no failures)
2) Every node eventually has the shortest distance to every other node.

Let’s suppose all nodes run this concurrently. The result of the above algorithm is that destination nodes know the shortest path from each possible source. To use this for routing, in practice, all the source nodes need to know the shortest distances and the next node on the path to each destination. In fact, for routing, source nodes build a table like:

<table>
<thead>
<tr>
<th>Destination</th>
<th>Next hop</th>
<th>Distance (to destination)</th>
<th>Go through</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This is non-trivial in a directed graph. The importance of this, these days, is that under certain circumstances wireless networks may have some unidirectional links. This table
should be initialized with all destinations, with go through=destination, and
distance=infinity.

Just the idea:
Then, all nodes could broadcast their own information in a message containing
quadruples <src, dst, parent, distance>. When a message arrives at the node X, for each
quadruple <X, A, B, n> in the message, update the distance for destination A to be equal
to n and if the entry in the “go through” column is A, then change it to B. If the entry is
not A, and the distance is less than the distance, save the quadruple. The go through
column works backwards along the path, until it gets back to an out-nbr of X. Each time
you update the go through column, check the set of saved tuples for one with A equal to
the value in the go through column; if found, change the go through column to B and
discard the tuple.

**RIP**
This is the Internet algorithm that is based on the above shortest paths algorithm.
As usual, in practice, we assume a bidirectional network and therefore an undirected
graph. here’s how the algorithm works:

Assume:
Each node knows the “distance” to each directly-connected neighbor router – a
link that is not there or down gets infinite cost.

Routing table:
A table $R_i$ of distances to routers, initially only to directly-linked
neighbors.

Initially, periodically, and whenever there is a change to the routing
table $R_i$, send routing table to all neighboring routers.

On receiving a routing vector $R_j$ from neighbor $j$, update your own routing table $R_i$
using the rule:
If $d(j) + R_j(k).distance < R_i(k).distance$ then
  change $R_i(k).to d(j) + R_j(k).distance$
  change next-hop (parent) to $j$

Example from Peterson & Davies, Figure 4.15, Table 4.5
RIP Example

Round 1. A sends routing table to B, C, E, F; B sends routing table to A, C; C sends
routing table to A, B; D sends routing table to C, G; E sends routing table to A; F sends routing table to A, G; G sends routing table to D, F.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>&lt;1,B&gt;</td>
<td>&lt;1,C&gt;</td>
<td>&lt;2,C&gt;</td>
<td>&lt;1,E&gt;</td>
<td>&lt;1,F&gt;</td>
<td>&lt;2,F&gt;</td>
</tr>
<tr>
<td>B</td>
<td>&lt;1,A&gt;</td>
<td>0</td>
<td>&lt;1,C&gt;</td>
<td>&lt;2,D&gt;</td>
<td>&lt;2,A&gt;</td>
<td>&lt;2,C&gt;</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>&lt;1,A&gt;</td>
<td>&lt;1,B&gt;</td>
<td>0</td>
<td>&lt;1,D&gt;</td>
<td>&lt;2,A&gt;</td>
<td>&lt;2,A&gt;</td>
<td>&lt;2,D&gt;</td>
</tr>
<tr>
<td>D</td>
<td>&lt;2,C&gt;</td>
<td>&lt;2,C&gt;</td>
<td>&lt;1,C&gt;</td>
<td>0</td>
<td>&lt;2,G&gt;</td>
<td>&lt;1,G&gt;</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>&lt;1,A&gt;</td>
<td>&lt;2,A&gt;</td>
<td>&lt;2,A&gt;</td>
<td>0</td>
<td>&lt;2,A&gt;</td>
<td>&lt;1,G&gt;</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>&lt;1,A&gt;</td>
<td>&lt;2,A&gt;</td>
<td>&lt;2,A&gt;</td>
<td>&lt;2,G&gt;</td>
<td>&lt;2,A&gt;</td>
<td>0</td>
<td>&lt;1,G&gt;</td>
</tr>
<tr>
<td>G</td>
<td>&lt;2,F&gt;</td>
<td></td>
<td>&lt;2,D&gt;</td>
<td>&lt;1,D&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Round 2.**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>&lt;1,B&gt;</td>
<td>&lt;1,C&gt;</td>
<td>&lt;2,C&gt;</td>
<td>&lt;1,E&gt;</td>
<td>&lt;1,F&gt;</td>
<td>&lt;2,F&gt;</td>
</tr>
<tr>
<td>B</td>
<td>&lt;1,A&gt;</td>
<td>0</td>
<td>&lt;1,C&gt;</td>
<td>&lt;2,D&gt;</td>
<td>&lt;2,A&gt;</td>
<td>&lt;2,C&gt;</td>
<td>&lt;3,A&gt;</td>
</tr>
<tr>
<td>C</td>
<td>&lt;1,A&gt;</td>
<td>&lt;1,B&gt;</td>
<td>0</td>
<td>&lt;1,D&gt;</td>
<td>&lt;2,A&gt;</td>
<td>&lt;2,A&gt;</td>
<td>&lt;2,D&gt;</td>
</tr>
<tr>
<td>D</td>
<td>&lt;2,C&gt;</td>
<td>&lt;2,C&gt;</td>
<td>&lt;1,C&gt;</td>
<td>0</td>
<td>&lt;3,C&gt;</td>
<td>&lt;2,G&gt;</td>
<td>&lt;1,G&gt;</td>
</tr>
<tr>
<td>E</td>
<td>&lt;1,A&gt;</td>
<td>&lt;2,A&gt;</td>
<td>&lt;2,A&gt;</td>
<td>&lt;3,A&gt;</td>
<td>0</td>
<td>&lt;2,A&gt;</td>
<td>&lt;3,A&gt;</td>
</tr>
<tr>
<td>F</td>
<td>&lt;1,A&gt;</td>
<td>&lt;2,A&gt;</td>
<td>&lt;2,A&gt;</td>
<td>&lt;2,G&gt;</td>
<td>&lt;2,A&gt;</td>
<td>0</td>
<td>&lt;1,G&gt;</td>
</tr>
<tr>
<td>G</td>
<td>&lt;2,F&gt;</td>
<td>&lt;3,F&gt;</td>
<td>&lt;2,D&gt;</td>
<td>&lt;1,D&gt;</td>
<td>&lt;3,F&gt;</td>
<td>&lt;1,F&gt;</td>
<td>0</td>
</tr>
</tbody>
</table>

Change weights, so that we don’t want to use the edge between F and G
RIP Example

Round 1.  A sends routing table to B, C, E, F; B sends routing table to A, C; C sends routing table to A, B; D sends routing table to C, G; E sends routing table to A; F sends routing table to A, G; G sends routing table to D, F.
<table>
<thead>
<tr>
<th></th>
<th></th>
<th>&lt;13,F&gt;</th>
<th></th>
<th>&lt;2,D&gt;</th>
<th></th>
<th>&lt;1,D&gt;</th>
<th></th>
<th>&lt;12,F&gt;</th>
<th></th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>G</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;1,B&gt;</td>
<td>&lt;1,C&gt;</td>
<td>&lt;2,C&gt;</td>
<td>&lt;1,E&gt;</td>
<td>&lt;1,F&gt;</td>
<td>&lt;3,C&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;1,A&gt;</td>
<td>0</td>
<td>&lt;1,C&gt;</td>
<td>&lt;2,D&gt;</td>
<td>&lt;2,A&gt;</td>
<td>&lt;2,A&gt;</td>
<td>&lt;3,C&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;1,A&gt;</td>
<td>&lt;1,B&gt;</td>
<td>0</td>
<td>&lt;1,D&gt;</td>
<td>&lt;2,A&gt;</td>
<td>&lt;2,A&gt;</td>
<td>&lt;2,D&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;2,C&gt;</td>
<td>&lt;2,C&gt;</td>
<td>&lt;1,C&gt;</td>
<td>0</td>
<td>&lt;3,C&gt;</td>
<td>&lt;3,C&gt;</td>
<td>&lt;1,G&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;1,A&gt;</td>
<td>&lt;2,A&gt;</td>
<td>&lt;2,A&gt;</td>
<td>&lt;3,A&gt;</td>
<td>0</td>
<td>&lt;2,A&gt;</td>
<td>&lt;14,A&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;1,A&gt;</td>
<td>&lt;2,A&gt;</td>
<td>&lt;2,A&gt;</td>
<td>&lt;2,G&gt;</td>
<td>&lt;2,A&gt;</td>
<td>0</td>
<td>&lt;12,G&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;2,F&gt;</td>
<td>&lt;3,F&gt;</td>
<td>&lt;2,D&gt;</td>
<td>&lt;1,D&gt;</td>
<td>&lt;14,F&gt;</td>
<td>&lt;12,F&gt;</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Round 2.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>&lt;1,B&gt;</td>
<td>&lt;1,C&gt;</td>
<td>&lt;2,C&gt;</td>
<td>&lt;1,E&gt;</td>
<td>&lt;1,F&gt;</td>
<td>&lt;3,C&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>&lt;1,A&gt;</td>
<td>0</td>
<td>&lt;1,C&gt;</td>
<td>&lt;2,D&gt;</td>
<td>&lt;2,A&gt;</td>
<td>&lt;2,A&gt;</td>
<td>&lt;3,C&gt;</td>
<td>&lt;2,A&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>&lt;1,A&gt;</td>
<td>&lt;1,B&gt;</td>
<td>0</td>
<td>&lt;1,D&gt;</td>
<td>&lt;2,A&gt;</td>
<td>&lt;2,A&gt;</td>
<td>&lt;2,D&gt;</td>
<td>&lt;2,A&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>&lt;2,C&gt;</td>
<td>&lt;2,C&gt;</td>
<td>&lt;1,C&gt;</td>
<td>0</td>
<td>&lt;3,C&gt;</td>
<td>&lt;3,C&gt;</td>
<td>&lt;1,G&gt;</td>
<td>&lt;2,A&gt;</td>
<td>&lt;3,A&gt;</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>&lt;2,A&gt;</td>
<td>&lt;2,A&gt;</td>
<td>&lt;3,A&gt;</td>
<td>0</td>
<td>&lt;2,A&gt;</td>
<td>&lt;14,A&gt;</td>
<td></td>
<td>&lt;12,G&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>&lt;1,A&gt;</td>
<td>&lt;2,A&gt;</td>
<td>&lt;2,A&gt;</td>
<td>&lt;2,G&gt;</td>
<td>&lt;2,A&gt;</td>
<td>0</td>
<td>&lt;12,G&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>&lt;2,F&gt;</td>
<td>&lt;3,F&gt;</td>
<td>&lt;2,D&gt;</td>
<td>&lt;1,D&gt;</td>
<td>&lt;14,F&gt;</td>
<td>&lt;12,F&gt;</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Round 3.