
FORE
SYSTEMS

Describing Network Link Costs

V1.3: Geoff Bennett

The Router's View of Routing

As humans we find it easy to discover routes through simple networks by just looking.

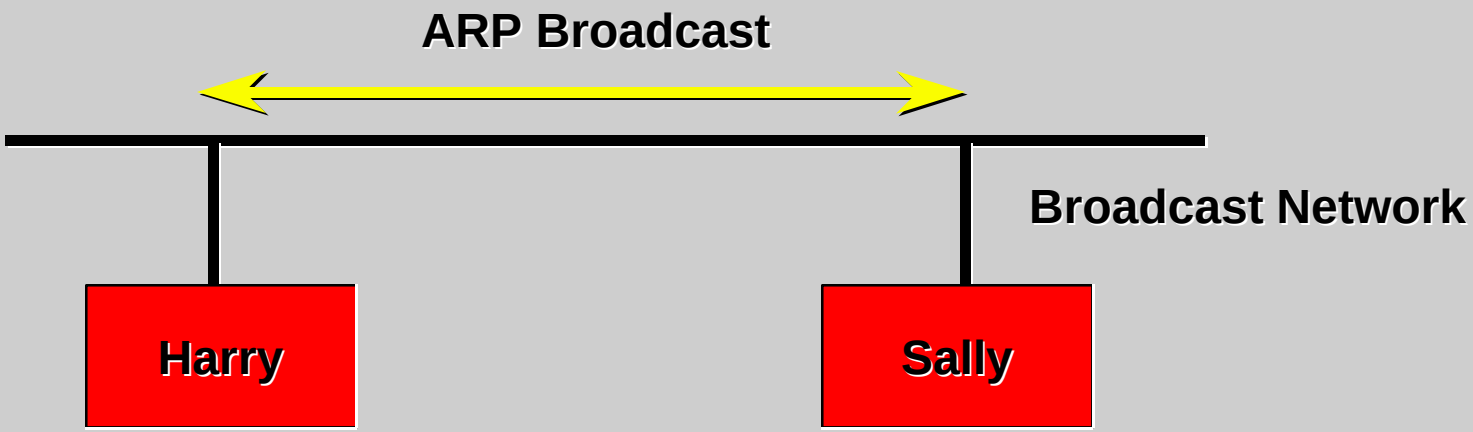
For routers to be able to do this, we need to represent network topologies in a form that a router can process mathematically. One way to do this is as a graph.

Routers also need to do calculations based on link costs, and a convenient data structure to use is a matrix.

- **Are two nodes connected directly to each other? In other words, are these two nodes ADJACENT?**
- **If two nodes are not actually adjacent, is it possible for traffic to flow from one node to another via one or more intermediate nodes? In other words, are these two nodes CONNECTED?**
- **If two nodes are connected, what is the SHORTEST PATH between them?**

Here are three concepts that can be represented using graphical methods.

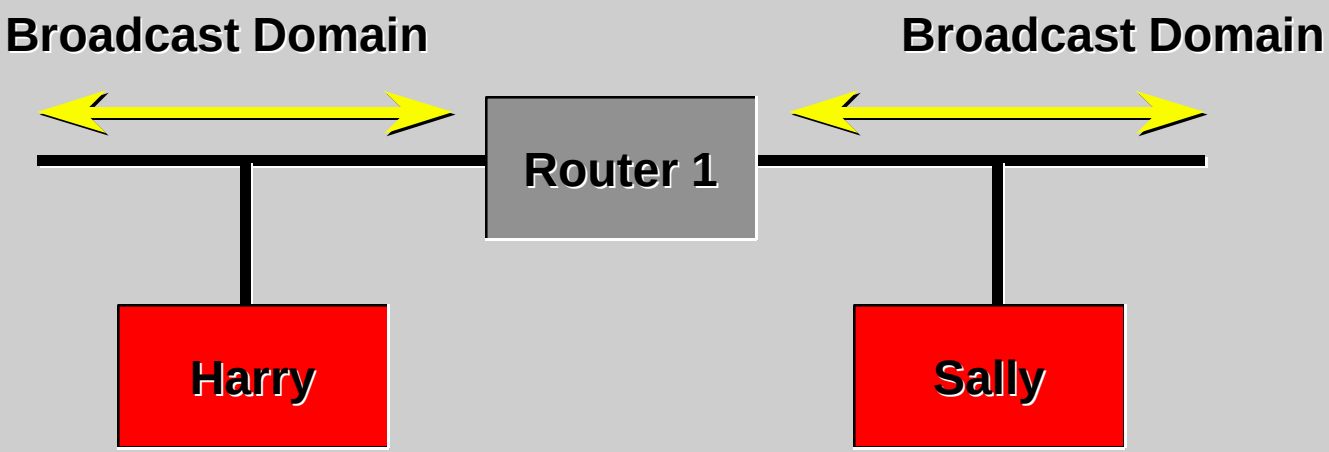
The graphical terms are rather abstract. I'd like to relate these properties more directly to network issues.



Harry and Sally are attached to an Ethernet LAN. Most LANs are broadcast networks because it's possible to send out a frame with a broadcast MAC address and this frame will reach all the nodes on the LAN.

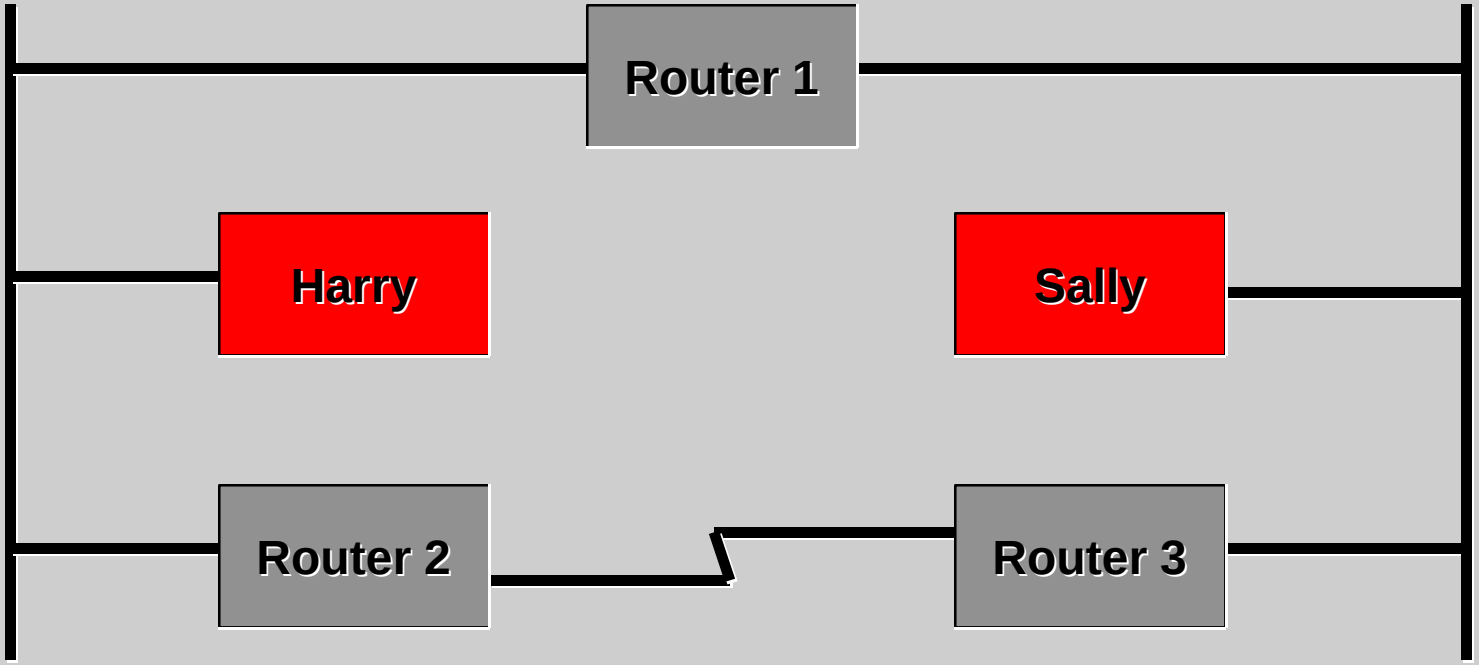
Local routing protocols such as ARP use just such a mechanism. So, in routing terms, Harry and Sally don't need an SPF algorithm because they're *adjacent.*

The extent to which the ARP broadcast will spread across the network is called the *broadcast domain*.



If Harry and Sally are *connected* by a router, then their *broadcast domain* is limited to the LAN on their side of the router. This is a vital aid to the scalability of internetworks, but it means that we need to find a way to route traffic from Harry to Sally.

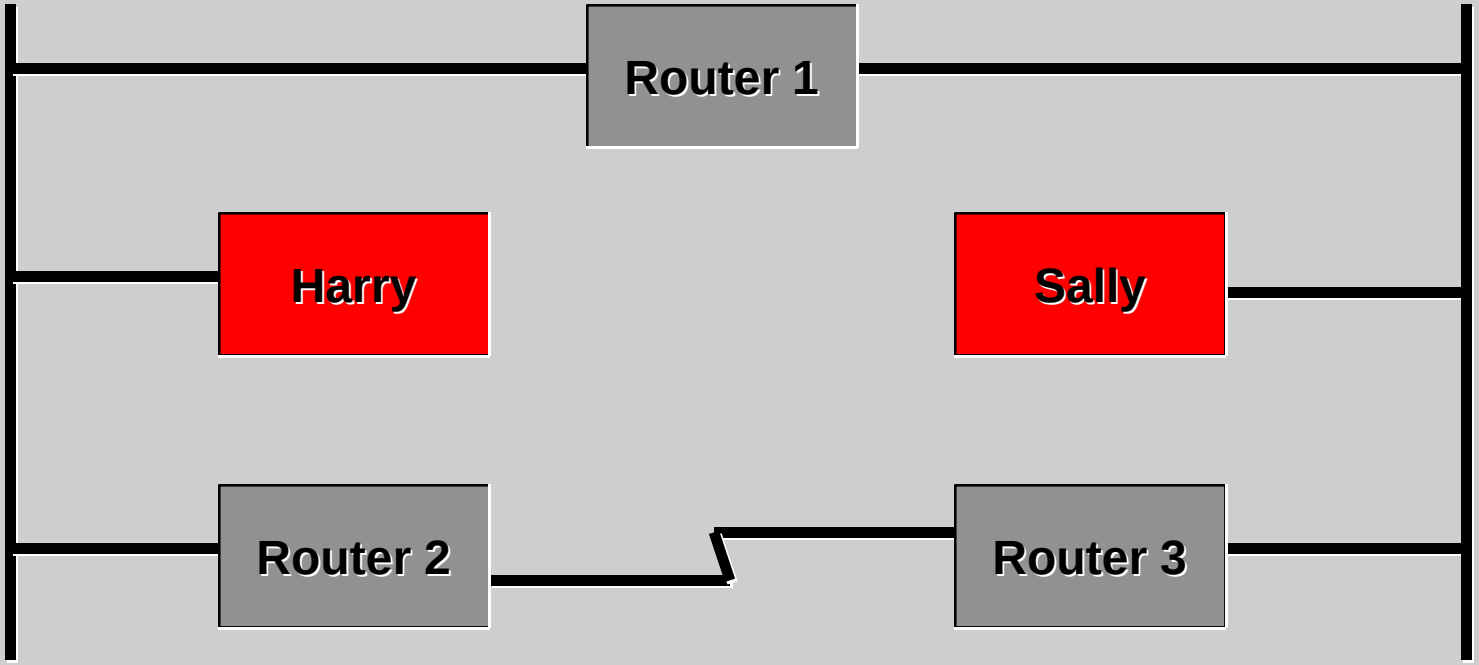
Our graphical method must allow the concept of *connection* beyond a single broadcast domain.



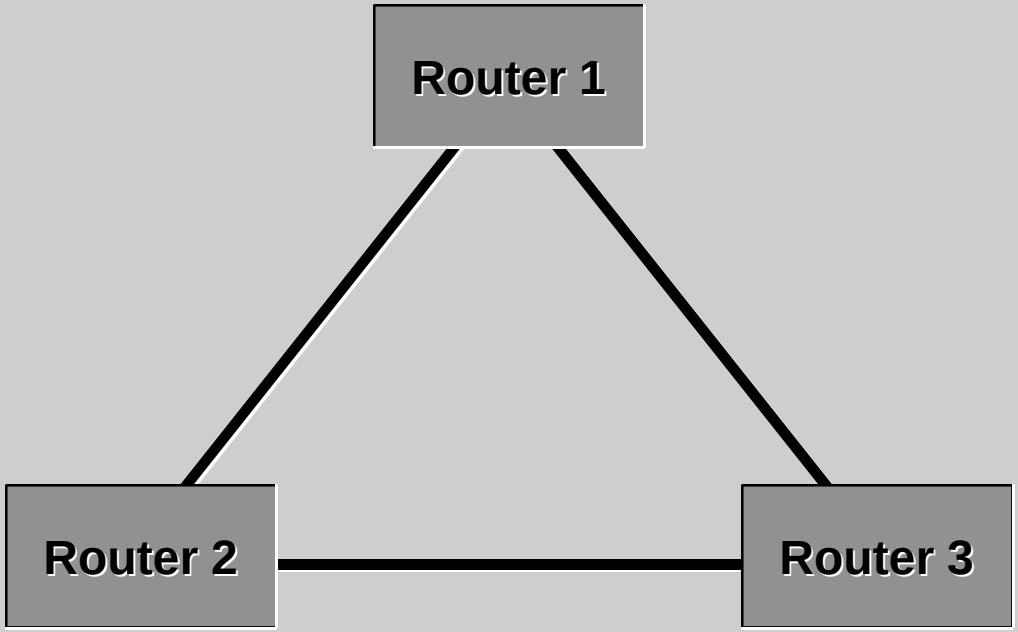
In this topology, there are two possible routes between Harry and Sally.

To enable the routers to understand the topology we must be able to indicate that Harry and Sally are *adjacent* to their respective routers, and that they are *connected* to each other by one or more router hops. Finally we need to give the routers a means to identify the *best route* or *shortest path* between the two hosts.

Hosts Are Not Routers!



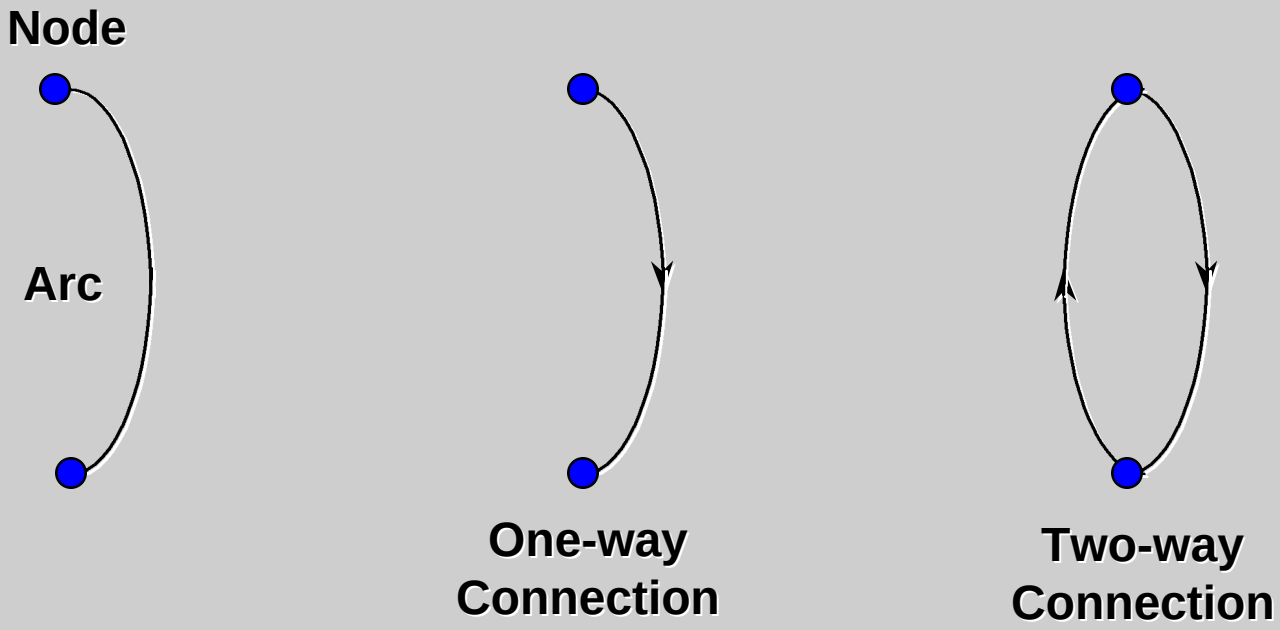
In a real router environment, the host discovers its preferred router using manual configuration, or by ICMP Router Discovery. Hosts should not take part directly in Routing Protocols such as OSPF.



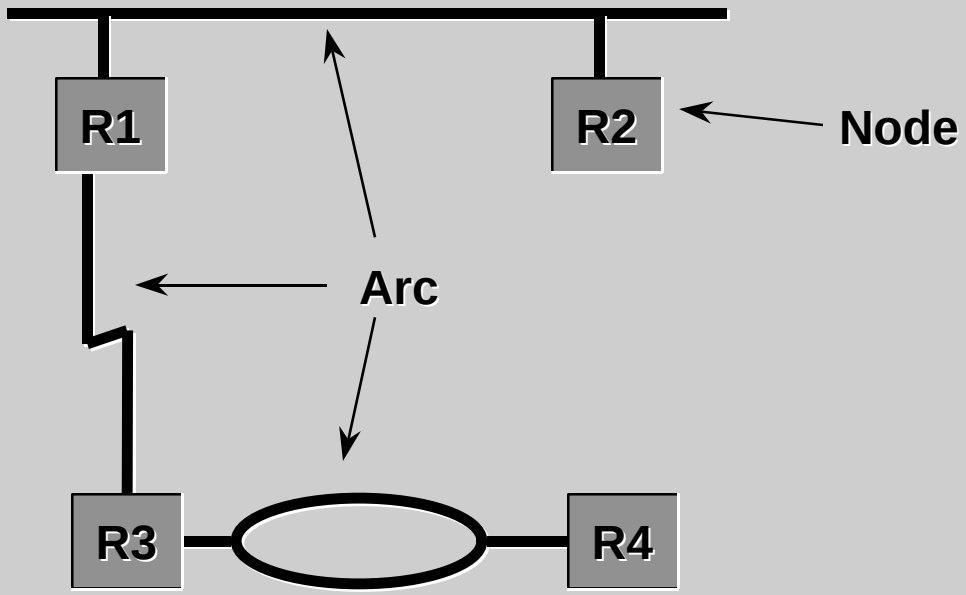
This means that we can simplify are router’s view of the world by removing the hosts.

Representing Networks as Graphs

In graphical terms, the router is a NODE, and the link between routers is an ARC.



Graph theory actually allows us to describe one-way or two-way connections between nodes.

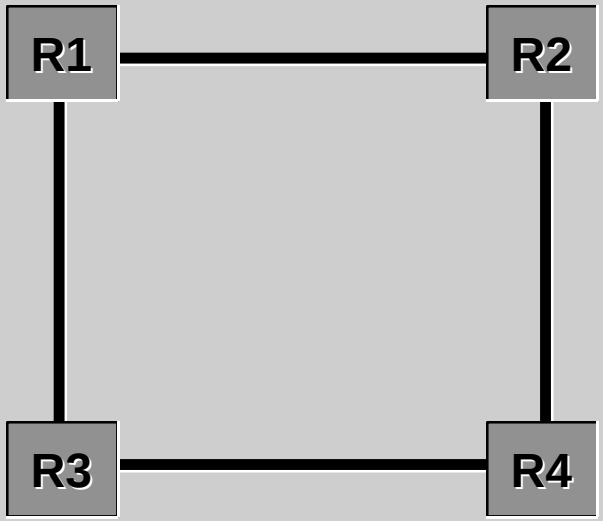


But in a real network, nodes are routers.

The arcs are LAN or WAN connections.

These connections are always two-way, and so I won't be discussing one-way connections very much.

However, a useful concept is to apply different costs for the traffic in each direction. This can be used very effectively in network designs, but it's a bit advanced for this discussion.

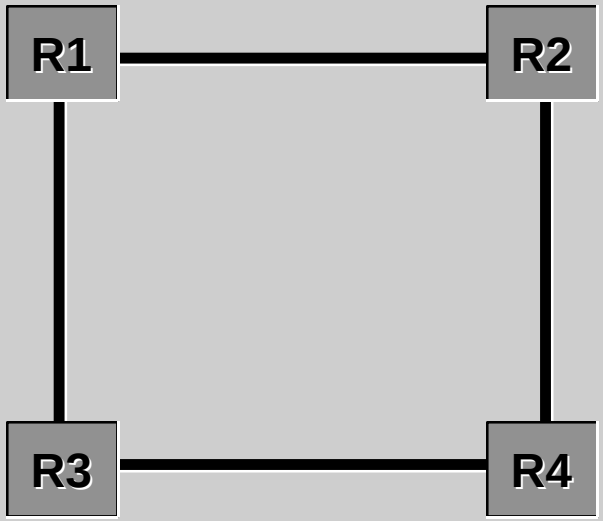


Rather than use lots of confusing indications of arcs, I'll stick to a simple concept of a point to point connection.

Representing Graphs as a Matrix

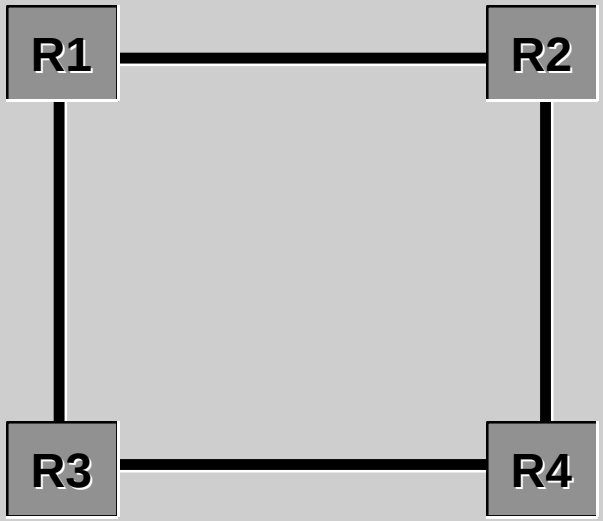
Graphs are an effective visual representation of network connections, but they are difficult to process mathematically. How can we represent the graph *inside* a router?

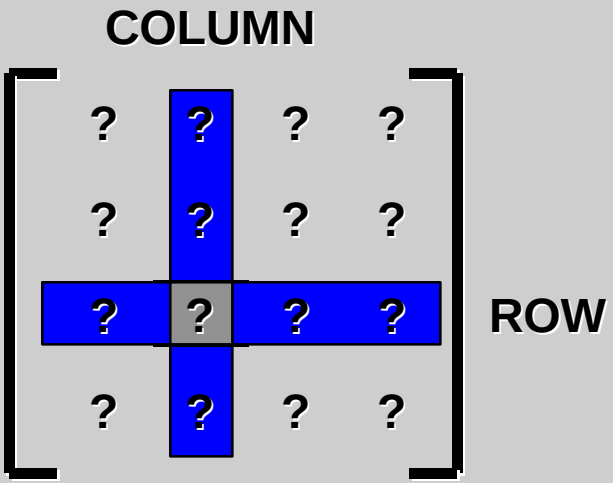
Using a matrix, we can represent the kind of graphs we'll encounter in router internetworks. Matrices are "computer-friendly" because we already have a tailor-made data structure called an *array* that is used in most computer languages.



Let’s look at the first stage of the topology problem. Using a matrix, how can we indicate that some of these routers are *adjacent*, while others are not?

There is a specific matrix type which can be used to represent the idea of connectivity between these units. It’s called an ADJACENCY MATRIX.



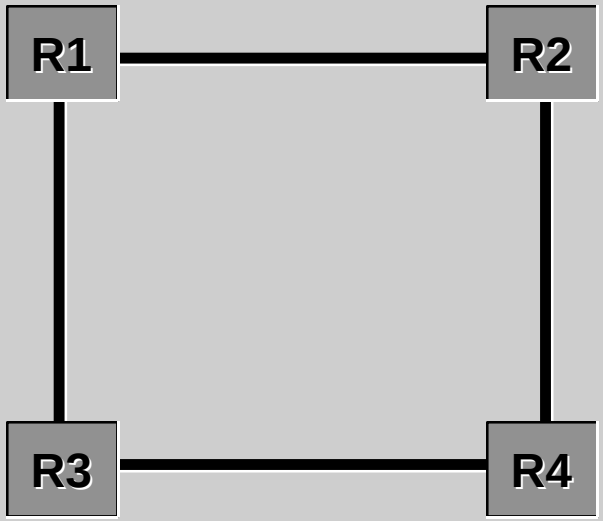


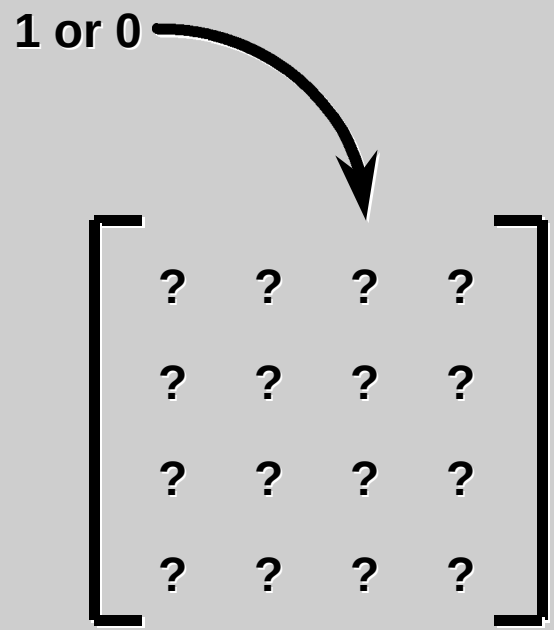
Adjacency Matrix

An adjacency matrix is always square - ie. it has the same number of rows as columns. Each node in the network has one row and one column, so for this example we would draw a 4x4 matrix because we have only 4 nodes.

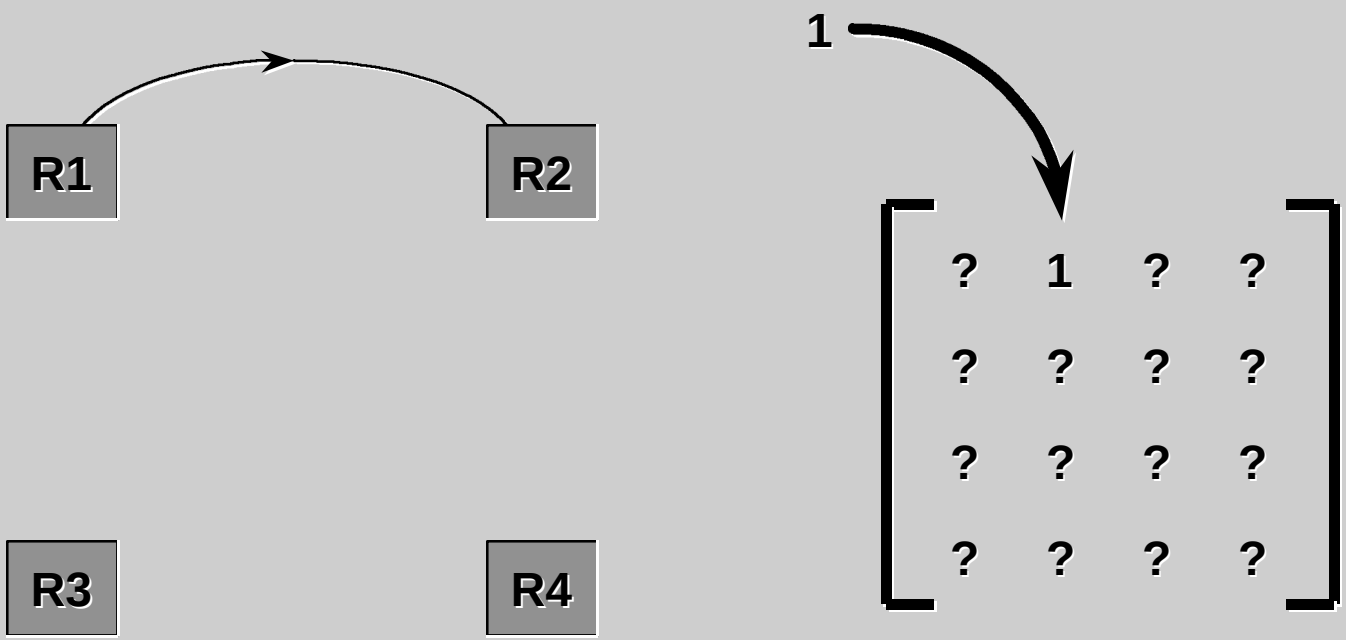
We can refer to individual matrix entries by using their row and column position. The question mark I've identified here is at position 3,4.

At this point I haven't said exactly what we'll put into the matrix.

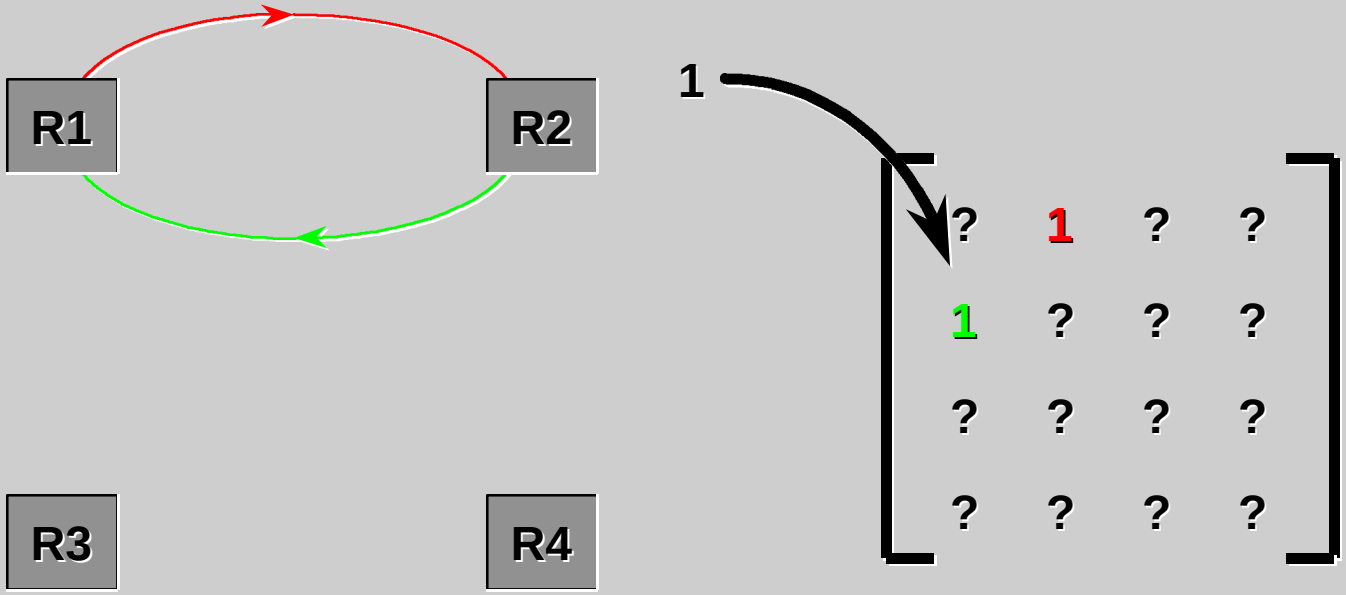




An adjacency matrix tells if two nodes are adjacent or not. So, the obvious values we can enter into the matrix are a “1” to indicate adjacency, or a “0” to indicate non-adjacency.



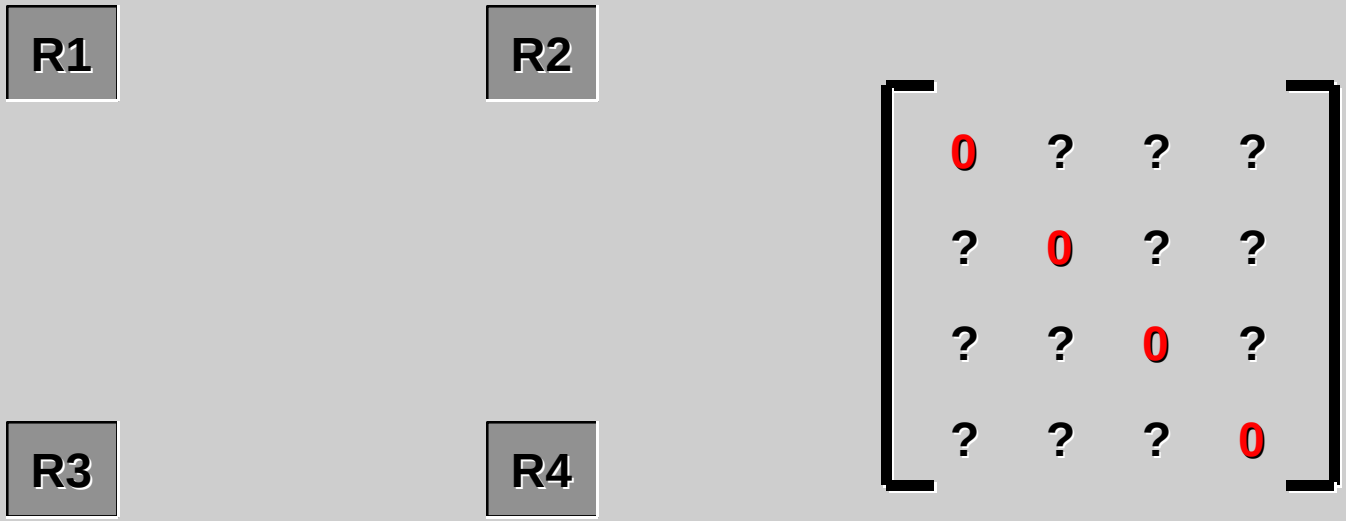
If there is an arc from R1 to R2, then we would insert a “1” into the matrix at position 1,2.



But I've already said that all the connections in the networks I'll be describing are bidirectional, and so there must also be an arc from R2 to R1.

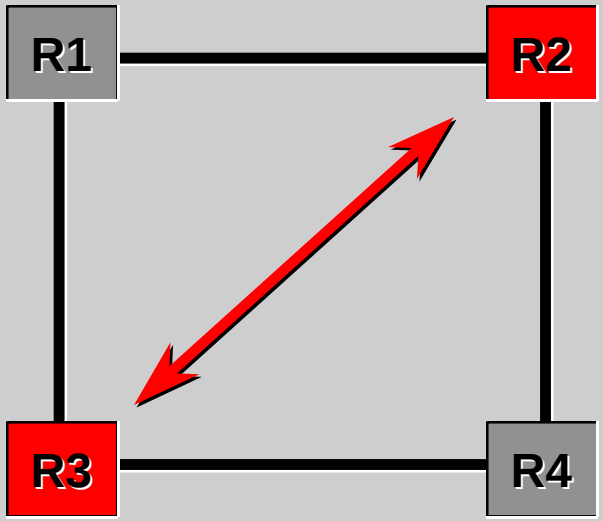
So in this case, we'd insert a 1 in position 2,1 of the matrix.

If we were really concerned about representing each direction individually, we could do it in the matrix. The red arc is represented by the red "1", and the green arc by the green "1".



There are 4 entries in this matrix - (1,1), (2,2), (3,3) and (4,4) - that indicate a node's connection to itself.

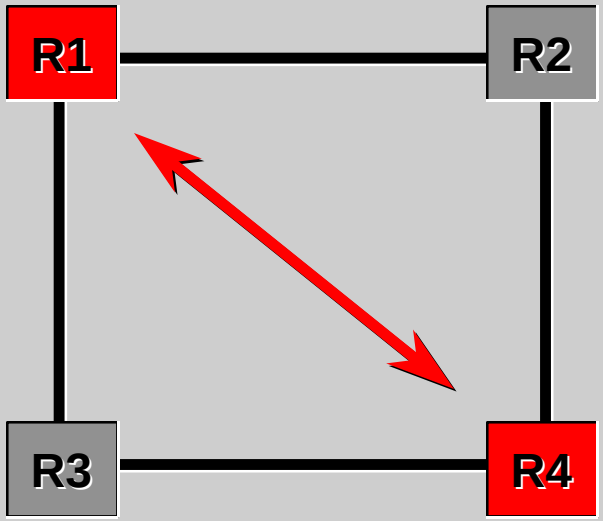
By convention, we insert a 0 at these points. In effect what we're saying is that a node cannot be adjacent to itself.



$$\begin{bmatrix} 0 & ? & ? & ? \\ ? & 0 & 0 & ? \\ ? & 0 & 0 & ? \\ ? & ? & ? & 0 \end{bmatrix}$$

Some node pairs are not adjacent.

R2 and R3...

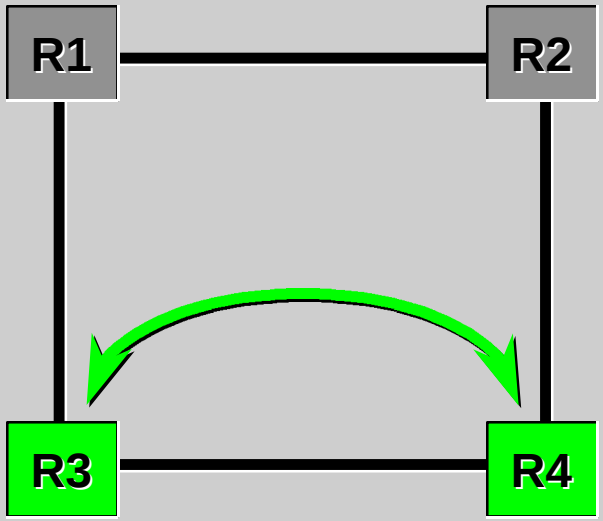


$$\begin{bmatrix} 0 & ? & ? & 0 \\ ? & 0 & 0 & ? \\ ? & 0 & 0 & ? \\ 0 & ? & ? & 0 \end{bmatrix}$$

Some node pairs are not adjacent.

R2 and R3...

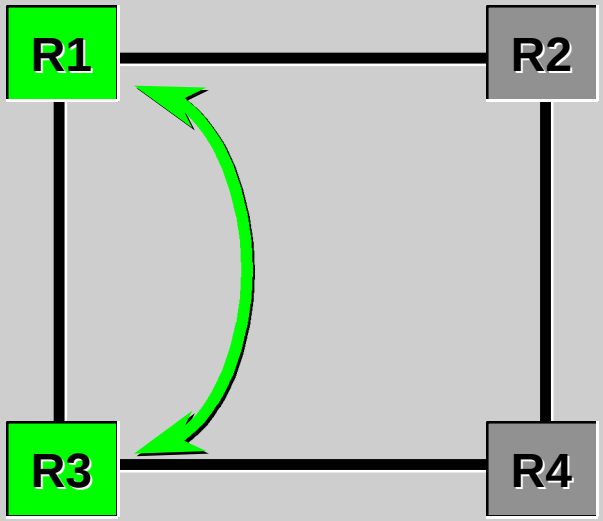
...R1 and R4.



$$\begin{bmatrix} 0 & ? & ? & 0 \\ ? & 0 & 0 & ? \\ ? & 0 & 0 & 1 \\ 0 & ? & 1 & 0 \end{bmatrix}$$

Some node pairs are adjacent.

R3 and R4...

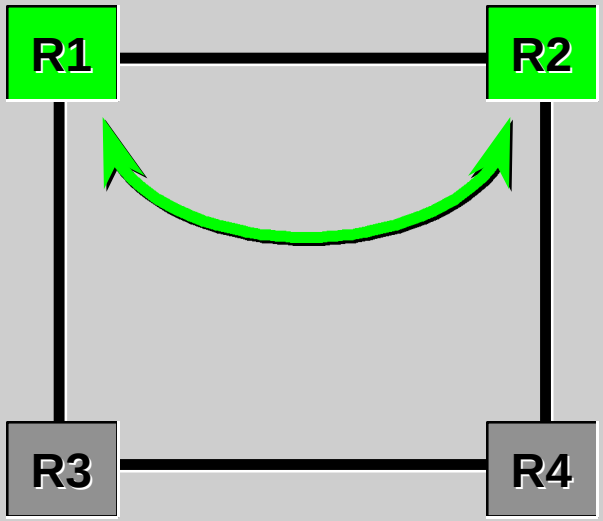


$$\begin{bmatrix} 0 & ? & 1 & 0 \\ ? & 0 & 0 & ? \\ 1 & 0 & 0 & 1 \\ 0 & ? & 1 & 0 \end{bmatrix}$$

Some node pairs are adjacent.

R3 and R4...

...R1 and R3...



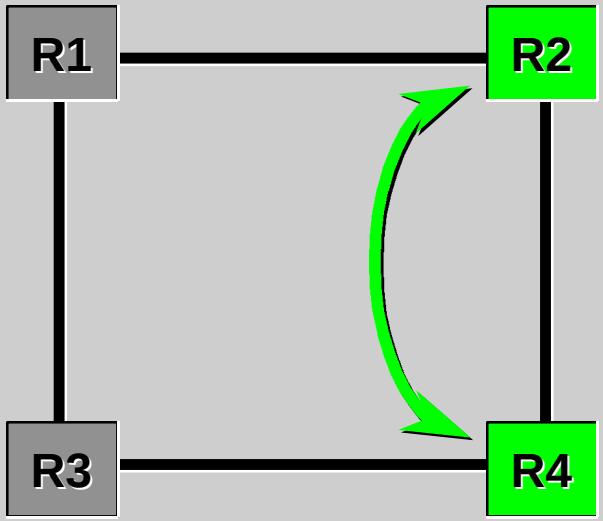
$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & ? \\ 1 & 0 & 0 & 1 \\ 0 & ? & 1 & 0 \end{bmatrix}$$

Some node pairs are adjacent.

R3 and R4...

...R1 and R3...

...R1 and R2...



$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

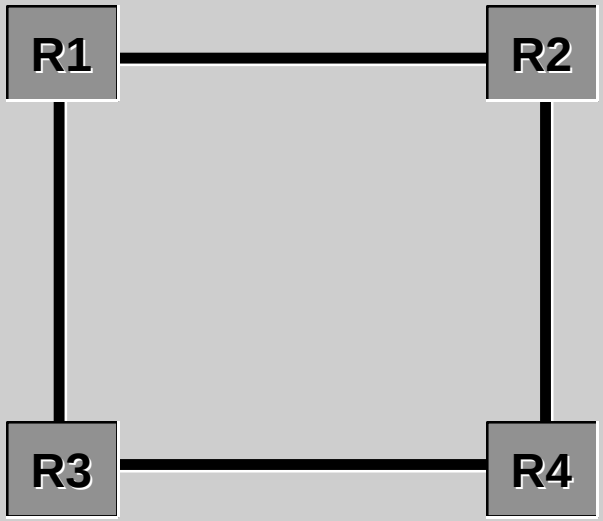
Some node pairs are adjacent.

R3 and R4...

...R1 and R3...

...R1 and R2...

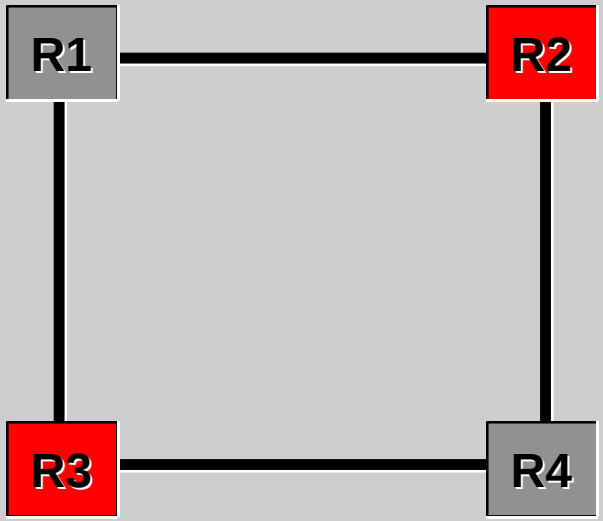
...R2 and R4.



$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

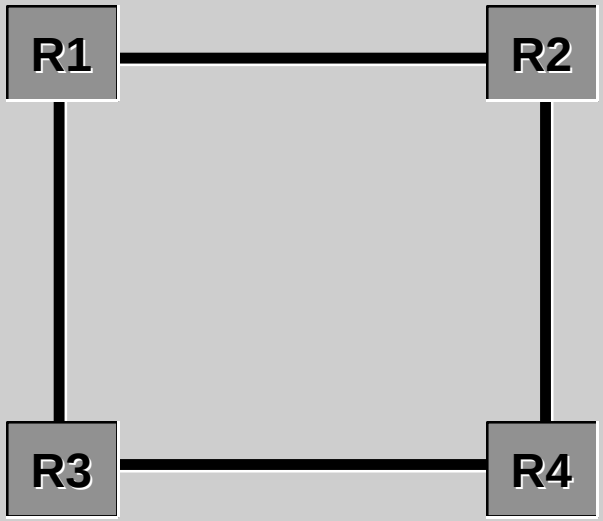
So here is the complete adjacency matrix for this network.

Using the Adjacency Matrix



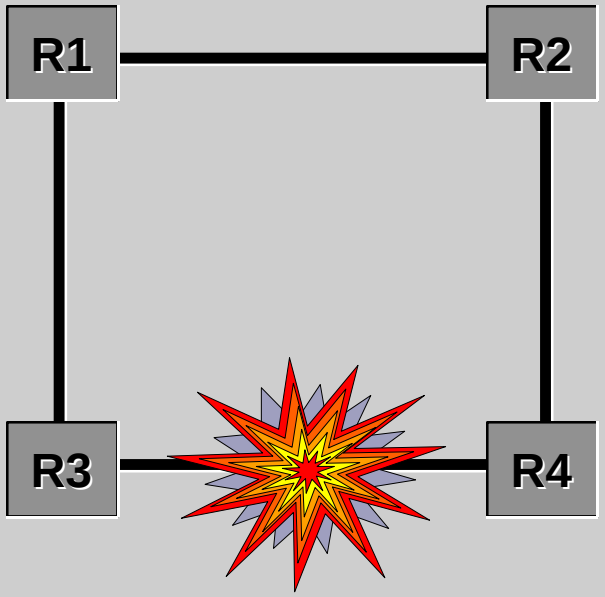
$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Looking at a couple of specific entries in the matrix, we can see that the red zeros indicate that R2 and R3 are not adjacent.



$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

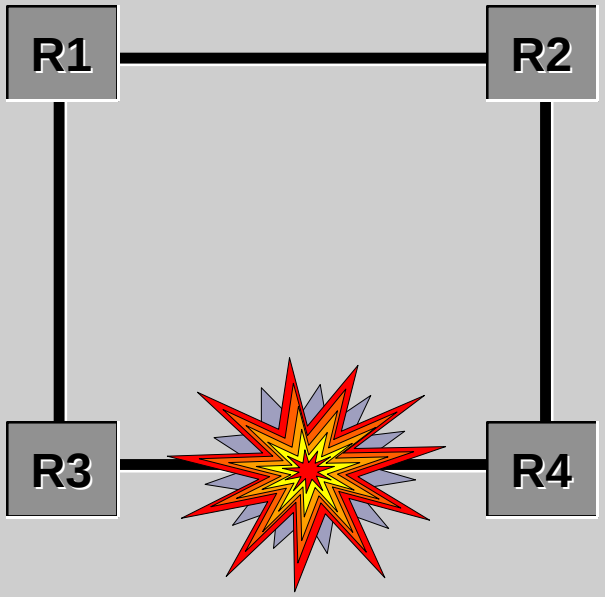
Now, to see if you understand the principle, I'm going to remove a link. This is the equivalent of a break in a connection.



Matrix Before Link Failure

$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Which of the entries in the table will change to 0?

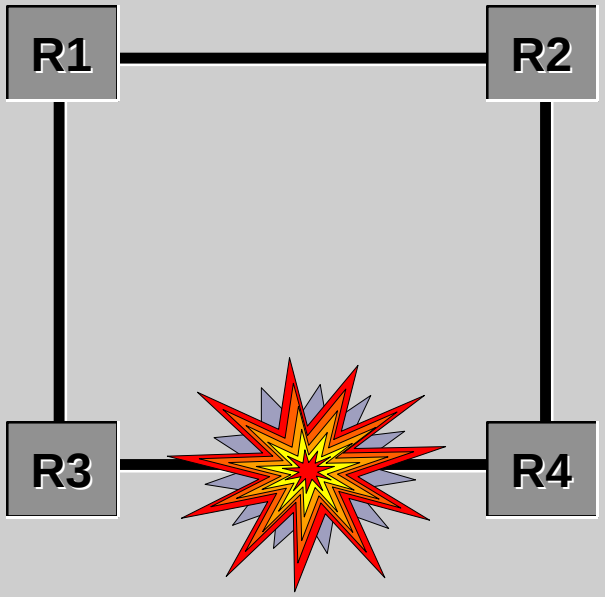


Matrix After Link Failure

$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Did you get it right?

If not, I recommend you read through the earlier slides again, because the rest of this tutorial, and the Dijkstra Algorithm Tutorial (“Finding the Shortest Path”) depend on a clear understanding of this notation.



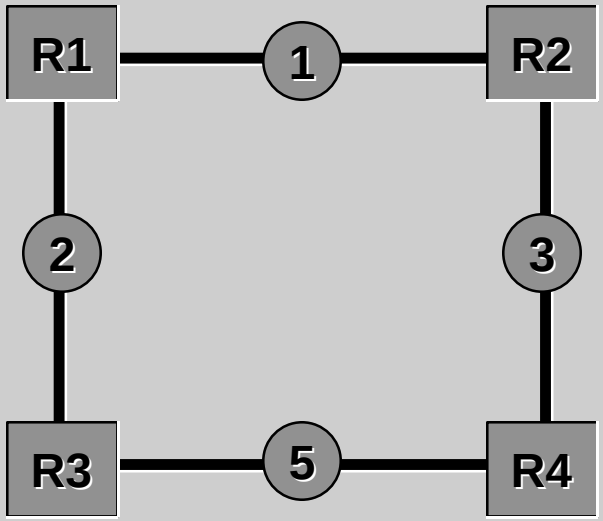
Matrix After Link Failure

$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

A key concept is that a *link state change* can be easily and quickly represented in this kind of data structure.

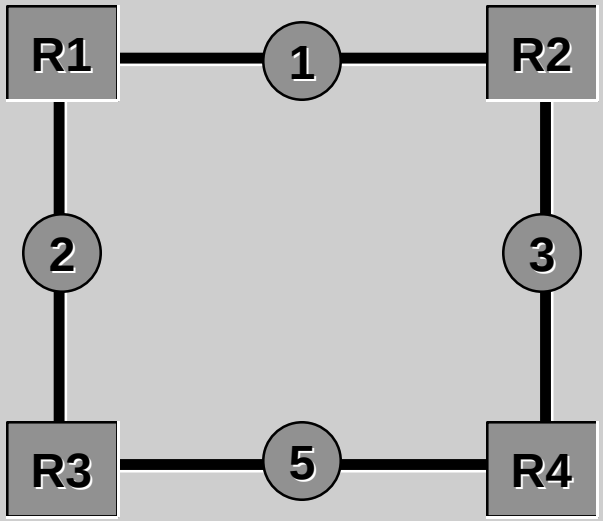
Representing the Cost of a Route in the Matrix

The adjacency matrix is a useful tool for the router, but it doesn't tell us anything about the cost of a specific route.



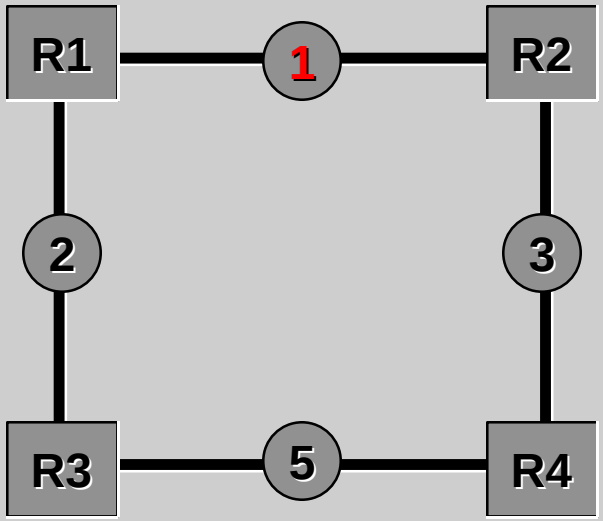
OSPF uses an abstract 16-bit metric to describe the route cost. For these examples, I'll use simple, single digit numbers instead.

In addition, I'll assume that the cost of any link is the same in both directions. However, like adjacency, we could use the matrix to represent a link with different costs in each direction.



Distance Matrix

The type of matrix used to represent route costs is called a *Distance Matrix*.

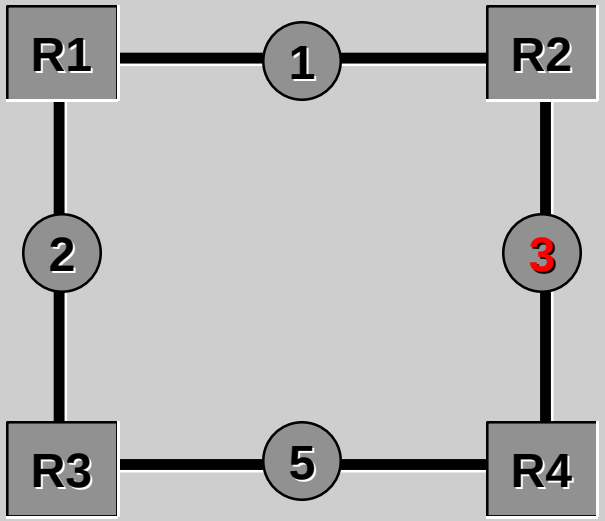


Distance Matrix

$$\begin{bmatrix} & 1 \\ 1 & \end{bmatrix}$$

The cost of the connection from R1 to R2 is one unit, so we'd enter a "1" in row 1, column 2 of the matrix.

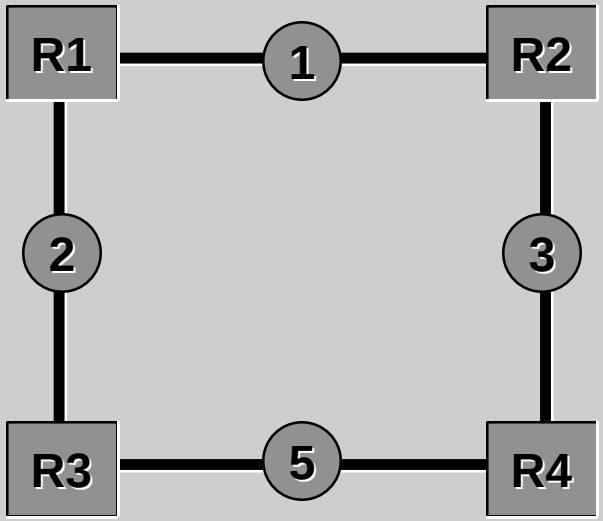
Similarly the cost of the link from R2 to R1 is one unit, so we'd also enter a "1" in row 2, column 1 of the matrix.



Distance Matrix

$$\begin{bmatrix} & 1 & \\ 1 & & 3 \\ & 3 & \end{bmatrix}$$

The cost of the bidirectional link between R2 and R4 is 3 units and so we'd enter a "3" in the matrix in row 2, column 4, and in row 4 column 2.



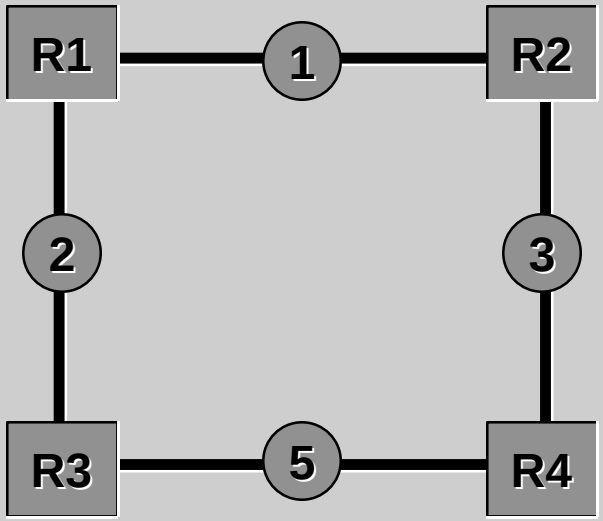
Distance Matrix

$$\begin{bmatrix} \infty & 1 & 2 & \infty \\ 1 & \infty & \infty & 3 \\ 2 & \infty & \infty & 5 \\ \infty & 3 & 5 & \infty \end{bmatrix}$$

This is the completed Distance Matrix. I've replaced the concept of a "0" entry with an infinity symbol.

What we're really saying with an entry of infinity is that the two nodes are not adjacent.

In a real router network, we need to be able to represent routes that are more than one hop. To do this with a Distance Matrix is not possible unless we can communicate links costs from one router to another.



Distance Matrix

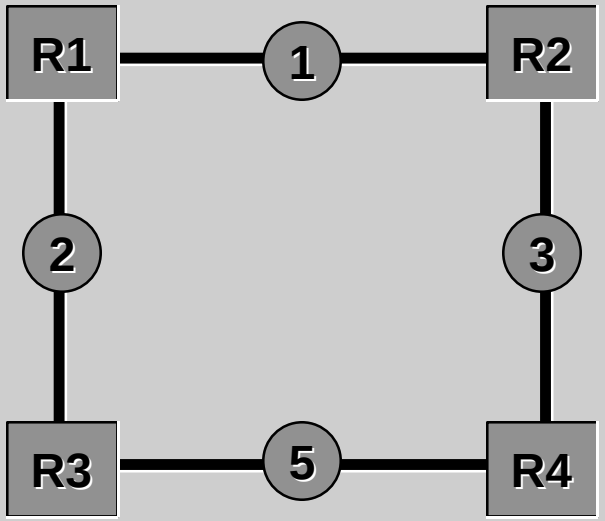
$$\begin{bmatrix} \infty & 1 & 2 & \infty \\ 1 & \infty & \infty & 3 \\ 2 & \infty & \infty & 5 \\ \infty & 3 & 5 & \infty \end{bmatrix}$$

As a human, we can see from the network diagram that there is a route from R1 to R4 via R2.

We can also see that this route is shorter that the alternative route via R3.

The router has some difficulty seeing this concept.

But, by using Routing Protocols we can propagate Distance Matrix information from one router to another.



Distance Matrix

$$\begin{bmatrix} \infty & 1 & 2 & \infty \\ 1 & \infty & \infty & 3 \\ 2 & \infty & \infty & 5 \\ \infty & 3 & 5 & \infty \end{bmatrix}$$

Also, by using a mathematical algorithm we can process the Distance Matrix to find the shortest route from one node to another.

The mathematical algorithm that is designed to do this is called the Dijkstra Algorithm.

This algorithm is covered by another tutorial in this series.

FORE SYSTEMS

The End

This concludes the tutorial.

If you aren't viewing this on the FORE Systems Web Site, then you can become a member of the ATM Academy free of charge and have access to many more of these tutorials.

You can find details at:

http://academy.fore.com/