Graphs and their applications to network modelling

Matthew Roughan
<matthew.roughan@adelaide.edu.au>
http://www.maths.adelaide.edu.au/matthew.roughan/

School of Mathematical Sciences,
University of Adelaide

July 17, 2013

A "Graph"

Mathematical Definition

- a set of **nodes** $N = \{1, 2, \ldots, n\}$ (also called **vertices**)

$$\left|N\right| = n$$

- a set of **links** $E \subseteq N \times N$ (also called **edges**)

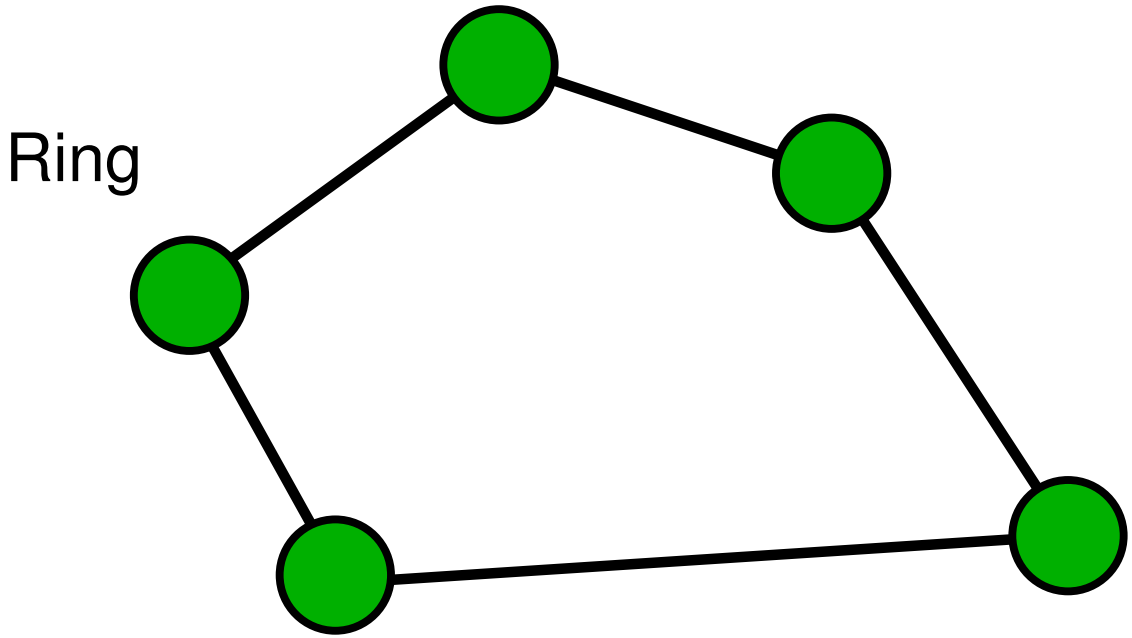
$$\begin{aligned} E &\subseteq \left\{(i,j) \;\middle|\; i,j \in N, i \neq j\right\} \\ \left|E\right| &\leq n(n-1)/2 \end{aligned}$$

- direction
  - undirected graph: links are symmetric
    - ★ $(i,j) \in E \Rightarrow (j,i) \in E$
  - directed graph: direction of link matters
- The network is defined by the **graph**,

$$G(N, E)$$

Examples

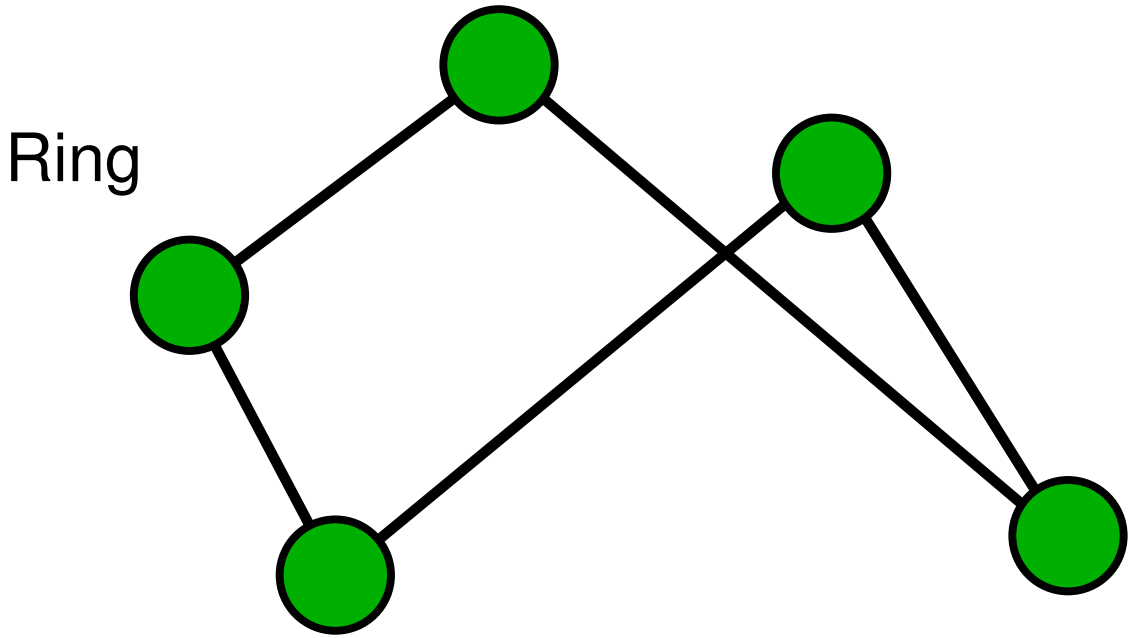
Ring



description: Every node has exactly two branches connected to it, so that they form a (logical) ring.

Examples

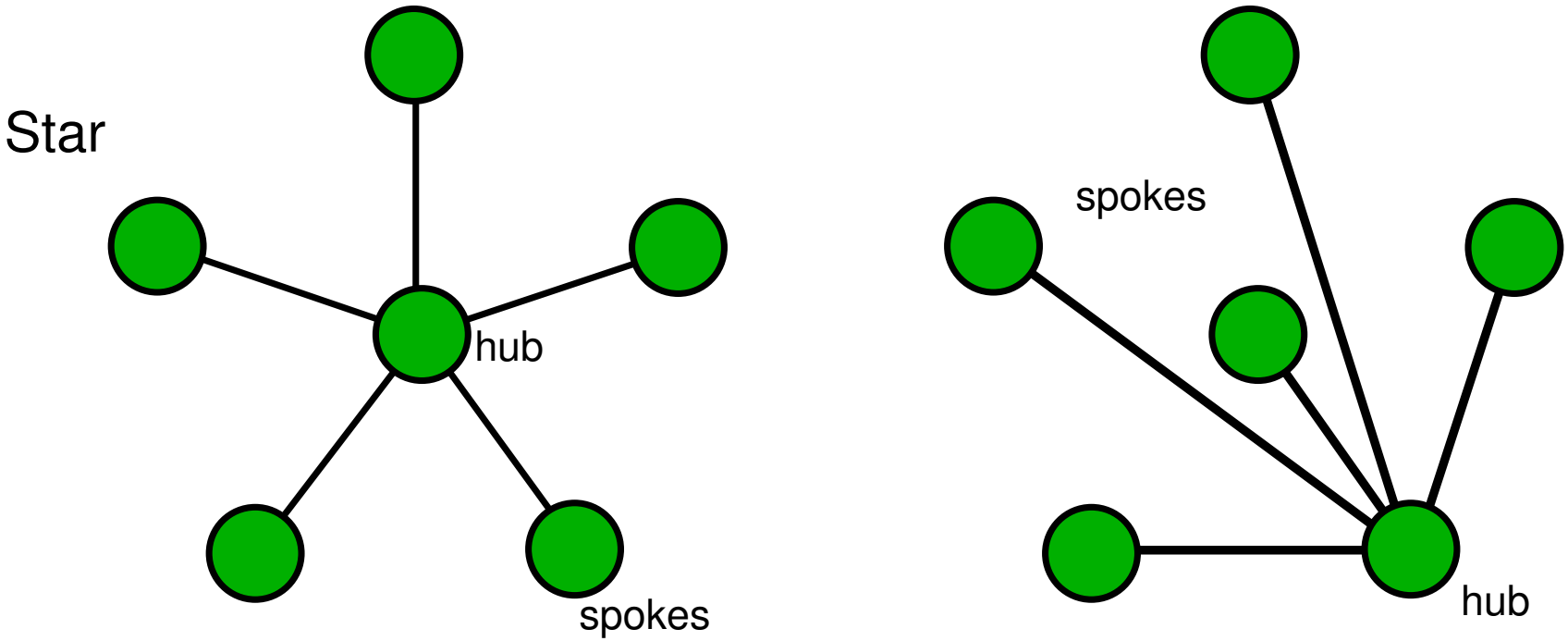
Ring



description: Every node has exactly two branches connected to it, so that they form a (logical) ring.

More Examples

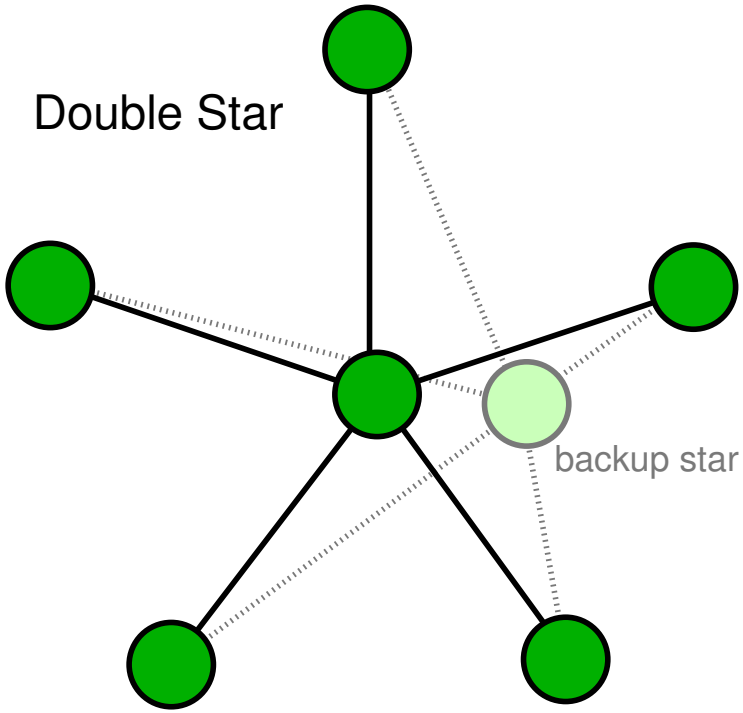
Star, or Hub and Spoke



description: peripheral (spoke) nodes are connected to a central (hub) node.

More Examples

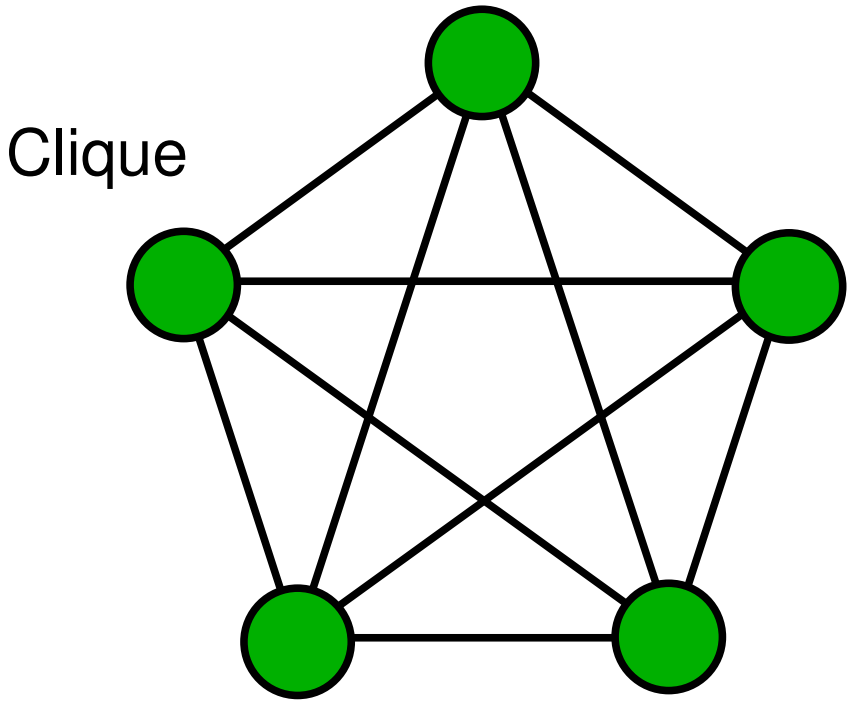
Star, or Hub and Spoke



description: peripheral (spoke) nodes are connected to a central (hub) node.

More Examples

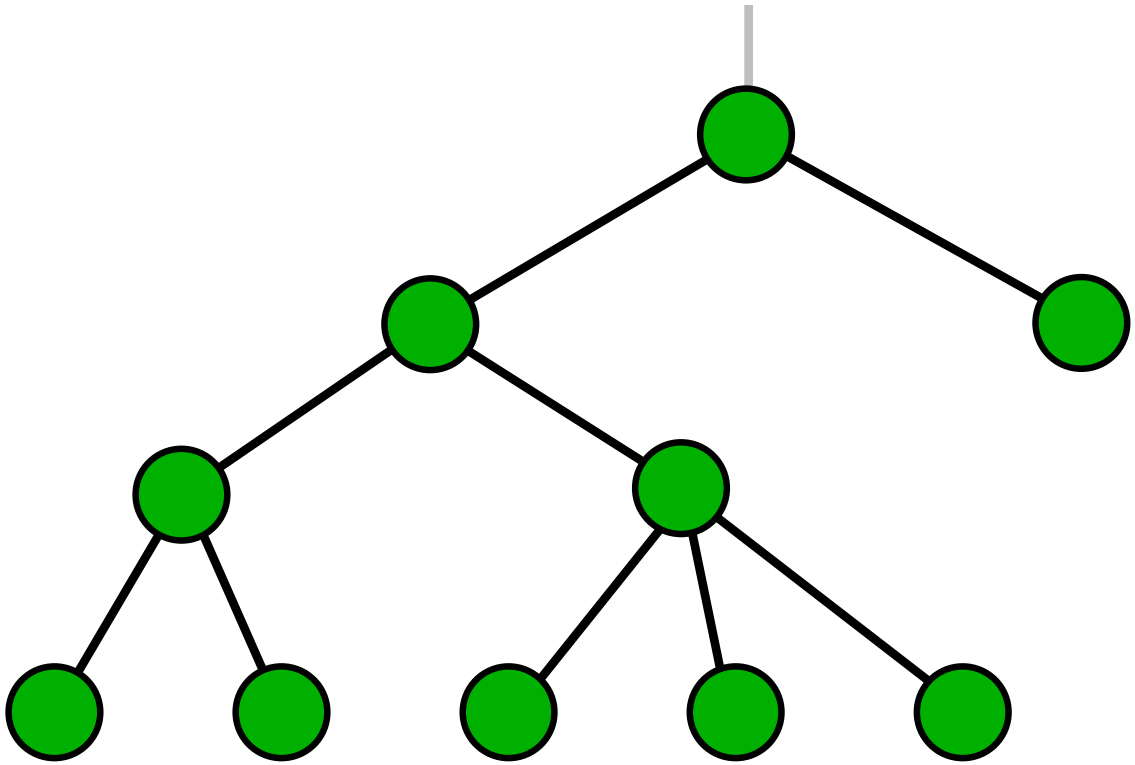
Fully connected or Clique



description: every node directly connected to every other node.

More Examples

Tree



description: nodes are arranged as a tree (no loops)

How can we use graphs?

Consider the following well known problem:

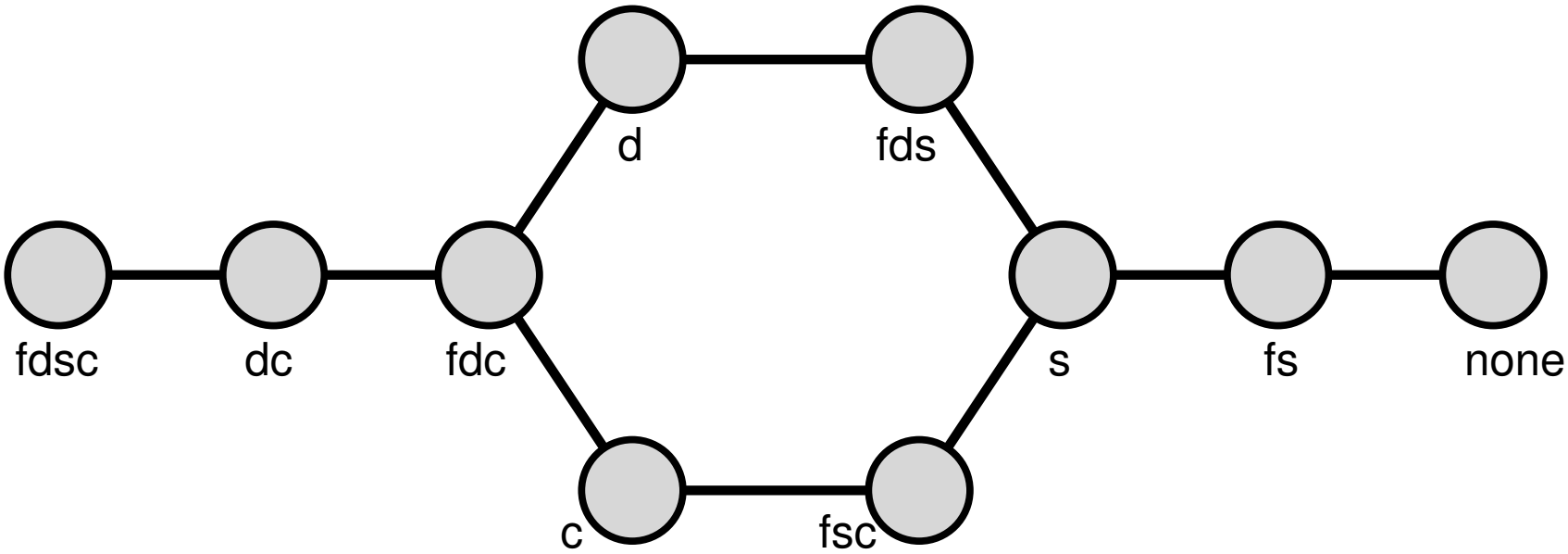
> *A ferry-man has been charged with taking a dog, a sheep, and a cabbage across the river. His rowboat can only take one at a time (plus himself). He cannot leave the dog with the sheep, or the sheep with the cabbage. How many ways can he make the transfer without repeating himself.*

Denote

- Ferry-man (f)
- Dog (d)
- Sheep (s)
- Cabbage (c)

Paths example

Label each possible state by who is on the left bank of the river. The following graph shows the only possible transitions.



There are two possible loop-free paths the ferry-man can take.

How can we use graphs

Star Trek

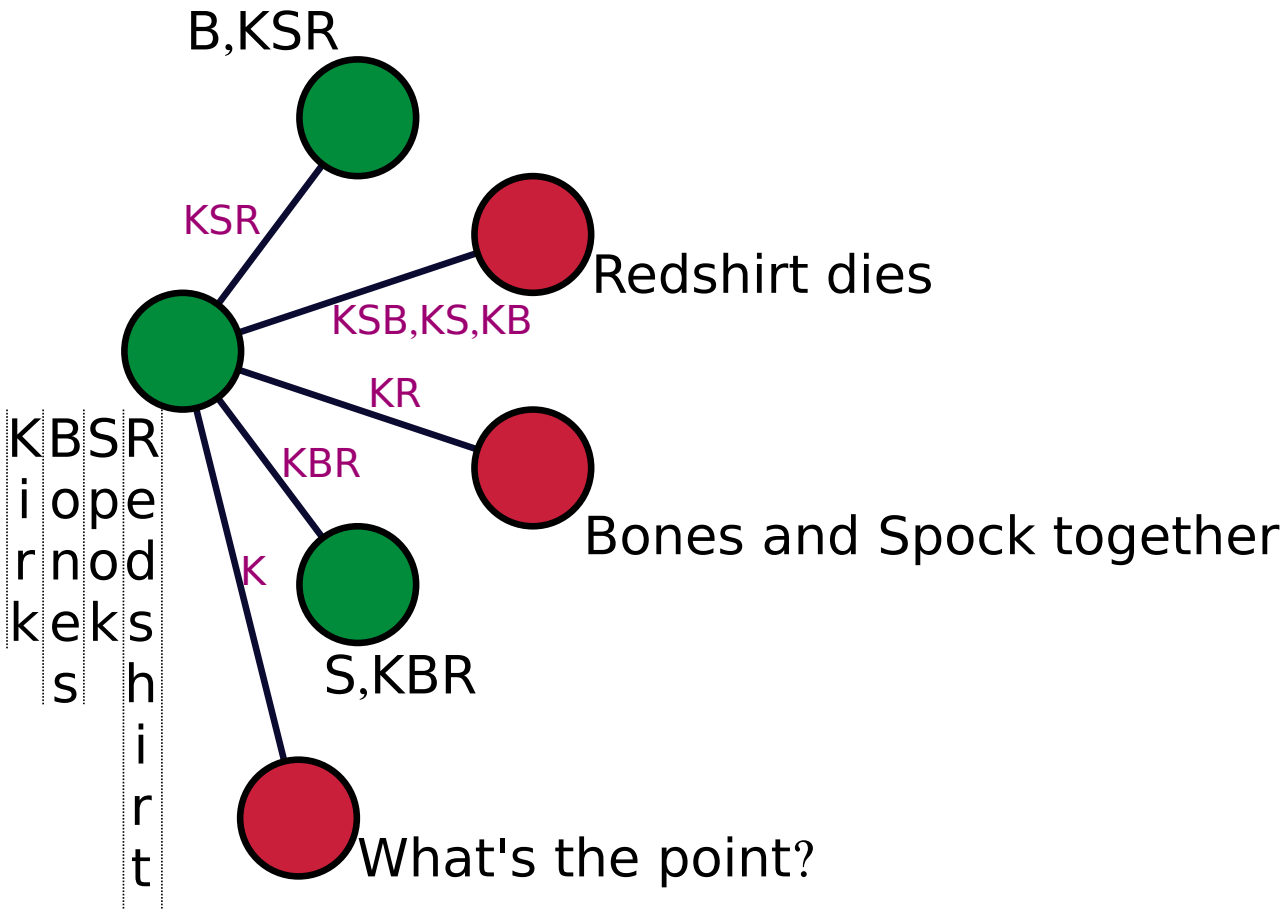
Captain Kirk has to get his away team up to an unmanned satellite from a planet before it explodes. The transporters are broken, and his shuttle is damaged — only Kirk can fly it, and he can take only two passengers. His away team is McCoy, Spock and one Redshirt.

- He can't leave Spock and McCoy together, because they fight.
- Bones is scared of being left alone on the Satellite.
- Kirk can't leave the Redshirt's side, because he will automatically be killed by some nasty plot device intended to make us understand the nastiness of the situation Kirk finds himself in. Anyone with the Redshirt (apart from Kirk himself) will also die.

How quickly can he get his crew to the satellite (alive please)?

Your turn

Start Trek



How can we use graphs?

- Scheduling
  - graph explains complex relationships, and constraints
- Representing complex relationships
  - social networks, e.g., Facebook
  - neurons in the brain
- Analysing complex processes
  - e.g. a Markov chain is often represented as a graph
- Analysing and designing computer networks

Network Topology

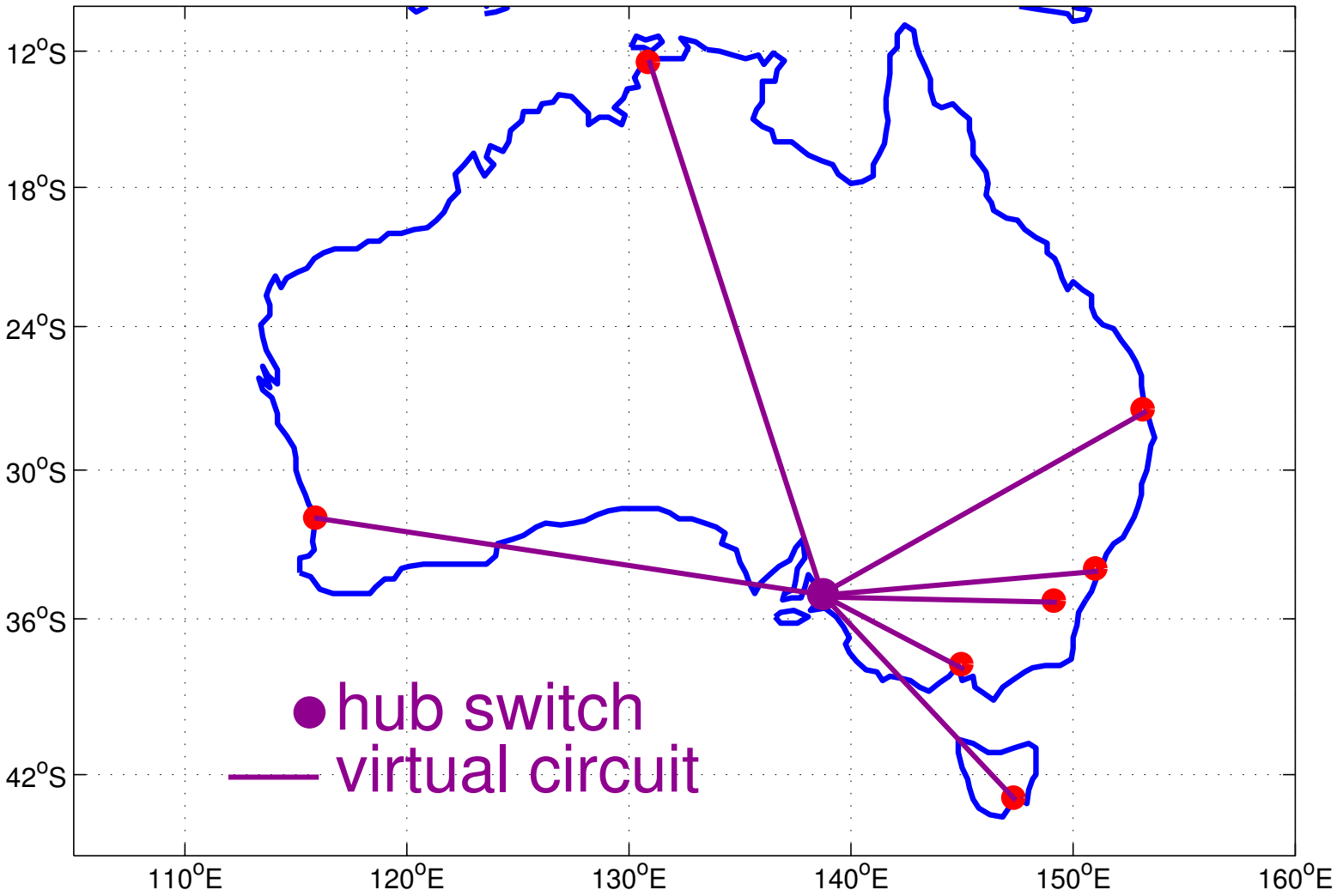
Routers and links

- The Internet is made up of devices (switches and routers)
- Connected by links
- So we represent it as a graph, where routers are vertices, and links are edges, and we call this the network topology

A router

A Juniper router in use.

An example networks



The Internet Topology Zoo

We've been collecting and transcribing networks into a common data format and making them available to people to play with:

http://www.topology-zoo.org/

For more info see New Scientist, 12th April 2013

GML

- Networks in the Zoo are stored in GML files
  - easy to read and write

```
graph [
  various_attributes "with values"
  node [
    id 0
  ]
  node [
    id 1
  ]
  edge [
    source 1
    target 0
  ]
]
```

- We can have a play with yED
  - download a map from the Zoo, open yED and edit away

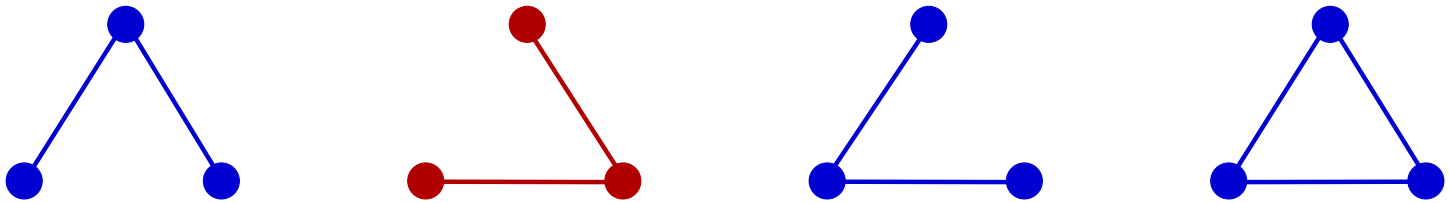
Why bother

Applications of graphs

- Network Analysis
  - e.g. debugging
- Optimisation
  - e.g. planning networks
- Routing
  - e.g. how to find your way through a network

Simple optimization example

Three node network has three acceptable designs:



Cost	2.2	2.0	2.4	3.1

- 4 possible network designs
  - associated costs have been worked out for each
- easy to choose the second network as the cheapest

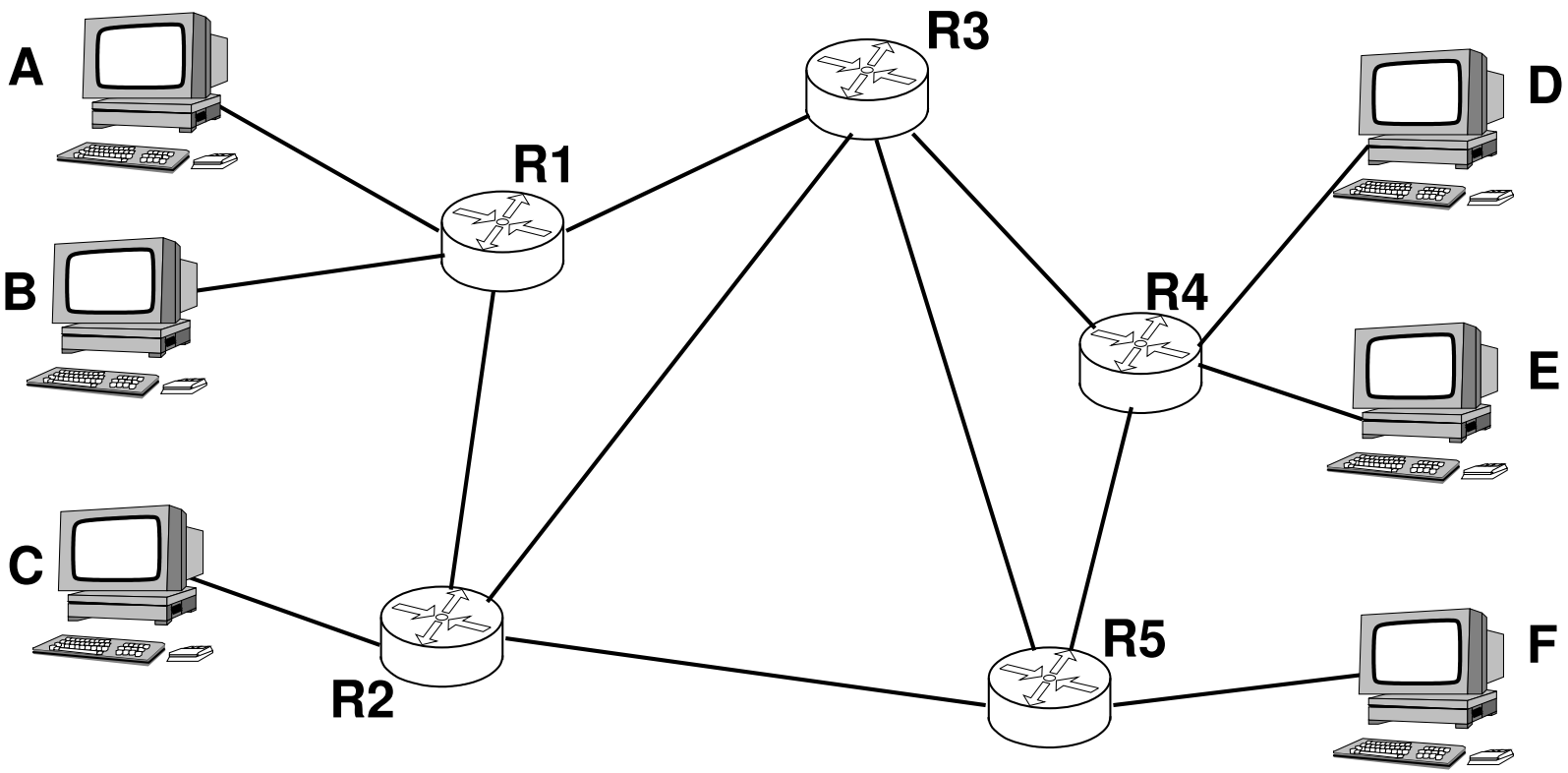
Bigger problems

- Node with n nodes
  - for small n we can evaluate all designs, and choose the best
- But $2^{n(n-1)/2}$ possible network designs
  - some aren't practical
  - but we still have to check that
- Even for $n = 20$ we can't evaluate all of these
  - at least not in the life-time of the Universe

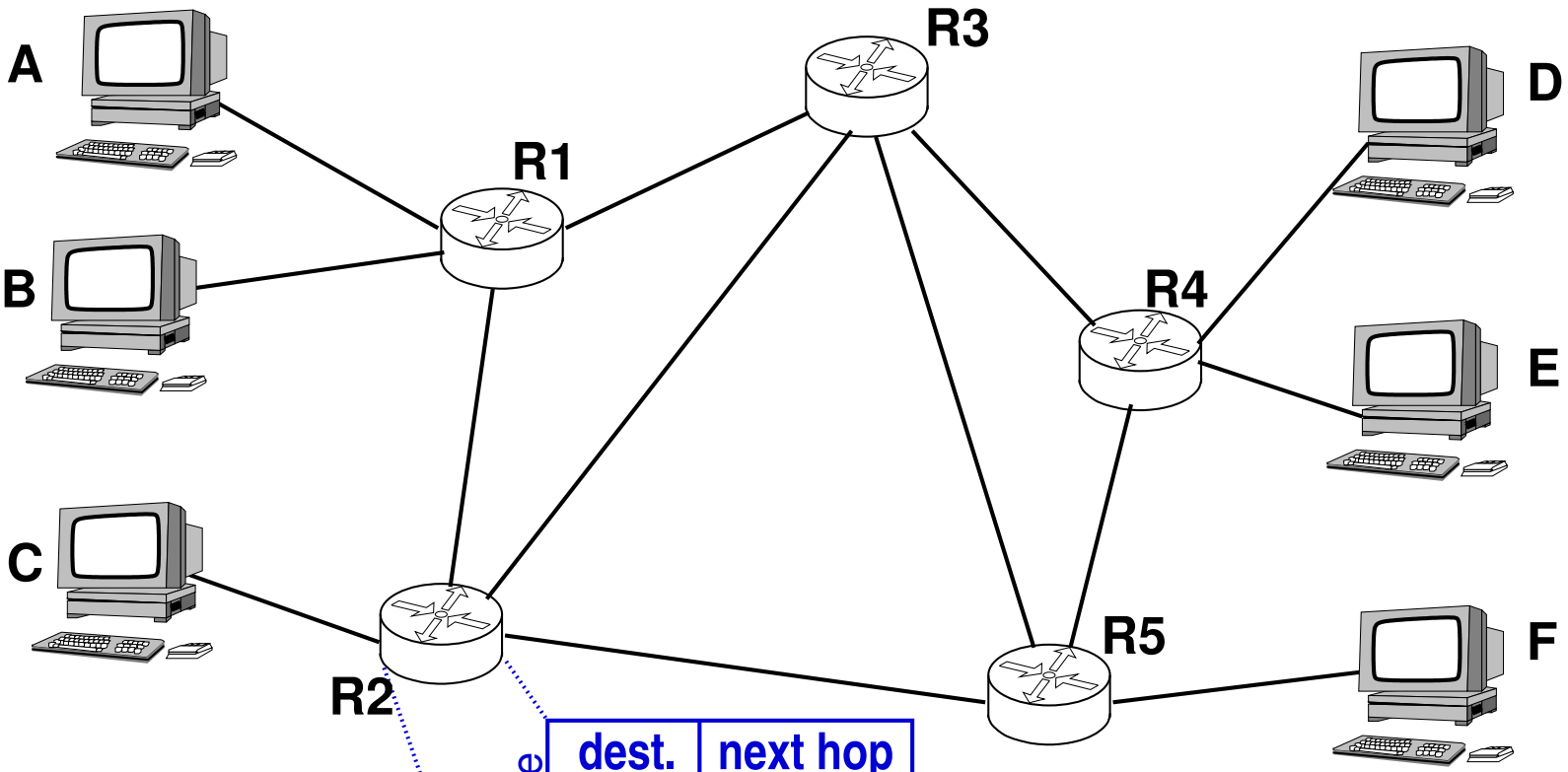
Forwarding

- In the real world, we find things using a map
  - look at the map, and plot a route
- The Internet is too big, and changes too often
  - there are no complete, up-to-date maps
  - do how do we find how to get somewhere?
- Routers keep a list of destinations, and how to get to each
  - packets are forwarded hop by hop
  - its a lot like the post office

Forwarding



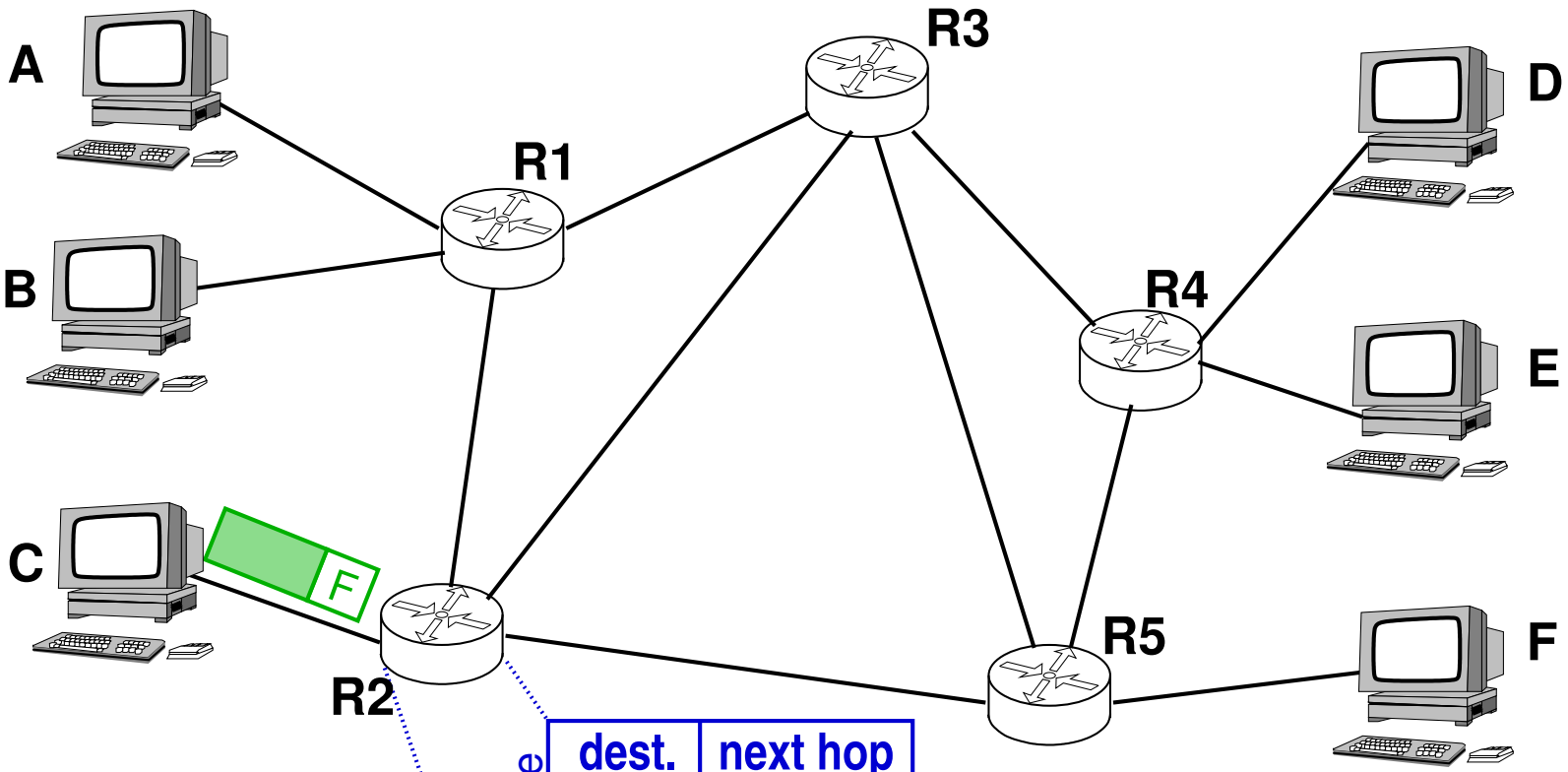
Forwarding



routing table

dest.	next hop
A	R1
B	R1
C	C
D	R3
E	R3
F	R5
default	R3

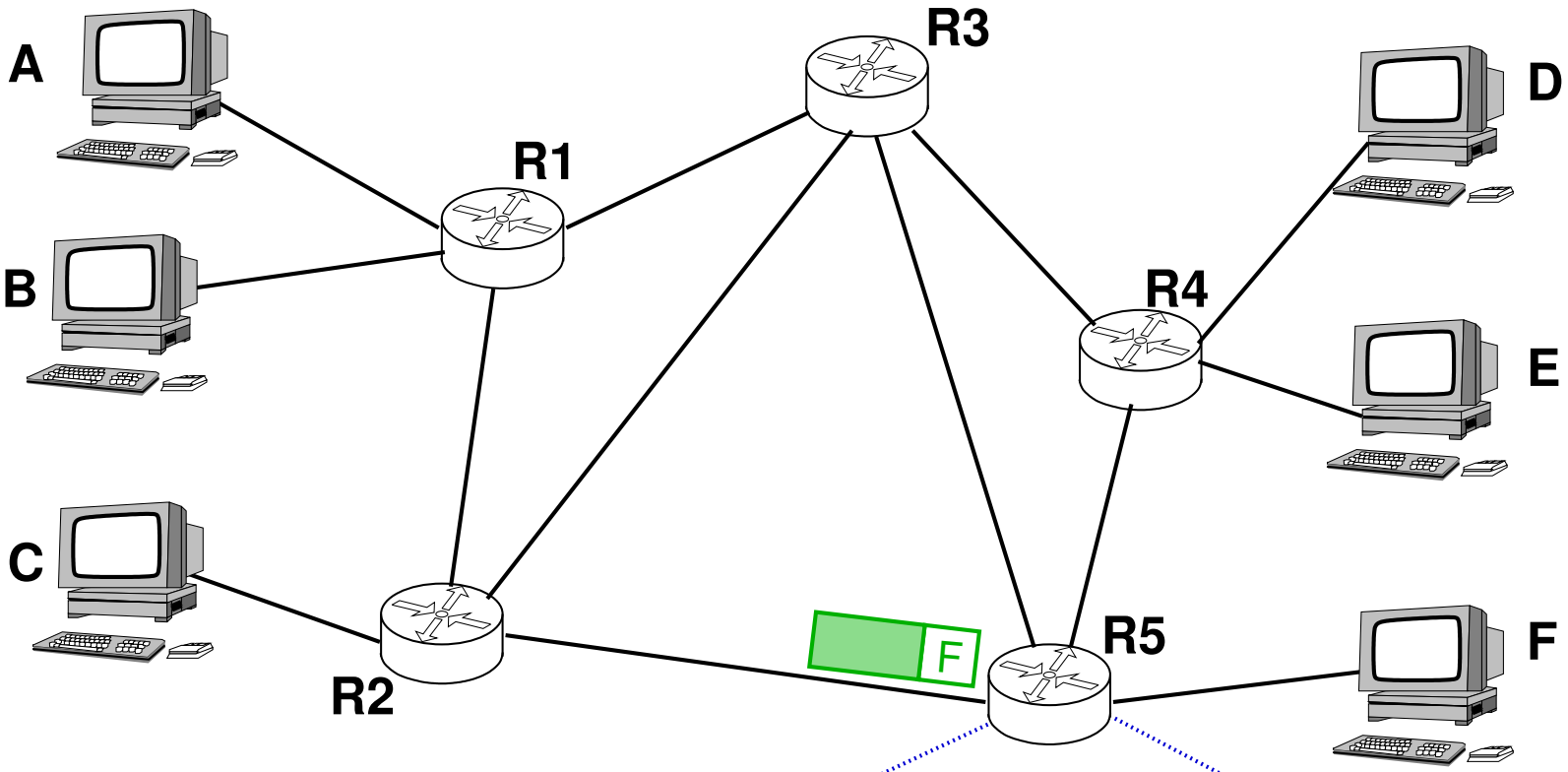
Forwarding



routing table

dest.	next hop
A	R1
B	R1
C	C
D	R3
E	R3
F	R5
default	R3

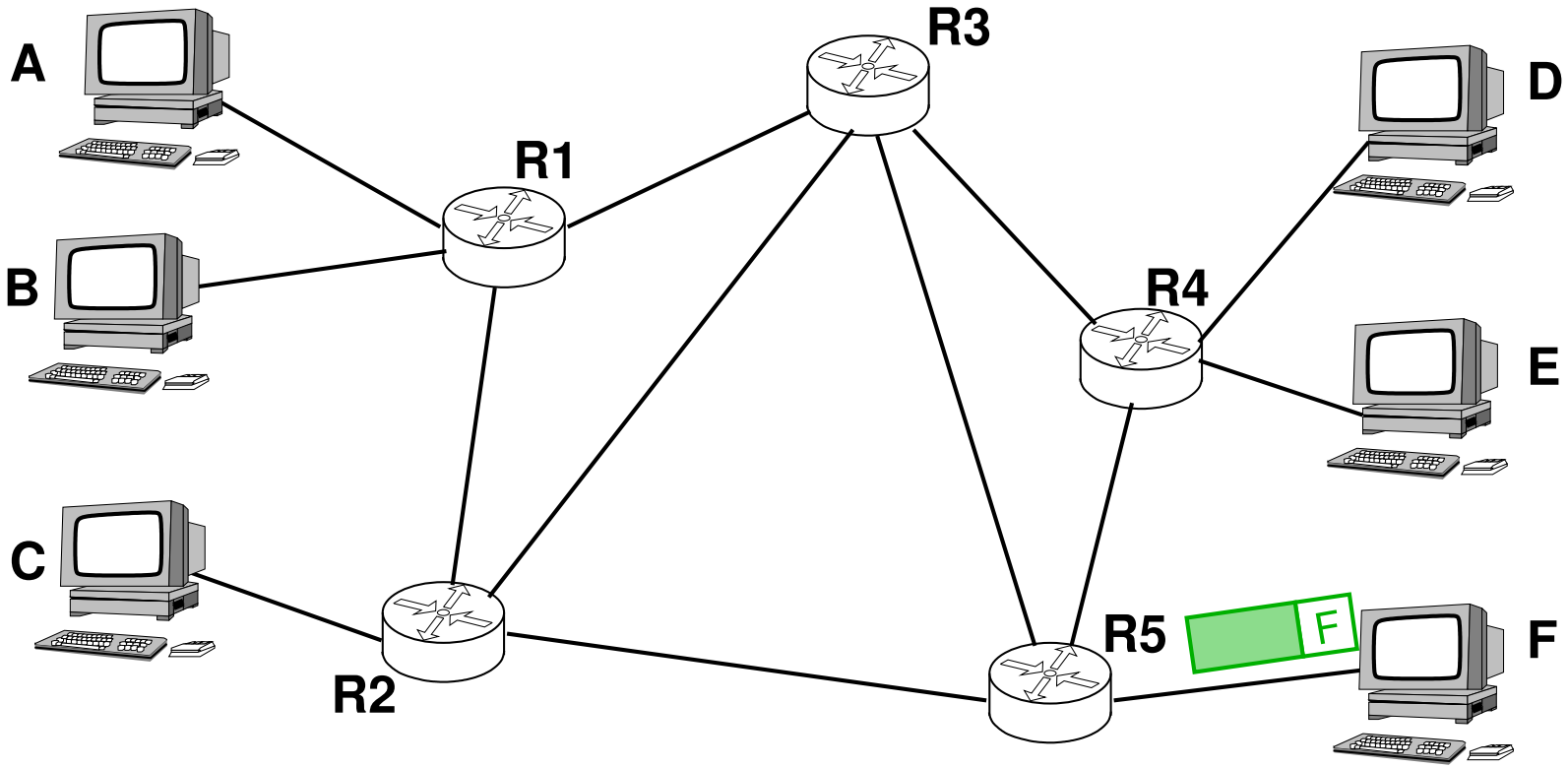
Forwarding



routing table

dest.	next hop
A	R2
B	R2
C	R2
D	R4
E	R4
F	F
default	R3

Forwarding



Routing

- In the real world, we find things using a map
  - look at the map, and plot a route
- The Internet is too big, and changes too often
  - there are no complete, up-to-date maps
  - do how do we find how to get somewhere?
- Routers keep a list of destinations, and how to get to each
  - how do they maintain the lists?
  - routing protocols
- Typical routing protocols look for the shortest path from A to B

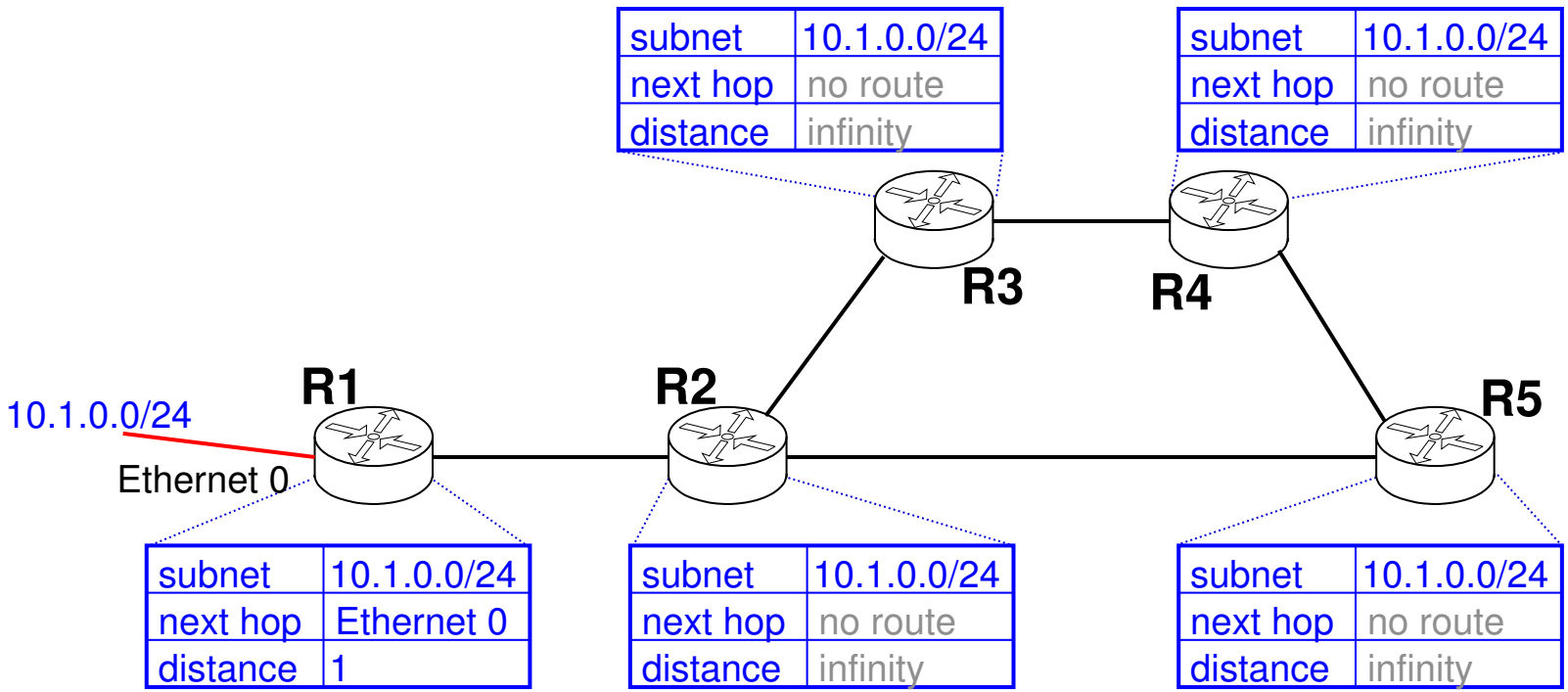
How do we find shortest paths?

- Algorithms (recipes)
  - Dijkstra's algorithm
    - ★ Edsger Dijkstra (1930-2002) (Dutch computer scientist, Turing prize winner 1972, ...)
  - Bellman-Ford algorithm
- Protocols
  - implementations of the algorithms
    - ★ OSPF implements Dijkstra
    - ★ RIP uses Bellman-Ford

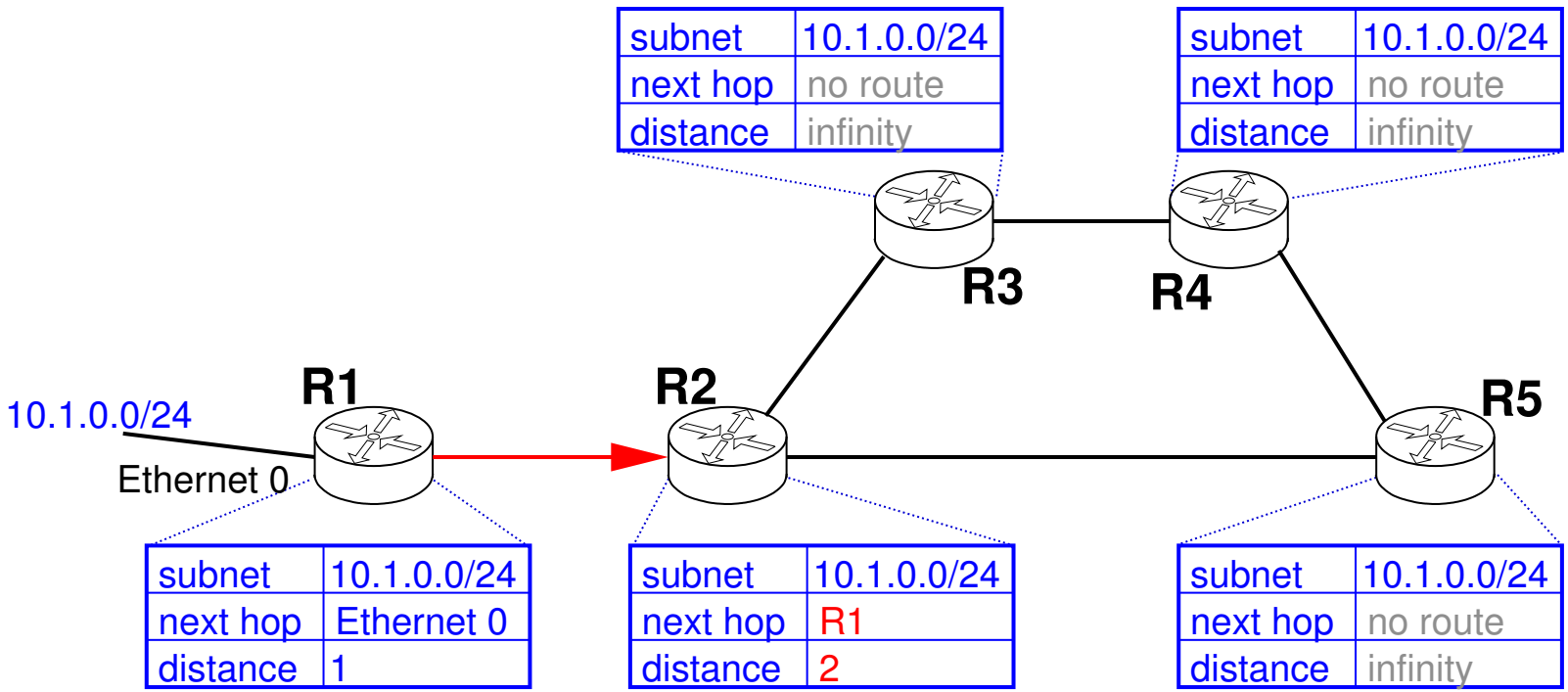
Bellman-Ford

- Make a list of destinations you can reach and the distance to these destinations.
  - Store in routing table
- Share this list with your neighbours
- Add to routing table new information gained from adjacent routers about the destinations they can reach
  - remember to increment their distance
  - keep the source as the next hop
- If two paths to the same destination exists, keep the shortest distance path.
- Repeat periodically (in RIP every 30 seconds).

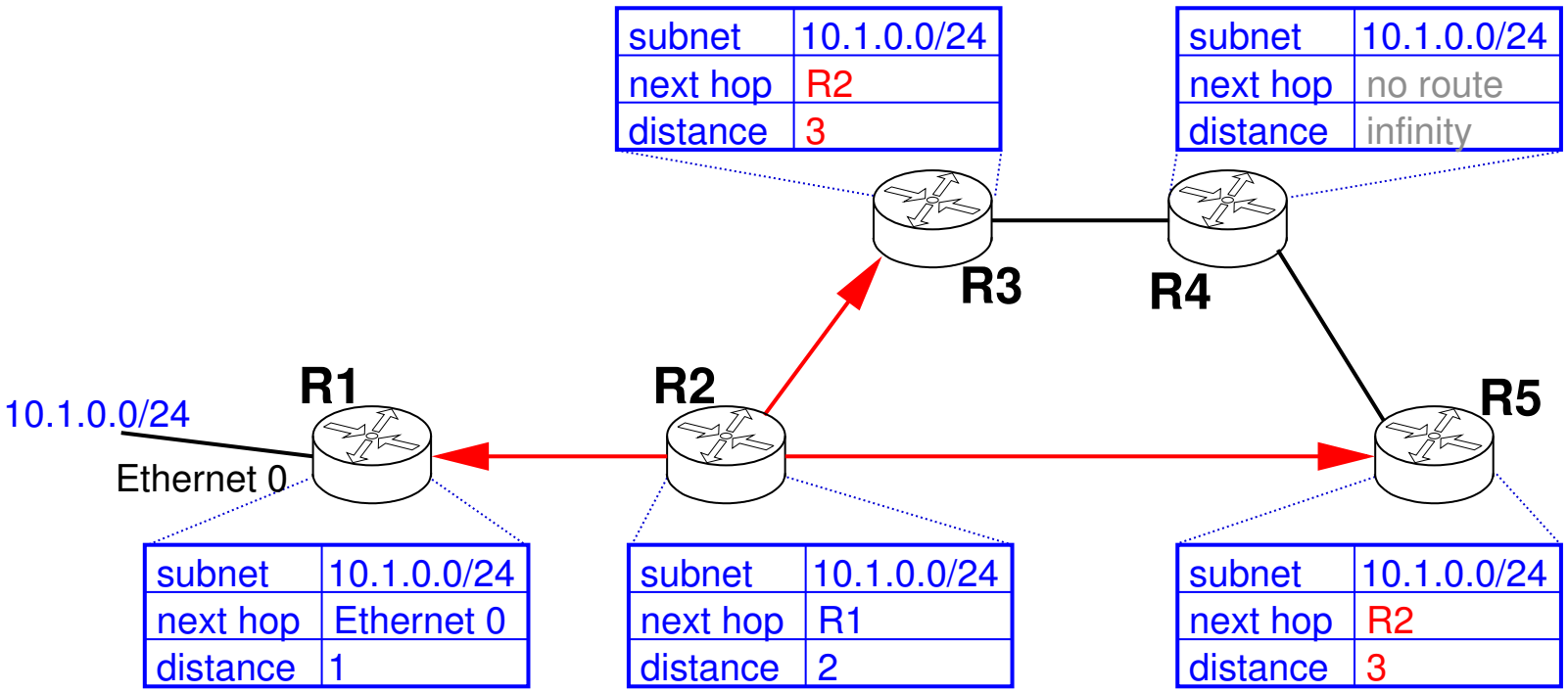
Bellman-Ford



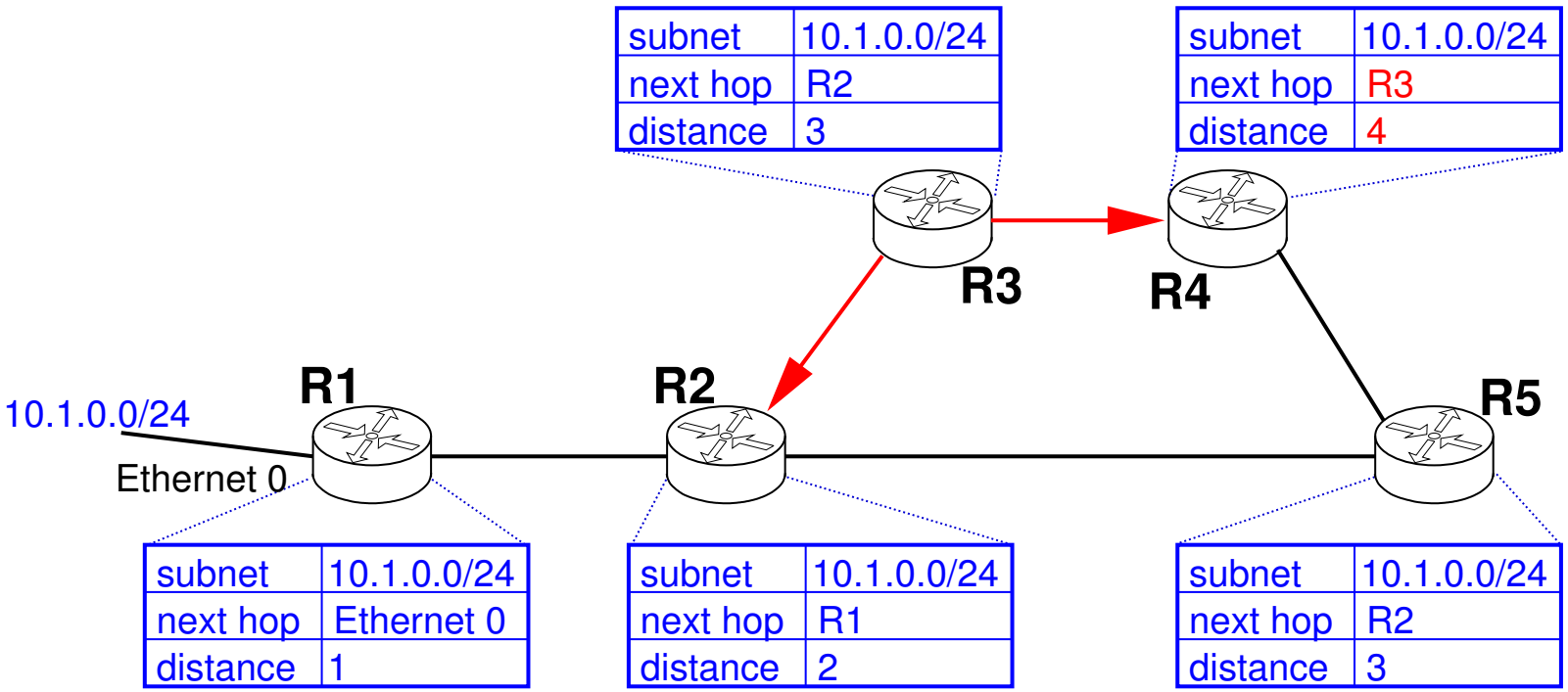
Bellman-Ford



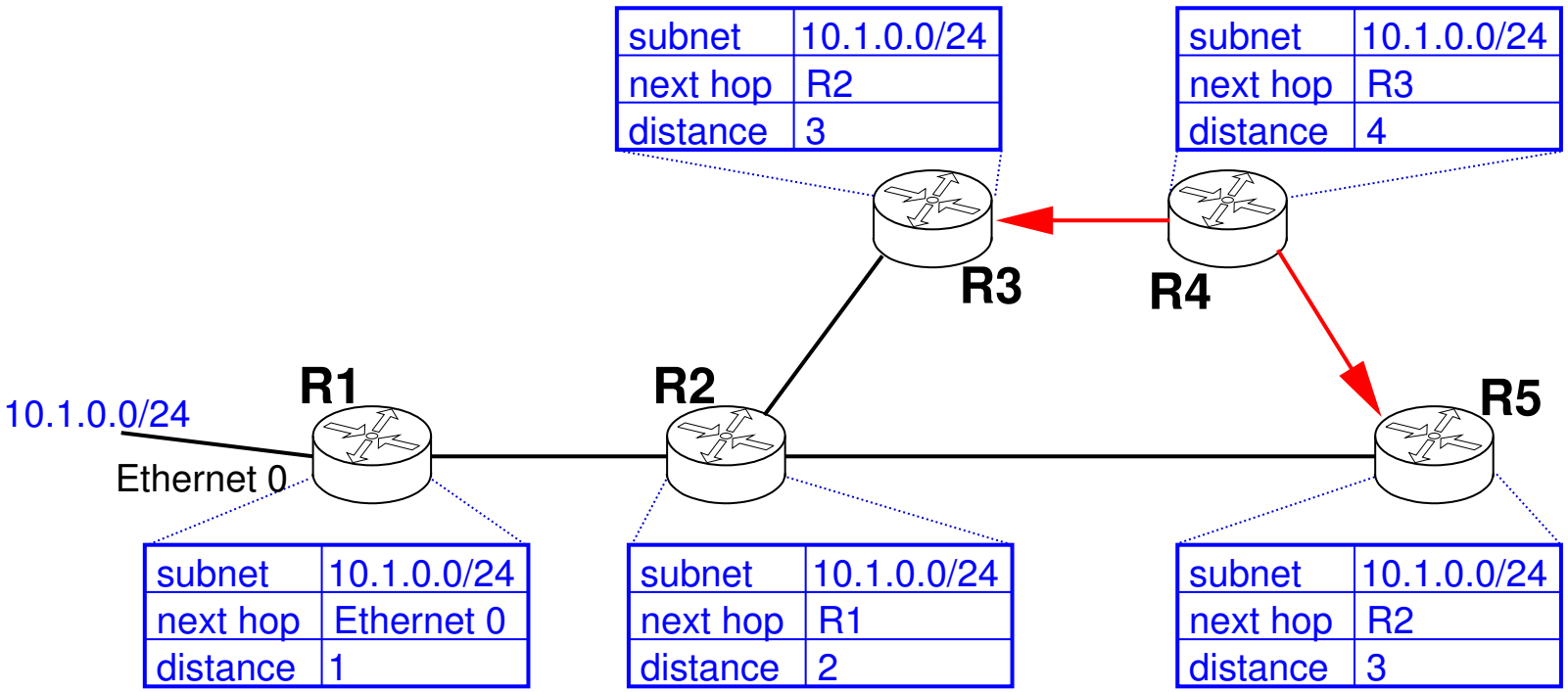
Bellman-Ford



Bellman-Ford



Bellman-Ford



The End

Questions?

Graph representations

- List of links: explicitly give
  - N: e.g., $N = \{1, 2, 3\}$
  - E: e.g., $E = \{(1, 2), (1, 3)\}$
- Adjacency matrix: define connectivity through a $(0, 1)$ matrix A defined by

$$A_{ij} = \begin{cases} 1, & \text{if } (i, j) \in E \\ 0, & \text{otherwise} \end{cases}$$

e.g.,

$$A = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

Some Graph Tools

- MatlabBGL http://www.stanford.edu/~dgleich/programs/matlab_bgl/
  - Graph libraries for Matlab,
  - using Boost Graph Library (BGL)
    http://www.boost.org/doc/libs/1_42_0/libs/graph/doc/index.html
- igraph http://igraph.sourceforge.net/
  - Libraries for working with graphs in R or Python
- GraphVis http://www.graphviz.org/
  - Toolkit for visualisation of graphs
- NetworkX http://networkx.lanl.gov/
  - Python toolkit for working with graphs
- GDToolkit http://www.dia.uniroma3.it/~gdt/gdt4/index.php
  - OO C++ library for handling and drawing graphs
- JUNG http://jung.sourceforge.net/
  - Java universal network/graph framework

Extensions

Various generalizations of simple graphs are useful in particular circumstances:

- hypergraph: links connect more than two nodes
  - e.g., where you have a connective medium (rather than a wire), for instance in a wireless network.
- multigraph or pseudograph: has multiple parallel links between two nodes
  - e.g. its easy to have two links between two routers
- self-loops: nodes can link to themselves
  - useful in a representive a Markov process

We'll exclude these cases unless explicitly stated.