IAC-06-B5.7.2
Ethernet over SpaceWire – Software Issues

Barry M Cook, Paul Walker

4Links Limited

Bletchley Park

MK3 6ZP

UK

Introduction

4Links™

Amongst other things …

Ethernet

- Is a long established technology with extremely wide take-up in computer networks;
- Has enabled a huge variety of applications through a rich set of supported protocols.

SpaceWire

- Is a newcomer which has, nonetheless, generated wide interest and significant take-up, worldwide, in the space industry;
- Is capable of being built into highly fault-tolerant networks and systems;
- Comes with a very limited, but growing, set of protocols targeted to the space industry.

Introduction (2)

4Links™

What if SpaceWire networks could offer Ethernet services as well as supporting the new Space-related SpaceWire, and other, protocols?

This would automatically support that rich set of protocols we are familiar with and its very wide variety of applications (and allow re-use of existing code).

This paper is based on a successful Linux proof-of-concept implementation.

Introduction (3)

4Links™

SpaceWire networks have been shown to be able to route Ethernet packets provided

- Ethernet destination addresses are translated to SpaceWire path or logical addresses
- Multicast / Broadcast is emulated by multiple unicast / distribution transmissions

This can be done in a device driver (or an interface unit between Ethernet and SpaceWire)

Requirements

4Links™

1. The principle need is to discover the routes from source to destination, with alternates for fault tolerance
   - Statically – for some applications
   - Dynamically
     - Ethernet is plug-and-play
     - Can tolerate more faults
2. To send multiple copies to emulate multicast / broadcast

SpaceWire Routing

4Links™

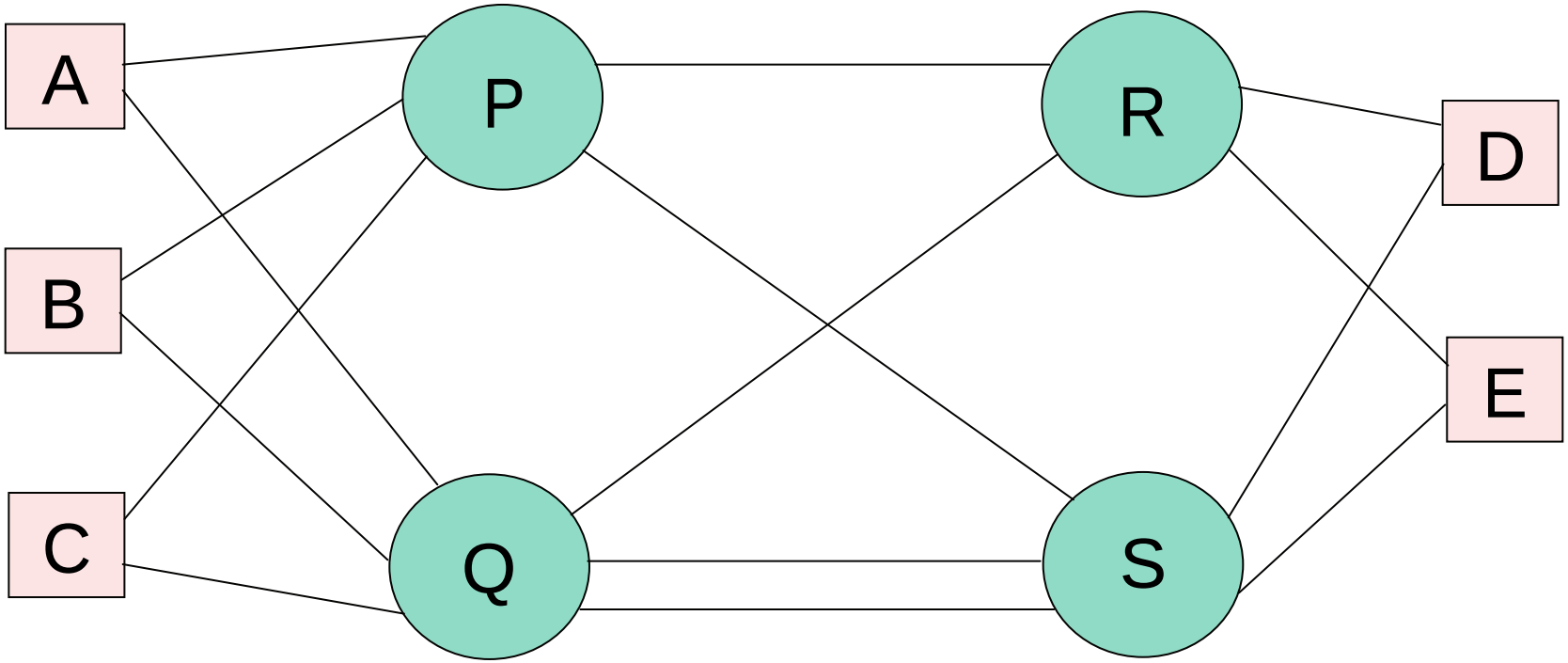
One, or more, bytes at the front of the packet are interpreted by routing switches in the network.

- The first data byte is interpreted by the first routing switch
  - 0 is communication with the switch
  - 1-31 sends the data to that physical port
    - Deleting the address byte
  - 32-255 indexes a table in the switch to determine disposition
    - The address byte may be retained or deleted
    - A set of acceptable output ports may be specified – group-adaptive routing
- The address table in the switch must be explicitly written
  - The switch has no learning capability

Routing – Logical Addresses

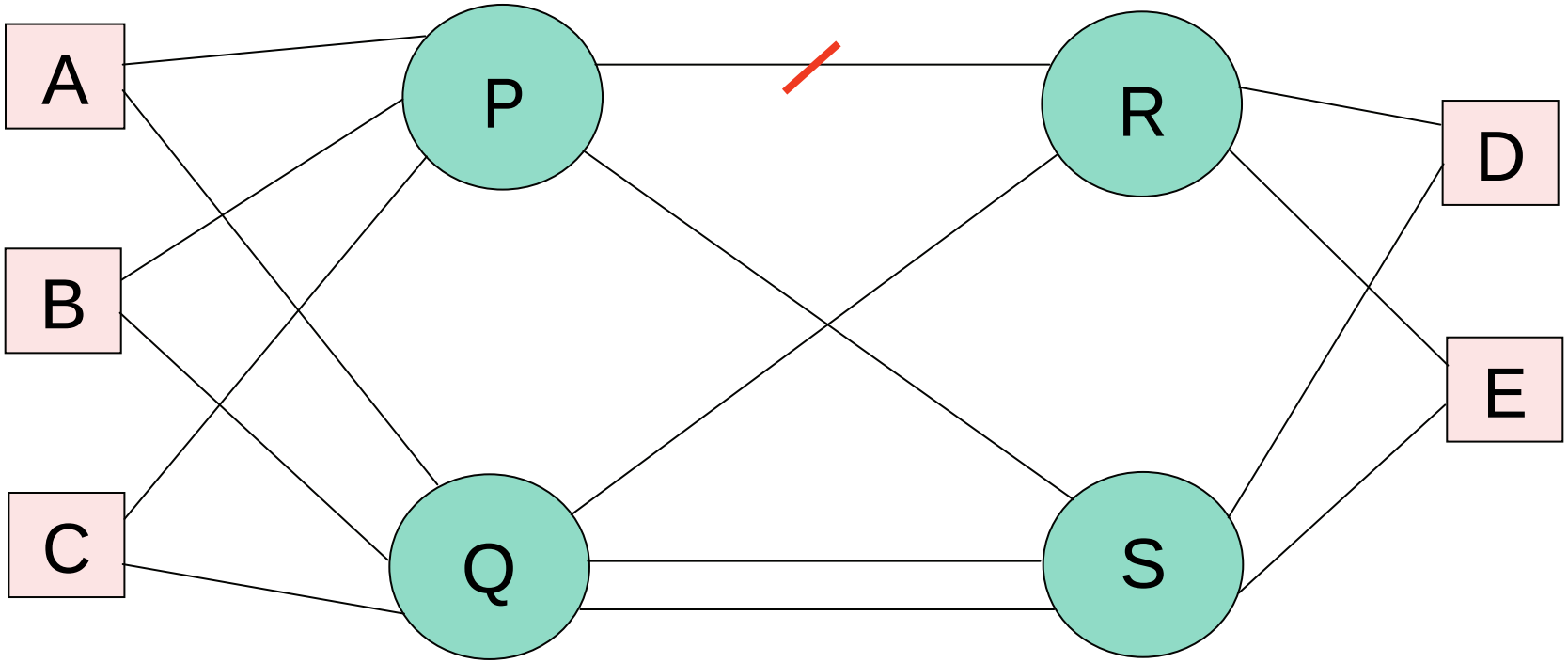
4Links™

- Logical address routing looks very attractive as it closely resembles Ethernet address routing
- Each destination is given a logical address
  - Only 224 directly available
- Each router has its table set to forward toward the destination
  - Using alternate routes with group-adaptive routing


A
B
C
P
Q
R
S
D
E

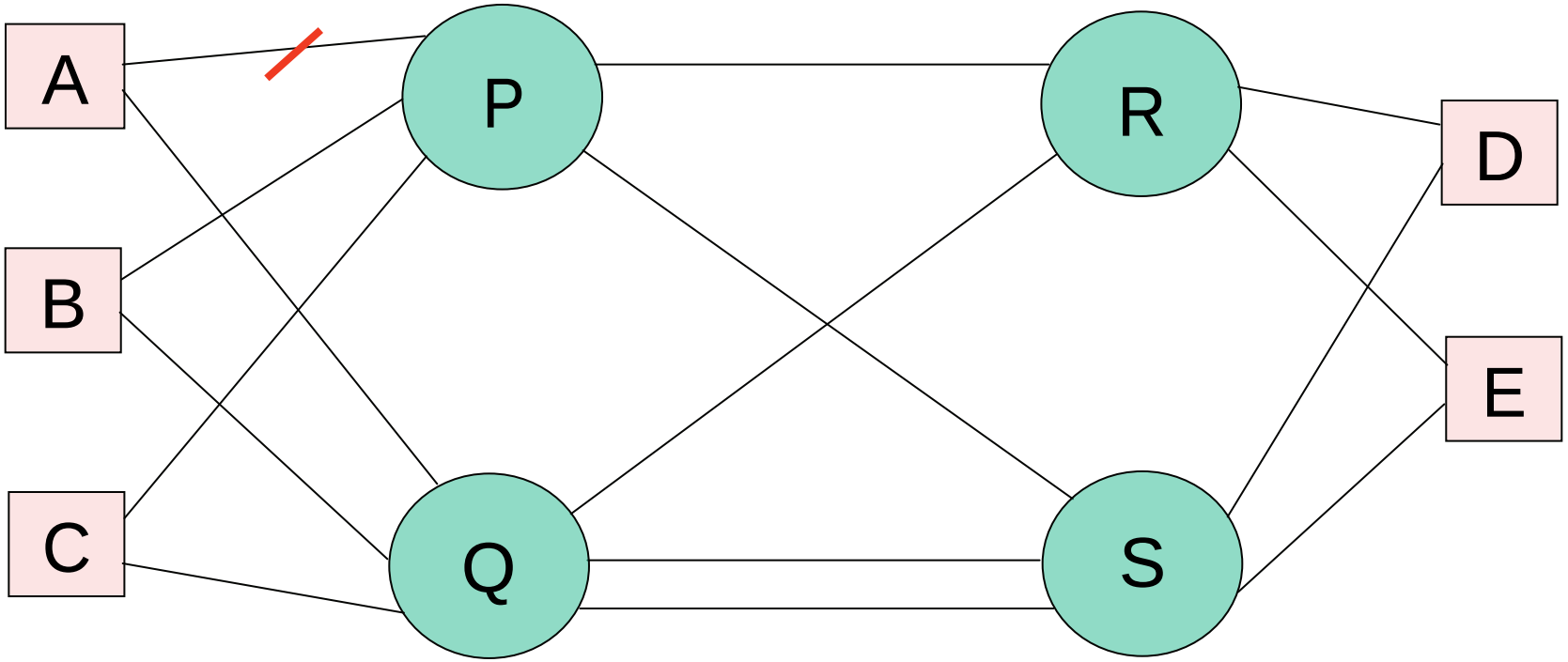
4Links™

Recoverable



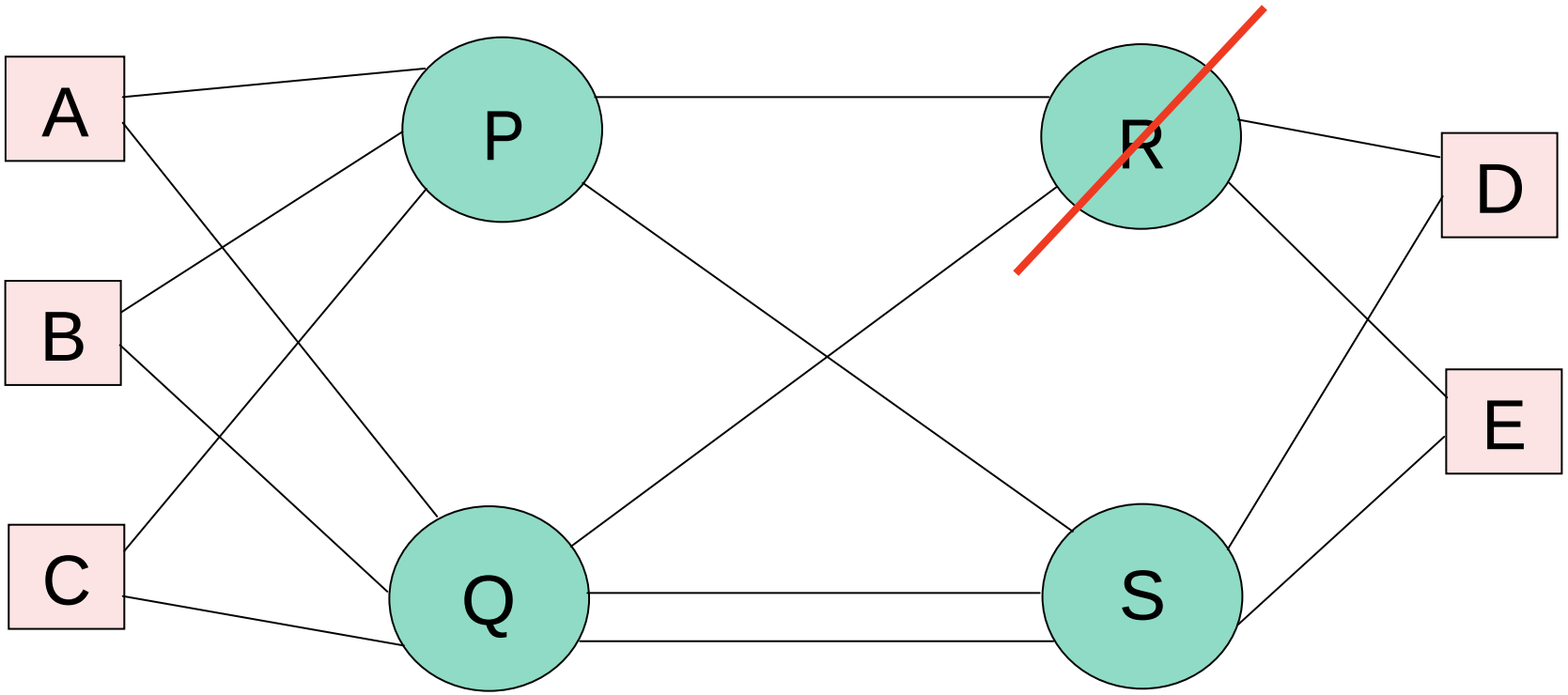
4Links™

Recoverable



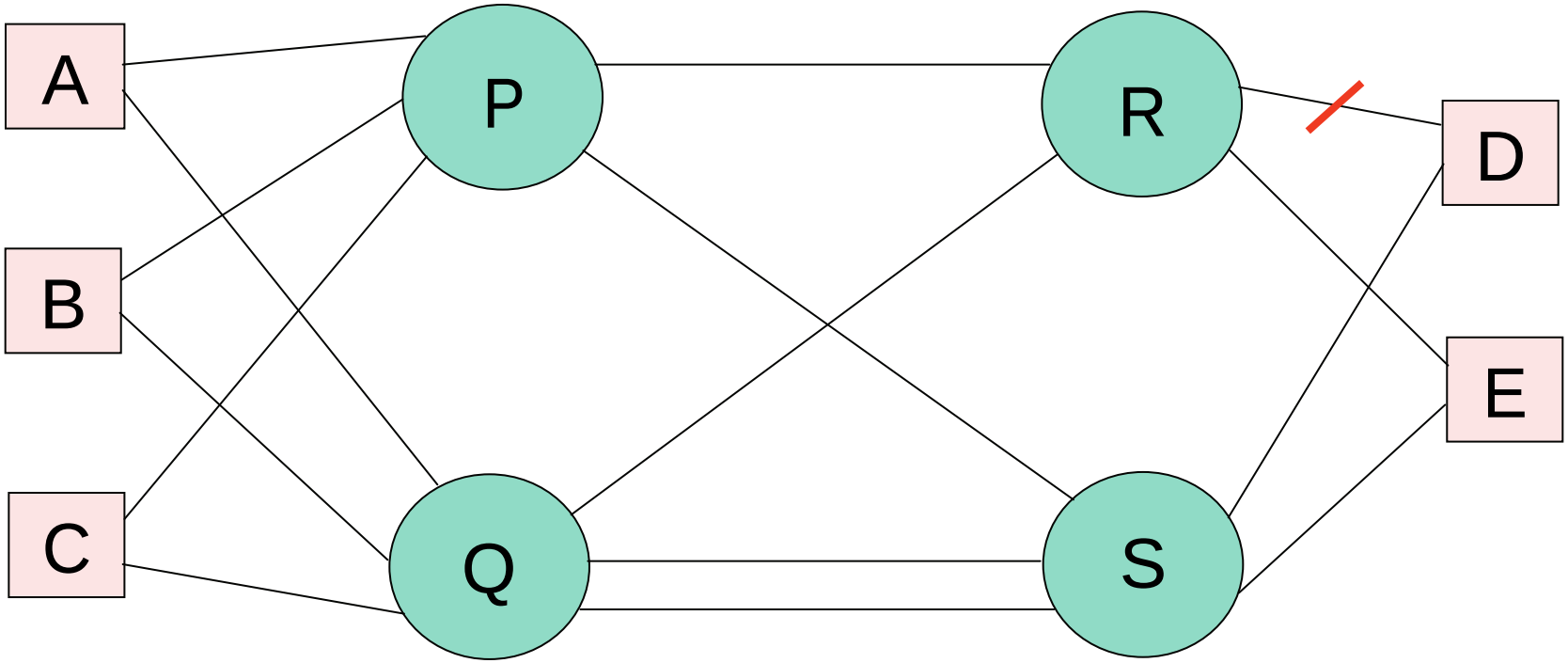
4Links™

Recoverable



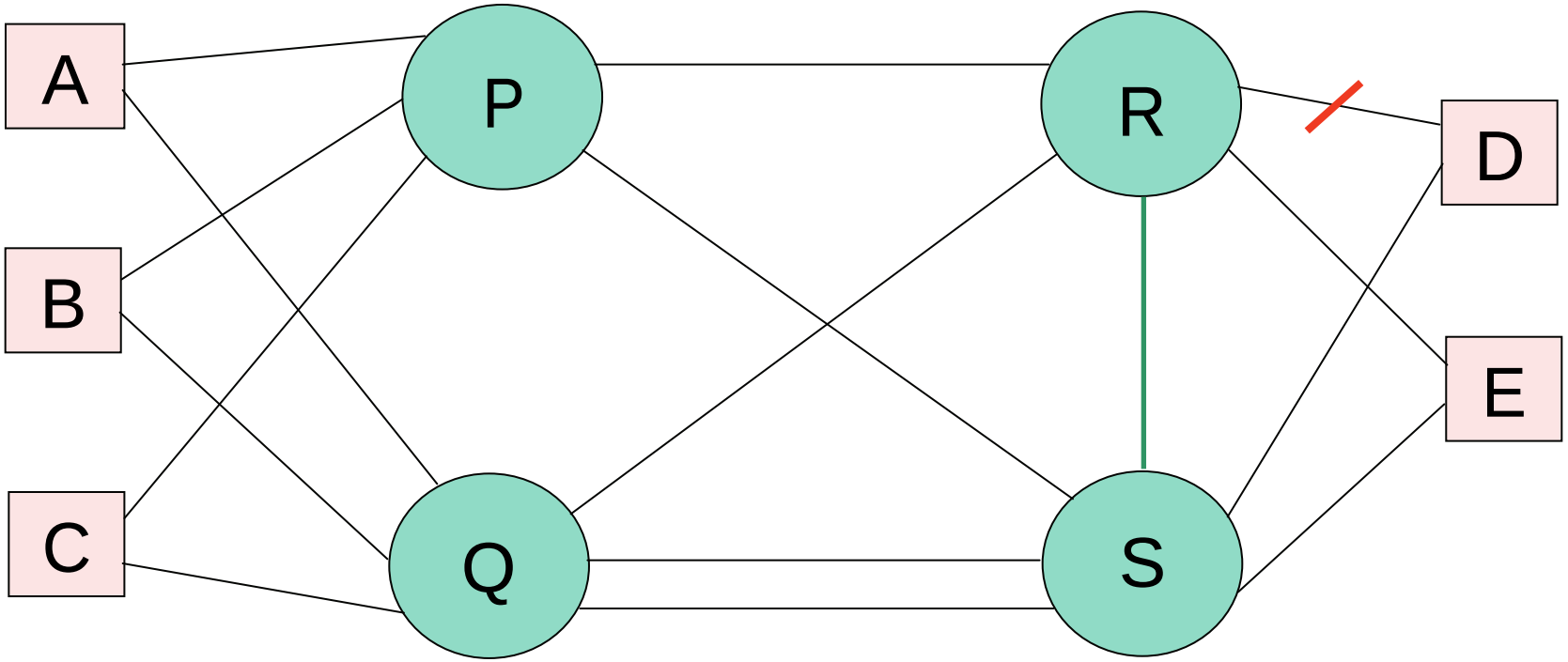
4Links™

Problem



4Links™

Solution?



Static?

4Links™

System design for static configuration is NOT trivial

- Simple logical addressing limits fault recovery capability

Faults change the network

- and then we need another configuration

Treating a system as dynamic acknowledges reality and can provide a more robust system

Static configuration limitations

4Links™

Dynamic configuration can provide a working network with a wider range of faults than static configuration

- Even with a static topology, dynamic configuration offers advantages
- Static becomes partially-dynamic when faults occur
- Enabling cold-redundant units produces a dynamic, not static, network

Dynamic configuration

4Links™

1. Discover the current network
   - Identify devices, switches and connections
   - (No need to identify non-Ethernet devices)
2. Select routes and configure devices and switches
   - Single controller
   - Multiple controllers (redundancy)
   - Distributed (most fault tolerant)
3. Repeat 1. & 2.
   - On demand – as changes are detected (e.g. ‘acks’ are missed)
   - Scheduled – decoupled from protocol

Network Discovery (1)

4Links™

Device/switch identification:

Each (participating) node must be able probed and return information to direct discovery

- Whether switch or device — may be implicit in …
- Number of ports
- Port receiving probe request
- Unique identifier
  - To allow detection of loops
    - Ethernet address for devices
    - Any scheme for switches

Network Discovery (2)

4Links™

From any point in the network …

Probe each immediately connected component

For each component that is a switch

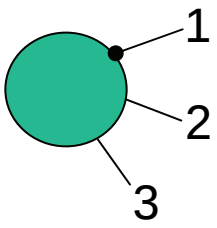
Probe what is connected to each port

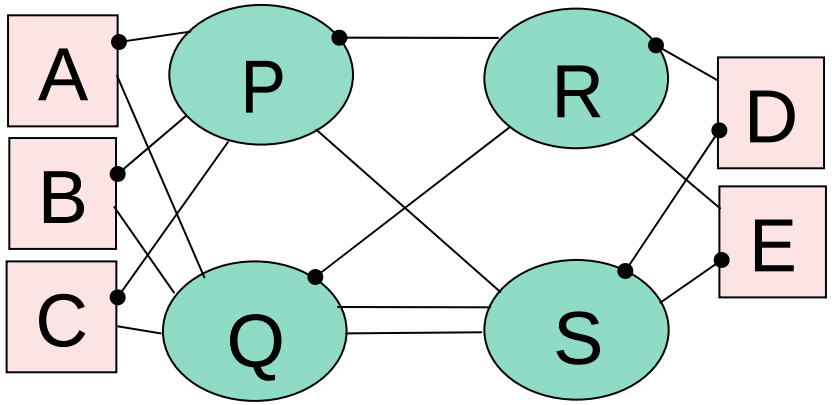
etc.

[Either breadth-first or depth-first search may be used.]

4Links™

A.1 = P.5
P.1 = R.4
P.2 = S.5
P.3 = C.1
P.4 = B.1
R.1 = D.2
R.2 = E.2
R.3 = Q.1
S.1 = D.1
S.2 = E.1
S.3 = Q.3
S.4 = Q.2
Q.4 = C.2
Q.5 = B.2
Q.6 = A.2





Paths from A to E
A.1 – P.1 – R.2 – E.2
A.1 – P.2 – S.2 – E.1
A.2 – Q.1 – R.2 – E.2
A.2 – Q.2 – S.2 – E.1
A.2 – Q.3 – S.2 – E.1
A.1–P.1–R.3–Q.2–S.2–E.1
etc.

Port:	Header
Port-1:	***1, 2***
Port-1:	***2, 2***
Port-2:	***1, 2***
Port-2:	***2, 2***
Port-2:	***3, 2***
Port-1:	***1, 3, 2, 2***

Routing

4Links™

Device ID from the probe is the Ethernet address of the device

The complete set of possible routes from source to destination can be discovered from any node

Preferred routes can be selected from the possible routes

Logical address tables can be set from a subset of the possible routes

OR

The physical address sequence can be used directly

- Each device is self-contained
- ALL possible routes may be used

Multicast and Broadcast

4Links™

Fault-tolerant SpaceWire networks will contain multiple paths – broadcast would result in deadlock

Broadcast is not supported by the hardware

Broadcast may be simulated by software as multiple unicast

- At some cost in network traffic
  - OK if broadcast is not too frequent

One exception – SpaceWire does allow broadcast of a time-code

Conclusions

4Links™

Ethernet over SpaceWire can be delivered

- On dynamic / fault tolerant networks
- Using very simple hardware
  - Routing switch much simpler than Ethernet switch
- With low-complexity software
  - Can be part of a device driver
- Distributed implementation
  - No centralized controller(s) required
- To provide a very robust system
- At the same time as other SpaceWire protocols