Communicating Process Architectures 2002
James Pascoe, Peter Welch, Roger Loader and Vaidy Sunderam (Eds.)
IOS Press, 2002

Configuration Discovery and Mapping of a Home Network

Keith PUGH
Computer Science Department, Keele University, Keele, ST5 5BG, UK.

Abstract. A home network comprising many heterogeneous devices requires a scaleable interconnect capable of satisfying the often vastly different network resource demands of the devices. In addition, it is likely that over time the configuration of the network will change as new devices are added to the network and older ones removed or replaced. While it is acceptable for enterprise networks to be managed by a trained network administrator it is unreasonable to expect a home owner to have such expertise. Consequently, a free topology network coupled with a capacity for plug and play that allows nodes to be added anywhere, and at any time without interruption to the operation of the networked system are essential requirements. IEEE 1355 standard technology has the potential to satisfy these criteria. The demand for a free topology and a capacity for plug and play require that the configuration of the network is re-discovered and mapped automatically, and at regular intervals. This paper describes such a configuration mapping process.

1 Introduction

The modern home has separate networks for heating, telephone, security entertainment, lighting, etc. Advances in digital technologies for consumer electronic devices and high performance networks have led to the possibility of uniting these dissimilar networks into a single ubiquitous home network that allows the interconnection of numerous heterogeneous devices. This diversity of devices imposes many different demands on the system's network design. For example, devices that perform status gathering or simple on/off control operations consume relatively little of the communication network's bandwidth – they can share bandwidth with other processes. On the other hand, real-time audio or video data transmissions need to be delivered at a high and sustained bit rate, which may require exclusive use of the network's bandwidth. A communication network performs best when the latency of the interconnect is low and the throughput is high. Achieving low latency and high throughput requires fast hardware, and efficient communication protocols that minimise communication software overhead.

Scalability and cost are important considerations for the design of an effective network. The network must continue to perform well as it grows in size, and the cost of the interconnect should not be allowed to dominate the cost of the system [1]. Many of the existing methods of interconnecting the components in a digital network within a limited area such as a room or a building are based on the restrictive model of a bus and do not meet these criteria. In a bussed system the bandwidth must be shared among the nodes of the network, consequently, performance declines as more nodes are added. More recent high performance network designs utilise routing switches to solve this problem by permitting simultaneous data transfers between disjoint pairs of nodes. A message generated at a source node travels through one or more routing switches to reach its destination node. To achieve the routing function knowledge of the network topology is needed, which, depending on the routing strategy, may be contained in routing tables in the source nodes or in the routing switches. In these networks the overall bandwidth grows as more nodes are added, thus greatly increasing the system's potential throughput. One method of interconnection that utilises routing switches is specified under the IEEE 1355-1995 standard [2]. This standard allows very low cost implementations, and is used in a wide range of applications. The space industry, for example, uses IEEE 1355 technology for its reliability and heterogeneity, and large parallel supercomputers exploit its performance characteristics.

Over time the configuration of the network will almost certainly change as new devices are added to the network and older ones relocated or removed. While it is acceptable for enterprise networks to be managed by a trained network administrator it is unreasonable to expect a home-owner to have such expertise. Consequently, in a network that allows nodes to be added anywhere, and at any time without interruption to the operation of the networked system the ability to *plug and play* is an essential requirement.

Cook argues that current home network architectures, which rely on a set of communications protocols defined to accommodate all present and future devices, are inefficient and restrict extensibility [3]. Any protocol that encompasses all devices is, by definition, one that must accommodate the most complex device envisioned. The definition of such a protocol will inevitably be overtaken by events. Furthermore, it is completely inappropriate to incorporate the support for an all-encompassing protocol into a simple device such as a lamp, where the processing capabilities are likely to be extremely limited. This approach suffers on the one hand from being short sighted and on the other hand being prohibitively expensive for the simpler case. Cook proposes instead an architecture in which each device supplies its own proxy driver to support the device's functionality, and also provides the, possibly unique, communication protocol to access it. The proxy device drivers are uploaded and run separately within a central controller. Using this scheme it is anticipated the central controller and the devices need only support a basic common communication protocol in order to provide a home network that is future proof. It is this concept that our work explores. But to begin, the combination of a free topology network and a capacity for plug and play requires that the configuration of the network is re-discovered and mapped automatically, and at regular intervals. This paper describes such a configuration discovery and mapping process.

2 IEEE 1355 Technology

The IEEE 1355 standard enables the construction of scaleable low-latency serial interconnect systems based on high-speed point-to-point links and routing switches. The standard specifies the physical media and a set of lightweight protocols (up to the packet layer), for a family of flow controlled, point-to-point, communication technologies with speeds and media ranging from 10 Mbps to 1 Gbps in both copper and optic fibres. IEEE 1355 links can be used as a reliable transport mechanism by higher level protocols, e.g. ATM.

The work presented in this paper focuses on the IEEE 1355 DS-link technology, operating at speeds from 10 Mbps to 200 Mbps. This technology was designed initially as a multiprocessor interconnect, but it is equally suitable for local area networks, and has demonstrated scaleable performance from one to over a thousand nodes [4], [5].

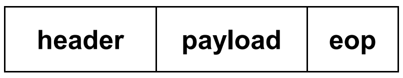


Figure 1: IEEE 1355 packet format

The packet format is shown in Figure 1. Information is transferred in packets comprised of a header, containing routing information which is used to route a packet through the switching fabric, followed by a payload of zero or more bytes of data e.g. 53 byte ATM packet, and finally an end-of-packet termination character. There is no prescribed packet length, however these should be limited in size to prevent them monopolising the network resources and blocking other packets.

2.1 Routing

A network's routing strategy determines how a message traverses its path through the network. In IEEE 1355 networks messages may be broken into a sequence of smaller more manageable packets, and individually routed from the source to their destination through routing switches. The packet header contains routing information that is interpreted by the routing switches when directing the packet to its destination.

2.1.1 Switch Architecture

A routing switch consists of a number of input and output ports, buffers to queue packets, an internal interconnect which connects the inputs to the outputs and an arbiter to resolve contention between the ports. The number of output ports is equal to the number of input ports. The internal interconnect in an IEEE 1355 routing switch is generally a non-blocking crossbar [6]. In a non-blocking routing switch any permutation of input and output ports can be connected. However, not more than one input port can be connected to the same output port simultaneously. Consequently, contention occurs if more than one packet arrives on different inputs at the same time all destined for the same output port. The arbiter in the routing switch allows one of the packets to proceed while the other packets are blocked and must be queued until the output port is free. Ultimately, on average, the input port selection algorithm provides each input port with an equal opportunity to send packets. However, if the packets are not all the same length, this selection scheme does not guarantee fair sharing of the output link bandwidth.

IEEE 1355 architecture routing switches improve the efficiency of the packet switching strategy by enabling the use of wormhole routing [7], and header deletion techniques, which provides very low switching latency and also requires minimal buffering in the routing switches.

2.1.2 Wormhole Routing

Usually, packet switching networks route packets using a store and forward technique. This requires each intermediate node to store an incoming packet and to decode the packet's header in order to forward the packet to the next node. This is undesirable for high speed, low cost interconnect systems because:

- Storage is necessary at each node to store packets. This limits the packet size and increases costs.

- Delay (latency) is increased because the complete packet must be received before it can be forwarded to the next node.

The wormhole switching technique ameliorates these problems. In wormhole routing networks only the first part of the packet is required to make a routing decision. The routing decision is taken as soon as the packet header has been read by the routing switch. The header is then sent to the chosen output link and the rest of the packet is copied directly from the input of the routing switch to its output. This means that a packet can be passing through several routing switches at the same time, and the head of the packet may be received by the destination before the whole packet has been transmitted by the source. This method can be thought of as a form of dynamic circuit switching, in which the header of the packet, in passing through the network, creates a temporary circuit (the `wormhole') through which the data flows. As the tail of the packet is pulled through, the circuit vanishes.

The ability to start outputting a packet while it is still being input can reduce delay significantly, especially in lightly loaded networks. The delay can be further minimised by keeping the headers short and by using fast, simple hardware switching to determine the link to be used for output.

2.1.3 Header Deletion

An approach that simplifies the routing of packets in a network is to provide two levels of header on each packet. The first header specifies the output link from the routing switch, and is removed as the packet leaves the routing system. This exposes the second header which tells the destination device which process this packet is for. Any output link can be set to do header deletion, i.e. to remove the routing header from the front of each packet after it has been used to make the routing decision. The first part of the remaining data is then treated as a header by the next node to receive the packet.

In summary, IEEE 1355 technology has many advantages over other interconnect methods [6], [8]:

- IEEE 1355 provides a set of lightweight protocols for bi-directional flow controlled, point to point communication.
- Networks of any size and topology can be built using routing switches to provide complete connection between all devices.
- The communications bandwidth does not saturate as more communicating devices are added to the system. Rather, the larger the number of devices, the greater the total communications bandwidth of the system.
- A credit based flow control mechanism built into the fabric of each communication link means packets are never lost (and have to be retransmitted), and there is no possibility of data overrun at the receivers. This simplifies the higher levels of the protocol, since it removes the need for retransmission unless an error occurs.
- A small protocol overhead makes the links very efficient, even for short packets.
- A flexible packet format allows IEEE 1355 links to be used as a carrier for other higher level protocols.

These features provide a solution to the bottleneck problem normally associated with a high-speed communication system.

3 The Network

The network depicted in the undirected graph shown in Figure 2 illustrates the components of a home network. The nodes represent one controller, four routing switches, and three devices. The edges between nodes are their interconnecting communication links.

Device are leaf nodes of the network. In a typical network there will be an assortment of devices of varying complexity. For example, simple devices such as lamps with minimal storage and processing ability may coexist with sophisticated devices such as video recorders and satellite receivers. The **Controller** is a device which in addition to its normal function has overall responsibility for the organisation of the configuration discovery process, and for maintaining a contemporary map of the network configuration. The controller may be any device with the necessary storage and processing capabilities, for example a digital set top box or a personal computer. **Routing switches** are multi-port switches that channel messages from their source node to their destination node. **Links** are point to point connections between each pair of nodes.

All nodes implement a full IEEE 1355 interface on each of their external ports. In addition, each node must provide its own hard coded unique identifier, and an internal management port (port #0) to enable access to the unique identifier and other relevant configuration information.

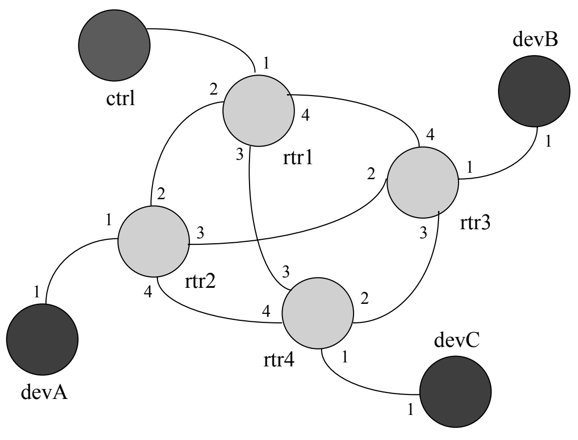


Figure 2: a typical network

4 Configuration Discovery Algorithm

The need for a free topology and a capacity for plug and play require that the configuration of the network is re-discovered and mapped automatically, and at regular intervals. In a typical switched network [9] a processor in each switch monitors the state of the network and triggers a distributed configuration discovery algorithm whenever a change in the network topology is detected. However, this approach is considered inappropriate for a home network because of the limited processing capability of many of the nodes in the network and the desirability of low cost. Instead, the discovery process is centralised at the controller from where the other nodes in the network are polled for their configuration details. The process is summarised by the following algorithm:

```
for each node on the network
    for each port at this node
        identify the node connected to this port
```

The mapping of a network is achieved by visiting all of the nodes in the network to acquire details of their configuration. Nodes are discovered in order, the closest nodes first. As the discovery progresses the configuration knowledge is extended by one link for each request. The discovery process begins by interrogating the node linked directly to the controller. If the information obtained relates to a routing switch, configuration details are requested from each of the nodes connected to the ports of that routing switch. This procedure is repeated until the discovery is complete. The information required from each node is its unique identifier, the number of ports at the node, and the transmission rate that each of its output port(s) can sustain. In addition, the interrogated node provides the number of the input port on which the identify request is received.

Table 1: adjacency list format

node identifier
route from controller to node
route from node to controller
number of ports
node (identifier) & port number connected to port #1
port #1 link speed
node (identifier) & port number connected to port #2
port #2 link speed
:
node (identifier) & port number connected to port #n
port #n link speed

From this information an adjacency list [10] is constructed to describe the network topology. The format of the information in each row of the adjacency list is shown in Table 1. At the start of the discovery cycle the only node of which the controller is aware is itself. Consequently, the only details registered in the adjacency list are those of the controller. As each new node is discovered the adjacency list is extended to register the details of the new node.

4.1 Concurrent Discovery

At any moment, during the discovery cycle, there may be one or more configuration inquiries in progress, but the order in which their responses arrive is unpredictable. A means of associating a response with the request that produced it is needed in order to locate the relevant entry in the adjacency list. To facilitate this tracability the node identifier and port number of the investigating node are included in the identify request. These data are echoed in the corresponding response, and used subsequently as an index to the relevant fields in the adjacency list.

4.2 Closed Loops

The freedom to link the nodes of the network arbitrarily in any configuration creates the possibility for interconnected nodes to form closed loops. In Figure 2 there are many closed loops – for example, the links between any two adjacent nodes, or the sequence of nodes *rtr1* to *rtr2* to *rtr3* and back to *rtr1*. While alternative links between nodes increases the bandwidth between nodes, and is beneficial for the robustness of the network it could lead rapidly to a situation in which the discovery process is continually rediscovering the same set of nodes. To avoid this situation a node's unique identifier is returned as part of the node's configuration details. The identifier enables the discovery process to detect whether a node has been visited previously during the discovery cycle. If the node's identifier is not recognised then the node must be a new discovery and a new entry can be created for it in the adjacency list. Conversely, if the identifier is recognised, it must have been returned in response to an earlier request, and the discovery process need only update the relevant port connection information in the existing adjacency list entry.

4.3 Disconnected Links

Routing switches return a link speed of '0' (zero), if a port is not connected. Identify node requests are not sent to ports that are not connected. However, a link could become disconnected during the interval between sending a request and the arrival of the corresponding response, in this situation the discovery process could wait indefinitely for the response. To resolve this problem the response to an identify node request must be received within a bounded time. If the discovery process does not receive a response within this time it is assumed that there is no connection on the selected port, and it is assigned a link speed of '0'. The use of a time out period, however, can lead to other difficulties if a response arrives too late.

4.4 Orphaned Responses

The transfer path lengths in a free topology network cannot be predicted. Consequently, the time needed for a packet to propagate through the network is indeterminate. It is possible, therefore, for a response to be received after the timeout period has expired. If there has been a change of configuration, since the previous discovery cycle, the information contained in the response may be incorrect and lead to numerous inappropriate discovery investigations. To avoid this problem, a process cycle identifier is included in each identify node request. This cycle identifier is returned by the target node in its response, enabling the discovery process to detect, and subsequently discard, late responses.

4.5 Deadlock

An essential property of a communication network is that it does not deadlock, i.e. arrive in a state where further progress of a packet between nodes is impossible. Whilst every attempt is made to ensure the configuration discovery process is free from deadlock (for example, applying the design rules for communicating processes advocated by Welch et al. [11]), the late arrival of a response to an identify node request, during the interval between the controller completing one discovery cycle and beginning the next, has the potential to cause a deadlock. To illustrate this deadlock problem consider the model of a routing switch shown in Figure 3.

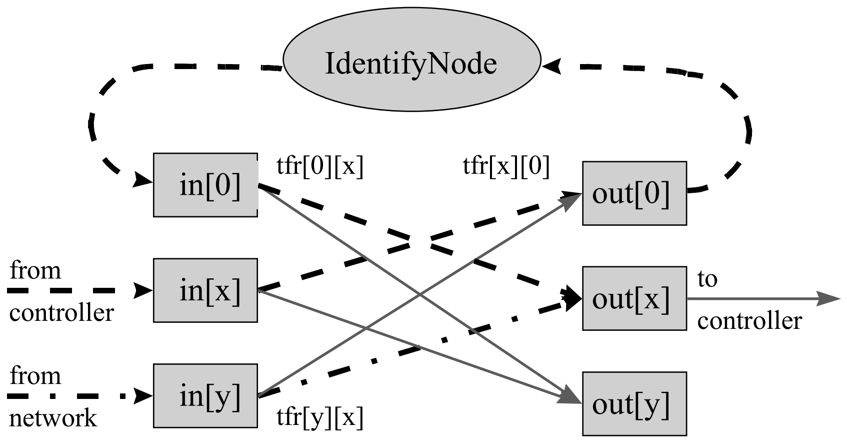


Figure 3: deadlock scenario

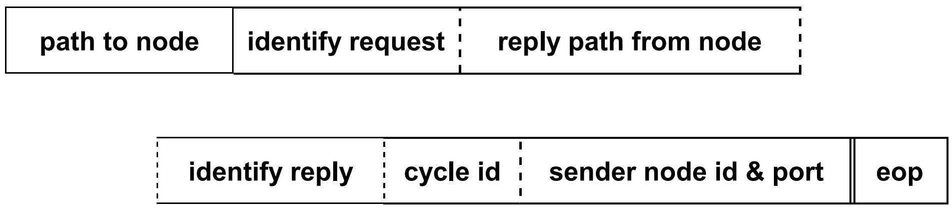
The controller's output process begins sending an identify node request. Meanwhile, a (late) response from the network arrives at port in[y], and is transferred to port out[x] en route to the controller. This response can make no further progress towards the controller since the controller is busy sending the identify node request and consequently has no input process ready to receive the response. The identify node request from the controller arrives at port in[x] of the routing switch, and is passed to the identify node process via the internal management port out[0]. Immediately, the identify node process begins its response by echoing the *return path from node* field, etc. The identify node response is sent to the internal management port in[0] from where it must be transferred to port out[x], and on to the controller. But port out[x] already contains data from the late response and cannot accept the new response data. The controller's output process is now blocked, and the system is deadlocked.

This deadlock problem can be avoided by either: ensuring that the packet containing the entire identify node request is delivered to the routing switch, or guaranteeing that the routing switch is able to forward the late response to the controller. In either case we must ensure the controller is able to complete its output process and proceed to its input process. Both remedies can be accomplished by providing a buffer to store and forward a packet. The decision of which solution to choose can be decided by the size of the buffer needed to store the packet. For a free topology network the length of the path to (and from) a node is variable, therefore the length of the routing information in the identify node request packet is unknown. Consequently, the size of the buffer needed to store the packet is indeterminate. Whereas, the size of buffer needed to store a response is known because the routing information will have been stripped off before the response is sent on the final stage of its journey to the controller. In addition, the use of a buffer to store incoming responses can be refined further by monitoring the cycle identifier of responses and discarding those that do not contain the current cycle identifier.

5 Configuration Discovery Protocol

Because of the constraints of many of the nodes in the network the discovery protocol is implemented using low level message passing techniques that require a minimum of processing resources in each node.

5.1 Identify Node Request



An *identify node* message is sent to the nodes of a network requesting configuration details of any node connected to their output ports. The information required from each node are its unique identifier, the number of ports at the node, and the transmission rate that each of its output port(s) can sustain. In addition, the interrogated node provides the number of the input port on which the identify request is received. In detail:

- The *path to node* contains the packet header to route the message through the network's routing switches to its destination node. The header is comprised of the series of output port numbers that lead from the controller to the node that is the object of the identify request. The final port number in the header is addressed to the internal management port #0.
- The *identify request* routes the message to the configuration discovery handling process once it has arrived at its destination node.
- The *reply path from node* contains the header needed by the object node to return the response through the network's routing switches to the controller. The header is comprised of the series of output port numbers that lead from the object node to the controller. The final port number in the header is addressed to the internal management port #0.
- The *identify reply* routes the response to the discovery handling process in the controller.
- The *cycle id* identifies a complete cycle of the discovery process to enable late responses to be detected.
- The *sender node id & port* identify the node, and its port, currently under investigation. There may be several identify node requests in progress simultaneously. The *sender node id & port* are returned in the response message to provide an index that enables the discovery process to locate the relevant entries into the adjacency list.

To simplify the processing needed at the destination node, the *reply path from node, identify reply, cycle id and sender node id & port* fields are embedded in the outgoing identify node request message, and are echoed by the destination node.

5.2 Identify Node Reply

The destination node responds to an identify node message with an identify node reply message. The *reply path, identify reply, cycle id and sender node id & port* fields are echoed from the identify node request message.

reply path	**identify reply**	**cycle id**	**sender node id & port**

responder node id	**incoming port**	**link speed(s)**	**eop**

The fields are:

- The *reply path* contains the header needed by the object node to return the response through the network's routing switches to the controller. The header is comprised of the series of output port numbers that lead from the object node to the controller. The final port number in the header is addressed to the internal management port #0.
- The *responder node id* is the identifier of the connected node.
- The *incoming port* is the port on which the responder node received the identify node request. This information is used to route subsequent responses to the controller.
- The *link speeds* declare the transmission capability of the nodes outgoing link(s). There is one link speed entry for each port on the responder node. If there is no connection at the port a link speed of zero is returned.

Table 2 shows the mapping table that is produced when the configuration discovery process is applied to the network in Figure 1.

6 Conclusions

This paper describes a simple discovery protocol suitable for mapping the configuration of an irregular home network comprised of routing switches and heterogeneous devices. In software simulations the discovery process has mapped a variety of irregular network configurations correctly, and has proven to be robust in its operation over a period of several weeks. In hardware demonstrations involving simple networks [12], the discovery process has proven effective also.

The configuration data obtained during the discovery process is essential for the tasks planned for the future, including, the evaluation of quality of service requirements, and the provision of both unicast and multicast message routing strategies.

Acknowledgements

The author is grateful to the EPSRC and to 4Links Limited for their sponsorship of this work.

Table 2: adjacency list for the Figure 2 network

Node identity	Controller to node route	Node to controller route	Number of ports	Links*			
				port 1	port 2	port 3	port 4
devA	[1, 2, 1]	[1, 2, 1]	1	rtr2/1, (10)			
rtr4	[1, 3]	[3, 1]	4	devC/1, (10)	rtr3/3, (70)	rtr1/3, (80)	rtr2/4, (250)
rtr3	[1, 4]	[4, 1]	4	devB/1, (10)	rtr2/3, (50)	rtr4/2, (60)	rtr1/4, (150)
rtr2	[1, 2]	[2, 1]	4	devA/1, (10)	rtr1/2, (30)	rtr3/2, (40)	rtr4/4, (200)
rtr1	[1]	[1]	4	ctrl/1, (10)	rtr2/2, (20)	rtr4/3, (90)	rtr3/4, (100)
ctrl	[ ]	[ ]	1	rtr1/1, (10)			
devC	[1, 3, 1]	[1, 3, 1]	1	rtr4/1, (10)			
devB	[1, 4, 1]	[1, 4, 1]	1	rtr3/1, (10)			

** The node's link(s) port details contain the node/port this port is connected to, and link transmission speed.*

References

[1] IEEE 1355 (ISO/IEC 14575) Rationale, http://grouper.ieee.org/groups/1355, 1998.

[2] IEEE 1355-1995 Standard for Heterogeneous InterConnect (HIC), "Low-Cost, Low-Latency Scalable Serial Interconnect for Parallel System Construction".

[3] B. M. Cook and N. H. White, "Java Joins IEEE 1355 in the Home Network", *Architectures, Languages and Techniques for Concurrent Systems*, Edited by B.M.Cook, IOS Press. ISBN 90 5199 480 X, 1999.

[4] S. Haas, D.A. Thornley, M. Zhu, R.W. Dobinson, B. Martin, "The Macramé 1024 Node Switching Network". *Microprocessors and Microsystems*, No. 21, pp. 511-518, Elsevier, 1998.

[5] S. Haas, D.A. Thornley, M. Zhu, R.W. Dobinson, N. A. Madsen and B. Martin. "Results from the Macramé 1024 Node Switching Network". *Microprocessors and Microsystems*, No. 21, pp. 206-210, Elsevier, 1998.

[6] Networks, Routers and Transputers: Function, Performance and Applications. Edited by M.D.May, P. W.Thompson and P.H.Welch, IOS Press, ISBN 90 5199 129 0, 1993.

[7] W. J. Dally and C. L. Seitz. The Torus Routing Chip, *Journal of Distributed Computing*, Vol. 1, No. 3, pp. 187-196, 1986.

[8] P. Walker, "ATM performance with RS232 cost for links between microprocessors – using the IEEE 1355 standard".

[9] Digital Equipment Corporation. "Autonet: A High-speed Self-Configuring Local Area Network Using Point-to-Point Links". *SRC Research Report 59*, 1990.

[10] A.V. Aho and J.D. Ullman, Foundations of Computer Science, C Edition. Computer Science Press 1995. ISBN 1 7167 8284 7.

[11] P.H.Welch, G.R.R.Justo, and C.J.Willcock. “Higher-Level Paradigms for Deadlock-Free High-Performance Systems” In R.Grebe et al. editors, *Transputer Applications and Systems '93*, Proceedings of the 1993 World Transputer Congress, Vol. 2, pp. 981-1004, Aachen, Germany. IOS Press, Netherlands. ISBN 90-5199-140-1. September 1993.

[12] 4Links Limited. http://www.4links.co.uk/, 2002.