Systems Aspects of the Cambridge Ring

R. M. Needham
Cambridge University

The Cambridge Ring is a local communication system developed in the Computer Laboratory of the University of Cambridge. It differs in various respects from some other local communication mechanisms such as Ethernet systems (Metcalfe & Boggs, 1976), and the purpose of the present paper is to describe the way in which the properties of the ring affect the general systems aspects of its exploitation.

The Ring

The ring has been described (Wilkes & Wheeler 1979) from an engineering viewpoint elsewhere; for the present purpose we need some relatively broad-brush facts about it. The ring is a communication system with a raw data rate of 10 megabits/sec which runs round the various buildings which constitute the Computer Laboratory. At intervals on the ring there is what will be referred to as a station (strictly a station unit and repeater) connected to a computer or other piece of apparatus by an access box. The stations are all identical except for their address and the access boxes are tailored to interface the station to whatever equipment is present. The ring functions as follows. The unit of transmission at the low level will be referred to as a "minipacket", and consists of a source byte, a destination byte, two bytes of data, and a few control bits. (The minipacket has been referred to in previous publications as a packet. A change of terminology is made so that the term "packet" is available for a slightly higher-level construct more analogous to packets in other systems.) To send a minipacket from station S to station R it is necessary to load the destination and the data into S and then to indicate that the minipacket is ready for transmission. The station itself inserts the source byte and the initial values of the control bits, and has the complete minipacket waiting in a shift register. As soon as an empty minipacket comes by on the ring, the station fills it up from the shift register. The minipacket then travels round the ring until the destination station R is encountered and then returns to S. The desired action is for the minipacket contents to be copied into R's register, but there are various reasons why this might not happen. Firstly, R might not be switched on. In this case the minipacket will return to S in its initial state as sent. This is interpreted at the sender as meaning 'ignored'. Secondly, R might have set a register known as the 'select' register so that minipackets from S are not accepted. The possible values for the select register are 'accept nothing', 'accept from anywhere', and 'accept from a specified station only'. If the minipacket is not accepted for this reason, it returns to S marked 'not selected'. Finally, the minipacket may be acceptable but the register in R not be free, R not having completed the processing of the previous minipacket received. In this case the minipacket is returned to S marked 'busy'. Assuming that none of these difficulties has occurred, the minipacket will return to S marked 'accepted'. Not until the minipacket has returned to S is S in a position to transmit again to R or anywhere else. The time taken for the minipacket to return is known as the ring delay and is currently of the order of 10 microseconds. Note that in all circumstances the minipacket is returned, and that the time taken for it to return is independent of what happened to it. When a minipacket has returned the sending station is notified and the access box is able to read the control information to determine the fate of the minipacket. The empty minipacket itself proceeds on its way suitably marked. If retransmission is needed for any reason the material is still standing in the station's register. It is not possible for the minipacket to be re-used by the sending station, thus producing an automatic anti-hogging control. The effect is that, where there are m stations, each station is guaranteed a minimum of 1/m+1 of the capacity of the ring. This guarantee is of more academic than practical interest, since all local communication systems are very much under-run practically all the time. To avoid the possibility of a sender generating too much traffic by simply repeating a transmission which receives 'busy', 'not selected', or 'ignored', the station becomes steadily slower and slower to inform the access box of what has happened, finally only doing so after 16 ring delays from the unsatisfactory return. Since retransmission

is entirely the responsibility of the access box (in practice usually of the computer behind it), overloading with spurious traffic is discouraged. Most systems on the ring have a time or repetition limit beyond which they give up retransmissions after "busy", and up to that limit they retry as fast as they can. To complete this sketch, we note that the error rate of the ring is low (about 1 in 5E11 bits).

The characteristics of the ring which affect our attitude to the systems aspects of its use are mainly these:

a) The chance of even a substantial block of material being damaged in transit is low, and

b) The timing requirements are not stringent, since failure to send at the maximum rate has no bad effects at all and failure to receive as fast as a sender is sending can do no worse than generate moderate and non-destructive busy traffic.

The direct consequence of these characteristics are firstly that large retransmission units may be used, which is attractive in the interest of protocol simplicity, and secondly as a result of b) above it is possible to connect rather simple devices which cannot sustain a data rate of anything like the regular megabit point-to-point, provided that the traffic is low-volume.

Low-level use of the Ring

If there is a very low level of intelligence at one or both ends of a transaction it may be reasonable to communicate using no protocol higher than the single minipacket. In such circumstances there is no flow control beyond that afforded by "busy" signals and little or no error control. This is only a reasonable way to proceed in rare cases in which there is an economic motive for avoiding even a minor amount of buffering to assemble larger units of transmission and in which data proceeds at a reasonably predictable rate. Some kinds of logging or monitoring equipment are suitable for use in this way, but the most obvious example is digital telephony in which a regular digitising chip may emit data steadily at a byte every 128 microseconds say. In this case we also notice that error control is not needed, since it is not a fatal matter if a few bytes are lost. If a piece of equipment has a known rate of reception of data, one can arrange to send to it at that rate; it would be possible to run say a 9600 baud terminal in this way. Although the ability to communicate at this very low level is a useful flexibility of the ring, in all ordinary circumstances there is sufficient intelligence available to handle higher level units, and we now turn to discuss these.

Packets

For most applications however it is expedient to assemble minipackets into larger units called packets (previously known as basic blocks). Packets are defined in such a manner that there is a considerable degree of flexibility available to the implementor in the amount of sophistication he puts in.

A packet consists of:

1. A header minipacket which consists of a fixed pattern of four bits, two control bits, and ten data bits. In its usual form a header minipacket contains the number of regular data packets which will follow it, together with an indication of whether the last data packet should be followed by a minipacket containing the checksum of all that has preceded it or whether it should be zero. Note the liberty given to the implementor to omit the checksum if it is too hard for him to compute one, but the check placed upon him by obliging him to put a fixed pattern, zero, instead.
2. A minipacket containing a port number which gives the logical destination of the packet in the recipient machine. A machine receiving a packet with an invalid port number is at liberty to discard it.
3. An appropriate number of data minipackets.
4. A minipacket containing a checksum or zero as appropriate.

The significance of this definition lies largely in what it does NOT say. A likely implementation is that a recipient machine sets its select register to receive from all senders, awaits a minipacket that looks like a header, sets its select register to receive from the source of the header minipacket only, and then receives the rest of the packet until it completes or times out. However it is quite legitimate for a high performance machine - here defined as one that can receive material, add to checksum, and store MUCH faster than the ring can deliver, to arrange to receive several packets at once. Notice that there is necessarily a restriction here to the effect that only one packet at a time may be received from a particular source, since there would be no way of knowing to which of two or more packets a particular minipacket belonged. The converse freedom is perhaps easier to exercise, namely to send several packets interleaved, provided they are to different physical destinations. This is potentially valuable, because some destinations may be known to be slow in receiving and therefore likely to generate busy traffic if bombarded as fast as possible.

Another effect of the loose timing implicit in the possibility of a "busy" return is that it is possible, if desired, to check the port number on the fly. It was stated above that a machine is at liberty to discard a packet with an unexpected port number; it may be of value to check the port number before the body of the block is read. This both avoids choking up buffers with material that will shortly be discarded by software and also makes it possible to arrange that certain material is transferred directly to its final resting place in memory, assuming that when a port number for reception is registered the desired memory address is given. Both of these facilities help to minimise the time after a packet has finished before another one can be read. It is of course possible to achieve this effect with most communication systems if one is fast enough, but the speed requirements can be very hard to meet if data arrives fast and synchronously. The ring makes it possible to take one's time within reason in examining the port number between receipt of the second minipacket of a packet

and the third. If one cannot be fast enough then there will be the odd "busy", which is not a serious matter.

We regard the freedom this gives to implement the same protocol on both fast and slow equipment as a great advantage. If we consider the way in which the packet protocol has been or is being implemented on various machines, the effect of the flexibility will be evident. On the CAP, a computer built for research into memory protection but now available as a general purpose shared machine, the packet protocol, including port number checking, is handled entirely by microcode. On two PDP11s almost the whole of the work is done by software, there being an interrupt per minipacket. On a Computer Automation LSI4 the work is partly done by a rather simple microprogrammed controller in the connection cable and partly by software. Some simple microcomputers mentioned further below spend much of their time polling the signals from the stations, and can only manage to receive a minipacket about every 40 microseconds while also implementing the packet protocol by program. On the other hand a much faster microcomputer which is being developed specifically to aid in high-performance connections for 16-bit minis certainly has the power in hand to multiplex packets if one wants to. The only penalty that seems to be paid for this flexibility is that time-out constants associated with the sending or receipt of a packet have to be large enough to avoid making communication impossible for the weaker brethren, and this means that they are perhaps rather lax for faster equipment. On the other hand the incidence of error is low enough for there to be really very little problem. It should also be remembered that for extremely slow devices the option remains of not using the packet protocol at all and working at the 2-byte minipacket level.

Peripheral Services

The ability to handle rather slow communication led us to a particular approach to the provision of peripheral services. If we wish to make, for example, a printing service available via the ring, there are several functions to be performed. There is low-level control of the device and there are also higher-level activities such as spooling, scheduling, accounting, and maybe some formatting. One way to proceed would be to attach the printer to a computer interfaced to the ring; this computer would then carry out all of the functions mentioned. This approach requires the construction of a device interface, the effective dedication of a machine, and the physical presence of the machine where the peripheral is. The first is essential but the second and third may be a nuisance especially where the device is used unpredictably but not very often - as for example a pointing machine used in the production of wire-wrapped prototypes. Another approach is being taken at Cambridge. Peripherals are connected to small, cheap, and simple microcomputers which carry out the device control functions and very little else beyond a simple ring connection. One of these microcomputers, consisting of a dozen and a half chips, is all that need be dedicated to a particular peripheral. The higher level functions mentioned above are left to "real" machines which do not have to be permanently dedicated or fixed in function. When a computer is assigned to carry out the higher-level functions, it is that machine which is regarded by the ultimate clients as being, for example, the printing server; they are unaware of the existence of the controllers and of the means of communication with them. There is a reliability advantage too because the computers are interchangeable.

Reliability and simple protocols

Other aspects of the use of the ring derive from its reliability. It is the practice, for example, to use it for archiving discs with a retransmission unit of one discful or 28Mbytes. This is not perhaps a practice to be recommended, but it makes the point. A consequence is that protocols are there for reasons of (end-to-end) flow control rather than of error management. It is possible to take the view that the major cause of bad communication is the failure of one of the computers involved, as for example by software crashes, a program not paying attention to ring signals, or someone unplugging the power, rather than loss or damage to bits in transit. The only way to deal with loss of the distant computer is by timing out, and one might as well implement the timeout at a reasonably high level in the software so as to deal with all lower-level errors indiscriminately. It thus becomes feasible to ignore packets which are in any way erroneous rather than to send an explicit negative acknowledgement of them. A further consequence is that in a number of circumstances protocols without flow control can be used and become very simple indeed because of the relaxed attitude that may be taken to errors. For example, in using the ring to load a program into memory, one would clearly not make the request unless the memory into which to load it was ready and waiting. Accordingly it is proper to respond to such a request at the fastest rate possible, and all that is necessary is for the recipient to be able to verify that what has arrived is correct in quantity and apparently correct in content. In a single ring minipackets cannot overtake one another and a fortiori packets cannot do so. Furthermore no packet will be sent twice, since there is nothing which would provoke a sender to retransmit one. It is thus extremely simple to check that all is well. In the case of the CAP implementation the packet protocol is handled by microcode, and the microcode facilities have been extended to the handling of multiple packets, on the same port, as a single request from a program. It is thus possible to hand over to the CAP microprogram a request to accept 43592 bytes from machine M into CAP's memory at place P using port 97. The microcode will do what is necessary, only raising an interrupt when the transfer is complete or has timed out or otherwise failed. Of the machines at present on the ring only the CAP has this facility at microcode or equivalent level, and there is no reason why other machines should necessarily do so. The decision is entirely up to individual designers. The rather elaborate CAP implementation is done the way it is because of a desire to use the ring for swapping of segments: hence the multiple block facility and the on-the-fly port number check mentioned earlier. What matters is that the sender to the CAP need not know about it. In the unlikely event that all is not well, the

simplest action is to repeat the entire operation, not to fuss about which bit of it failed. It is along these lines that the ring protocol for use of the file server (Birrell and Needham, 1979) is being constructed. With a little ingenuity all operations made available to clients by the file server can be made repeatable in the sense outlined, and the overhead of establishing connections is avoided. The main technique for making things repeatable is illustrated by the following snatch of hypothetical dialogue between the CAP and the File Server:

CAP program (to ring microcode) "Be prepared to receive x bytes of of material on port z from machine M (which is the file server) and put it at address y in memory"
CAP program (to file server via ring) "Please rush me x bytes of file F on port z"
We suppose that nothing happens and the CAP program gets fed up; its action is:
CAP program (to ring microcode) "Cancel port z; be prepared to receive x bytes of material on port z' from machine M and put it at address y in memory"
CAP program (to file server via ring) "Please rush me x words of file f on port z' "

If the file server was in fact merely being sluggish and was about to honour, or was actually honouring, the original request, the cancellation of the port number z will cause any material arriving for it to be discarded as fast as it comes, and will not interfere with the second attempt.

The example of the file server has been chosen because it is slightly complicated; for many services a single packet exchange will do, without having to bother with cancellation of ports before a retry. This applies to lookup services (including bootloading the various microcomputers), date and time services, and so on. In order to keep some regularity in such programs, a single-shot protocol has been defined. This simply consists of a set of recommended standards for the layout of packets used to request services; although this protocol only takes a few lines to define its existence has been highly beneficial.

Conclusions

The approaches to local communication that have been described in this paper depend thoroughly on the properties of the Cambridge Ring itself. It is sometimes considered undesirable to exploit such properties, it being thought preferable to obscure the nature of the medium being used as much as possible in order to simulate some ideal system which may in principle be implemented using many different media. Although it is necessary to do this in dealing with wide-flung communication, it is not so necessary in a local system, and to do so may lose advantages which can flow from the proper exploitation of particular equipment. Local communications may be treated as being seriously different from others, and the work reported is part of an effort to explore the consequences of doing so. If it turns out as the result of practical experience that there is serious advantage to be gained from exploiting the ring's special properties, then attention will be paid to such questions as the interconnection of rings in ways which preserve the simplicity which exists for a single ring.

The methods and suggestions discussed here owe much to colleagues at Cambridge, especially Martyn Johnson and Robin Walker, and also to conversations with David Boggs of Xerox Palo Alto Research Center.

References

Birrell,A.D and Needham,R.M. "A Universal File Server", to appear in IEEE Transactions on Software Engineering
Metcalfe,R and Boggs,D. "Ethernet: Distributed Packet Switching for Local Communication Networks", CACM July 1976
Wilkes,M.V and Wheeler,D.J. "The Cambridge Digital Communication Ring", Local Area Communication Networks Symposium, Mitre Corp. and National Bureau of Standards, Boston, May 1979