Research Task Force. 1988

Proposed Protection Scheme.

I thought we should start our design with a protection scheme.

Objective.

~~To permit same degree of dynamm~~

(1) Provide mechanism for loading into a user machine (portions of library packages from various sources, before running the user program

(2) Provide a means for a library loaded into the same transputer as its user program package to communicate with shared resources located semipermanently in ~~fixed~~ fixed regions of a transputer network.

(3) To provide protection of library code against the user program, and other portions of library code loaded with it.

(4) To ensure that proper termination messages reach the [redacted] shared library resources in the event of expiry of time limits or other detected misbehaviour

of the user program.

(5) To allow library packages to offer (software) a facility for dynamic allocation and release of virtual resources.

(6) To provide a method for dynamic updating and reloading the code of the service modules.

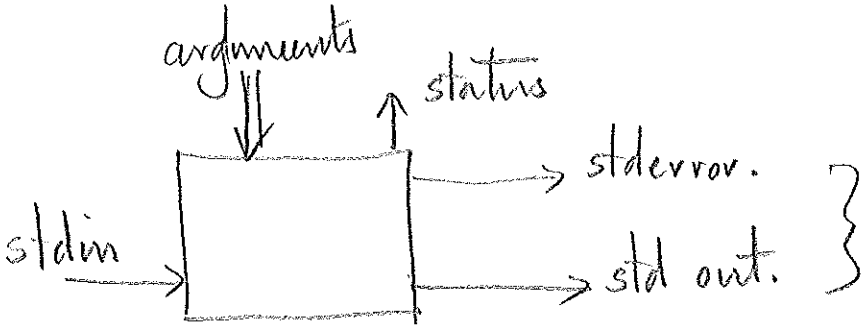
Method.

Based on the virtual resource paradigm of Pascal Plus (Structured Systems Programming: Welsh and McKeag), adapted for communication by the theory developed in Chapter 6 of my book.

Proposal.

A library module should be written like a PASCAL PLUS monitor, with ~~none or more~~ a collection of processes to run permanently

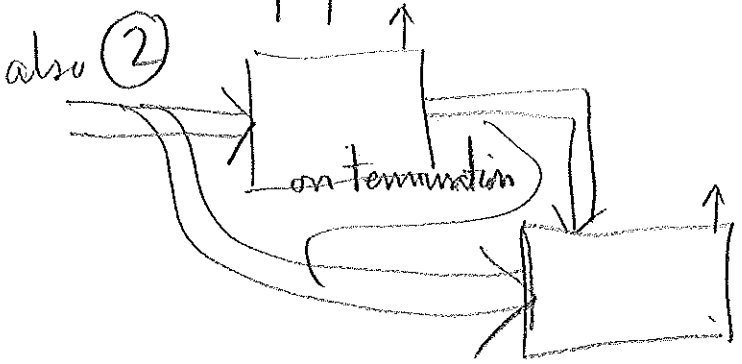
Program as black box

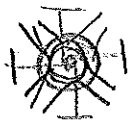


why not put arguments at head of input stream.

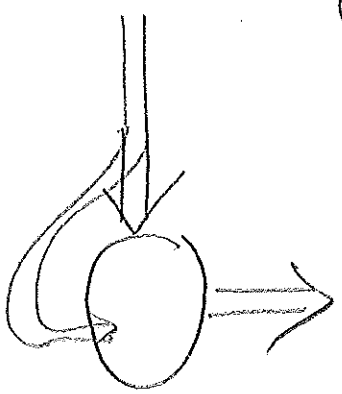
Combining form

(1) pipe in version 6





(3) stdin from arguments



global variables set in shell.
default parameters.

Objects

programs code

typed external channels

typed free variables

domain: processors
memory
links

Operations are:

bind: joining channels
supplying context for free vbls.

run: allow a (partially?*) bound program to

*rest of binding done by program itself

need to add to old unisc.

- Need richer binding environment
  better than parameter list.

- Combining forms
  richer than local pipelines.
  (can't bind a channel to existing program).

need to subtract

files & devices — replaced by
persistent processes.

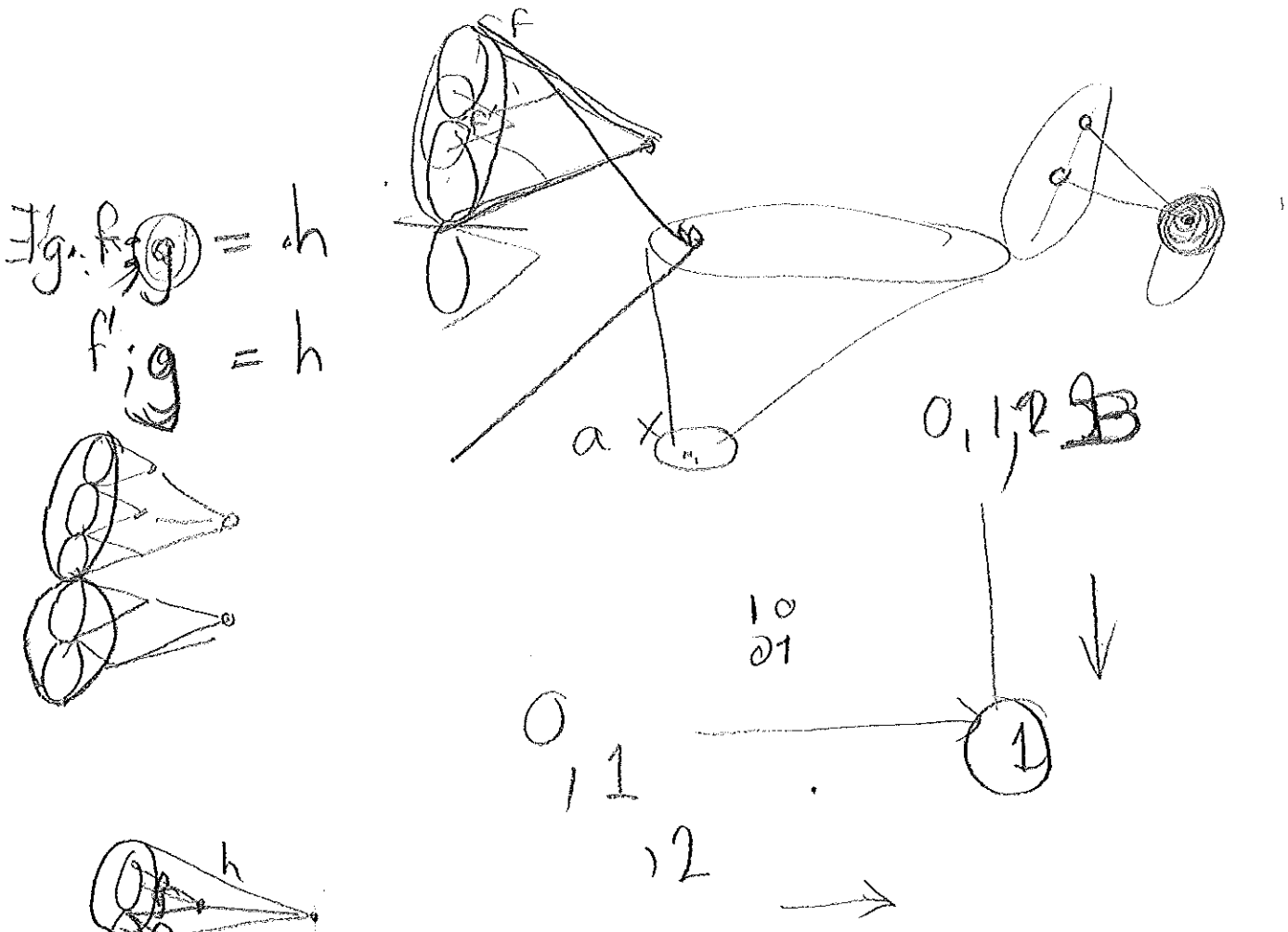
signals
replace by messages and
a recovery mechanism.

don't ∴ don't need to be part of os
kernel.

grumbles about developments of unisc.
how to prevent the same rot.

41 system calls. in original UNIX.
1000 lines assembly + 9000 C.

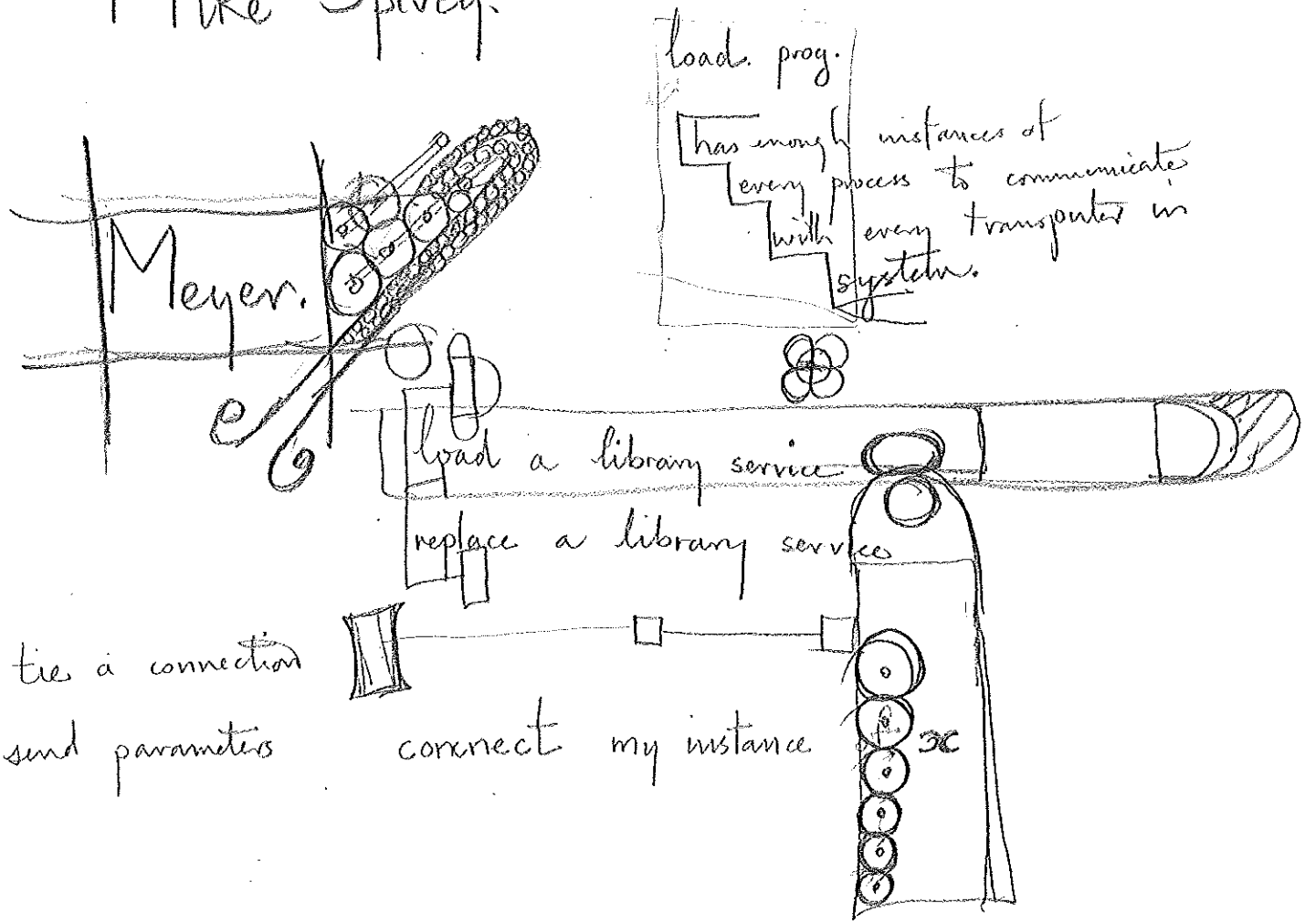
Berkisc 158 system calls
>> 327 000 lines of source.



questions: is dynamic binding really necessary what kind of language.

Bill & Mike? their book ☺

Mike Spivey.



~~Synopsis~~

We must send Profiles of

Myself

Reed

Roscoe

Gardiner.

to Jos Baeten.

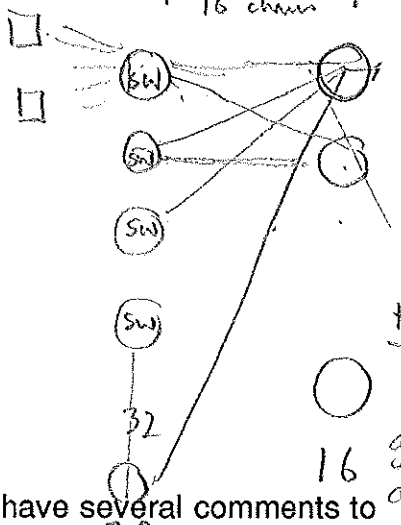
too much effort learning others under 3.1

inmos

INMOS Limited
1000 Aztec West
Almondsbury
Bristol BS12 4SQ
(0454) 616616
Telex 444723
Fax (0454) 617910

cc: Les Valiant
Richard Miller
Bill McColl

512 processors
32 groups of 16
each group has link into 16 chans



4 bits
+5 bits
9 to get routing among 512 pro

total no switches:
48 per link
x8
400?

Professor C A R Hoare,
Programming Research Group
8-11 Keble Road
Oxford OX1 3QD

Dear Tony,

Thank you for your letter to Roger Shepherd. I have several comments to make. Firstly, it seems that there is some confusion: we do not intend to build transputers which can only be employed in two dimensional arrays. It seems unwise for Inmos to choose a specific system structure at present.

I intend to use the routing system initially in two products:

1 A high performance transputer (probably upwards compatible with the T800) with 8 links

2 A routing switch with 32 links

These two components should support a very wide variety of system architectures including small fully connected systems, hypercubes and large arrays. I would not be at all surprised if 'general purpose' concurrent computers will need to employ ten times more routing switches than transputers.

wires. 100 X wires.

Even in systems with a relatively large amount of interconnect, I do not expect to remove the need for good (software) allocation techniques. It is clearly desirable that the physical length of (most of) the virtual channels is short.

I agree that the experience with the ARPAnet is of no relevance to us. The interval routing system was chosen because it appears to efficiently use message address bits and can be implemented efficiently on silicon. The techinque of stripping (say) the first 4 bits of the incoming message and using them to select the outgoing link requires many more address bits in any realistic system; on the other hand it is these additional bits that provide the many possible routes from one node to another. I can see that it would be attractive to allow two virtual channels between the same pair of nodes to take different physical paths (although I find it difficult to assess the importance of it). Intuitively I am not very happy with either scheme and would welcome a better suggestion.

no, its optimal

completely

inmos

I agree that whatever system is chosen it must deal effectively with input and output - and with disks. However, if it is true - and will continue to be true - that disks have intermittent transfers of high data rate then it would be best to connect them using one of the existing configuration switches. A neater scheme still would be to use the routing switch mentioned above.

I enclose a note by Peter Thompson and a copy of the paper about the interval routing system.

I look forward to discussing these issues further.

Best wishes,

David

David May, January 14, 1988

OXFORD UNIVERSITY COMPUTING LABORATORY
PROGRAMMING RESEARCH GROUP

Professor of Computation:
C.A.R. Hoare, FRS
Professor of Numerical Analysis:
K.W. Morton

8 - 11 Keble Road
Oxford OX1 3QD

Tel: Oxford (0865) 273840

5th January, 1988

Dr. R. Shepherd,
INMOS Ltd.,
1000 Aztec West,
Almondsbury,
Bristol. BS12 4SQ

Dear Roger,

Thank you for coming to visit us just before Christmas, to discuss prospects of switching algorithms for computer networks. This seems an admirable development for transputers, and will greatly ease problems of building them into large nets.

However, we were concerned about the possible implications of some of the remarks which you made in passing, and we are writing to see if we can help you converge rapidly on a design of high quality.

We understand that you have adopted the policy that all actual routing and the correspondence of actual and virtual channels, will be done at compile time, giving (at least initially) some degree of control to the programmer via a configuration statement. This means that the "header directive" technique of determining the route of a message must be the first candidate for consideration. This was the recommendation made to INMOS by David Wheeler many years ago, and has been used in switches for some time.

The interval routing technique was (I am told) originally designed and engineered for a completely different environment, more like ARPAnet, where the objective is to permit dynamic reconfiguration without any central administration. Experience with this technique does not seem relevant to your requirement. In fact, the restrictions of interval routing can only make the task of designing a virtual/actual correspondence more difficult.

Our second concern is that of thrashing. The definition of thrashing is that a high proportion of the resources of the system are occupied by processes which are waiting for additional resources. As soon as a block of resources is released, they get immediately redistributed to the remaining processes. In the extreme, this leads to deadlock; but even if deadlock has been proved impossible, the whole system rapidly declines to a state in which only one person at a time can make progress (i.e. only one message at a time gets through!) This state is remarkably persistent.

The only ways of reducing this risk are (1) to reduce the amount of resource needed by each task, and (2) to share each resource evenly among as few competing tasks as possible. This is the main reason why it is important (1) to keep the physical length of the virtual channels quite short, and (2) to ensure that different virtual channels (even between the same pair of physical processors) should take different routes. We fear that your suggestion of connecting processors on a two-dimensional grid will fail to achieve (1); and that your interval routing technique will fail to achieve (2). Some denser interconnection technique such as hypercube is a solution which is widely adopted, and will be widely expected by your customers.

It is possible that the success of the two-dimensional grid connection of the present generation of transputers has encouraged you to set aside some of the problems which have not yet been widely encountered. One such problem is the use of high-volume disc stores for long-term storage. These will presumably be connected to a subset of the transputers in a system. The load pattern of disc transfers is extremely unfavourable - very intermittent, but with high data rates. It is therefore essential that each processor should have at least one very short path to each of the discs. The same problem arises in sharing one or more visual displays, perhaps using windows to enable many processors to communicate simultaneously with a user.

These comments may well be based on a misunderstanding of your current thinking; and we look forward then to removing the misunderstanding, and helping you in future in other more constructive ways.

Yours sincerely,

Tony

C.A.R. Hoare

cc: Dr. D. May

P.S. Les Valiant has seen this letter in draft, and supports the general conclusion

T.

SMITH ASSOCIATES LTD.

SURREY RESEARCH PARK GUILDFORD SURREY GU2 5YP ENGLAND TEL: GUILDFORD (0483) 505565 TELEX: 859057 SACSEL G FAX: (0483) 506976

TONY HOARE

Professor C.A.R. Hoare,
Oxford University Computing Laboratory
Programming Research Group,
8-11 Keble Road,
OXFORD,
OX1 3QD.

Our Ref: CJE/bew:M/41
Date: 6th June 1988

Dear Tony,

I was very interested to hear of your ideas to use a spacecraft environment as a model problem to define work on a distributed operating system. I should be very interested to help define the "boundary conditions" imposed by the environment and contribute in any other way that I can.

I shall contact the satellite engineering team at the University of Surrey to ask if they would help draw up the definitions. If they are willing, I shall contact you again to arrange a meeting at Smith Associates in Guildford.

Yours sincerely,

Dr. C.J. Elliott,
Director.

paradoxical to share wires by preemption.
<.1 ms,
allocate store on loading > 10 s
proc

Tony Hoare

larger systems will use standard hardware and protocols. for loose coupling

Notes on Meeting with David May

Jonathan Bowen

12th May, 1988

A 32-processor system with shared virtual memory would be a much more manageable software problem.
1024 +32 Links
32 of them omnidible

Winterbottom.
Data Base on Transputers

Research Task Force
Oxford University Computing Laboratory
Programming Research Group
8–11 Keble Road
Oxford OX1 3QD

A meeting was held with David May of INMOS and members of the Programming Research Group in Tony Hoare's office at Keble Road, on 12th May. The following were present: David May, Prof. Tony Hoare, Richard Miller, Geraint Jones, Jonathan Bowen.

These notes are intended to record the main points covered at the meeting and to provoke further discussion. The meeting considered the design of a distributed system using transputer-style processors and some sort of *link* switching chip (e.g. 32-way).

A number of design choices must be made. The following were suggested:

- Fixed routing rather than dynamic routing configuration.
- N-way switch better than hypercube.
- Randomisation gives approximate send on first free channel.
- Time out can be used instead of explicit messages — a channel must not be reallocated till both ends are cleared up.
- Message passing preferable to remote memory access (cheaper).

It was suggested that a hierarchical approach should be adopted. Intercommunicating trusted monitor processors should each control a number of child user processors (or a number of other monitors for a large system). There may be a (possibly conceptual) "grand-daddy" monitor in charge.

?? I wonder why very flat tree wired in. ✓ one level may be best for hardware/software reasons.

Failures should be split into two concerns:

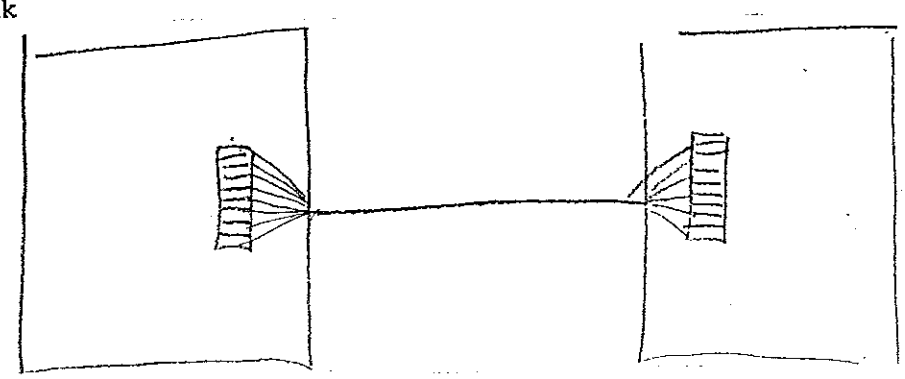
1. Low level protocol — e.g. board down.
loss of a bit. hardware maintenance.

But "shared store" has less software problems.

Separation of concerns

In order to make progress in the many inter-related topics we have chosen to study, it will be essential to find a way of separating concerns, preferably in a manner corresponding to physical separation of chips and wires in a distributed system. Here is a preliminary draft, just to illustrate what I mean, separating two concerns in the design of a new switch chip.

1. The multilink



The multilink is a device for multiplexing say eight permanent virtual links through one physical link which connects two transputers.

The links are configured by the occam compiler exactly as if they were eight physical links.

The virtual links are treated by the rest of the hardware of the transputer in exactly the same way as if they were physical.

2. The multiswitch

The multiswitch should have almost exactly the same logical behaviour as the existing C00 switch. The only differences are

(1) A multiswith may be constructed with any number of physical link connections, instead of only thirty two.

(2) Any fixed(?) subset of the physical links may be multilinks.

The control link may be used to establish one-to-one (expensive, fast, dedicated) physical connections between any subset of the unilinks (just like the present switch).

It can also establish a larger number of one-to-one (cheaper, slower, multiplexed) virtual connections between any subset of the virtual ports provided by the physical multilinks.

The allocation of virtual channel identifiers and the routing algorithm is entirely invisible from outside.

The principle of separation of concerns requires that neither the occam compiler nor the transputer hardware should know that the multiswitch is constructed from a larger number of 32-way switches.

This achieves the essential requirement that no transputer can ever send messages into the network except on channels that have been set up for it by the transputer connected to the control link.

Research Task Force
Programming Research Group

The next meeting will be at 11.30 a.m. on Tuesday 24th May in Tony's office (KG3) at Keble Road.

Robert Isherwood will talk about structures and mechanisms for supporting the distributed operating system which is being developed at INMOS.

Jonathan Bowen
17th May, 1988

To:	Robert Isherwood	Inmos
	Miles Chesnay	Meiko
	Roger Gimson	Hewlett Packard
	Tim Gleeson	Cambridge University
	Geoff Barrett	
	Michael Goldsmith	
	Jeremy Jacob	
	Geraint Jones	
	Les Valient	
	Bill McColl	
	Quentin Miller	
	Richard Miller	
	Ian Page	
	Martin Raskovsky	
	Mike Reed	
	Bill Roscoe	
	Bernard Sufrin	

Cc: Tony Hoare

Research Task Force
Programming Research Group

The next meeting will be at 2.00 p.m. on Tuesday 17th May in the conference room (KT2) at Keble Road.

Topics

1. Resumé of meeting with David May on 12th May.
2. Separation of concerns (Tony).
3. Case study: the Amoeba distributed operating system — process management.
4. Topics for next meeting.

Jonathan Bowen
16th May, 1988

To:	Miles Chesnay	Meiko
	David May	Inmos
	Roger Gimson	Hewlett Packard
	Tim Gleeson	Cambridge University
	Richard Miller	
	Geraint Jones	
	Michael Goldsmith	
	Les Valient	
	Jeremy Jacob	
	Quentin Miller	
	Bill McColl	
	Bernard Sufrin	
	Ian Page	
	Mike Reed	
	Martin Raskovsky	
	Bill Roscoe	
	Geoff Barrett	
Cc:	Tony Hoare	