A Direct Communication Model for Process Management in an Open Hypermedia System

Rob Wilkins
Ian Heath
Wendy Hall

CSTR 93-14

© University of Southampton

Department of Electronics and Computer Science
University of Southampton
Southampton SO9 5NH

Abstract

This paper describes the design and implementation of a new, direct communication model for the Microcosm open hypermedia system. Both the current and new communication models are described in detail, along with the major differences between the two, and how those differences will affect the overall performance of the system. In addition, it describes the effect that the new model will have on user interaction with the system. A prototype system constructed from a small subset of the existing Microcosm modules is described. Results derived from comparing the two systems and conclusions are then presented.

Contents

1. Introduction. 2
2. Current segregated uni-directional chain based model. 3
   2.1. Communication pathways. 3
      2.1.1. Viewer to filter 4
      2.1.2. Filter to viewer 4
      2.1.3. Filter to filter 5
      2.1.4. Viewer to viewer 5
   2.2. User interaction with the current system 6
   2.3. Problematic areas with the current system. 6
3. New direct communication model. 9
4. Prototype construction. 11
   4.1. Potential for cleaner process interaction. 12
   4.2. New user interaction with the system. 13
   4.3. Synchronising processes 14
5. Results 15
6. The Need for a "Process Manager" 16
7. Summary 16
8. Conclusions 17
Bibliography 18

1. Introduction.

Microcosm is a distributed open hypermedia system [Davis92][Davis93][Fountain90] designed to support the integration of many different applications. It is not a single application; but instead composed of many concurrently running communicating processes. It is by the interaction of these processes that the Microcosm functionality is achieved.

Currently these processes are sub-divided into two basic groups, the viewers and the filters. The viewers allow the user to interact with the information within the system and the filters provide a variety of functions ranging from navigation tools to link construction and storage tools.

The majority of the processing is carried out amongst the filters, communicating via a uni-directional chain. A filter receives messages, it may then interpret them, but in addition it must pass them on to the next filter in the chain. One filter cannot communicate with another which is positioned earlier in the chain. This leads to problems in ordering the filters; if they are in the wrong order then the system may not perform as desired.

Any communication between viewers and filters is routed through the document control system (DCS) before it can be sent to the filters. In practice this produces an unwanted communication bottle neck slowing down any interaction between the two process types.

The new direct communication model replaces the chain with a system of direct bi-directional channels. A system of abstracted connections is constructed on top of the basic bi-directional channels effectively connecting any process to any other process. This system of abstract connections can be dynamically modified by any process. In the new model there is no distinction made between the viewer and filter processes.

Messages are intelligently routed by the communication system and are only sent to the processes which need to receive them. Bottle necks and single channel sequences, two features which restrict the speed and scalability of the current system, have been replaced by a dynamically configurable bi-directional network of channels.

Although the new model represents a significant improvement over the current system it still contains scope for further development. The areas that need such development are discussed and the paper then draws a variety of conclusions from the work already carried out.

2. Current segregated uni-directional chain based model.

The current Microcosm communication model [Hill93] is clearly segregated into two halves, the viewers and the filters. Each half has its own control process; the DCS for the viewers and the Filter Management System (FMS) for the filters. All communication is routed through one of these two processes, or in some cases both. This design comes from extensions to an earlier system, described in [Hall 92][Heath92]. The diagram below shows the conceptual communication channels which exist between the processes.

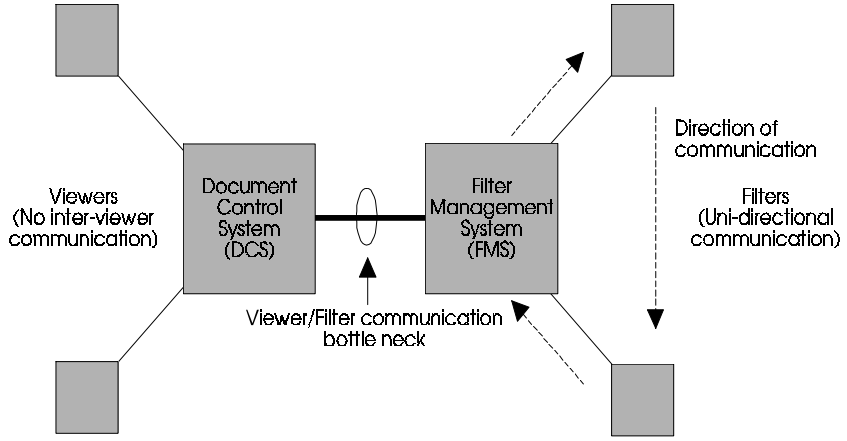


Figure 1 : Topological view of current communication

In contrast, figure 2 shows the actual channels which exist. The topological differences between the two diagrams are due to the two routing processes, which dictate the distribution of each message. Different message pathways exist, depending on the locations of the communicating processes. The four pathways are discussed in more detail in the section below.

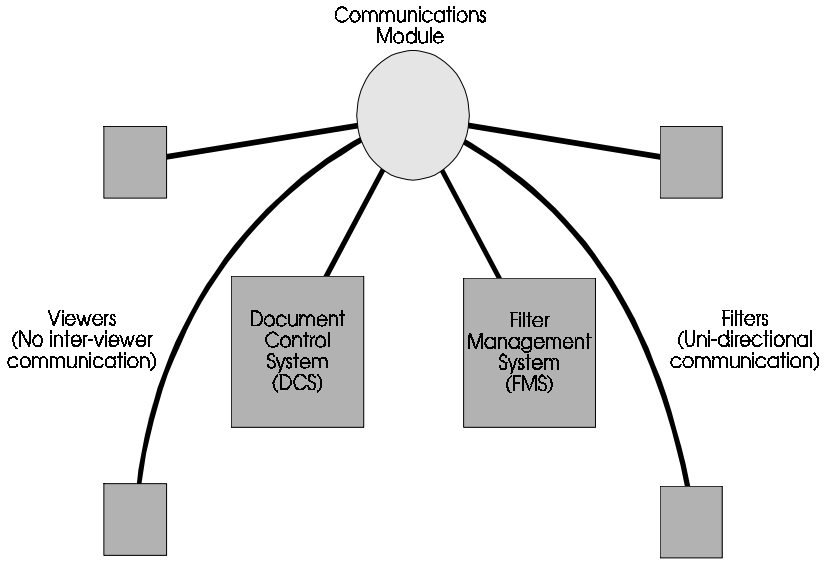


Figure 2 : Current communication model.

2.1. Communication pathways

The current system contains four different communication scenarios, which are due to the filter/viewer distinction (the two types of process imply four possible combinations). These are:-

- Viewer to filter.
- Filter to viewer.
- Filter to filter.
- Viewer to viewer.

The sections below detail the routes through which messages must currently travel (and the problems with those routes) for each scenario. The sending and receiving processes in the diagrams are marked with bold outlines. The solid arrows shows the path which a message has to travel if there is a receiving process. If there is no receptor, the extra path along which the message travels, before it is deleted, is shown by the dashed arrows. The following four diagrams consider a two filter configuration. The addition of more filters will result in longer message pathways. This will have the most impact on the message pathway length in the viewer to viewer scenario (see section 5).

2.1.1. Viewer to filter

When a viewer wishes to communicate with a filter it must send a message to the DCS. The message is then sent to the FMS and on down the filter chain until it reaches the appropriate filter. This route is shown in the diagram below.

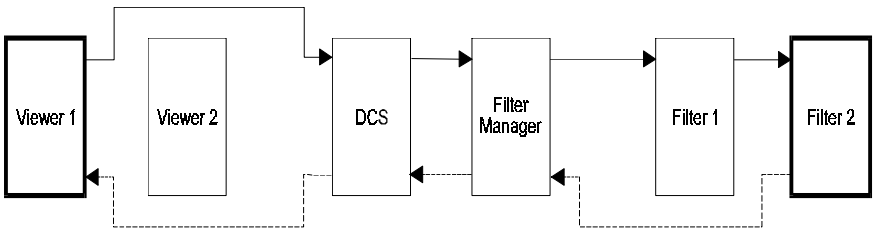


Figure 3 : Route for a message from a viewer to a filter.

If there is no receiving process for the message, upon reaching the end of the chain it will be sent back to the DCS. The DCS will then send the message to the viewer that originated the message. As the viewer was sending the message to a filter the latter half of the message path is always redundant.

For example when a document is opened within Microcosm, the document viewer sends out a request for any links that need to be displayed. This request is received by all link databases (linkbases) in the system.

2.1.2. Filter to viewer

When a filter wishes to send a message to a viewer it must first pass through the remainder of the filter chain. The FMS then sends the message on to the DCS, the DCS then attempts to send the message to a viewer. This route is shown below.

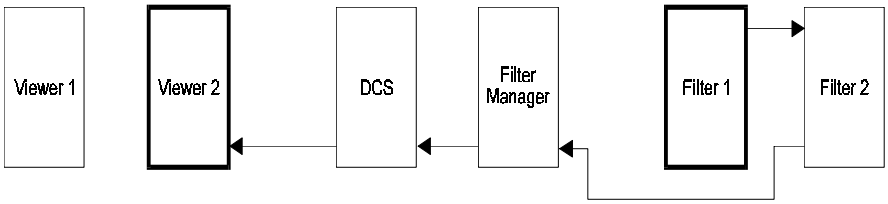


Figure 4 : Route for a message from a filter to a viewer.

The DCS determines which viewer the message will go to by examining its contents. If the DCS is unable to do this then the message is deleted. In the current system there is no explicit method for uniquely identifying a viewer which greatly increases the complexity of this scenario.

For example when a linkbase receives a request for links from a document viewer it responds, sending back one message for each link found. The linkbase is able to extract the viewer's identity from the original link request message. This identity is used to route the link messages back to the correct viewer.

2.1.3. Filter to filter

When one filter wishes to communicate with another it sends the message to the next filter in the chain and so on until it arrives at the receiving filter. This route is shown in the diagram below.

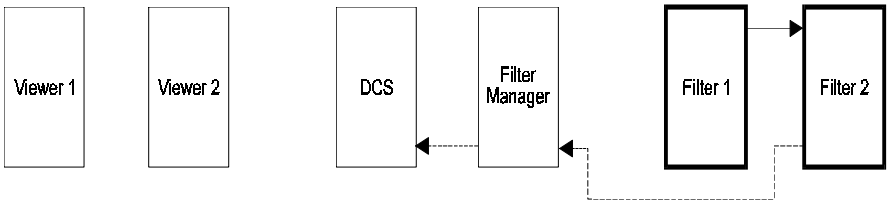


Figure 5 : Route for a message from a filter to a filter.

If the receiving filter is not later in the chain then the message will follow the same path as an unwanted message in the viewer to filter scenario (see section 2.1.1).

For example when a linkbase receives a message to search for a set of links which match a specific selection it replies by sending off a set of messages. These messages are received by another filter which displays them as a list of destinations for the user to choose.

2.1.4. Viewer to viewer

If a viewer wishes to send a message to another viewer then the message has to go to the DCS. From there it goes to the FMS and along the complete filter chain. It then is sent back to the DCS and then finally to the viewer. The message route is shown in the diagram below.

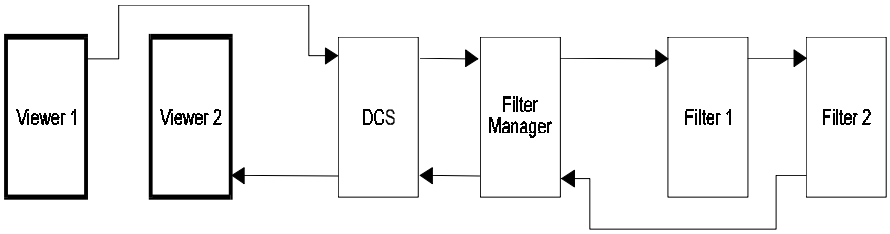


Figure 6 : Route for a message from one viewer to another viewer.

The message has to travel through the filter chain because the DCS has no mechanism for directly routing it to the other viewer. This means that the message has to travel along a much longer path than is necessary.

The current Microcosm system contains no examples of viewer to viewer communication. It is envisaged that such communication will become necessary if the system is to allow for viewers to be synchronised.

2.2. User interaction with the current system

In the current system the only method available to the user for beginning a Microcosm session is to start the DCS process. This then starts the FMS which starts the filters. The user chooses a document from the DCS icon and the relevant viewer is started up. This method of interaction is significantly different to the user's normal interaction with the windows environment. Users mostly locate a file from the file manager which is associated with a certain application. The mere act of double clicking on that file starts up the relevant application on the desired document. This method of opening documents is supported within the proposed new system.

2.3. Problematic areas with the current system.

The majority of the communication overhead in the current system is due to the messages travelling along the filter chain. This is not due to the generation of a large number of messages, but to the serial nature of the chain mechanism. Filters must pass on all messages to the next filter in the chain.

Consider a multi-tasking environment in which viewer processes can be accurately synchronised. Suppose the user causes five viewers to open at the same instant. Upon initialisation those five viewers send messages to all linkbase filters present (e.g. requesting the locations of any text buttons). All of those

messages must be funnelled through the channel between the DCS and the FMS. They must then be bounced through all the filters to the linkbases themselves. The linkbases process the five messages, producing perhaps hundreds of messages which must be returned to the five viewers. The messages produced in parallel must be serially sent down the filter chain, then between the FMS and the DCS. They are only distributed at the last stage when the DCS sends the messages to each of the viewers. The net effect of this is to slow down the whole system. Such slowing down is a direct result of the number of messages generated within the system. The following formulae shown below can be used to calculate the number messages generated (in the current system) for the events outlined above. Formula (1a) is the number of messages generated when the *v* viewers send off link requests to the linkbases. Formulae (1b) is the numbers of messages generated when the linkbases at consecutive positions *p*, *p+1*, *p+2* in the filter chain respond with *k* links. Formula (1c) is the total number of messages generated in the above situation.

(a) $v[2f+3]$

(b) $vk\sum_{q=(L-1)}^{q=0} 2(f-(p+q))+3$

(c) total = (a) + (b)

f = number of filters in filter chain
v = number of viewers
k = number of links returned per linkbase
L = number of linkbases
p = position of linkbase 1 in filter chain

Formula 1: Calculating no. messages generated in the existing system.

The following formulae represent the calculations shown above but for the direct communication model. Formula (2a) represents the same calculation as (1a), formula (2c) represents the same calculation as (1e). As the new model contains no chain, hence no ordering, formulae (1b) can be represented by the simplified formula (2b).

(a) vL

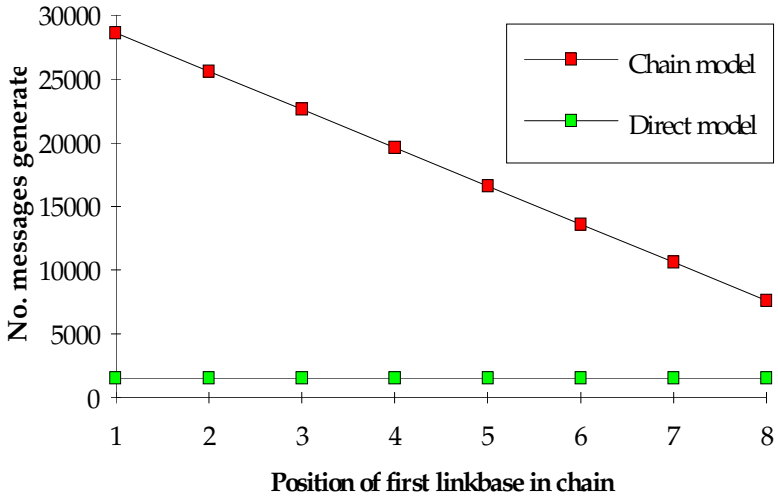
(b) vLk

(c) total = (a)+(b)

f = number of filters in filter chain
v = number of viewers
k = number of links returned per linkbase
L = number of linkbases
p = position of linkbase 1 in filter chain

Formula 2: Calculating no. messages generated in the direct communication model.

The above sets of formulae can be applied to equivalent instances of both models. For example suppose we have ten filters (including three linkbases) and five viewers in both systems. For simplification we can assume that each viewer will open the same document and each linkbase will contain the same set of links. The graph below shows the number of messages produced in the following situations (in all cases k=100 and l=3). The graph shows the number of messages generated against the position of the first linkbase in the chain.



Graph 1: Number of messages against position of first linkbase in chain.

In the current system there is no concept of direct viewer to viewer communication. For example, if one viewer wishes to synchronise itself with another, messages must travel along the path illustrated in section 2.1.4. The extra time lag and processing overhead will make such synchronisation of viewer processes very difficult.

If a filter wishes to hold a bi-directional conversation with another, it must physically exist in two positions within the filter chain, since the chain is inherently uni-directional. The first instance must exist in front of the receiving filter and the second instance after the receiving filter. An example of this is an intelligent search engine, which queries the linkbase for a particular set of links, filtering the results before passing them to the rest of the system.

In the current communication model there is no concept of either a secure channel or a broadcast message. A secure channel is a channel along which one process can send a message to another ensuring it is not changed or deleted. A broadcast message is a mechanism that allows one process to send a message to all the other processes in the system. In the current system a broadcast message would be impossible as there is no mechanism for distributing a message to multiple viewers. A secure channel is impractical as the filter chain model means that a message must pass through every filter. The only way this could be achieved is by ensuring the co-operation of all the process en-route. Without this, any filter is able to change or delete the message as it passes along the chain.

The various components of the current model can be categorised as either message routers or interpreters. The message routers (DCS and FMS) rigidly define the communication topology. The

message interpreters (viewers and filters) send and receive information down a single channel; they have no understanding of the rest of the system. This separation is advantageous because it makes the development of message interpreters easy. However, the performance of the system is heavily dependent on the implementation of the message routers and the topology of the communication system is static.

3. New direct communication model.

The new direct communication model contains no distinction between filters and viewers, as far as any communication is concerned they are identical. Any process can register or un-register with the communication module at any time. Conversations can take place between any of the registered processes. Thus in the new model, for instance, a viewer can talk directly to a filter and vice-versa.

The new direct communication model is implemented as a single module which allows processes to talk to each other via bi-directional channels between the process and the module itself.

The module contains the concepts of topics and process names (e.g. Linkbase, Available Links). A topic is an abstract label used by the communication module to route messages to groups of processes. For example a linkbase will register 'FOLLOW_LINK' as a topic. Any messages sent about this topic will be automatically routed to the linkbase. Process names and topics serve the same purpose as far as the system is concerned, but they differ at a conceptual level. The process names were introduced to allow the prototype to more accurately match the existing model.

In order for a process to register with the module it must provide a registration message. This contains its name, and a list of topics which can be used by the module to distribute messages during conversations. The process will receive a unique identifier when the module has completed its registration. There is no reason why a process can not register more than once in a session. Each registration will be given a new identifier. To the rest of the system each registration of the process will appear as a separate entity.

The process name and topics which are contained within the registration message need not be unique as each registration is given a unique identifier. For example when two bitmap viewing processes register they both use the same process name and topics, but each will receive a unique identifier. This identifier is then used by the rest of the system to identify the process.

A process can communicate with other processes, or conceivably with itself via four different routes, these are :

1) **General broadcast** - one process sends a message to all the other processes.

2) **Name oriented** - one process sends a message to all other processes which are registered under the given process name.

3) **Topic oriented** - one process sends a message to all other processes which are registered under the given topic.

4) **Process to process** - one process sends a message to another process using its identifier.

Every message has data (which can be an existing style message for compatibility), an information message and an identifier for the process that sent it. The information message contains the identifiers and process names of the receiver and the sender of the message. If the communication is topic oriented then the information message will also contain the topic name. The data is a pointer to a block of shared memory which can contain any information which the process wishes.

The communication module can be used to maintain the state of a process. A process can un-register a topic while it is performing the required action. Upon finishing it will register the topic once more, thus allowing its re-use. An example of this is given in section 4.1.

The processes are able to access information (contained within the communication module) about the rest of the system. This allows them to have complete control over the message routing. This has the advantage that topology of the system is completely dynamic, changing from message to message. However, it does require the processes to become more aware of the system leading to an increase in their complexity.

The direct communication module provides additional functionality to allow a process to register to receive messages in the following cases :

- Every message sent about a particular topic.
- Every message sent to a particular process name.
- Every message ever sent.

This functionality facilitates monitoring of the system by one or more of the processes. In addition a process can register to receive messages that are sent in the following cases :

- To processes which are not currently running.
- About a topic which does not exist.

This allows the easy inclusion of processes to handle error cases. For example if the text viewer attempts to retrieve links from a linkbase and none are registered then a third process could start one up.

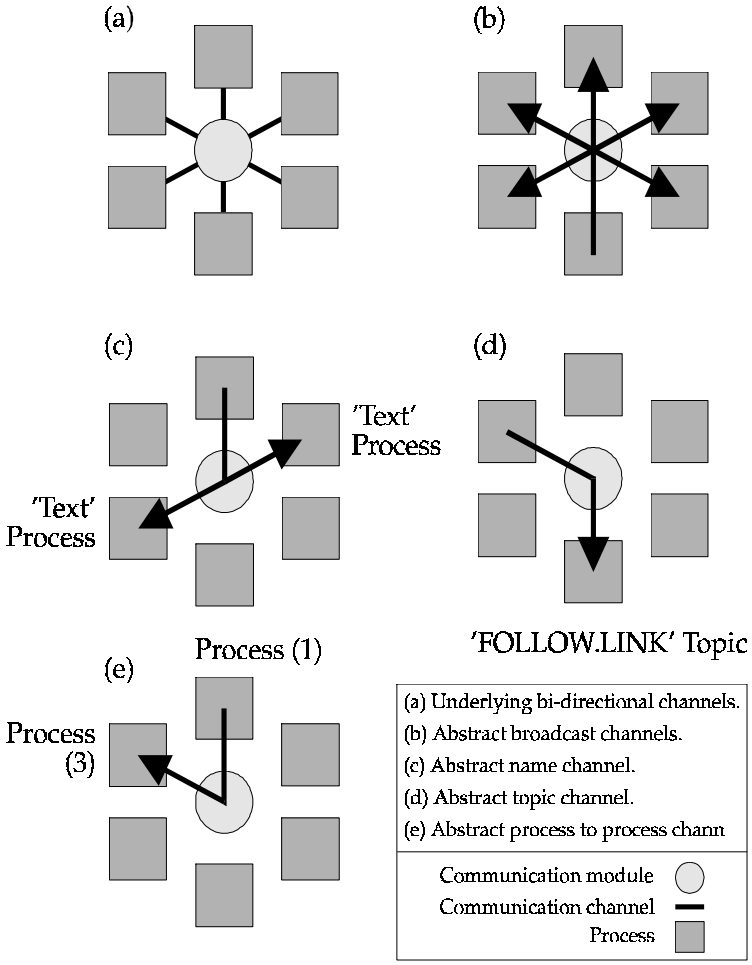


Figure 7 : Real and abstract communication networks.

At the lowest conceptual level the direct communication system is represented by the diagram (a) of figure 6. The radial lines on the diagram represent the actual bi-directional channels between the processes and the communication module. Diagrams (b) to (e) of figure 6 show the different types of communication which are supported by the new communication model.

4. Prototype construction.

A direct communication Microcosm prototype has been constructed by adapting a subset of the current Microcosm system. The prototype consists of an Available Links process, DCS, Linkbase and Text Viewer process. These are the minimum

set of processes needed in order to provide a simple hypertext/hypermedia system within the Microcosm design. The processes provide the following functions in the new model :

- **Available links process** - provides the user with the ability to choose a destination from a set of destinations when following a link.
- **DCS** - provides the ability to open up document viewers if needed and direct access to any document which Microcosm is aware of. Unlike the DCS in the existing system the new DCS contains no communication or state management functionality. It exists at the same level as any of the other processes within the system.
- **Linkbase** - provides the link storage and retrieval mechanism for any links stored in the system.
- **Text viewer process** - provides the user with the ability to read text documents and follow links from within them.

Each process registers with the direct communication module when it starts up. It registers a process name and the topics on which it wishes to receive conversations. For example the text viewer registers with the process name TEXT and the topic 'CAN_REUSE'. Viewers within the Microcosm system can be locked by the user, indicating that they do not want the system to use the viewer to display a new document (they are in effect setting a bookmark at the current document). The text viewer dynamically un-registers the 'CAN_REUSE' topic upon being locked by the user, and re-registers it upon being unlocked. Thus, the DCS is able to locate viewers which are not locked (able to display a new document) as only these will be registered with the 'CAN_REUSE' topic.

4.1. Potential for cleaner process interaction.

In the current system the DCS has to maintain a table of all the viewers currently open and which of those are unlocked. In the new system the onus is on the viewers themselves to remember their current locked state. This state is available to the rest of the system through the dynamic registration of the 'CAN_REUSE' topic. By using this mechanism to reflect the general state of a process, information about the system as a whole is now available to every process.

A further example of cleaner interaction is the manner in which the Available Links process now works. In the current system, the Available Links process has no knowledge of the number of linkbases currently executing. It has to rely on each linkbase sending control messages along the filter chain. The Available Links process maintains a counter which is incremented every time it receives a 'PROCESSING' control message and decremented every time it receives a 'COMPLETED' control

message. In addition the DCS itself sends an extra 'COMPLETED' control message. When the counter reaches -1 it knows that all the linkbases have finished producing links.

This is a very complex protocol which requires special case coding within the DCS. In the new model a much simpler protocol was implemented. Every module which wishes to produce links, registers the same topic (for example, 'LINK_PRODUCER'). The Available Links process is able to ask the communication module about the current number of link producing processes within the system. Each link producer can send a single control message to denote it has finished processing (for example, 'COMPLETED'). The Available Links process can count these and thus ascertain when all of the link producers have finished. This is a much simpler protocol which requires no special case code.

The length of the message pathways are considerably reduced in the new system. For example, a viewer requests the positions of buttons within a document when it is first displayed. The viewer can send the request to all of the linkbases, which have registered the 'FIND_BUTTONS' topic. Each linkbase in turn, responds *directly* to the viewer using the process-to-process communication facility. This method is much more direct than the current filter chain topology, as only the originating process is involved in the reception of the message.

4.2. New user interaction with the system.

There are many different ways of interacting with information within GUIs. The most common metaphor used is that of the desktop. Here, the user is presented with documents which are operated on by applications. Various methods are provided within the windowing environment to allow the user to do this :-

- Executing an application and using commands within it to locate and open a document.
- Executing a command with the document as a parameter.
- Interacting with the file system directly. Users can associate particular files (documents) with certain applications. By activating the file, the application is started and the document loaded.
- Dragging a document from one application and dropping it on another.

The current Microcosm system only supports the first of these interactions. The user must first start Microcosm and then use it as an interface to the documents. This approach is well suited for naive users, as they only need to learn about the system itself. It is, however, restrictive in that it does not allow more proficient users to interact with Microcosm in the same manner as other applications. The inclusion of these interaction styles will allow

Microcosm to become more of a background process than is currently the case. They do not rely on the user interacting directly with Microcosm itself.

The new model no longer requires the central control of a single process as routing of messages happens dynamically. It is therefore possible for the remaining three methods of interaction to be incorporated into the system.

For example, the user can start the text viewer and then select the document to be displayed using the text viewer's menu. Or, associations can be made, so that Microcosm processes can be started by browsing the file system. The act of selecting a file will cause the associated application to start and process it. For example, if the link file is associated with the linkbase process, users can choose to use the links by selecting the relevant file. New links will appear in any of the already open documents.

The new system is more flexible, as it provides different interaction styles catering for the different types of user.

4.3. Synchronising processes

The new communication module allows direct communication from any process to any other process. In an multi-process system, it is conceivable that a process will be unable to receive a message because it is busy. The communication module must be able to accommodate this eventuality, providing a method for re-transmitting the message. This can be achieved by queuing the messages that are to be sent.

Messages are removed from the head of the queue and resent at set time intervals. If this delivery fails then the message is placed on the tail of the queue. This mechanism ensures that messages are not lost, but it does not ensure that they will be received rapidly.

The above approach does not allow for the synchronisation of processes, or the delivery of messages in a time-critical situation. In both cases, the messages must be delivered as quickly as possible. One solution is to allow the message senders to flag the message as important. In this case the communication module does its utmost to ensure that the message is sent as soon as possible. If delivery is unsuccessful, the message will be placed at the head of the queue instead of the tail as is the norm, ensuring the message will be given priority. A suitable time-out period will ensure the system does not deadlock.

If this does not provide the required decrease in delivery time, more drastic steps can be taken. These rely on knowledge of the underlying implementation of event processing within the GUI. The dangers with using such an extreme approach are that, it would lead to a tighter coupling between the communication module and the processes (possibly leading to degradation in

communication performance), and make the system more difficult to port to other GUIs and architectures.

5. Results

The following section details both qualitative and quantitative results gained from comparing the current Microcosm system with the direct communication prototype.

The following table shows the message pathway lengths in both the current system and the prototype. The four scenarios discussed previously are considered (i.e. viewer to filter, filter to viewer, filter to filter and viewer to viewer). Each scenario considers the best and worse cases for both successful and unsuccessful message delivery. There are three different configurations considered:-

1. The original system with two filters and two viewers
2. A generalisation of the original system, with n filters and m viewers.
3. The new direct model, with any number of viewers and filters.

The results, drawn from the prototype, show that in the current system, the lengths of the message pathways are independent of the number of viewers, but in most cases proportional to the number of filters. From this we can conclude that the main overhead in communication within the current system is due to the filter chain. In the new model, however, all messages are delivered directly to the process, and so all message pathways are the same length. This provides a delivery time independent of the number of processes.

If there is no process to receive a message, then nothing is actually sent in the prototype; the communication module simply returns an error to the sending process. This is radically different to the current system, where the message pathway is, in most cases, longer.

Model	Case		V to F	F to V	F to F	V to V
1. Current, small	Best	Received	3	3	2	8
		Not Received	8	2	4	7
	Worse	Received	5	5	2	8
		Not Received	8	4	4	7
2. Current, (n+m)	Best	Received	3	3	2	2n+4
		Not Received	2n+4	2	2n	2n+3
	Worse	Received	2n+1	2n+1	2n-2	2n+4
		Not received	2n+4	2n	2n	2n+3
3. New, any size	Best	Received	1	1	1	1
		Not Received	0	0	0	0
	Worse	Received	1	1	1	1
		Not Received	0	0	0	0

Table 1: Table showing number of messages in system

6. The Need for a "Process Manager"

In the current system, the DCS and FMS contain code to both control processes and handle communication between them. However, communication within the prototype is now handled by the communication module. The control processes have been removed from the new system, and as a result the user has to explicitly start any processes they require. This approach is more flexible, but requires the user to be more aware of the functionality of the different processes available within the system. If the process control features of the current system are required, then an additional process which mimics those features within the DCS and Filter manger is required: a Process Manager.

This Process Manager will have to be started transparently, whenever the user starts a Microcosm process. By taking advantage of message trapping built into the communication module, it can detect when messages are sent to non-existent processes and so start them up. For example, the user may wish to compute links without the relevant process running. The Process Manager would detect a message being sent to the 'COMPUTE_LINKS' topic, determine the name of the process, start it, and forward the message. The user would see the compute links process start up and the results of their text query appear.

The Process Manager would exist at the same level as the other processes within the system. It would use the facilities provided by the communication module to detect when to start up new processes etc.

7. Summary

In the new model there is no concept of explicit process ordering. In contrast, the current model provides ordering as a side effect of the uni-directional chain mechanism. Whether such an inherent ordering is needed is still open to discussion. In addition, the centralised user interface that allows a user to shut down or start-up processes has been removed. Once again, it is not immediately obvious whether such an interface is essential to the operation of the system.

The destination of messages within the current system is dictated by a pre-defined field within the message. All messages are routed by the DCS and FMS and so follow the same path. Each process has to examine the contents of the message to decide if it will interpret the rest of it or not. With the new system, the destination is decided by abstract topic identifiers.

Communication within the current system is based around the client-server model. The DCS and the FMS are clients for the viewers and filters respectively. The client server model prevented the viewers and filters communicating directly. The model employed within the prototype is a peer-to-peer communication network. Each process is at the same level and can communicate with any other. The routing of the messages is controlled by the processes themselves rather than by a single, controlling process (the FMS or DCS).

The current model contains two areas which restrict its scalabilty, namely the DCS to FMS bottle neck and the filter chain. Every message sent from any viewer to any filter has to pass along the DCS to FMS channel. The system is entirely dependent on the robustness of these processes. In contrast the new model does not contain either of these restrictions. As such it is capable of being scaled up in size without significant performance degradation. If one process stops working for any reason, then it will stop receiving messages. No other processes will be affected, however. Processes also have access to more information about their environment, and can explicitly direct their conversations. This effectively moves the control of the system from the DCS and the FMS into the processes themselves.

8. Conclusions

It is not possible, within the new design, to determine whether or not there is a receiving process for a message. The new communication module allows a process to register the topics it will listen to (receive), but not ones it will talk about (send). The system can only detect a message being sent to a non-existent process at the time of sending. The Process Manager (see section 6) can only warn the user, or start up the missing process, when the error is detected. If processes register the topics that they wish

to send then the system can detect the error as soon as they start. The Process Manager can then start the missing process or warn the user at the earliest opportunity.

The use of abstract topic identifiers de-couples the content of the message from the destination and provides a further level of abstraction. For example, all processes that understand the concept of link following register for the topic, 'FOLLOW_LINK'. The routing of messages has become dynamic, as the message pathways are computed whenever a message is sent.

Because of the new dynamic topology, new protocols need to be designed to allow process interaction. For example, the protocol used within the current system when a link is followed has already been described in section 4.1, along with a possible new protocol. Due to the dynamic nature of the new model, other protocols must be designed to provide new services within the system. For example, an abstract channel can be defined through which information about link services can be exchanged. Any process interested in such information need only to register for that topic (say 'LINKS'). A linkbase might notify other process that it is starting or shutting down. A text viewer registered for this topic, is able to monitor this abstract channel and adjust accordingly. It might remove from the display all buttons from a particular linkbase when that linkbase closes down.

The introduction of a radically different communication model to the Microcosm system avoids the current message bottle-necks leading to greater system speed, improved system security and less communication redundancy. In addition the new model enables a far greater degree of flexibility for user interaction. However, the new direct communication model does set new challenges which must be met.

Bibliography

[Davis92] Hugh C. Davis, Wendy Hall, Ian Heath, Gary Hill and Rob Wilkins, *Towards an Integrated Information Environment with Open Hypermedia Systems*, In: D. Lucarella, J. Nanard, M. Nanard and P. Paolini (eds), *ECHT '92. Proceedings of the Fourth ACM Conference on Hypertext.* ACM Press, 1992, 181 - 190.

[Davis93] Hugh C. Davis, Wendy Hall, Adrian Pickering and Rob Wilkins, *Microcosm: An Open Hypermedia System*, In: S. Ashlund, K. Mullet, A. Henderson, E. Hollnagel & T. White (eds), Human Factors in Computing Systems. INTERCHI '93. Proceedings of Human Factors in Computing Systems, Amsterdam, ACM Press, April 1993, 526.

[Fountain90] Andrew M. Fountain, Wendy Hall, Ian Heath and Hugh C. Davis, *MICROCOSM: An Open Model for Hypermedia With Dynamic Linking*, In: A. Rizk, N. Streitz and J. Andre (eds),

Hypertext: Concepts, Systems and Applications. Proceedings of the European Conference on Hypertext, INRIA, France, November 1990, Cambridge University Press, 1990, 290 - 311.

[Hall92] Wendy Hall, Ian Heath, Gary Hill, Hugh Davis & Rob Wilkins. The Design and Implementation of an Open Hypermedia System, CSTR 92-19, Computer Science Technical Report, University of Southampton, UK, 1992.

[Heath92] Ian Heath, *An Open Hypermedia System: Abstracting Links From Documents*. PhD thesis, Dept. Electronics and Computer Science, University of Southampton, 1992.

[Hill93] Gary Hill, Rob Wilkins and Wendy Hall, *Open and Reconfigurable Hypermedia Systems: A Filter Based Model*, Hypermedia, Vol 5 Num 2, 1993.