7

Inter-Process Communication

7.1 Introduction

Inter Process Communication of PARAS Microkernel deals with asynchronous and synchronous modes of communication, port management, packet management, system thread and receiver thread. It gives the details of how messages are handled within and outside the node communication. It gives the functionality of system thread and receiver thread.

7.2 Packet Management

Packet management deals with the buffer space in the kernel. An array of maximum number of packets are initialized as linked list and each packet points to its buffer space, which is maximum packet size. Get and put macros will take packets from and release packets to linked list. Receiver thread copies the message to free packet and queues the packet to the port, if receive is not posted on that port.

7.3 Port Management

Port management deals with creation, deletion, name, unname, locate and poll calls for ports. For name services micro-kernel contacts NameServer through remote-procedure-call(RPC) mechanism.

7.4 Communication Primitives

Send primitives, async, sync, csend, msenddma and Receive primitives, block, mwaitdma are supported.

Async_send initiates the send and returns to user mode immediately. It does not wait for the message to be delivered at the destination. Async send is complete after the message is copied to the user buffer or queued to the port or the message leaves the node.

Sync_send sends the message and waits for the ack to be received from the destination. The ack will be sent after the message has been processed by the receive call at the destination.

Csend is used to send a word of control message. Csend accommodates this word in the header without a message body.

MsendDma is used to send large messages after getting the physical address of the buffer from the destination. It sends the message by splitting the message in to packets. The physical address of the buffer where this packet has to be copied mentioned in the dma header. Receiver thread copies the dma message into the physical address given in the dma header. The destination buffer address is assumed to be continuous.

MwaitDma is used to receive long messages sent by using MsendDma. This call is blocked till complete message has been delivered and this thread is waken up by receive thread.

Block_receive checks whether any message is pending on the port, if the message is present, it will be copied in to the user buffer; otherwise it waits on the port for the message to arrive. After the message is received, it checks for the type of the message and if it is a synchronous message, it sends ack to the sender.

Send calls check whether the destination process is on this node or outside the node. If the message is for the same node, the message will be copied to user buffer, if the request is already posted for the message; other wise the message is copied to a packet and it is queued to the port. If the message is for the some other node, it will be transferred to the destination by programming the CCP.

7.5 System Thread

System thread is started as kernel thread at kernel initialization time. Its functionality include service the remote function return value and to start multi-cast server thread for multi-cast group initialization.

7.6 Receiver Thread

Receiver thread is started as kernel thread at kernel initialization time. It receives messages from the CCP FIFO. The message arrival is indicated by CCP with hardware interrupt. The interrupt handler wakes up the receiver thread. The receiver thread processes all the messages present in the FIFO. Till the receiver thread is active CCP interrupts are disabled. For each message receiver thread checks whether any request is pending for the message. If the request is already posted, the message is copied to the user buffer and the thread waiting for this message is woven up; other wise the message is copied to a packet and is queued to the port. In case of the thread waiting for long message using MwaitDma, the thread is waken up by receiver thread after complete message is received. In case of Dma messages, the message is directly copied into user buffer as the destination address is part of the dma header.

7.7 CCP Programming

CCP contain two buffers, transmit(Tx) and receive(Rx) buffers. Each buffer is of 64 Kbytes size and each buffer is configured as 32 packets and each packet is of 2 Kbytes size. Filling of these buffers is done in circular way.

The packet format is as follows,

a. CCP header(4 bytes).
b. Length of the message(4 bytes).
c. System header and message.

7.8 Multi Cast Server

Multi cast server thread is started by system thread whenever request for multicast server is made by multicast group root. This thread has to handle multi cast messages delivery and to delete the server thread, when deletion of multi cast group request comes.

7.8 Implementation Details

/* port_id format and other format assumptions */

/* What issues to be discussed */

7.9 Machine Independent Issues

7.10 Port Group Management