Layered Modeling of Hardware and Software, with Application to a LAN Extension Router

Peter Maly and C. M. Woodside
Department of Systems and Computer Engineering
Carleton University, Ottawa, Canada, K1S 5B6
{maly, cmw}@sce.carleton.ca

Abstract - *Understanding the interactions between hardware and software is important to performance in many systems found in data communications like routers. Responsibilities that traditionally were programmed in software are being transferred to intelligent devices, and special purpose hardware. With more functionality being transferred to these devices, it becomes increasingly important to capture them in performance models. Modeling hardware/software systems requires an extended queueing model like LQN. This paper describes a layered architecture model which represents hardware and software uniformly and which emphasizes resources and performance, called a Resource-based Model Architecture (RMA). The approach is demonstrated on a remote access or LAN extension router. The model is created by a systematic tracing of scenarios and is used to explore the router capacity for different workloads, and to analyze a re-design for scaleup.*

1.0 Introduction

Interaction between hardware and software resources are important to the performance of many systems, for example in data communications, where logical operations are partitioned between software tasks and intelligent interface devices or processors. One difficulty in modeling these systems is they are not well represented by ordinary queueing models, and require an extended queueing model. Layered Queueing [17] provides a systematic framework which simplifies the model construction and solution.

This paper describes a “Resource-based Modeling Architecture” (RMA) for modeling this class of system. It demonstrates the approach with a study of a modest sized LAN Extension Router (LAN/ER), in which the model is used to estimate performance and evaluate design trade-offs.

The model is created in two stages. First the layered Resource-based Model Architecture (RMA) is created by inspecting the hardware and software and tracing the involvement of components in scenarios. Then the parameters of the architecture are determined by various means, including profiling. The architecture itself gives guidance as to what to collect and how to combine the parameter information. The model is completed by adding environmental information.

The RMA model is an abstraction that shows how the software and hardware together create delays and bottlenecks under different workloads. The model is used to predict performance characteristics like capacity and delay, using either analytic or simulation techniques [17]; in this case simulation was used.

The first purpose of the model is evaluation of the capability of the given design. The second purpose is to expose performance bugs, by establishing expectations for performance tests. The third purpose is to provide insight to guide the evolution of the design. The special value of the layered

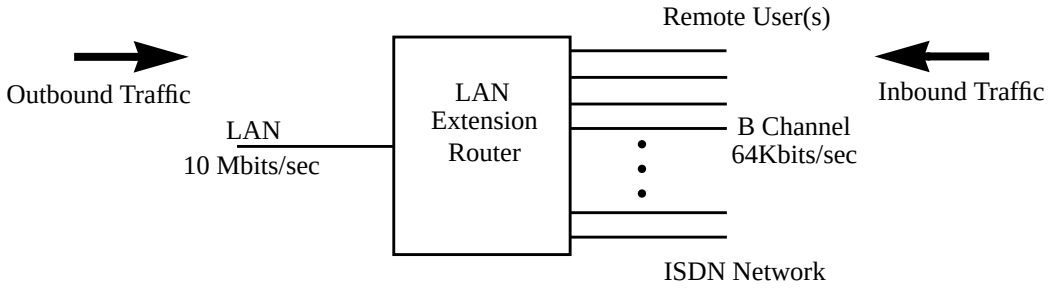
structure of RMA is that it conceptually organizes a complicated flat assembly of queues and functions, according to how they compete for resources. This gives insight into the effects that software and hardware have on each other.

2.0 Application of RMA to a LAN Extension Router

Routers make extreme demands on the performance of software and hardware, to execute communication protocols and direct the traffic to the correct destination network. To increase the performance of routers, research has been done in improving various aspects of the software and hardware. Examples of software research include router table look ups [6], [11], [15] to significantly speed up routing decisions, heuristic techniques for implementation of protocols [16], general ways of handling packets in routers efficiently [4], and comparisons of queueing strategies [10]. Hardware research has proposed different architectures to deal with the increasing demands [2], [7], [9], [10], [12], [13]. Each of these studies is focused on using a single aspect of router design to enhance performance. However the overall performance comes from the interaction of many factors, and RMA provides an approach to study the overall combination.

The router studied here and shown in Figure 1 is a small device, not necessarily at the cutting edge of technology, but it nicely illustrates the use of the modeling approach. It extends the LAN on the left so that the remote users on the right (which are connected to the router by ISDN links) operate as if they were connected to the LAN directly. The users have a Basic Rate Interface ISDN service, which can consist of one or two B channels. The LAN/ER is capable of handling up to 120 B channel connections at one time.

FIGURE 1. High level diagram of the LAN/ER's major input and output ports

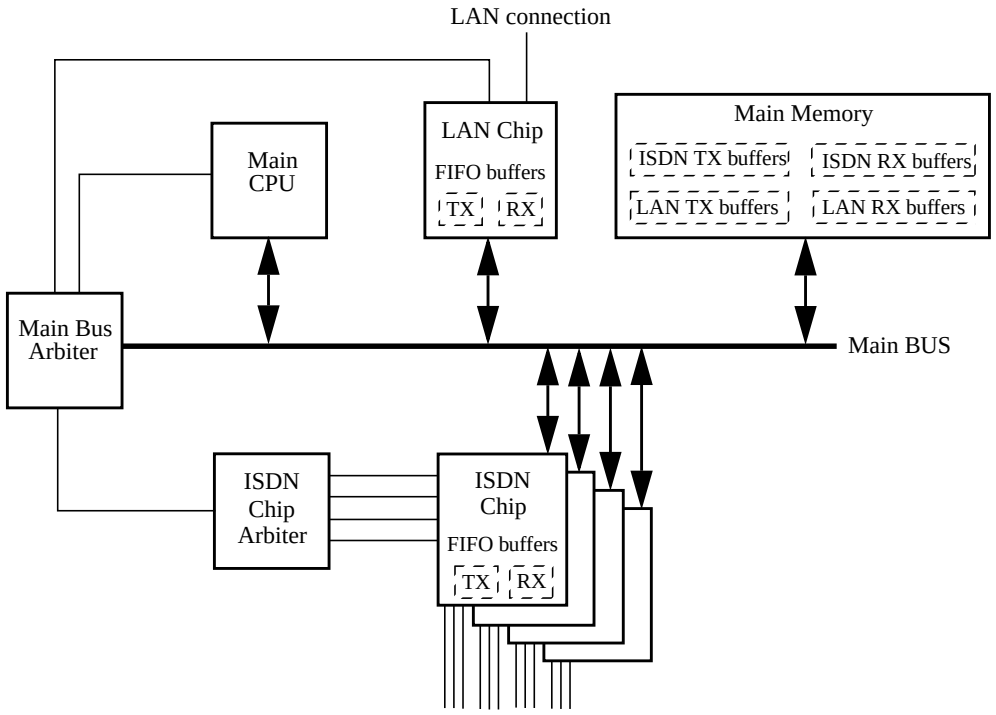


The LAN/ER is also connected to an 10 Mbits/sec LAN network which uses a standard Ethernet protocol. Frames being received on the ISDN network side can be bridged to the LAN, or be routed back to another B channel on the ISDN network side. Conversely packets arriving on the LAN will be routed to the appropriate B channel. Each B channel uses a synchronous Point to Point Protocol (PPP) at the network layer, and a HDLC protocol at the data link layer.

2.1 A more detailed description of the LAN/ER

The LAN/ER has a single bus, single processor architecture as shown in Figure 2.

FIGURE 2. Hardware interconnection in the LAN/ER system



The two network interface devices in the system are the ISDN Line Control chips and the LAN chip. The ISDN chip provides a full duplex ISDN interface, with the ability of performing some of the processing for the HDLC protocol. There are four of these chips in the LAN/ER system, each capable of supporting 32 B channels simultaneously. Frames which are received and frames to be transmitted are placed in the ISDN buffers in main memory. When the ISDN chip is not busy it polls the transmit buffers in main memory every 125 μs to check if any new frames to transmit have appeared. The FIFO buffers within the ISDN chip hold 8 bytes for each direction (receive and transmit) for each B channel. To empty its receive FIFO buffer, or fill its FIFO transmit buffer, the chip makes a request for the bus and executes a DMA operation on the main memory.

The other network interface is the LAN chip which supports the 10 Mbits/sec IEEE 802.3 Ethernet protocol. The LAN chip operations are similar to the ISDN chip, except that it does not poll when it is idle. Instead the LAN chip receives commands from a queue in main memory which tells the chip what operations to perform. The LAN chip has an on board FIFO buffer of 32 bytes for each of the receive and transmit directions.

The CPU executes a Worker task to process frames found in the ISDN buffers and transfer them into the LAN buffers, and vice versa. It runs a real-time kernel that manages the Worker task and ten other tasks which do the management and maintenance of the system. Their CPU utilization is much smaller than that of the Worker task.

Main memory is a single centralized memory unit in the LAN/ER, holding the code for the CPU and all the data and data buffers that it manipulates. The same data buffers are also accessed by DMA

operations performed by the interfaces (the ISDN chips and the LAN chip). Access to the main memory is controlled by the bus arbitrator.

The bus arbitration is a pre-emptive “hold till done” priority scheme. The priority order, from low to high is CPU, LAN chip, ISDN chip. The DMA devices can pre-empt the CPU in use of the bus, but once any DMA device has the bus, it is not pre-emptable. A special ISDN chip arbiter is used which blocks ISDN requests for the bus after 4 requests from 4 different ISDN chips. This gives the LAN chip higher priority and prevents the ISDN chips from dominating the main bus.

The freehand lines running through Figure 3 trace four important scenarios. Each scenario starts from a filled circle, representing a triggering event, and shows the order in which components are involved in completing it. This is based on the Use Case Map notation [3].

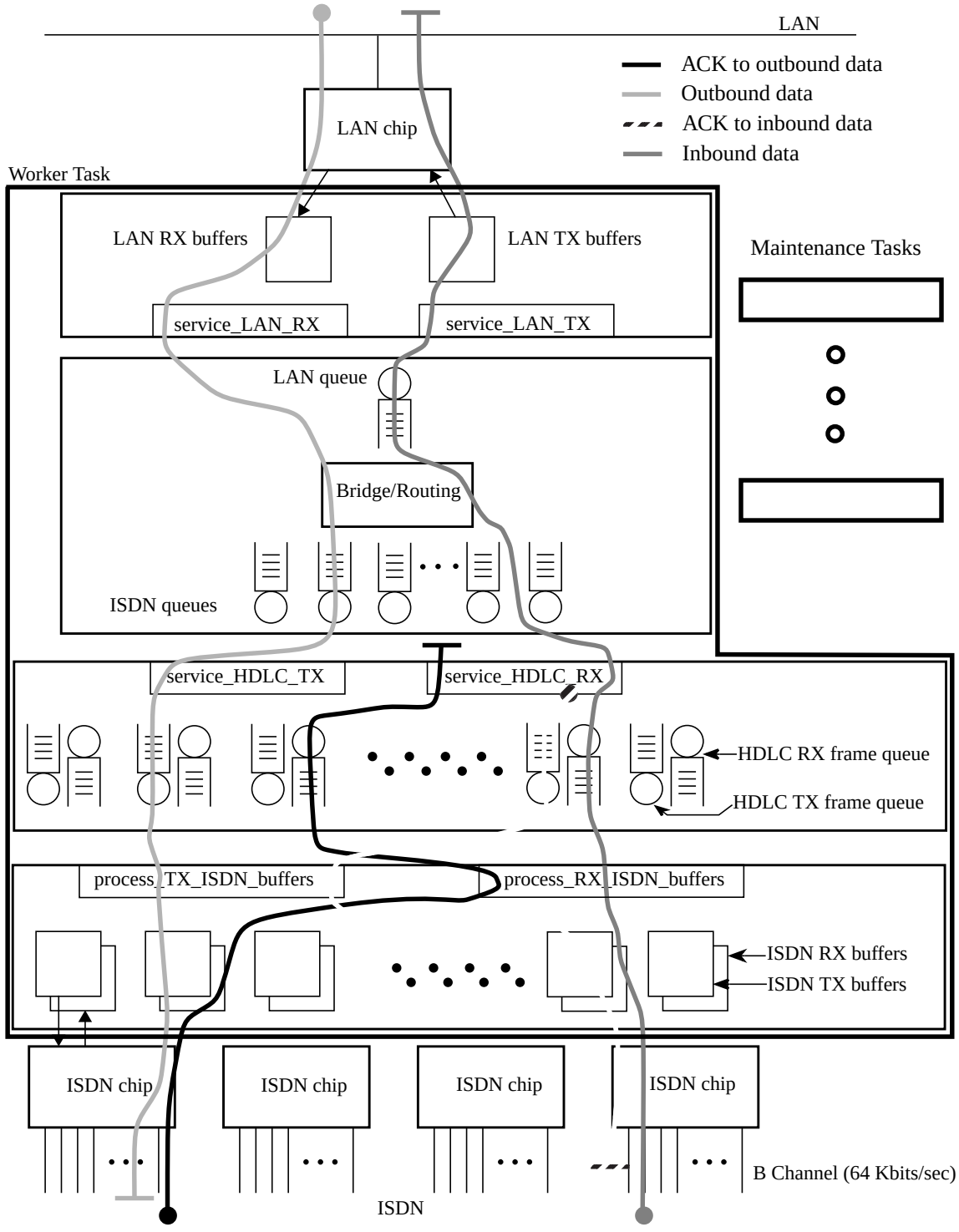
2.1.1 Outbound data scenario (LAN to ISDN)

The handling of traffic going out from the LAN to a remote user begins when a packet is received by the LAN chip. Incoming bytes are stored in the on-chip FIFO buffer and then transferred to the LAN RX buffers. Eventually the Worker task will check the LAN RX buffers for complete received packets, and process them. First it determines which B channel the packet is destined for, by calling the bridge/routing functions. Then the packet is queued to its appropriate B channel queue (ISDN queue). When the B channel is able to accept data, the HDLC portion of the PPP stack (service_HDLC_TX) processes the packet to a frame. The frame is converted by the process_ISDN_TX_buffers operation into a format that the ISDN chip can understand and is stored in the ISDN TX buffers. The ISDN chip polls these buffers, and when it finds a new frame ready to transmit, it DMAs the frame 8 bytes at a time into its appropriate B channel buffer. Depending on the sliding flow control window of HDLC on the link, an acknowledgment frame will be received eventually. This frame is processed the same way as an inbound frame to the point when it reaches the HDLC_service_RX function, where the acknowledgment is processed and the frame is discarded.

2.1.2 Inbound data scenario (ISDN to LAN)

The ISDN to LAN scenario starts when a frame arrives from a remote user, on a B channel. The incoming bytes are stored by the ISDN chip into a FIFO buffer for that B channel. As the FIFO buffer fills up, the chip will transfer its contents into one of the ISDN RX buffers in the main memory. When the frame is complete, the Worker task will eventually assemble the frame for that B channel through the process_ISDN_RX_buffers function. The frame is then sent to the service_HDLC_RX function, which processes the packet, and updates any state information. Depending on the sliding window of HDLC on the link, an acknowledgment may be sent to the remote user for the successful reception of frame(s). The frame is then sent to the upper layer of the PPP stack for bridging/routing, and is deposited into the LAN transmit queue. The frame is taken from the LAN queue and moved into the LAN chip transmit buffers, and the LAN chip is instructed by the Worker task to send the packet off to the LAN.

FIGURE 3. Use Case Map of four scenarios



2.1.3 Worker task operation

The Worker task polls the various queues and buffers shown in Figure 3, using a polling sequence described by pseudo-code in Figure 4. Each service function is repeated until the given buffer is empty.

FIGURE 4. High level pseudo code describing the Worker task polling order

```
while (TRUE)
{
   for(x=1; x <= number of active channels; x++)
   {
      hdlc_service_RX(x);
      hdlc_service_TX(x);

      process_ISDN_RX_buffers(x);
      process_ISDN_TX_buffers(x);
   }

   service_LAN_TX();
   service_LAN_RX();

   sleep until awakened by Null task or timer tick
}
```

3.0 Derivation of the model

Once the scenarios have been defined, as they have been in section 2.0, the layered model can be developed in four steps:

1. Obtain sub-models by tracing out scenarios and grouping responsibilities to resources.

2. Merge sub-models to get a model of the system identifying all the entities but without full operational detail.

3. Add any missing operational detail.

4. Optionally perform any simplifications on the model.

3.1 Step 1: Obtaining sub-models from scenarios

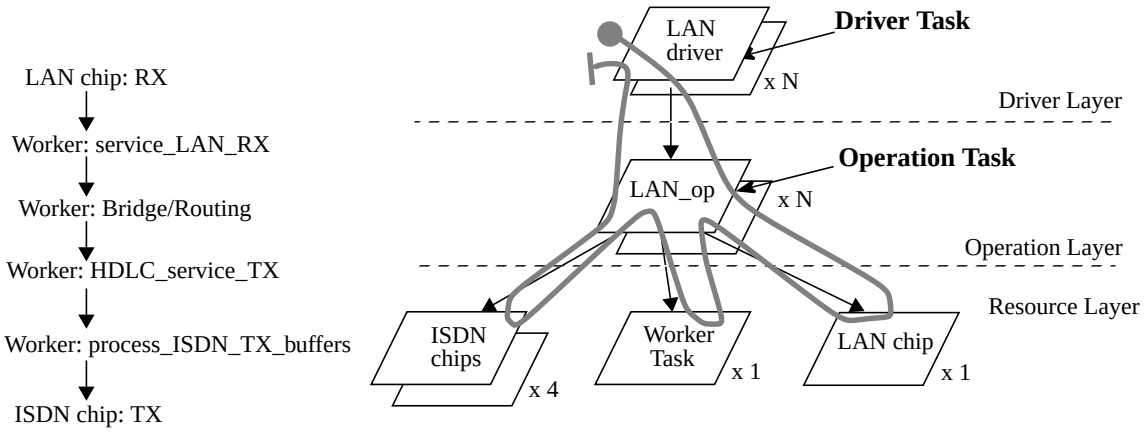
The transformation starts by identifying all the operations that have to be performed to complete a scenario. In the model each operation becomes an “entry” of a “task” that represents the entity (software or hardware) which executes the operation. Each entity is a resource with some level of resource provisioning. The ISDN chips for example have are provisioned at a level of four, for the four chips. The Worker task is single threaded and thus is provisioned at a level of one. In the LAN to ISDN scenario from Figure 3, three resources are identified that play a role in the completion of the scenario.

These resources are the LAN chip, CPU, and ISDN chip. The entry is just an identifier for the operation, as executed by that task.

To model the collection of responsibilities in the scenario, an artificial “operation task” one layer above the resource entities is introduced. This is a pseudo task related to one execution of the scenario, which calls all the entries that belong to the scenario. To set the rate at which the scenario is triggered, a driver task is introduced in a layer above the operation task.

The transformation gives the model shown in Figure 5. Note each call is synchronous which means the entry that is making the call waits for it to be completed. Also notice that Figure 5 does not show the entries in the diagram, to save space. The synchronous calls are denoted by solid arrows in Figure 5. If the driver task has a multiplicity of N as shown it represents N separate sources of input events, such that up to N copies of the scenario can be active at once.

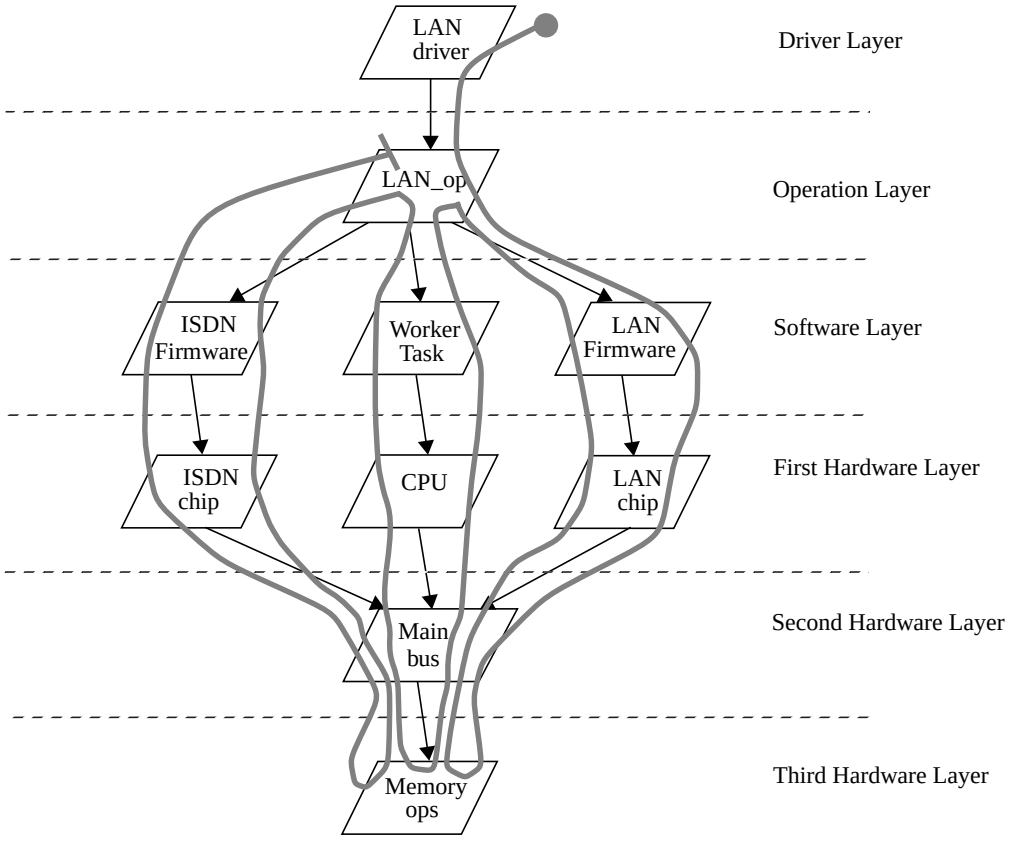
FIGURE 5. Step 1 (first part): Grouping of responsibilities to resources for LAN to ISDN scenario



The second part of this step is to expand the model by including lower level resources that may introduce contention into the system. In an embedded system, three types of contention can be found. One is software competing for software resources, the second is software competing for hardware resources, and the third is hardware competing for other hardware resources. In order to encapsulate contention for software resources, it is necessary to add one or more layer(s) into the model. Any hardware that acts like a master on a bus will in most cases have a set of operations to complete, and conceptually is no different from software. Therefore any firmware in hardware components can be represented in the software layer(s) also.

The hardware/software contention patterns in the LAN/ER can be obtained from the descriptions of the software and hardware in section 2.0 to expand the model to what is shown in Figure 6. (Again as in Figure 5 the entries are not shown.) The first contention point is the bus, as shown previously in Figure 2, and represented in Figure 6 by having the three device tasks call a main bus entity, which calls an entity that represents the memory. The Worker task running on the CPU sees no contention for the CPU (in this scenario), which is also the case for the firmware running on the ISDN and LAN chips. In fact the Worker task does contend for the CPU with ten other maintenance tasks, which are omitted here but are shown later, in Figure 8.

FIGURE 6. Step 1 (second part): Expanded submodel which includes all resources for LAN to ISDN scenario

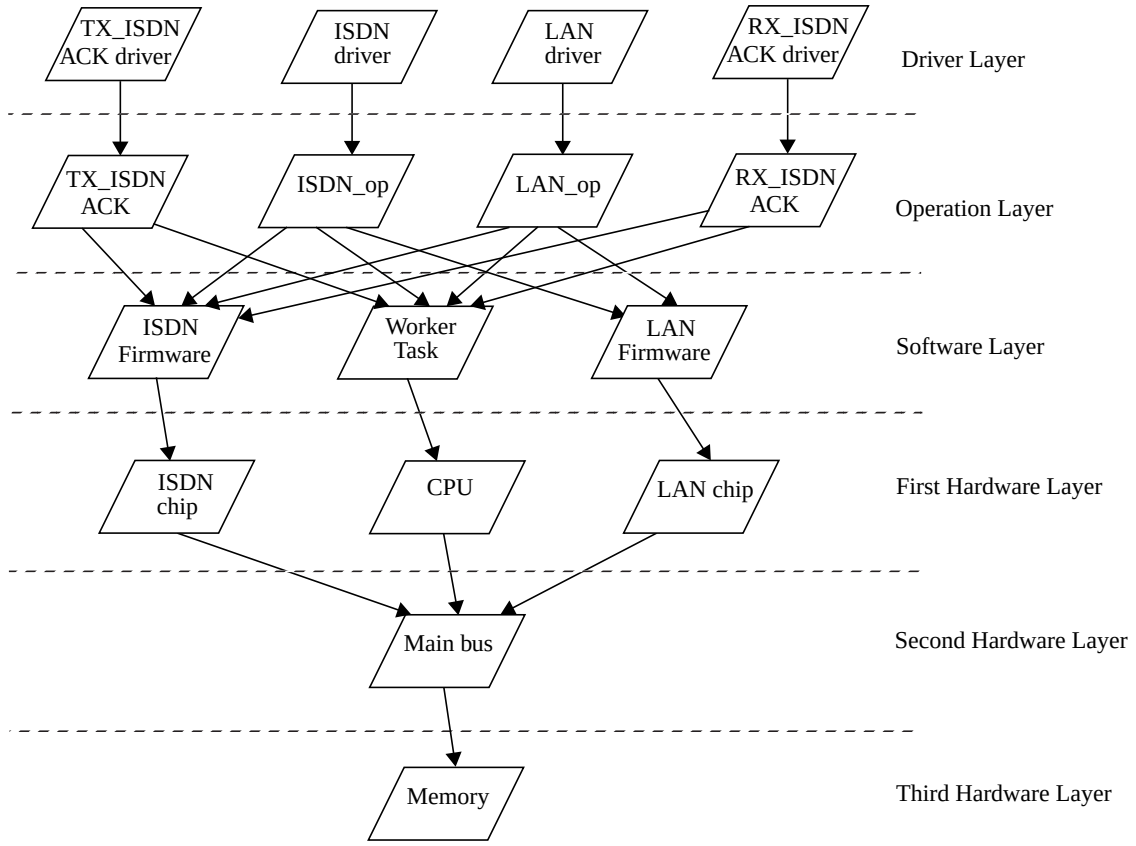


Step 1 has to be applied to all the scenarios in the system before proceeding to step 2. The application of step 1 on the other three scenarios is not shown.

3.2 Step 2: Merge sub-models to create a single model of the system

Merging of the sub-models is done by identifying the same task (by name) in different sub-models. Entries defined in different sub-models for a given task are grouped together in the task. The merged model of the LAN/ER is shown in Figure 7. To save space the entries in the merged model found in Figure 7 are not shown (entries representing software operations will be shown in Figure 8). Since entries are not shown in Figure 7 one must keep in mind that entries within tasks call other entries. Therefore Figure 7 can mislead one into thinking that tasks are calling other tasks, when it is the entries within the tasks making and receiving requests which represent certain services (responsibilities) offered by the entity. These entries have a one to one correspondence with the responsibilities defined in section 2.0. This is how the merged model retains all the scenario operations and call structure encapsulated in the other submodels.

FIGURE 7. Layered model covering all four scenarios



3.3 Step 3: Adding behavioral detail to model

The transformation step has captured the behavior defined by the Use Case Maps found in Figure 3. This step will now add the behavioral detail which was ignored in the Use Case Maps.

Maintenance tasks

In section 2.0 there was mention of other tasks (maintenance) which run on the system when the Worker task is idle. The model represents them by a single aggregate Maintenance task which is triggered once a poll cycle by an entry Sleep in the Worker task.

Main bus arbitration

The arbitration mechanism for the main bus as described in section 2.1 has priorities with some pre-emption. This is approximated by a pre-emptive priority scheduling which is supported by the layered modeling tools. In the model when the ISDN, CPU and LAN “tasks” make requests to the memory/bus “task”, their requests are handled by priority. The ISDN and LAN bus “tasks” are given equal priority because they cannot pre-empt each other, but the CPU bus task has lower priority because it can be pre-empted at any time. The priority is indicated by a number at the lower left corner of ISDN, LAN, and CPU tasks in the first hardware layer.

Adding detail to the ISDN chip

There is one important feature of the ISDN chip that has to be included in the model which is not represented in Figure 7. This property will be represented in the software layer, because it is a description of the ISDN chip's firmware.

The ISDN chip polls all the ISDN TX buffers periodically, for the channels which are not currently transmitting a frame. A poll of all the idle channels is initiated every 125 μs, a delay which will be the think time of each ISDN chip task in the model. The probability that a channel buffer is idle (and thus is polled) is P, given by

$$P = 1 - \frac{1}{f} V\lambda$$

f Maximum transmission speed of a B channel, being 8000 bytes/sec.

V Average size of the frame to be transmitted.

λ Frames per second to be transmitted on a B channel.

To represent the polling overhead for empty polls, an "ISDN Poll" task is introduced which activates the empty polls. One ISDN poll task on average requests $N*P$ polls if N ISDN channels served by the ISDN chip are connected.

Polling Discipline of the Worker Task

The data waiting to be processed by the worker task is contained in several queues, and the task itself polls these queues, emptying each queue before going on to the next. When it completes a cycle it waits for a timer to trigger the next cycle. In the model all the queues are attached to the worker task, and the execution of each operation is represented by an entry whose workload includes the polling overhead and the operation itself. On the other hand the work defined above, to poll the empty queues which are not connected to processing a request, is represented as part of the workload of the "Sleep" entry, which is triggered once per poll cycle.

Concurrency in scenarios and scaling of the model

It is important to identify concurrency in scenarios as this can have a significant effect on the results the model gives. Up to Figure 7 the model exclusively uses synchronous calls which is not entirely correct. When a frame or packet is completely processed, it is left in memory for the DMA device to handle. The operation as far as the CPU is concerned is complete, and the CPU can continue other processing, while the DMA device concurrently will deal with the frame or packet. Therefore any driver task that calls a DMA device will call it asynchronously. An asynchronous call means a task that makes the call does not wait for a reply to continue doing other work. These asynchronous calls are denoted by the open arrowhead in Figure 8. Also since the LAN/ER system is an open system, requests from the driver tasks to the operation tasks are also asynchronous.

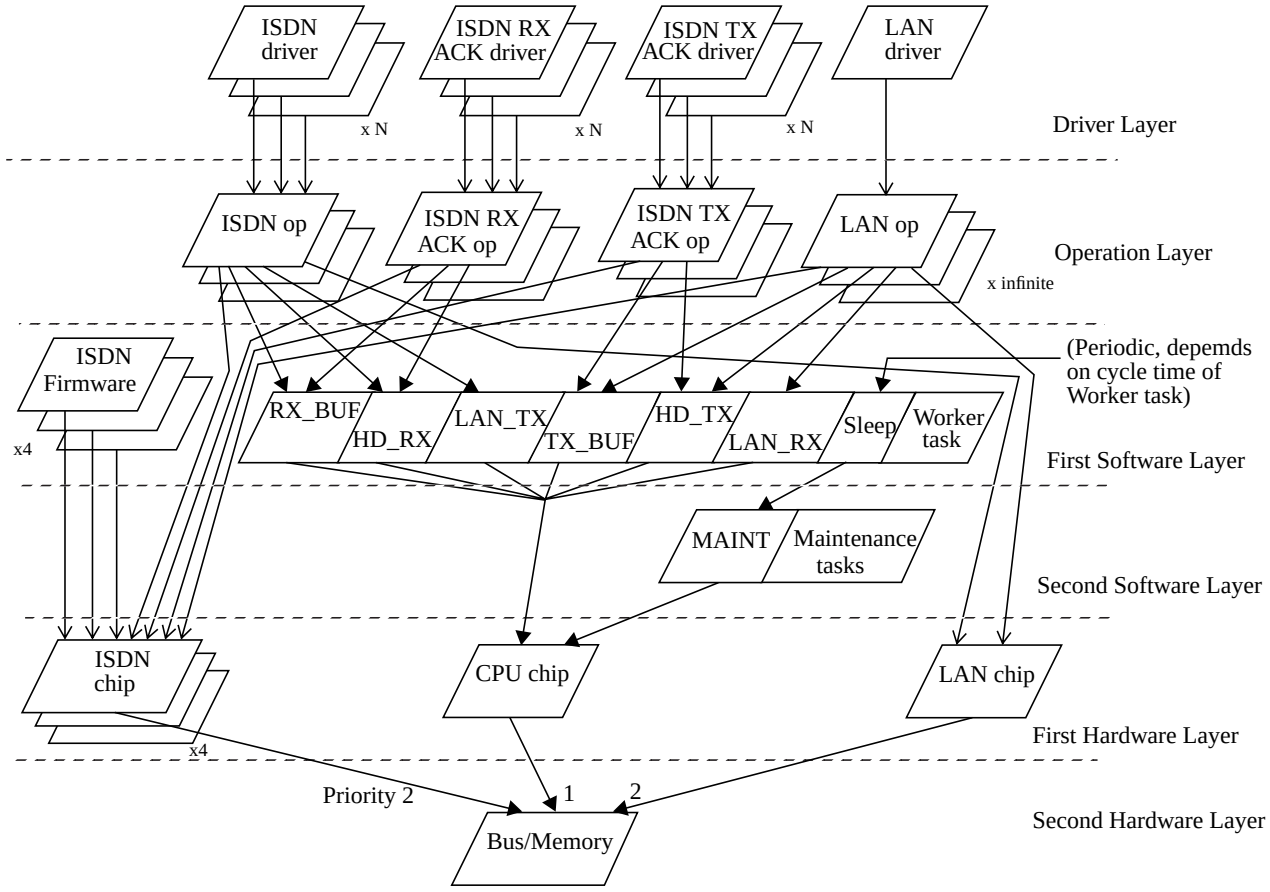
The frequency with which the driver tasks make requests determines the packet or frame arrival rate. For the ISDN network side, each active channel in the model is represented by a copy of the ISDN driver task. Each time the ISDN driver task sends a frame, it is handled by an ISDN operation task through its processing. Thus there is a busy ISDN operation task for each frame being processed, at a given moment. Since the model assumes the LAN/ER has infinite memory, there can be an infi-

nite number of packets or frames going through the system, hence the model allows infinite number of copies of the operation tasks.

3.4 Step 4: Simplifying the model

A final optional step is to perform any simplifications on the model, which will not effect the results. A motivation for doing this is to reduce solution/simulation time of the model, by reducing the number of separate entities in the model. The final model is given in Figure 8. The Main Bus and Memory entities have been merged since the bus does not contend for memory. Similarly the LAN firmware does not contend for the LAN chip device and has been merged with the device entity. The ISDN chip logic has been kept separate because it has the polling logic.

FIGURE 8. Complete model with Asynchronous Calls, Software Detail and Hardware Simplification (Merged Layers)

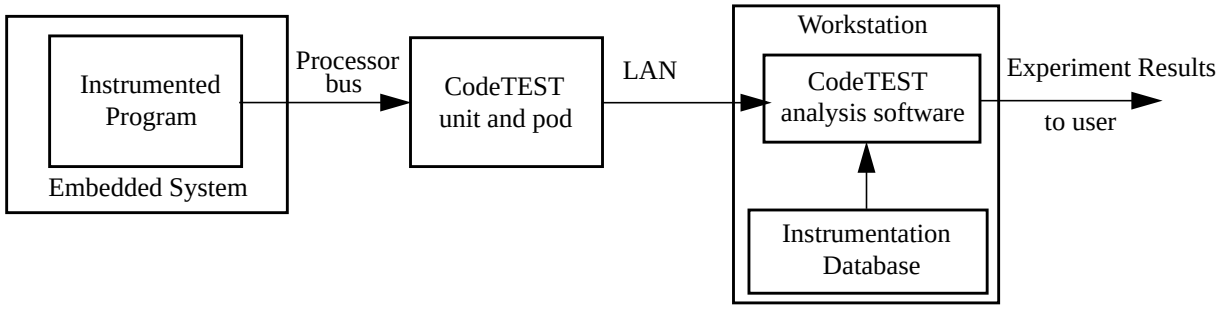


4.0 Model parameters and validation

4.1 Parameter gathering

Execution times of code were measured with a profiling tool called CodeTEST [1], which has a pod that piggybacks the pins on the processor chip. The software is instrumented by calls inserted in the code with probe IDs, which are also put into a database along with the location of the probe (that is, the source file and line number). When the instrumented code runs, the probes write the IDs to a reserved memory location. The CodeTEST unit logs these writes as pairs of values of (ID, time), and sends the information periodically to the CodeTEST analysis software. The analysis software, with the help of the instrumentation database, outputs results a human user can understand. The entire operation is illustrated in Figure 9.

FIGURE 9. Data gathering when code is running



Using the CodeTEST tool, software service demands for the Worker task entries HD_RX, HD_TX, TX_BUF, RX_BUF, LAN_TX, LAN_RX, and MAINT were obtained for packet and frame sizes of 64, 128, 256, 500, 1000, 1500 bytes. Polling overhead (cost of an empty poll) for the entries above were also measured using the CodeTEST tool and put into the model. These measurements were done under a moderate load with 30 active channels.

The parameters for the interface chip operations were obtained from data books. This gave the time for the ISDN and LAN chips to read and write data to main memory, and in the case of the ISDN chip, also provided the time for one poll to the ISDN transmit buffers.

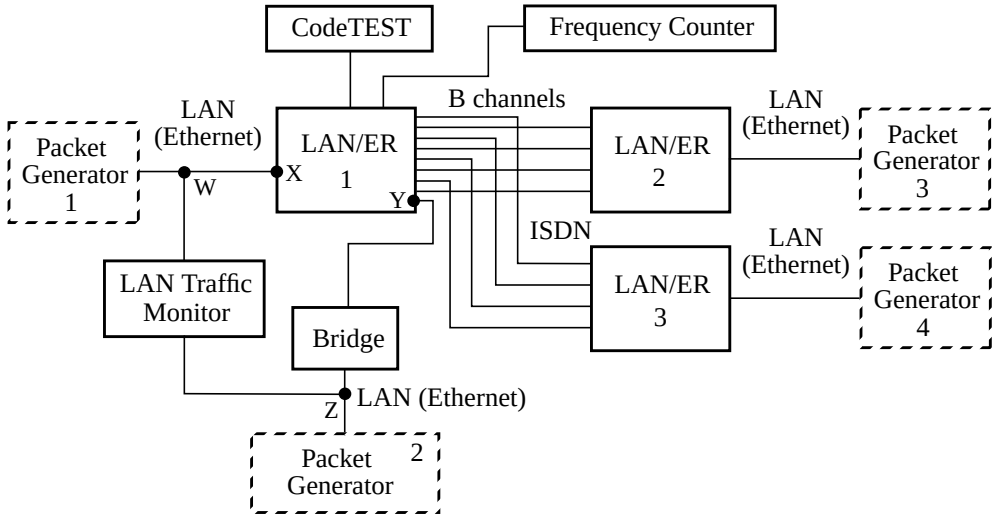
4.2 Model validation experiments

Validation was done by comparing predicted results from the model with measured delays across the LAN/ER, for both the inbound (LAN to ISDN) or outbound (ISDN to LAN) scenarios. Different packet sizes and number of active channels were used.

The measurement setup is shown below in Figure 10. It consists of three LAN/ER systems, two of which are connected back to back to the third LAN/ER. This was done to ensure that LAN/ER 1 will always bottleneck first by splitting the traffic load across LAN/ER 2 and 3. In Figure 10 there are 4 different slots where packet generators can be inserted. During the experiment three combinations of slots were used. Throughout the paper the direction in which traffic is flowing is always with respect to LAN/ER 1. In order for traffic to be flowing in the inbound direction, a packet generator will be

used in slot 1. In order to generate outbound traffic, packet generators will be used in slots 2, 3 and 4. For bi-directional traffic there is a packet generator in every slot.

FIGURE 10. Diagram of experiment setup for model validation



One of the problems with obtaining delay measurements is that the bridge (which converts ISDN packets to ethernet packets) skews the delay measurements. This bridge was required because the LAN traffic monitor used to measure packet delay could only measure delay across 2 ethernets. Therefore any delay measurement made will include the propagation delay across the bridge and the B channel. Without understanding the performance characteristics of the bridge, the experiment setup has to be calibrated as shown in Figure 11. The calibration is based on the assumption that with a single channel, delay through the LAN/ER (the delay from X to Y) is the sum of the execution times recorded by the CodeTEST. Subtracting this from the delay measured from W to Z gives the calibration adjustment for other measurements. This calibration adjustment is then subtracted for all the experimental measurements to estimate the delay from W to Y.

The calibration adjustment was determined for each packet size and for B channel utilizations of 45% and 90%.

FIGURE 11. Measurement calibration

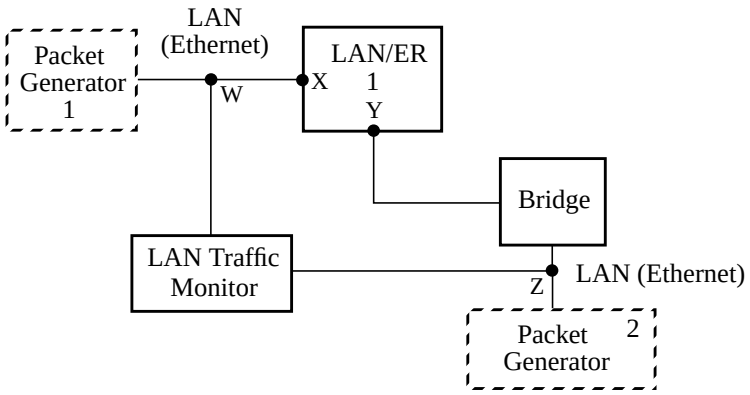
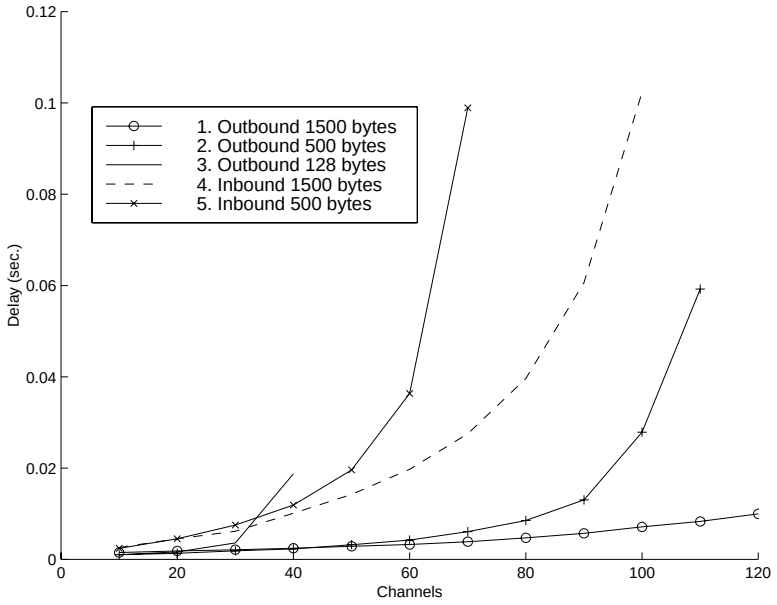


FIGURE 12. Uni-directional measurement results for (X - Y) delay, with every B channel 90% utilized

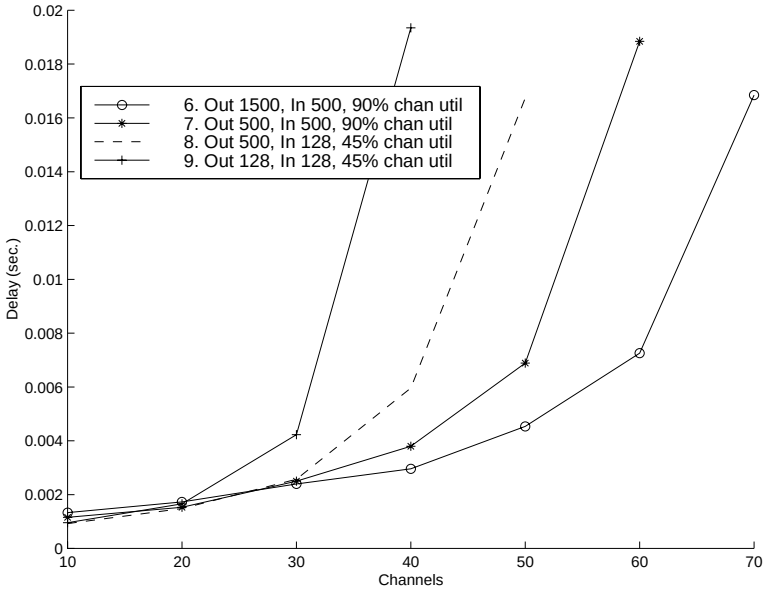


4.3 Model validation results

Figures 12 and 13 show the results of the end to end delay measurements. The delays are given for the outbound traffic, except for the two inbound traffic experiments in Figure 12. In all cases the number of active channels was increased up to the point where the router dropped 1% of the packets. This number of active channels was taken to be the router capacity for that traffic profile. The number of active channels can only go as high as 120 because this was a limitation of the LAN/ER.

Table 1 gives a summary of the model's accuracy for a wide variety of experiments. In each experiment the number of active channels was varied, and the table reports the results for the largest number of active channels that gave acceptable model error (which was taken as 12%). In the experiment

FIGURE 13. Bi-directional measurement results for (X - Y) delay, with either 45% or 90% B channel utilization



there was an upward trend in the error in all cases as the number of active channels increased. Thus the first row of Table 1, corresponding to the lowest line on the graph in Figure 12, shows that the accuracy of the delay predictions is within 10% over the entire range of active channels. The same is true for the last line in Table 1, and is true for all but the last point in the curve for the other lines except for rows 2, 3 and 4. The experiments of rows 2, 3 and 4 had high CPU utilization and system congestion at the top ends of their ranges. Row 3 shows saturation for a relatively small number of channels because each channel carries a lot of short packets, and the CPU effort is dominated by a fixed amount per packet.

The source of the errors appears to be over simplification in the modeling of the bus. When the CPU is waiting for the bus it is effectively busy and an error in modeling the bus, which inflates the time the CPU waits for the bus, will inflate the CPU utilization in the model. When the utilization is already high, this error has a magnified effect on the end to end delay. The bus modeling simplifications are:

- the ISDN chip arbiter shown in Figure 2 was not modeled, however this should not influence these results;
- the CPU locks memory in critical sections (not modeled) and this will prevent higher priority devices from taking the bus; this could influence the results at saturation;
- the bus arbitration time used in the model was the maximum value, and is a significant fraction (approaching 10%) of the memory access time which is the service time of the bus/memory element at the bottom of Figure 8. The actual arbitration time could be much less, which would affect the saturation point.
- the priority arbitration between the LAN chip and the ISDN chips was more complex than described in Section 3.1.2., which might have an effect.

TABLE 1. Summary of End-to-End Delay Errors: Range of system operation for adequate prediction accuracy. Higher utilizations gave errors over 10%.

Experiment Run		Max. Active Channels for Error <= 12%	Max Delay Error (%)	Max CPU Utilization
1	Outbound 1500	120	9.8	0.512
2	Outbound 500	80	9.6	0.64
3	Outbound 128	30	8.3	0.61
4	Inbound 1500	80	9.3	0.411
5	Inbound 500	60	5.1	0.643
6	Outbound 1500, Inbound 500	60	9.7	0.56
7	Outbound 500, Inbound 500	50	9.0	0.59
8	Outbound 500, Inbound 128	40	9.3	0.588
9	Outbound 128, Inbound 128	40	11.5	0.80

In all cases in Table 1, the bottleneck was the main bus. This result is not surprising, because every device in the system requires the use of the main bus/memory combination. Overall the inbound traffic capacity for the LAN/ER predicts roughly 45 to 60 channels less capacity than the outbound case given same load. This is attributed to the ISDN chip wasting bus time polling empty TX buffers, when all the traffic is going the other way. For the outbound case the ISDN polling has no significant effect, because when the chips are busy transmitting frames they do not poll the buffers as often.

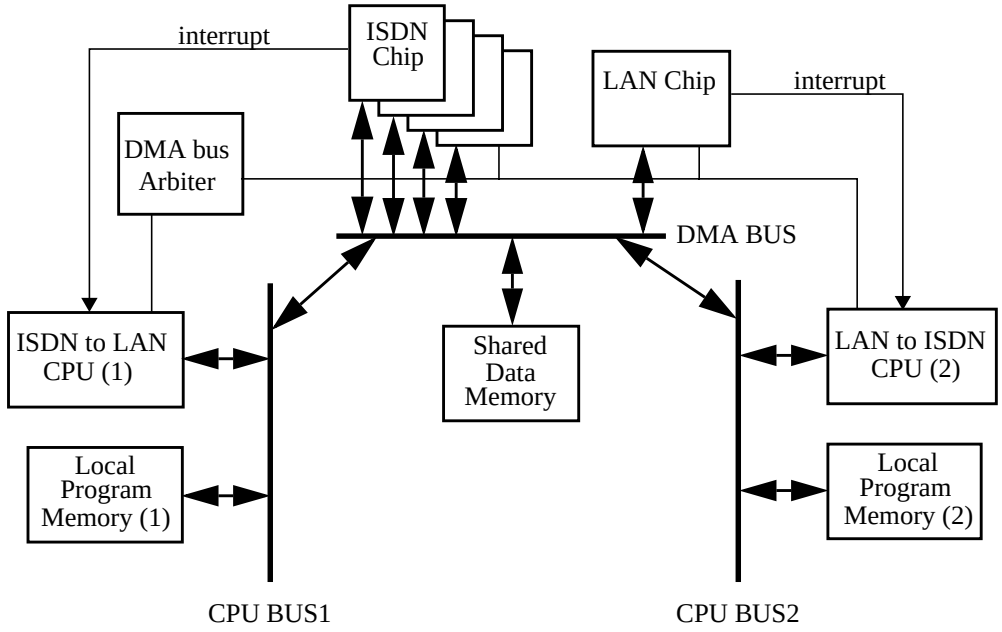
5.0 Model-based re-design and evaluation

A validated model can be used both to suggest and to evaluate a proposed re-design. Let us consider a system re-design to increase the capacity of the LAN/ER system so it can handle more B channels. The previous section showed two characteristics which hampered servicing more channels. First, the main bus was found to be the bottleneck in the system; second, ISDN polling occupied a lot of time on the bus when no frames had to be transmitted.

It was decided to consider a re-design which adds a second processor to the system and replaces the main bus by three buses, a main “DMA bus” for accessing all data and a local bus for each processor. The ISDN and LAN chips are on the main bus. This should increase capacity by reducing contention on the main bus. Each CPU will handle all the responsibilities of pushing data through the LAN/ER in one direction, and will contend only for access to shared data. Also the LAN and ISDN chip will not hinder the execution of code by the CPU to the same extent as in the old design.

The worker task no longer polls, and the ISDN and LAN chips send interrupts to the appropriate processor to indicate that a frame or packet is ready to be processed. A diagram of the new hardware design is given in Figure 14.

FIGURE 14. Hypothetical hardware/software re-design of the LAN/ER



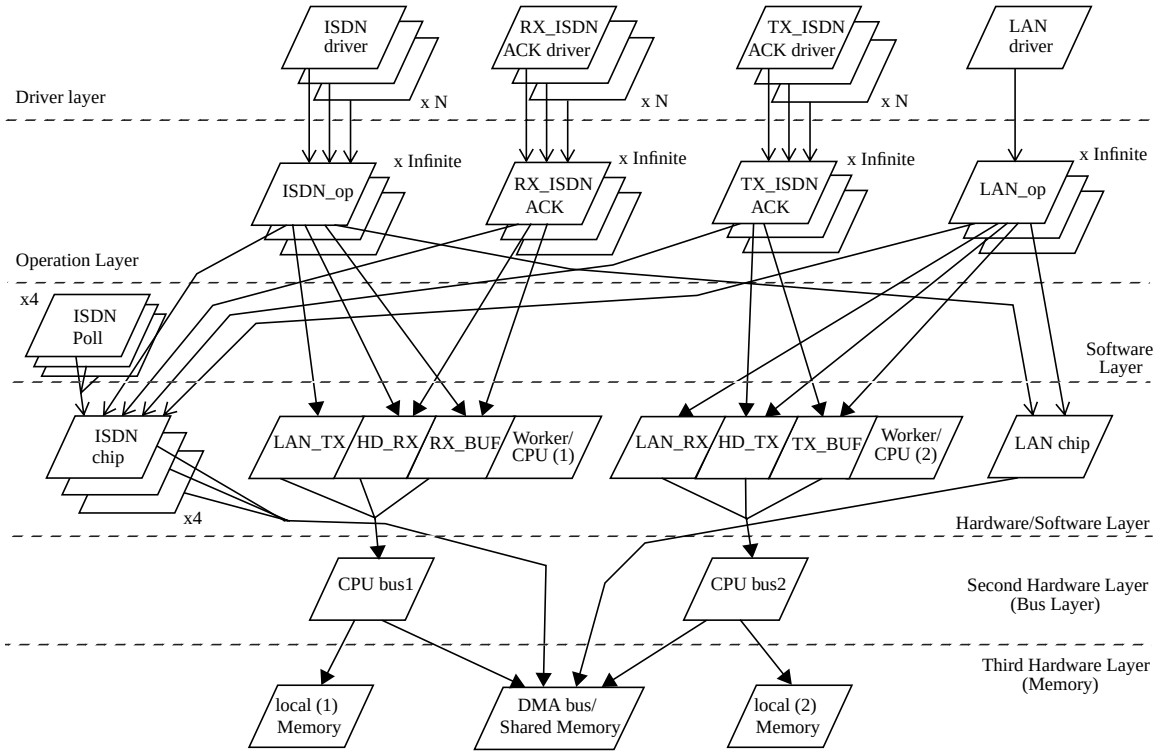
A layered model of the new hardware/software design is shown in Figure 15. One will notice looking at Figure 15 how the layered model captures the new system design. The reason for this is the layered model includes all the resource access patterns in a organized way, so one can look at the model and understand the new design without looking at Figure 14.

The new model is different in various ways from Figure 8. The maintenance tasks were left out because they only contribute 0.5% - 2.5% CPU utilization. The Worker task was divided into an Inbound Worker task on CPU (1) and an Outbound task on CPU (2). Then these tasks were merged, in the model with their CPUs (since they have no contention for the CPU) and the resulting merged hardware/software entities labelled Worker/CPU (1) and Worker/CPU (2). The bus/memory entity was partitioned into three bus entities and three memories. Then the shared memory was merged with its bus. Notice how the CPU buses access the DMA bus.

With the new model, Worker task polling has been removed and interrupt overhead has to be accounted for. It is assumed that total CPU cost of a single interrupt is 80 μs. This value was obtained from a context switch measurement on the iRMK RTOS kernel, which turned out to be 40 μs. Since every interrupt has two context switches, a value of 80 μs is used. The interrupt overhead is added into the service time for entries RX_FM and LAN_RX in Figure 15. It was also assumed that for every operation the CPU performs, it will spend half its execution time in the shared memory.

The new design was evaluated by solving the model for traffic consisting of 1500 byte packets or frames, either all inbound or all outbound. The results are shown in Figure 16. They are compared to the results found in Figure 12 for the same frame or packet size to show the improvement in capacity over the old design.

FIGURE 15. Layered model of a hypothetical hardware re-design



In the re-design, the overall average end to end delay has dropped significantly, because the system does not perform any polling. The bottleneck device now changes when the traffic switches direction. For the outbound case, the bottleneck is the LAN to ISDN CPU, CPU(2), but for the inbound case, the main memory is the bottleneck. In the experiments on the original design in section 4.2, the bottleneck device was always the main memory and main bus combination.

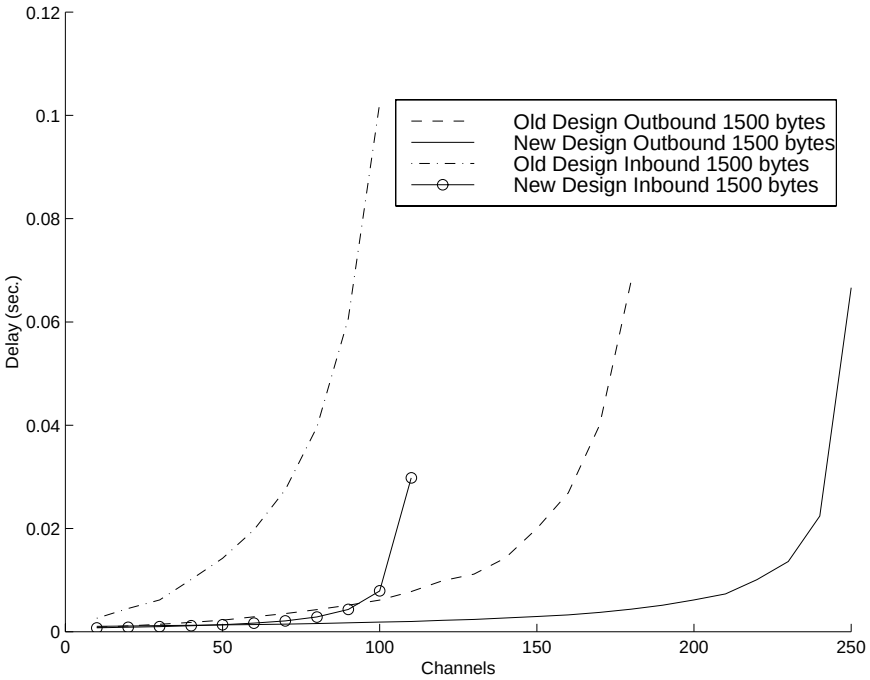
The capacity of the re-designed system is about 50% higher for both outbound and inbound traffic. Taking the capacity to be the number of channels giving 20 msec. delay, the outbound capacity has been improved from 150 channels to 230 channels, and the inbound capacity has increased from 60 channels to 90 channels.

6.0 Conclusions

This paper has demonstrated a layered Resource-based Model Architecture (RMA) approach to modeling router software and hardware together. The layered model shows the dependencies of resources on other resources, such as the dependencies of CPU (1) bus on the DMA bus to access shared memory in Figure 15. By showing these dependencies, the model will also show how contention effects spread upward from a congested resource, such as the bus in the original system.

The process of creating the model is systematic and straightforward. Structure and traffic parameters were found by tracing scenarios, and demand parameters were found by a form of profiling. Of these, the demand parameters required the greater effort. However once an established measurement process is in place the effort is less.

FIGURE 16. End to end delay (X - Y) comparison for new and old design, 1500 byte packets at 90% B channel utilization.



The measurement effort to find the demand parameters took 3.5 man-weeks starting from scratch with just a basic understanding of the system. The validation measurements took about 7 man-weeks, but they could be less detailed and less time-consuming in a mature process. Sufficient validation data could be obtained from routine stress testing, to track the modeling accuracy. Therefore the RMA approach combined with an established process of gathering measurements is a practical way of tracking the performance of the system at any point of the development cycle.

Layered modeling has been used before this for distributed software systems, but this is its first application to a system with layered hardware resources. High-capacity systems like routers often have significant functionality in hardware and require modeling of the interactions among the hardware components. Layered modeling provides a strategic level of detail. It captures the dependencies which affect performance, without the high cost of greater detail.

7.0 References

1. Applied Microsystems Corporation, *CodeTEST Users Guide,* Applied Microsystems Corporation, February 1996.

2. Mats Bjorkman and Per Gunningberg, “Performance Modeling of Multiprocessor Implementations of Protocols”, *IEEE/ACM Transactions on Networking*, vol. 6, no. 3, June 1998, pp 262 - 273.

3. R.J.A. Buhr and R.S. Casselman, *A Use Case Map Approach To High Level Design Of Object Oriented Systems*. Prentice Hall 1996.

4. G. P. Chandranmenon and G. Varghese, "Trading Packet Headers for Packet Processing", *IEEE/ ACM Transactions on Networking*, vol. 4, no. 2, April 1996, pp. 141 - 152.

5. Greg Franks, Alex Hubbard, Shikharesh Majumdar, John Neilson, Dorina Petriu, Jerome Rolia, Murray Woodside, "A Toolset For Performance Engineering And Software Design Of Client-Server Systems", *Performance Evaluation*, no. 24, 1995, pp. 117 - 136.

6. Pankaj Gupta, Steven Lin, and Nick McKeown, "Routing Lookups in Hardware at Memory Access Speeds", *Proceedings of the IEEE Infocom 98*, April 1998, pp. 1240 - 1247.

7. A. R. Hajare, "Performance Modelling of LAN Bridges and Routers", *Proceedings of the 16th Conference on Local Computer Networks*, IEEE Computer Society Press, 1991, pg. 554 - 561.

8. A. Jirachiefpattana, R. Lai, P. County, T.S. Dillon, *Proceedings of 1994 IEEE Region 10's 9th Annual International Conference on: "Frontiers of Computer Technology"*, pp. 1085 - 1089 vol.2.

9. O. G. Koufopavlou, A. N. Tantawy, M. Zitterbart, "A Comparison of Gigabit Router Architectures", *IFIP Transactions in Communications C-26*, pp. 107 - 121, 1994 Netherlands.

10. A. Kumar, T. V. J. Ganech Babu, S. V. R. Anand, "Comparitive performance of queueing strategies for LAN - WAN routers in packet data networks", *IFIP Transactions in Communications C-13*, 1993, pp. 103 - 116.

11. Butler Lampson, V. Srinivasan, and George Varghese, "IP Lookups using Multiway and Multicolumn Search", *Proceedings of the IEEE Infocom 98*, April 1998, pp 1248 - 1256.

12. Peter Newman, Greg Minshall, Tom Lyon, and Larry Huston, "IP Switching and Gigabit Routers", *IEEE Communications Magazine*, January 1997, pp. 64 - 69.

13. Craig Partridge et al, "A 50-Gb/s IP router", *IEEE/ACM Transactions on Networking*, vol. 6, no. 3, June 1998, pp 237 - 248.

14. H. Salwen, R. Boule, J. N. Chiappa, "Examination of the Application of Router and Bridging Techniques", *IEEE Network*, Vol.2, No. 1, pp. 77 - 80, IEEE, New York, 1988.

15. V. Srinivasan, and George Varghese, "Faster IP Lookups using Controlled Prefix Expansion", *Perfromance Evaluation Review Sigmetrics 1998*, Madison, Wisconsin, June 1998, pp. 1 - 10.

16. George Varghese, “Techniques for Efficient Protocol Implementations”, *Tutorials Sigmetrics/ Perfromance 1998*, Madison, Wisconsin, June 1998.

17. C. M. Woodside, J.E. Neilson, D. C. Petriu, and S. Majumdar. “The Stochastic Rendezvous Network Model for Performance of Synchronous Client-Server-like Distributed Software”, *IEEE Trans. Computers*, vol 44, no. 1, January 1995, pp. 20 - 34.