

IETE Journal of Research

ISSN: 0377-2063 (Print) 0974-780X (Online) Journal homepage: http://www.tandfonline.com/loi/tijr20

Modelling and Performance Analysis of a Digital Switching System—Case Study of the C-DoT DSS

Anurag Kumar FIETE, B Hareesh Kumar & B Madhavi

To cite this article: Anurag Kumar FIETE, B Hareesh Kumar & B Madhavi (1993) Modelling and Performance Analysis of a Digital Switching System—Case Study of the C-DoT DSS, IETE Journal of Research, 39:2, 77-97, DOI: 10.1080/03772063.1993.11437102

To link to this article: http://dx.doi.org/10.1080/03772063.1993.11437102

Published online: 02 Jun 2015.

Submit your article to this journal

View related articles

Full Terms & Conditions of access and use can be found at
http://www.tandfonline.com/action/journalInformation?journalCode=tijr20

Download by: [University of California Santa Barbara] **Date:** 16 June 2016, At: 08:43

Journal of IETE
Vol 39, No 2, March-April 1993, pp 77-97

Modelling and Performance Analysis of a Digital Switching System – Case Study of the C- DoT DSS

ANURAG KUMAR, FIETE, B HAREESH KUMAR
Department of Electrical Communication Engineering, Indian Institute of Science, Bangalore 560 012, India
AND
B MADHAVI
Texas Instruments (India) Pvt Ltd, Sona Towers, Bangalore 560 052, India

It is now well recognised that modelling and performance analysis is an important and necessary part of the development cycle of any large telecommunication system. In this paper we demonstrate some of the techniques that are used, and the kinds of results that can be obtained, by describing a detailed performance analysis that we have done for the Centre for Development of Telematics' Digital Switching System (C-DoT DSS). We have used a simulation model to evaluate the performance of the Base Module (BM). This model yielded an estimate of BM capacity, helped to design and evaluate a new overload control, and showed how BM capacity can be improved by a scheduling change. We have used analytical queuing models to estimate the capacities of the Administrative Processor, and the Space Switch Controller. We have formulated the system overload control problem as a constrained optimisation problem, have solved the problem, and have used simulations to evaluate various adaptive overload control algorithms.

Indexing terms : Overload control, Distributed systems, Switching system, Queuing models

Downloaded by [University of California Santa Barbara] at 08:43 16 June 2016

THE transmission, switching, and control in a modern telephone network can be viewed in terms of a hierarchy. The lowest level comprises transmission systems (trunks and local loops), and switching fabrics (located at the telephone "exchanges") that interconnect transmission channels in order to create end-to-end voice circuits. Further, in an electronic switching system, the switching fabric, and the line and trunk termination equipment is controlled by a network of special purpose computers. These computers constitute the call processing subsystem of the switching system. Within the switching system, these call processors communicate by passing messages over a simple message switching network. Finally, at the highest level in this hierarchy, the switching systems are themselves networked via data links and large packet switches. This network facilitates the setup of end-to-end voice circuits, network management, routing, and congestion control. The latter network is called the Common Channel Interoffice Signalling (CCIS) network.

A telephone network is a service system, and consequently has quality of service requirements that are characterised mainly in terms of two performance measures: probability of call blocking and call setup delay. A call is blocked if the network is unable to find a free voice path between the origin and the destination. Even if a free circuit exists, a call setup may be excessively delayed (or even blocked) owing to congestion in the call processing and signalling systems. The quality of service requirements are usually expressed as a maximum blocking probability, a mean setup delay, and a delay value that should be exceeded with a specified low probability. The capacity of a telephone network is the call attempt rate upto which all the performance requirements are met.

In a modern electronic telephone exchange, the switching fabric (or the interconnection network) is usually designed to be nonblocking, *ie*, if the terminals that need to be connected are free, then there exists a path through the interconnection network that can be used to connect them. As mentioned earlier, call requests can get blocked owing to the unavailability of a free series of channels linking the source and the destination. The modelling of, and blocking probability calculation in, circuit switched networks is a well formulated, though computationally difficult problem. It has received considerable attention in the performance analysis literature[1, 2]. We do not concern ourselves with this problem here.

Even if a network path is available between the source and the destination, a call can be delayed, or even blocked, during setup, owing to congestion in the call processing subsystem of an exchange, or in the CCIS network. In fact, since the transmission and switching resources are managed on the basis of demand blocking, the delay in call setup is governed entirely by the two upper layers in the hierarchy discussed earlier. The performance requirements on call setup delay can be met if appropriate performance requirements are made of each subsystem that participates in the call setup.

In this paper our concern is with the call setup within a processor controlled digital exchange. Performance requirements can be specified for the exchange, and the capacity of the exchange is the maximum call attempt rate upto which the requirements are all met. Models can be developed for the call processing architecture. These models are used to estimate

Paper No 154-A; Copyright © 1993 by IETE

Downloaded by [University of California Santa Barbara] at 08:43 16 June 2016

the performance yielded by the architecture, to estimate the capacity, to design congestion (or overload) control algorithms, and to suggest improvements. The whole process can be referred to as performance analysis and engineering. It basically involves the following steps

(*i*) The call processing architecture of the switch is carefully studied. This involves understanding the hardware blocks, the interprocessor communication, the call processing software on each module (down to the process scheduling level), and the flow of interprocessor messages during call setup.

(*ii*) Models are then developed for the various subsystems. These models are usually in the general area of queuing systems, and involve the theory of stochastic processes for their analysis [3,4]. The models should capture the essentials of the problem, ignoring aspects that would make the models computationally intractable without yielding any additional information. The models may need to capture some of the finer details of process scheduling within the processor, or may represent each processor as a queue in a queuing network.

(*iii*) The models are analysed either directly by stochastic analysis (queuing theory), or if such analysis is intractable, then via simulation.

(*iv*) The models are used to calculate delay distributions, and whatever other performance measures there may be for the subsystem being modelled. These calculations are used to determine the load beyond which performance requirements are violated.

(*v*) A very important outcome of the modelling process is that it may reveal problems in the architecture, and may suggest changes that would help improve the performance. Further, the models can be used to answer "what if" questions about the system.

Such a performance engineering effort is a necessary part of the development cycle of any large telecommunication system. It has been carried out for many large digital switching systems, developed all over the world. References [5,6] are two ext books that summarise some of these studies.

We have carried out a detailed modelling and performance nalysis of several subsystems of the digital switching system nder development by the Government of India's Centre for)evelopment of Telematics. In this paper we will describe this vork and present its results, in order to expound the techniques hat are used, and demonstrate the kinds of results and onclusions that can be drawn from such an analysis. The iodels and their analysis have yielded capacity estimates, and nproved congestion control and scheduling algorithms. In articular, the following is a list of analysis done and results btained. C-DoT DSS architecture is described after this itroduction.

(*i*) A detailed simulation model was developed for call processing in the Base Processor (BP). The model incorporates subscriber behaviour and details of process scheduling in the BP.

This model was used to estimate the call carrying capacity of a single Base Module (BM). We estimated a capacity of 6600 BHCA (Busy Hour Call Attempts). It was shown that without any overload control, the performance degrades dramatically beyond this attempt rate. The existing overload control was modelled and was shown to yield a performance worse than without the control. Reasons for this poor behaviour were revealed. A new overload control was developed and modelled. This yielded the estimated capacity, and throughput could be sustained even upto 30% overload.

The model also suggested a modified process schedule in the BP that yielded an estimated capacity of 8700 BHCA.

(*ii*) Queuing models were developed for the Space Switch Controller (SSC) and the Administrative Processor (AP). These were analysed to obtain the capacities of these modules. The significant effect of using non-intelligent link controllers was revealed by this analysis.

(*iii*) A fully configured DSS (32 BMs) may be able to accept 200,000 to 300,000 BHCA. This may be sufficient to overload the centralised modules such as the AP or the SSC. Thus a system level overload control strategy is needed. Such a strategy should prevent any subsystem from getting overloaded.

Using a somewhat idealised model for the entire call processing system, a system overload control strategy was devised. The strategy limits load by blocking calls at origination, but the extent of blocking exercised depends on periodic measurements obtained from all the subsystems to be protected. For a given level of offered load, the control level to be applied is obtained by solving a constrained nonlinear optimisation problem that seeks to minimise mean call setup delay while keeping call throughput as large as possible.

(*iv*) Modelling of the Input-Output Processor (IOP), the man-machine interface to the DSS, is currently underway at the time of this writing.

This paper is meant to be an exposition of modelling and performance analysis techniques through a practical application of these techniques to a real problem. Owing to the considerable detail to be modelled, we have used a simulation model for the BP. Our discussion demonstrates how a hardware and software description is converted into a system model, making appropriate assumptions. We show how insights obtained from queuing theory are used to interpret the simulation results, and to suggest system improvements.

On the other hand, the SSC and the AP are much simpler

Downloaded by [University of California Santa Barbara] at 08:43 16 June 2016

central resources. The call processing traffic to them is more easily characterised, and they are not affected by customer impatience for dial-tone. Consequently, we have used analytical queuing models to study the performance of these two modules. These models are analysed using well known techniques available in the queuing theory literature.

Finally, in developing a system overload control, it would have been impossible to handle the problem if we had modelled each module using its full performance analysis model. Our objective here was to study overload control methodologies in a network of processing modules. Consequently, we used single queue models for each processor and did not model the detailed process interactions within each module. Even this simple model has allowed us to evaluate various alternatives, and has yielded important insights.

We begin by briefly describing the C-DoT DSS architecture and call flow. We then discuss the modelling and analysis of the Base Module. We obtain the performance without overload control, and with the existing overload control, and show how the overload control and the BM capacity can be improved. Next we develop a queuing model for the Space Switch Controller, and analyse it to obtain an estimate of the capacity of this module. Then we do the same for the Administrative Processor. Finally, we develop a methodology for system level overload control of the entire call processing system.

C-DoT DSS ARCHITECTURE, AND TYPICAL CALL FLOW

Here we give an overview of the C-DoT DSS call processing architecture, and trace the flow of a typical call through the switch.

Overview of the architecture

The CDOT DSS has a modular and distributed architecture (Fig 1) consisting of a number of processor based modules, namely an Administrative Module (AM), a Central Module (CM), and a number of Base Modules (BM).

The AM provides the functions of overall system initialisation, systemwise maintenance, call routing and trunk allocation.

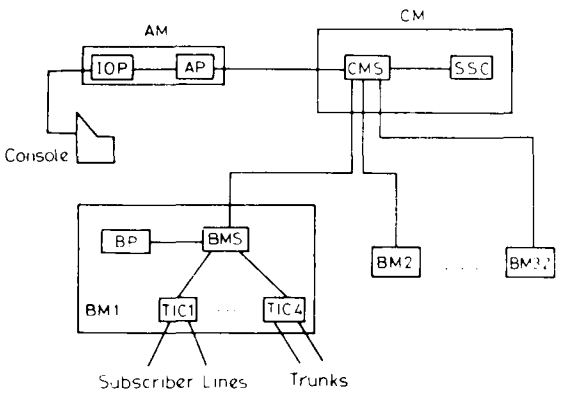


Fig 1 Architecture of the C-DoT DSS

The system administration consoles are connected to the Input-Output Processor (IOP), which interfaces to the system via the AM.

The CM consists of a Space Switch (SS), the Space Switch Controller (SSC), and a Central Message Switch (CMS). The space switch provides the digital paths for voice connections between BMs. The SSC allocates and deallocates inter-BM speech paths through the space switch. The CMS is basically a simple packet switch, and is connected to the BMs and the SSC by 64 kbps links and to the AM by a 128 kbps link. The various modules communicate with each other by sending messages that are switched between the links by the CMS. A simple stop-and-wait ARQ [7] protocol is used on each link. End-to-end message integrity is maintained by an end-to-end go-back-n ARQ protocol [7].

Lines and trunks terminate on the Terminal Interface Controllers (TICs) in the Base Module. Each Base Module comprises four TICs, each providing access to 128 subscriber lines or trunks. Hence the maximum configuration of 32 BMs supports 16,000 lines/trunks. A TIC performs functions such as origination detection, digit reception, answer detection, digit outpulsing, etc, for the line and trunk terminations. A BM also contains a Time Switch (TS) (time slot interchanger), a Time Switch Controller (TSC), a Base Message Switch (BMS) and a Base Processor (BP). The time switch is part of the overall time-space-time switching fabric, and is also used for switching intra-BM calls. The BMS provides message switching between the TICs and the BP. The BP processes call stimuli in the BM.

The processor modules run the CDOS operating system. This is a multitasking operating system, in which processes interact by passing messages to each other. The setup of a call involves the interaction of several software processes running in the processors of the various modules. Each of these processes can either be eternal or dynamic. Eternal processes are created during system initialization, whereas dynamic processes are created on demand by the operating system and they terminate on completion of their task (typically on the completion of a call). The processes communicate between themselves through messages. As mentioned above, when the processes are on different modules, the interprocessor messages are switched by the CMS.

Each process has associated with it a message queue. When a process gets its turn to run on the processor, it processes a message in its message queue. Upon completion of this work, the process joins the back of a queue of processes waiting for service at the processor. The operating system permits process priorities, but the priority is nonpre-emptive. Further, the queue of messages waiting at a process may also be ordered by message priorities. Note here that, although priorities between software processes are nonpre-emptive, hardware priority is pre-emptive. Thus, for example, an interrupt

by a link controller will interrupt a software process. This has important implications for central resources (*eg*, AP and SSC) that must communicate frequently with all the other modules.

Typical call setup

We shall describe the setup of a line-to-line call.

Consider a subscriber A placing a call to a subscriber B. When the subscriber A goes offhook, the terminal handler software residing in the TIC sends an origination message to an eternal process called the Status Control Process (SCP) in the BM to which A is connected. The SCP maintains the busy/idle status of the terminals (lines and trunks) in the BM. The SCP creates a dynamic process called the Originating Terminal Process (OTP) for the new call. A terminal process handles signalling and other details peculiar to the particular type of terminal it controls. This new OTP finds a digit receiver, and dial tone is fed to A via the time switch.

Upon receipt of the first digit, dial tone is disconnected, and the digit receiver is connected to A. The digit receiver collects the digits and sends them to the OTP. When digit collection is complete, the receiver is released.

The OTP now creates a Call Manager (CMR) process for this call. This process controls the overall progress of the call. This new CMR now sends a routing request along with the routing digits to the Global Routing and Resource Allocation (GRRA) process, an eternal process in the AM. The GRRA performs routing for both local and outgoing calls. If it finds that the call is an intra-BM call, it sends back a message to the CMR. If the call is inter-BM, the GRRA sends a message to the Global Path Control process (GPC), an eternal process residing in the space switch controller (SSC) in the central module. The main task of the GPC is to assign and switch space switch paths in the central module.

The GPC now sends a call request to the SCP process in the BM to which the called party B is connected. This call request signals the arrival of the terminating "half call" into the terminating BM.

This SCP on receiving a call request from GPC/GRRA, depending on whether it is an inter/intra BM call, checks B's idle/busy status. If it finds B free, it marks B as busy and creates a Terminating Terminal Process (TTP). This new TTP now sends a message to the CMR, indicating that B is free. The TTP then sends a message to the TSC in B's BM to operate a path and connect the ringing tone. Upon receiving the message from the TTP, the CMR sends a message to the OTP indicating that B is free. The OTP sends a message to the TSC in A's BM to operate its path. A and B are now connected via the time-space-time switch, and A and B get the ring back and ringing tones respectively.

When B lifts his handset, ringing is disconnected and the TTP sends an answer message to the CMR and enters conversation state. The CMR sends a message to the OTP and enters conversation state. The OTP upon receiving this answer message also goes into conversation state. Now A and B are in conversation.

Conversation can be terminated by either party, and this is handled in an analogous way.

PERFORMANCE ANALYSIS OF THE BASE MODULE

Scheduling of processes in the BM processor

All processes in the BM are run on the Base Processor (BP). When an origination message arrives at the SCP, the SCP, being an eternal process, may already be queued up at the BP. The origination joins the message queue at the SCP, while the SCP is itself waiting for service by the BP. Whenever the SCP goes into service, it takes the first call request from its queue and serves it. That is, the serving of call requests in the SCP queue is FIFO. After a call request is served, if the SCP queue is nonempty, the SCP joins the back of the BP queue. Thus one can think of the SCP as a server to the SCP queue, with one call request being served each time the SCP cycles through the BP queue.

It must be noted that the SCP must process not only the origination but also terminations which are terminating parts of calls. The terminations are also filed in the SCP queue. Since the SCP can serve a single request each time it cycles through the BP, this could be either an origination or a termination. Again the strategy for serving terminations in the SCP is FIFO.

The behaviour of the dynamic processes in the BP can be inferred from the call setup described earlier.

Modelling approach and assumptions

A queuing model that captures all the details of the above service system would be analytically intractable. Consequently we resorted to a simulation. The following modelling assumptions were made in developing this simulation.

(*i*) The Base Module is assumed to be a line BM with only subscriber lines connected to it. This implies that our model will not capture the effect of outgoing or incoming calls. A model for a BM with such calls can easily be obtained by changing some service time parameters.

(*ii*) The waiting room in the BP queue is assumed to be infinite. All resources both software and hardware, except for BP real time are assumed to be infinite. No call fails to succeed because of lack of a resource other than the BP. This is, in fact, how the actual system is engineered.

(*iii*) The input traffic into the system due to originations is assumed to be Poisson. Also the traffic due to terminations is assumed to be Poisson, and independent of the origination process. The latter assumption ignores the fact that some terminations are due to intra-BM calls.

Downloaded by [University of California Santa Barbara] at 08:43 16 June 2016

(*iv*) The inter-digit time, *ie*, the time between the dialling of digits is assumed to be uniformly distributed between 0 and 0.5 seconds.

(*v*) The traffic to the overall system is assumed to be uniformly distributed among all the BMs.

(*vi*) If a subscriber does not get dial tone before he loses his patience, he either retries or abandons the call. It is assumed that a subscriber retries with a probability of 0.8. The probability density function of the patience time of the customer is assumed to be as shown in Fig 2.

We say that a call is good or successful if it is successfully handled by the switch. A good call may fail to complete because the called party is busy, or not answering, but nevertheless it is considered to have been successfully switched. Every good call has an originating part on the originating BM, and a terminating part on the same or different BM depending on whether the call is intra-BM or inter-BM.

We have assumed that 65% of the total first attempts are good attempts (*ie*, if they were to be successfully processed by the switch, they would result in busy, no answer or completed calls) and the other 35% calls are a mix of no dial calls (15%), partial dial calls (15%), or calls that fail due to network congestion (5%).

If λ is the total arrival rate of originations to the BM, then the rate of good attempts is 0.65λ. Observe that a termination is generated for every good origination processed. Assuming that originations and terminations are uniformly distributed over the BMs, it follows that the termination rate into a BM is 0.65λ.

With the above assumptions, the model is implemented in SIMSCRIPT II.5 [8].

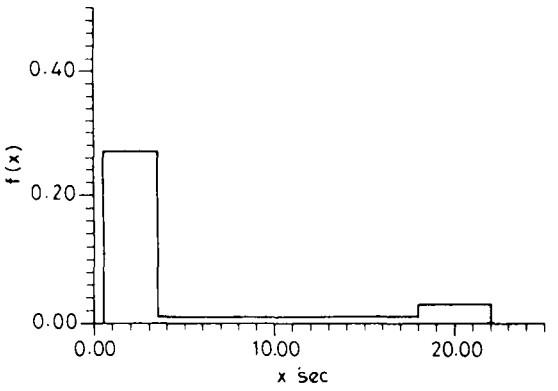


Fig 2 Probability density function of customer patience time

Performance of the system without overload control: capacity estimation

Simulation studies were carried out on the model described earlier, to study the performance of the system without any overload control. As discussed above, termination parts of the calls are generated at a rate 0.65 times the generation rate of originations. No controls are exercised on the originations or the terminations. Dial tone delay, and the rate of successful calls, were observed as a function of the total call attempt rate.

The delay requirements are: P (dial tone delay >2.5 sec) <0.015 and P (dial tone delay >1 sec) <0.05.

Figure 3 shows the results for the throughput and Fig 4 shows the mean dial tone delay and mean termination delay versus the offered load. The probability of dial tone delay exceeding 1 and 2.5 seconds is shown in Fig 5. From Fig 3, it can be seen that the throughput increases with arrival rate till 120 calls/min, beyond which the system goes unstable and the throughput drops down to zero.

The reason for the drop in throughput is that, as the attempt rate increases, eventually dial tone delay becomes so large that the subscribers (whose call originations are pending

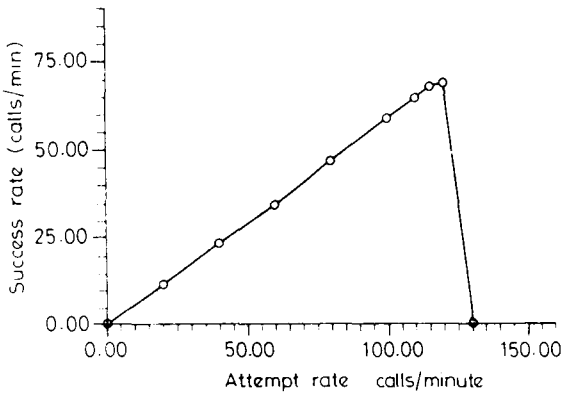


Fig 3 Throughput of successful calls with no overload control

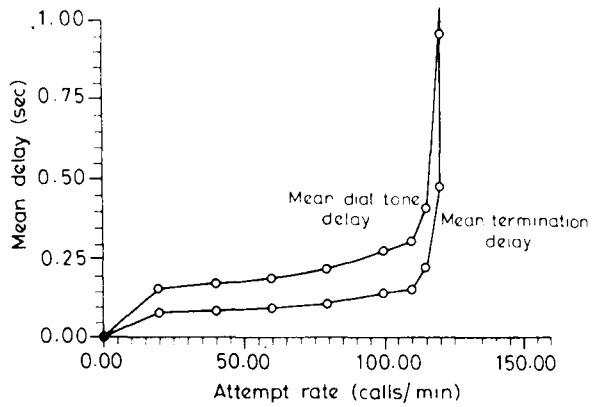


Fig 4 Mean delays for no overload control

Downloaded by [University of California Santa Barbara] at 08:43 16 June 2016

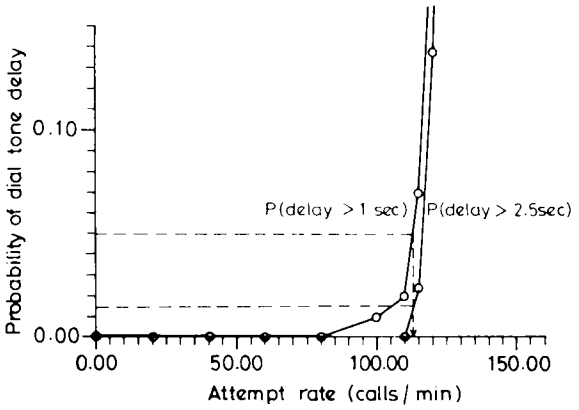


Fig 5 Probability of dial-tone delay for FIFO schedule

in SCP) become impatient, and they may respond by dialling even before dial tone is connected, or by reattempting or abandoning the call. Each of these actions results in an unssuccessful attempt even though the BP may have spent some time on the call. As load increases, the rate of reattempts increases, and the processor does more and more useless work. Ultimately the amount of useful work it does begins to decrease and eventually becomes zero [5].

From Fig 5, we can see that for arrival rates greater than 113 calls/min, the delay specifications given above are violated. Notice, though, that the delay is increasing very sharply at this load. Hence, in practice, one should operate at less than 113 calls/min, say 110 calls/min. We also found that the BP's occupancy at 110 calls/min is 0.63. Later on we will see how the capacity is affected by the service discipline.

To prevent the drop in throughput under overload, an overload control is employed. The overload control blocks excess calls if necessary, and attempts to maintain delay requirements for the processed calls. The overload condition of the system is detected by monitoring some load indicator such as the processor occupancy. Next, we describe the overload control strategy presently existing in the DSS and study its performance.

Performance of the existing BP overload control strategy

The load indicator used by the overload control mechanism is BP idle time. Every T seconds (chosen to be 60 seconds), the BP idle time over the past T seconds is obtained. This periodic measurement is used to determine the level of control to be applied. There are several levels of load control from "normal" to "critical". The degree of control action is gradually increased as the control level moves from normal, where there is no control action, to critical where maximum control is exercised.

At each control level, the control action essentially consists of blocking (shedding) of calls at various points in the call flow, as given below

(*i*) Origination control at the SCP: This is a proportional control in which after an origination is processed by the SCP, it is allowed to proceed further with a probability p and is rejected with probability $(1-p)$. The acceptance probability p depends on the control level.

(*ii*) Termination control at the SCP: This is also a proportionate control, similar to the origination control in SCP, but here the conrol is on the terminations, instead of the originations.

(*iii*) Blocking of originations at the TIC: This control is the rejection of calls right at the TIC, *ie*, calls are blocked at the ports and are not allowed to enter the system. The ports are viewed as being arranged in groups of 8, and the degree of this control is given as "n out of 8", implying that n out of 8 ports in the TIC have to be blocked. The value of n depends on the level of control. If any port is blocked, all the calls that originate on that port for the next control adjustment interval (T sec) are blocked. In order to be fair, the "window" of ports that are blocked is rotated at the end of every control adjustment interval, *ie*, at the end of every T sec.

The existing control strategy applies the above control actions when the measured BP occupancy exceeds a threshold. Based on the capacity estimation discussed above, we would like to maintain BP occupancy at 0.63, hence the threshold chosen in the simulation was 0.64. When the occupancy exceeds this threshold, the control actions are applied with progressively increasing severity, as long as the occupancy stays above the threshold, until the load entering the BP is completely blocked. The controls are relinquished gradually when the occupancy falls below 0.62. Notice that a certain amount of "hysteresis" is allowed; this is to prevent excessive triggering of the controls due to statistical fluctuations in the occupancy. Details of the existing algorithm may be found in [9].

Simulations were carried out with this control. Figures 6 and 7 show, respectively, the throughput of successful calls and mean dial tone delay as functions of the offered attempt rate.

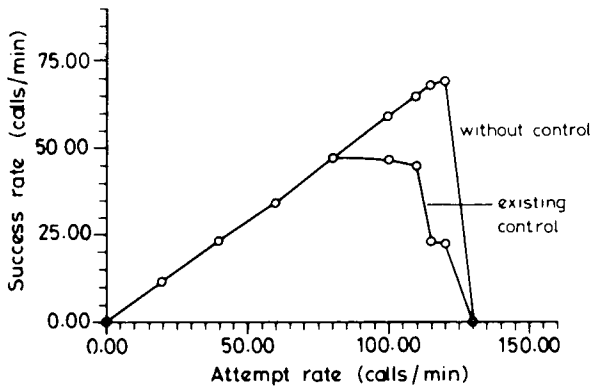


Fig 6 Throughput of successful calls without overload control and with existing overload control

Downloaded by [University of California Santa Barbara] at 08:43 16 June 2016

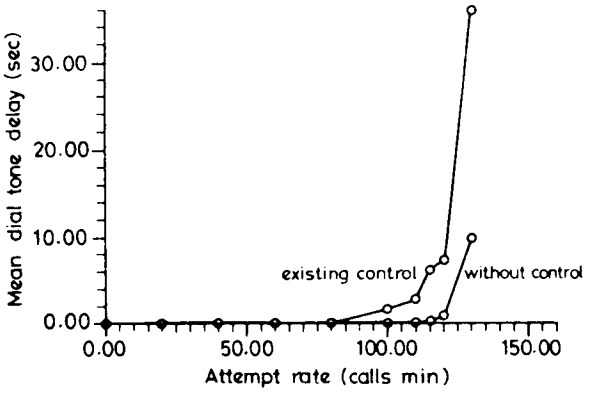


Fig 7 Mean dial tone delay without overload control and with existing overload control

We see that the performance of the existing control scheme is very poor both in terms of throughput as well as dial tone delay. The control begins to get triggered at just 80 calls/min, far below the capacity load of 110 calls/min. Thus, the throughput of successful calls never reaches the level that it should without any control. The other problems are also evident; large dial tone delays and reduction in termination throughput. The reduction in termination throughput is very undesirable since it implies that overload in one module will manifest itself in increased reattempts on modules that feed terminations to it, thus possibly driving them into overload.

One reason for the poor performance of the control is the large control update interval of 1 minute. Because of the large update interval, a false triggering of the conrol at lower loads than the capacity can only be corrected after a minute. This is the reason for the poor throughput at loads less than capacity.

The TIC control is a "bang-bang" control. Once overload control is triggered, the control level fluctuates so as to attempt to control the BP occupancy to the desired level. Since the TIC control is a bang-bang control with 4 discrete levels, the only way originations can be controlled to a desired rate is to let in either too much load or too little load, such that the long run occupancy is at the desired level. This is inevitable, but the long adjustment interval makes matters worse since the input is then very "bursty" causing large delays, customer abandonments, reattempts and thus a further increase in load.

To see this effect of bursty arrivals, consider a simple M/G/1 queue [3] as shown in Fig 8. Suppose that the actual arrival rate to the queue is $\tilde{\lambda}$, but we wish to control the rate to $\hat{\lambda}$ by switching the offered rate between (say) $\bar{\lambda}$ and $\underline{\lambda} = 2\hat{\lambda} - \bar{\lambda}$, for equal intervals of time. What effect does this have on customer delays?

If the actual arrival process to the queue was Poisson with a certain rate λ, then the mean delay versus λ curve is well known to be convex and increasing. In the control situation just

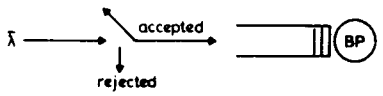


Fig 8 Bang-bang control of the input to a queue

described, if the durations for which the offered rate stays at $\bar{\lambda}$ and $\underline{\lambda}$ are long enough (so that the queue becomes stationary in that duration), then the actual mean customer delay will lie on the chord of the mean delay curve joining the points at $\bar{\lambda}$ and $\underline{\lambda}$. Thus the mean delay is more than that due to a Poisson process of rate $\hat{\lambda}$. The solution clearly is to switch between $\bar{\lambda}$ and $\underline{\lambda}$ more rapidly.

Further, if control adjustments are based on only the previous interval then sufficient history is not being kept, and it may be forgotten that extra load was let in and must be compensated for in the future [10].

Another problem is the origination control applied by SCP. By continuing to allow calls to enter the BM even under overload, the delays actually becomes worse, and then some of the patient calls that the SCP does manage to process, and which are likely to succeed, are rejected by the SCP origination control. Thus the throughput and delays become worse with this control, not serving the purpose it is meant for.

Improving the overload control

We have seen that the problems of the existing control strategy lie in the long measurement interval and the SCP origination control. Simulations were performed with the measurement interval reduced to 5 seconds, and also with the SCP origination control removed. It became clear that elimination of SCP origination control improved performance, but it was not sufficient to just reduce the measurement interval and keep the original TIC control algorithm [9].

We must adjust the control frequently (this helps to reduce arrival burstiness, and prevents an incorrect control level from lasting too long), but in order to reduce measurement "noise" the measurements on which the control adjustment is based should be taken over a sufficiently long time interval. The latter interval should not be too long, as that will affect the responsiveness of the control. We adopt a strategy that was proposed in [10].

The control level is adjusted every τ seconds. Under normal load, the occupancy of the BP over the previous W_m intervals (*ie*, over the previous $W_m\tau$ seconds) is monitored. When the overload threshold is crossed, the far end of the measurement window is anchored at the point at which it was when the control was triggered. Thus after each additional control interval, the measurement window grows by τ seconds and the control level adjustments are based on the occupancy measurements over this window. This anchoring of the window

Downloaded by [University of California Santa Barbara] at 08:43 16 June 2016

Downloaded by [University of California Santa Barbara] at 08:43 16 June 2016

is necessary in a "bang-bang" type control, since the only way to achieve a given input load into the processor is to alternately let in either too much load or too little. The anchoring allows the control to remember that it had let in too much load, and must now compensate for it.

The above scheme is studied to find suitable values for W_m and τ. To get the best $W_m\tau$, simulations were carried out on the model with the above control strategy, keeping τ at 5 sec and varying W_m. It was found that $W_m\tau$ = 120 seconds, yielded the best results, *ie*, a history of 120 seconds is sufficient for the system to perform well. Now with $W_m\tau$ = 120 seconds and varying τ (*viz*, 2.5, 5, 10 seconds), the throughput was studied. It was found that τ = 2.5 seconds, gave the best performance. It is clearly not advisable to make τ too small, as this will result in excessive overhead due to the overload control process.

Recall that in the TIC control, the window of ports blocked is rotated every control adjustment interval which is now very small. This quick rotation of the window causes calls that are waiting on a blocked port to leak in when the port is unblocked. Such originations have already spent some time waiting at the TIC, and hence are more likely to result in bad calls than fresh originations. To reduce this leakage effect, we implemented the following two actions, which reduce the rate of leaked originations.

(*i*) The window of ports blocked is rotated every $W_r\tau$ seconds, instead of every τ seconds. In our simulations results discussed below, $W_r\tau$ was taken to be 1 minute.

(*ii*) Even after a blocked port is unblocked, *ie*, the port is put back on scanning, an existing origination waiting at the port is not accepted until a time out period $W_t\tau$ of a few seconds.

We found $W_r\tau$ = 1 min (*ie*, W_r = 24) to give a much better performance than rotating the window of blocked ports at the end of each control update interval (*ie*, $W_r\tau$ = 2.5 sec).

The scheme described above implies that when overload control is active for a long time, the measurement window can grow indefinitely large. Thus if there is a sustained overload and then the load decreases, the control will take a long time to recover. To prevent this, the measurement interval is allowed to grow only to a maximum of $W_c(>W_m)$ during overload, and then a constant window of W_c is maintained, until the load returns to normal. We found a good value for W_c by increasing W_c (from a small value) until the performance was indistinguishable from $W_c = \infty$. This yielded a value of W_c = 240 (*ie*, $W_c\tau$ = 10 min).

We incorporated the above changes in the simulation. The origination control at the SCP was removed. At each control level the control applied is shown in Table 1. Under normal loads the control level is set at $L0$, and is increased by 1 level, from $L1$ to $L4$, for each τ for which the measured occupancy is above threshold.

From Figs 9, 10 & 11 which show respectively, origination

TABLE 1 TIC and SCP controls applied at each level of overload control

Control level	TIC control	Termination control at SCP
L0	0	0
L1	2/8	0
L2	4/8	0
L3	6/8	0.5
L4	8/8	1.0

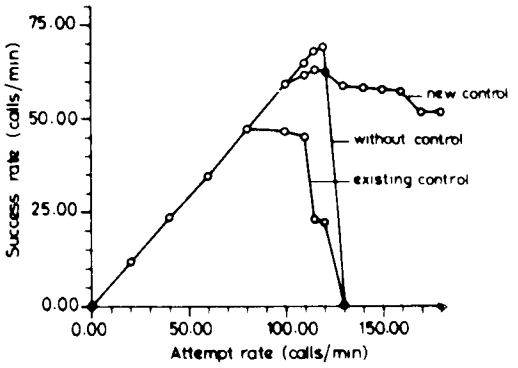


Fig 9 Throughput of successful calls with new and existing overload controls

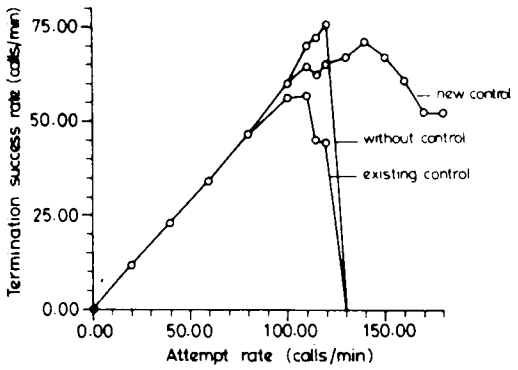


Fig 10 Throughput of successful terminations with new and existing overload controls

throughput, termination throughput, mean dial tone delay versus the offered load, the improved performance of the above control strategy can be seen. Observe that even with this control, at capacity load the throughput of good calls is slightly below that obtained without control, but the difference is small. This slight reduction is inevitable because noisy measurements cause false triggering of the control. The throughput and delay performance are, however, maintained very well even upto about 145 calls/min, *ie*, a 30% overload.

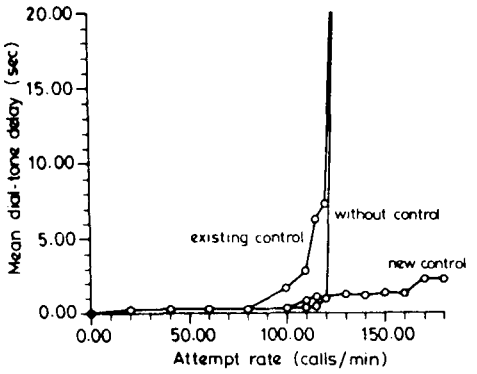


Fig 11 Mean dial tone delay for new and existing overload control

Observe that, unlike the existing control, termination throughput is also maintained. Dial tone delay is somewhat worse than that without control. The larger delays are contributed by those originations that do manage to leak in despite the precaution taken (the customer patience model allows customers to wait upto 20 seconds, but with a small probability). The only way to reduce delays is to somehow implement a mechanism by which originating customers can be given a congestion tone by the TIC, so that they hang up and try later.

An improved schedule in the base processor

With the existing service schedule in the BP, the estimated capacity is 110 calls/min at 63% occupancy as seen earlier. Hear we will first briefly discuss a well known result about customer behaviour and processor service discipline, and then we will propose a new service discipline for the BP.

We have observed above that the principal reason for the drop in throughput at high call attempt rates is customer impatience behaviour. Clearly there is a need for a mechanism which allows the BP to serve those originations that are more likely to result in successful calls. With the current schedule, at high loads the processor is more likely to serve an origination that will not result in a good call.

This problem has been studied analytically [11], and it was shown that in an M/G/1 queue, if the complementary distribution function of the customer patience time is convex, then it is optimal to serve customers last-in-first-out (LIFO). Here optimality is with respect to the throughput of successful customers. If the complementary cdf is concave then FIFO service is optimal. For general patience distributions, a mixture of LIFO and FIFO disciplines is optimal.

We proceeded with the above result in mind. Note that the complementary cdf of our chosen customer patience time distribution (Fig 2) is neither convex nor concave, but appears to be more "similar" to a concave function. For simplicity, therefore, we experimented with LIFO service for originations in the SCP queue. Note also that once a customer has received dial tone, and has dialed digits, he will tend to be more patient. We therefore, serve the terminations first-in-first-out in the SCP queue. We shall refer to this as the LIFO-FIFO discipline.

Figures 12,13 and 14 show the performance of the changed schedule, *ie*, LIFO-FIFO. It can be observed from Fig 14 that the capacity has gone up from 110 calls/min, for the FIFO-FIFO schedule, to 145 calls/min for the LIFO-FIFO schedule. The improvement in delays is also evident. The BP occupancy at the new capacity is found to be 0.82.

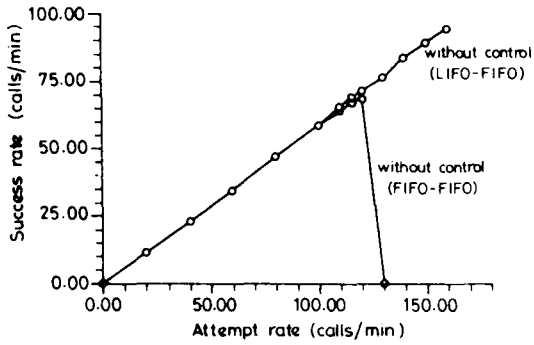


Fig 12 Throughput of successful calls for different SCP schedules

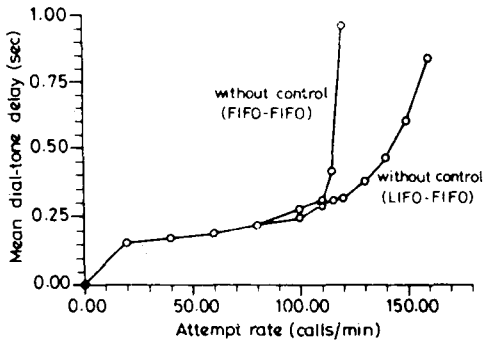


Fig 13 Mean dial-tone delay for different SCP schedules

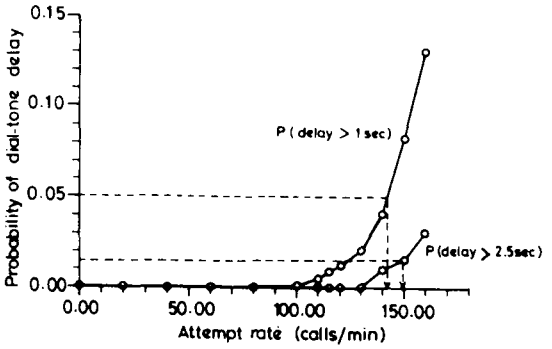


Fig 14 Probability of dial-tone delay for LIFO-FIFO schedule in the SCP

Downloaded by [University of California Santa Barbara] at 08:43 16 June 2016

Downloaded by [University of California Santa Barbara] at 08:43 16 June 2016

Results of implementation by C-DoT

The overload control, as described earlier, was implemented by C-DoT engineers in their laboratory BM. Load was applied using a load generator. Throughput, BP occupancy, and dial-tone delay measurements were taken (*i*) without any control, (*ii*) with the old overload control, and (*iii*) with the new control. Detailed customer behaviour could not be modelled in the load-generator; a 5 second dial-tone delay time-out and 100% reattempt probability was used.

Without overload control, the typical precipitous dropping of the throughput was observed as predicted. The BP occupancy shot up to close to 100%, indicating that the BP was doing a lot of useless work.

With the optimistic "customer behaviour" implemented in the load generator (a fixed 5 seconds "patience" time) the new overload control yielded a capacity of about 9000 BHCA at 80% BP occupancy. The discrepancy between these measured and our estimated results appears to be entirely due to the difference in the customer behaviour models. The average call service time in the BP is about 400 msec. Considering just the dominant part of the patience time density in Fig 2, a customer's patience time is uniformly distributed over 2.5 to 8.75 service times. If the caller simply uses a dial-tone delay time-out of 5 sec this implies a patience of exactly 12.5 service times. Even a simple M/M/1 model [3] now reveals that in the former case customer abandonments become a serious problem at around 60% occupancy, whereas in the latter model the problem begins to show up around 80% occupancy.

With the old control the measured throughput increased upto about 9000 BHCA, but then decreased under overload, staying well below that due to the new control.

The new overload control has been incorporated in the field version of the switch software.

PERFORMANCE ANALYSIS OF THE SPACE SWITCH CONTROLLER

The SSC and its activities

Recall that the Central Module houses the Space Switch which is a part of the overall time-space-time switching fabric and is used for switching inter-BM calls. The SSC is concerned with establishing and tearing down paths through the Space Switch. This is the only call setup related activity of the SSC.

The SSC communicates with the various module processors through messages handled by 64 kbps links attached to the Central Message Switch (CMS). These message transmissions and receptions cause hardware interrupts to the SSC. Each link interrupt needs service from an interrupt service routine that is executed by the SSC processor. The protocol used for message exchange between processors is such that each message results in 4 interrupts to a processor. A message reception/transmission is followed by its link acknowledgement, then by an end-to-end acknowledgement and finally by a link acknowledgement for the end-to-end acknowledgement. Recall that hardware interrupts are served at the highest priority and are preemptive.

In the current software version, some maintenance and audit processes are run in the SSC. The SSC periodically receives sanity monitor punches (SMPs) from the various processors, namely the BPs, the AP, the CMS. These are served at a higher priority than calls. In addition, the SSC performs tests that are served at a lower priority than call processing tasks. There are three types of tests (*i*) Sequential Memory Access test, (*ii*) Voice Path test, and (*iii*) Link test. Lastly the SSC is required to perform auditing functions that are also lower in priority than call processing. One such audit function is the detection and release of hanging buffers.

SSC processor occupancy as a function of load

We have seen earlier that tests and audits are performed only when the processor is not involved in the processing of calls, sanity punches or interrupts. Such work is said to be deferrable. There are three non deferrable tasks in the SSC, namely, processing of calls, processing of sanity punches and processing of interrupts. We shall refer to the fraction of time that the SSC is busy performing non-deferrable tasks as SSC occupancy.

Let

ρ = SSC processor occupancy (due to non deferrable work)

λ = Busy Hour Call Attempt (BHCA) rate

σ = Fraction of attempts at the BMs that are good attempts (see the section on Modelling assumptions)

λ_c = Arrival rate of requests for path setup and tear down = $2\sigma\lambda$

b_{setup} = Service time of a path set up request

$b_{teardown}$ = Service time of a path tear-down

λ_I = Arrival rate of link interrupts to the SSC processor

b_{int} = Interrupt service time

ρ_{SMP} = Occupancy of the SSC processor due to sanity monitor punches (about 1-2%)

We note that b_{setup}, $b_{teardown}$ and b_{int} are modelled as fixed times and are obtained from measurements. It follows that the SSC processor occupancy is given by

$$\rho = \rho_{SMP} + \sigma\lambda(b_{setup} + b_{teardown}) + \lambda_I b_{int} \quad (1)$$

Note that the interrupt arrival rate λ_I comprises the interrupts related to calls and also the interrupts due to other

Downloaded by [University of California Santa Barbara] at 08:43 16 June 2016

messages. Recall that the message protocol is such that each message (sent or received) causes 4 link interrupts. Each call generates 3 messages at the SSC: a switch path request from the AP, a termination message from the SSC to the destination BP, a switch path release message to the SSC.

Letting

λ_{I_o} = interrupt rate due to non-call related messages

we have,

$$\lambda_{I_o} = 12\sigma\lambda + \lambda_{I_o}$$

λ_{I_o} is easily calculated by accounting for the message traffic due to sanity monitor punches, and the various tests. It turns out that, for a 32 BM configuration, and for the current software version. λ_{I_o} = 150 per second. For details see [12].

QUEUING MODEL FOR THE SSC

The SSC processor has to serve 5 types of customers at different priority levels. This would yield a model as shown in Fig 15. The SSC occupancy due to the sanity monitor process is of the order of 1-2%. We remove the SMP out of the model by inflating all service times by a factor of 1/0.98 or 1/0.99.

Note that tests and audits are performed at the lowest priority, but are not indefinitely deferrable. There is an attempt to perform tests and audits in segments of 10 ms approximately. Though this traffic is not always present, we model it as though it is always queued for service at the SSC and call it "fill work". It then becomes a point of interest to know the exact number of fill work segments (of 10 ms each) that can be served by the SSC in a given time.

Thus we have a 3-priority queue model consisting of interrupts, path setups/tear downs and fill work segments as shown in Fig 16.

Note that interrupts can occur either during a call service time or the service time of a fill work segment. These interrupts receive pre-emptive service at the highest priority. This suggests that we can have a further simplified 2-priority model consisting of calls and fill work segments both of whose service times are modified appropriately to take into account the effect of the pre-empting interrupts.

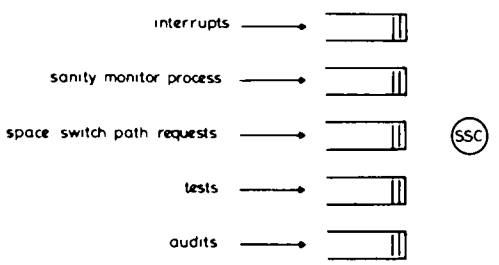


Fig 15 Five priority queuing model for the SSC

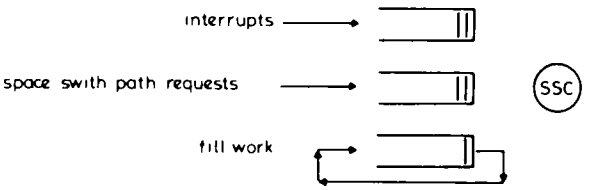


Fig 16 Three priority queuing model for the SSC

Moreover, since fill work segments are performed only in the absence of calls we can model these as nonpre-emptible "vacations" taken by the server whenever the call queue is empty.

If the arrival process of path setups and teardowns into the SSC, is assumed to be Poisson, then the above modification of service times due to interrupts can be done exactly, and we have the standard M/G/1 queue with vacations as the model for SSC [3]. This final model is shown in Fig 17.

Performance criteria

We first note that the call sojourn times at the SSC directly add up to the end-to-end set up delay. Hence, an important performance measure is switch path request sojourn time at the SSC.

Though tests are deferrable, there is a requirement that they cannot be delayed by more than a specified amount of time. It follows from our 2-priority queue model that this requirement will be met if there is a limit on the lenghts of call busy periods. Similarly, a certain number of audits must be completed within a specified amount of time. Thus, completion time for a cycle of tests and audits is another performance measure of interest.

These performance measures generate the following performance requirements, where t_1, t_2 and t_3 are specified time constraints, and α_1, α_2, α_3 are probabilities (close to 1) with which these constraints must be met.

$$P \text{ (Call setup or teardown time } < t_1) > \alpha_1 \quad (3)$$

$$P \text{ (Call busy period } < t_2) > \alpha_2 \quad (4)$$

$$P \text{ (Cycle time for } F \text{ fill work segments } < t_3) > \alpha_3 \quad (5)$$

We see that the performance requirements are specified in terms of probability distributions of the corresponding random variables. Hence we resort to the determination of Laplace Stieltjes Transforms [3] for the distributions of:

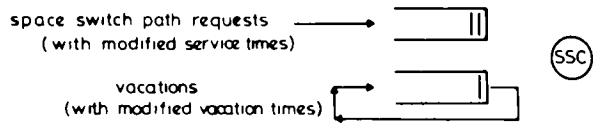


Fig 17 Two priority queuing model for the SSC

- Sojourn times in the call processing queue
- Busy periods in the call processing queue
- Completion time for F fill work segments

These are then numerically inverted using a numerical technique [13] to obtain the probability measures.

Analysis of the queuing model

Modification of call service times and fill work segments to incorporate interrupts

Let X denote the call service time random variable, Y the interrupt service time random variable, and V the vacation time random variable. Thus, for the workload model described earlier

$$X = \begin{cases} b_{setup} \text{ with probability } ½ \\ b_{teardown} \text{ with probability } ½ \end{cases}$$

Further, $Y = b_{int}$ with probability 1. Letting

b_{fill} = fill work segment length (= 10 msec in the current system)

we have $V = b_{fill}$ with probability 1.

Proceeding more generally, however, let $\tilde{x}(s)$ denote the Laplace Stieltjes Transform (LST) of the cumulative distribution function (cdf) $X(t)$ of X. Let X_I denote the call service time modified by hardware interrupts, and define $X_I(t)$ and $\tilde{x}_I(s)$ analogously. Similarly, define $Y(t)$, $\tilde{y}(s)$ and $V(t)$, $\tilde{v}(s)$, V_I, $V_I(t)$, $\tilde{v}_I(s)$.

Assuming that the interrupts arrive in a Poisson process independent of the call arrival process, we now proceed to find $\tilde{x}_I(s)$.

By definition,

$$\tilde{x}_I(s) = E(e^{-sX_I})$$

Since the interrupts receive pre-emptive service at the highest priority, the interrupt service queue is just an M/G/1 queue. Let G_I denote its busy period distribution, and $\tilde{g}_I(s)$ the LST of G_I [3]. It follows that

$$\tilde{g}_I(s) = \tilde{y}(s + \lambda_I - \lambda_I \tilde{g}_I(s))$$

Conditioning on the length of the unmodified call service time X, we get,

$$\tilde{x}_I(s) = \int_0^\infty e^{-sx} \sum_{k=0}^{\infty} \frac{(\lambda_I x)^k}{k!} e^{-\lambda_I x} (\tilde{g}_I(s))^k dX(x)$$

$$= \int_0^\infty e^{-x(s+\lambda_I(1-\tilde{g}_I(s)))} dX(x)$$

ie,

$$\tilde{x}_I(s) = \tilde{x}(s + \lambda_I(1 - \tilde{g}_I(s))) \quad (6)$$

Similarly, we have

$$\tilde{v}_I(s) = \tilde{v}(s + \lambda_I(1 - \tilde{g}_I(s))) \quad (7)$$

For the work load model discussed above

$$\tilde{x}(s) = 0.5e^{-sb_{setup}} + 0.5e^{-sb_{teardown}}$$

$$\tilde{y}(s) = e^{-sb_{int}}$$

$$\tilde{v}(s) = e^{-sb_{fill}}$$

Also it follows that,

$$EX_I = EX + EX \frac{\lambda_I b_{int}}{1 - \lambda_I b_{int}} = \frac{(b_{setup} + b_{teardown})/2}{1 - \lambda_I b_{int}} \quad (8)$$

$$EV_I = EV + EV \frac{\lambda_I b_{int}}{1 - \lambda_I b_{int}} = \frac{b_{fill}}{1 - \lambda_I b_{int}} \quad (9)$$

Call sojourn times

Now we have an M/G/1 vacation model for the SSC with arrival rate λ_C, service time LST $\tilde{x}_I(s)$ and vacation time LST $\tilde{v}_I(s)$. Let T_C be the random variable for the sojourn time of a call at the SSC. Let $t_C(s)$ denote the LST of the cdf $T_C(t)$ of T_C. Let T_{C_o} and $t_{C_o}(s)$ denote the random variable and its LST for call sojourn time in an M/G/1 queue without vacations. It follows from the Pollaczek - Khinchine formula [3] for sojourn time in an M/G/1 queue (without vacations), that

$$t_{C_o}(s) = \frac{s(1-\rho_{CI})\tilde{x}_I(s)}{s + \lambda_C - \lambda_C \tilde{x}_I(s)} \quad (10)$$

where,

$$\rho_{CI} = \lambda_C E(X_I) \quad (11)$$

Clearly ρ_{CI} is the processor occupancy due to calls and the interrupts that occur during call processing.

It now follows from the stochastic decomposition property of the M/G/1 queue with server vacations [14] that

$$\tilde{t}_C(s) = \tilde{t}_{CI}(s) \frac{1 - \tilde{v}_I(s)}{sEV_I}$$

$$= \frac{s(1-\rho_{CI})\tilde{x}_I(s)}{s + \lambda_C - \lambda_C \tilde{x}_I(s)} \cdot \frac{1 - \tilde{v}_I(s)}{sEV_I} \quad (12)$$

where $\frac{1-\tilde{v}_I(s)}{sEV_I}$ is the LST of the cdf of the residual life-time random variable of V_I [4].

Call busy periods

For calculation of call busy period distribution, we consider a vacation in which at least one arrival has occured.

Note that the presence of vacations in the model causes the

Downloaded by [University of California Santa Barbara] at 08:43 16 June 2016

Downloaded by [University of California Santa Barbara] at 08:43 16 June 2016

elongation of call busy periods because an arrival to an empty system does not immediately start service, but waits for the completion of the vacation time. Other arrivals may occur during this residual vacation and these will contribute to the following busy period. Let G_{CV} denote the random variable for the busy period due to calls arriving during such a vacation and $\tilde{g}_{CV}(s)$ be its LST.

Let $\tilde{g}_C(s)$ denote the LST of the call busy period distribution in the model without vacations. We have [3].

$$\tilde{g}_C(s) = \tilde{x}_I(s + \lambda_c - \lambda_c \tilde{g}_C(s)) \tag{13}$$

Conditioning on V_I and then unconditioning we get

$$\tilde{g}_{CI}(s) = \frac{\int_0^\infty \sum_{q=1}^\infty \frac{e^{-\lambda_c v}(\lambda_c v)^q}{q!} \tilde{g}_C(s)^q \, dV_I(v)}{(1-\tilde{v}_I(\lambda_C))}$$

$$= \frac{\int_0^\infty e^{-\lambda_c v} \, (-1 + e^{\lambda_c v \tilde{g}C(s)}) dV_I(v)}{(1-\tilde{v}_I(\lambda_C))}$$

$$= \frac{\tilde{v}_I(\lambda_C - \lambda_C \tilde{g}_C(s)) - \tilde{v}_I(\lambda_C)}{1-\tilde{v}_I(\lambda_C)} \tag{14}$$

where $\tilde{v}_I(s)$ is as in the previous section.

Cycle time for F fillwork segments (vacations)

Let U denote the random variable denoting the time until one cycle of F fillwork segments is completed and a fresh cycle starts and let $\tilde{u}(s)$ denote its LST.

A vacation is followed by a call busy period if arrivals take place during the vacation. Otherwise, the server goes on another vacation. Owing to the assumption of Poisson arrivals, we have a renewal process with a renewal epoch defined at the start of a vacation.

Let Z denote the random variable for the inter renewal time and $\tilde{z}(s)$ its LST.

Clearly

$$U = \sum_{i=1}^{F} Z_i$$

$$\tilde{u}(s) = \tilde{z}(s)^F$$

We now proceed to obtain $\tilde{z}(s)$. Conditioning on the vacation length random variable V_I and recalling $\tilde{g}_C(s)$ from Equation, 13, we have

$$\tilde{z}(s) = \int_0^\infty e^{-sv} \Big(\sum_{k=0}^\infty \frac{(\lambda_C v)^k}{k!} e^{-\lambda_c v} \tilde{g}_C(s)^k \Big) dV_I(v)$$

$$= \int_0^\infty e^{-v(s+\lambda_c(1-\tilde{g}_c(s)))} dV_I(v)$$

$$ie, \quad \tilde{z}(s) = \tilde{v}_I(s + \lambda_c(1-\tilde{g}_C(s))) \tag{15}$$

Finally, $\tilde{u}(s)$ is obtained from $\tilde{u}(s) = \tilde{z}(s)^F$.

Occupancy measurements in the SSC

We note here briefly our resolution of an anomaly that was observed by C-DoT engineers when they measured SSC processor occupancy. Whereas processor occupancy should increase linearly with λ, as is clear from Eqn. (1), the measurements yielded a nonlinear, convex, increasing curve.

Upon investigation, it turned out that the measurement process was measuring the fraction of time the SSC queue is nonempty. This is what we have denoted as ρ_{CI} (Eqn. (11)), and is clearly nonlinear in λ. The reason for the nonlinearity is clear: as λ increases, so does λ_I, thus causing an increase in the effective mean service time of a call request (*ie*, EX_I).

Numerical results

The transforms obtained in the previous section were inverted using a numerical inversion technique for Laplace transforms [13]. The numerical data used are as follows

$\sigma = 0.735$; $b_{setup} = 10ms$; $b_{teardown} = 6ms$; $b_{int} = 0.6ms$; $b_{fill} = 10ms$; $t_1 = 200ms$; $t_2 = 5$ seconds; $t_3 = 10$ seconds; $F = 100$ and $\alpha_1 = \alpha_2 = \alpha_3 = 0.95$.

In Fig 18 we plot ρ, the SSC processor occupancy, and ρ_{CI}, the occupancy measured by the built in idle time measurement process in the system. The discrepancy between the two is significant at low to medium BHCA. Note that ρ is not zero at $\lambda = 0$, since interrupts at the rate λ_{I_o} continue to arrive at the processor.

The cumulative distribution functions for the various performance measures were obtained for a set of λ values corresponding to 70%, 75%, 80%, 85% and 90% occupancy of the SSC. These values of ρ correspond, respectively, to λ = 36, 39, 42, 45, 48 call attempts per second.

Figure 19 shows the probability distribution of the call setup/teardown delay in the SSC for the various values of BHCA mentioned above. It is clear that space switch path request delay requirements are easily met upto an attempt rate of 48 calls/sec, *ie*, 172,800 BHCA, which corresponds to 90% processor occupancy.

Figure 20 shows the distribution of call processing busy period. Clearly this performance requirement does not constrain SSC capacity at all.

Figure 21 shows the cycle time for completing 100 fill work segments. This graph shows that at 45 calls/sec this requirement is met, but at 48 calls/sec it is not.

Owing to certain operational requirements, the possibility of adding other kinds of work to the SSC, and due to just plain caution, it is not considered appropriate to operate at above 70% occupancy. All performance requirements are, of course, met at this occupancy corresponding to a load of 129,600 BHCA.

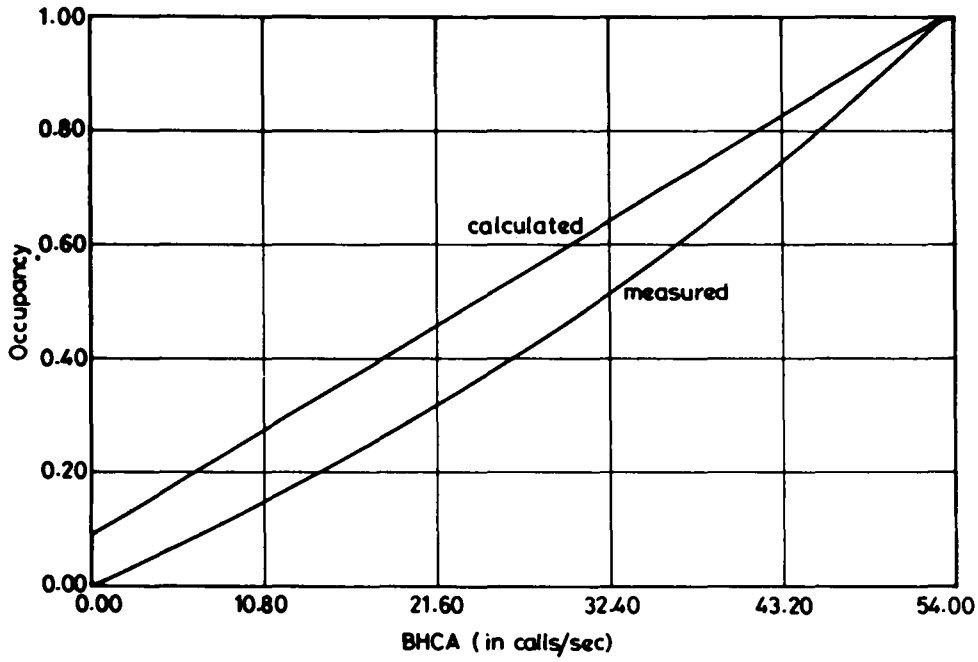


Fig 18 SSC processor occupancy, as calculated from analysis, and as measured by the built in system measurement

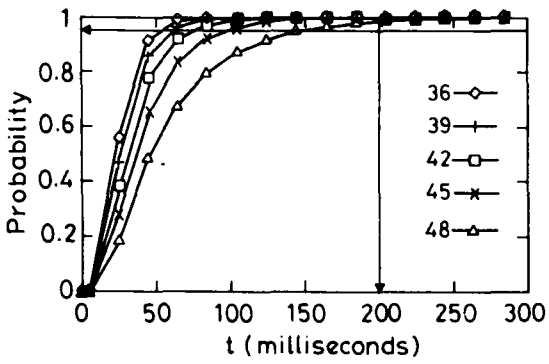


Fig 19 SSC performance: probability (path setup/teardown delay $< t$) vs t. $\alpha_1 = 0.95, t_1 = 200$ ms

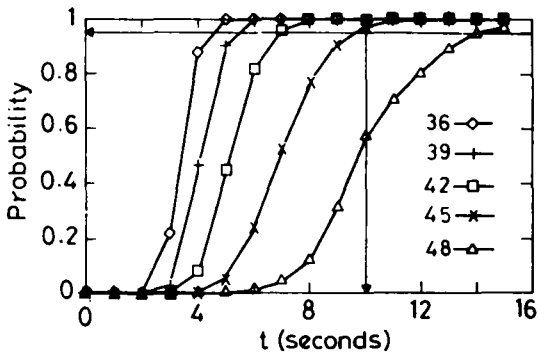


Fig 21 SSC performance: probability (completion time for 100 fill work segments $< t$) vs t. $\alpha_3 = 0.95, t_3 = 10$ sec

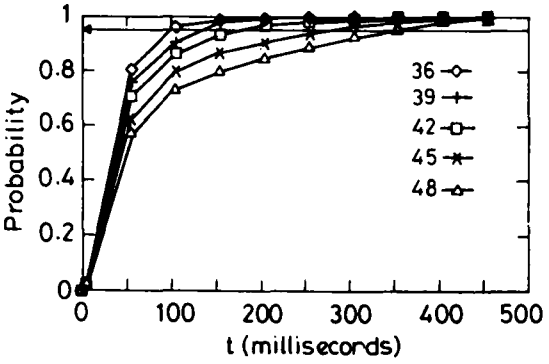


Fig 20 SSC performance: probability (call processing busy period $< t$) vs t. $\alpha_2 = 0.95, t_2 = 5$ sec

PERFORMANCE ANALYSIS OF THE ADMINISTRATIVE PROCESSOR

From a modelling perspective, the AP is very similar to the SSC.

The AP and its activities

Recall that the AP is connected to the CMS through a 128 kbps link and is the interface between the switch and the system administration consoles. Call processing work at the AP involves routing and allocation/deallocation of trunk resources. In addition, the AP acts as a conduit for flow of billing and traffic information from the BMs to the secondary storage at the Input Output Processor (IOP). The AP

Downloaded by [University of California Santa Barbara] at 08:43 16 June 2016

Downloaded by [University of California Santa Barbara] at 08:43 16 June 2016

communicates with the other module processors through messages that are switched by the CMS. The AP involves itself in processing the hardware link interrupts generated by these message transfers. Like every other processor, the AP participates in basic maintenance activities which include the reception of sanity punches from its copy (the AP is duplexed), the SSC, the IOPs and the Alarm and Display Unit (ADP). The AP in addition performs audits during idle time.

AP occupancy

We define the activities of call processing, sanity punch processing and interrupt processing as constituting the nondeferrable work at the AP. Deferrable work comprises the collection and delivery of billing/traffic records to the IOP, and also audits.

AP occupancy is the fraction of time the AP spends on nondeferrable work. The methodology for expressing the occupancy as a function of the call attempt rate is similiar to that of the SSC.

Since the discussion here is entirely about the AP, we can adopt the same notation used earlier without causing any confusion

Let

b_{alloc} = service time for routing and trunk allocation

$b_{dealloc}$ = service time for trunk deallocation

From the description of the call flow described earlier, it is clear that each intermodule call generates 3 messages at the AP. It follows, as in the case of the SSC, that

$$\rho = \rho_{SMP} + \sigma\lambda (b_{alloc} + b_{dealloc}) + \lambda_I b_{int} \qquad (16)$$

and

$$\lambda_I = 12\sigma\lambda + \lambda_{I_{dump}} + \lambda_{I_o}$$

where $\lambda_{I_{dump}}$ is the interrupt rate due to billing and subscriber monitoring dumps from the BMs to the IOP via the AP, and has the form

$$\lambda_{I_{dump}} = 0.3\sigma\lambda + 0.16\lambda$$

The coefficients in this equation are obtained from analysis of the AP interrupts due to dumps, and $\lambda_{I_o} = 8.5$ per second for a 32 BM configuration [12].

Queuing model for the AP

For the purposes of modelling, work at the AP is considered to be at three priority levels

(*i*) Interrupts, caused by message transmission and reception over the links, are served at hardware priority level. These are pre-emptive in nature.

(*ii*) Call processing work that involves routing and trunk allocation/deallocation is served at the highest software priority level.

(*iii*) Billing, traffic dumps and audits are served at the lowest priority.

Thus we again have a three priority model as with the SSC. Again, we treat the low priority work as fill work that is done in 10 ms segments. These segments are modelled as server "vacations". By modifying call service times and vacation times to incorporate the interruptions in service, the three priority model is shrunk to the standard M/G/1 queue with server vacations. The analysis is identical to the one described for the SSC.

The performance measures identified are call sojourn times and the probability that F fill work segments are completed within a given duration, which translate into the following performance requirements

$$P \text{ (call sojourn time at the AP } < t_1) > \alpha_1 \qquad (17)$$

$$P \text{ (cycle time for } F \text{ fill work segments } < t_2) > \alpha_2 \qquad (18)$$

Numerical results

The numerical data used for obtaining occupancy and performance measures are as follows

$\sigma = 0.735$; $b_{alloc} = 7$ms; $b_{dealloc} = 3$ms; $b_{int} = 0.6$ms;

$b_{fill} = 10$ms; $t_1 = 100$ms; $t_2 = 3600$sec; $\alpha_1 = \alpha_2 = 0.95$.

In Fig 22 the cdf of call sojourn time in the AP is plotted for $\lambda = 46, 50, 54, 58, 62$ calls/sec, which correspond, respectively, to $\rho = 0.60, 0.65, 0.70, 0.75, 0.80$. It is seen from the graph that sojourn time requirements are met even upto $\rho = 0.80$.

In Fig 23 we plot the probability of performing $100N$ fill work segments in 3600 seconds versus N, for $\lambda = 46, 50, 54, 60, 62$ calls/sec. The amount of fill work to be performed in one hour depends on λ, since the billing and traffic dumps are traffic related. Consequently, the value of F in the performance requirement (eqn (18)) depends on λ. For each λ the

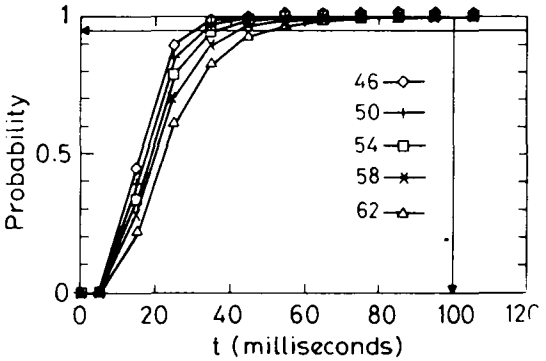


Fig 22 AP performance: probability (call sojourn time at AP $< t$) vs t. $\alpha_1 = 0.95$, $t_1 = 100$ ms.

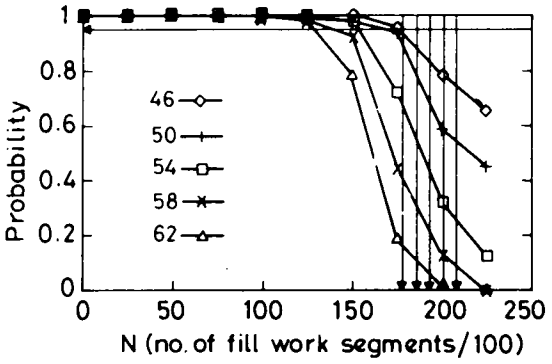


Fig 23 AP performance: probability (number of fill work segments done in 60 min $> 100N$ vs N. $\alpha_2 = 0.95$. (The vertical arrows point to the fill work requirement at each BHCA. This requirement increases with BHCA)

corresponding value of F is shown as an abscissa in Fig 23; the values increase with BHCA. It is clear that only for $\lambda = 46$ ($\rho = 0.6$) is the performance requirement met.

It follows that the capacity of the AP is 165,600 BHCA at 60% processor occupancy.

SYSTEM OVERLOAD CONTROL

Here we propose and partially develop a methodology for designing an overload control strategy for the entire switch.

Earlier it was seen that the maximum load a processor can carry is limited by the various performance constraints in that processor. The maximum load upto which these constraints are satisfied was defined as the capacity of the processor.

Recall also that the capacity of each processor module corresponds to a maximum occupancy. If the load offered causes an occupancy exceeding this, an overload control mechanism must respond quickly and attempt to shed the excess load. The objective of such a load shedding strategy will be to maximise carried load, while giving satisfactory performance to all the carried load.

A model for system overload control

The overload control mechanisms that are used in the BP are rather ad hoc. Driven by the need for a quick and easy implementation, the earlier work on performance analysis of the Base Module simply adopted the basic structure of the existing strategy, and modified it so that it yielded the desired performance.

In the work that we now report here, we have taken the first step towards devising an integrated control strategy for protecting all the processor based modules. For simplicity, we begin by considering only the BPs and the AP. This motivates the model in Fig 24 with which all the subsequent discussions are concerned.

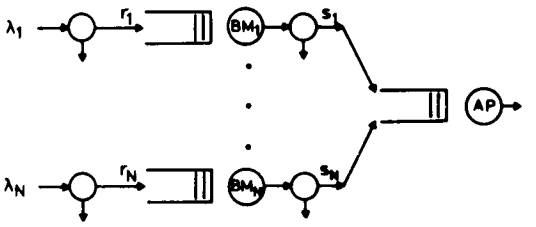


Fig 24 Queuing model for the overload control problem. The "valve"-like symbols denote load shedding points

There are N BMs. In Fig 24, λ_i denotes the rate of calls entering BM_i. Prior to each BM, there is a pre-dial control that controls the rate into BM_i to r_i ($\leq \lambda$). Between each BM and the AP there is possibly a post-dial control that further limits the rate r_i from BM_i to s_i ($\leq r_i$). Thus, the total input rate into the AP is $\sum_{i=1}^{N} s_i$.

The mean service times per call at the BP and AP are denoted respectively by b and a. We are given $0 < \rho^*_{BP} < 1$ and $0 < \rho^*_{AP} < 1$, such that the occupancies of each BP and AP must be maintained less than ρ^*_{BP} and ρ^*_{AP} respectively. Thus, all the offered load can be handled by each of the processors, if $\lambda_i \leq \rho^*_{BP}/b$, for all i, and $\sum_{i=1}^{N} \lambda_i \leq \rho^*_{AP}/a$. It follows that the maximum carried load $\bar{\gamma}$ for given $(\lambda_1,...,\lambda_N)$ is given by $\bar{\gamma}(\lambda_1,...,\lambda_N) = \min\left(\sum_{i=1}^{N} \min(\lambda_i, \rho^*_{BP}/b)\right), \rho^*_{AP}/a)$.

The problem is to decide at which points control should be exercised, and for a given set of values $(\lambda_1, \ldots, \lambda_N)$ what the values of $(r_1, \ldots, r_N)$ and $(s_1, \ldots, s_N)$ should be. We will impose the constraint that the throughput of the system should be as large as possible, *ie*, $\bar{\gamma}(\lambda_1, \ldots, \lambda_N)$.

Clearly there are many ways of sheding load to achieve this maximum throughput. We take a simple example to illustrate the point. Let $N = 2$, $\rho^*_{BP} = \rho^*_{AP} = 0.7$, $b = 0.35$ sec, $a = 0.2$ sec, $\lambda_1 = 2$ calls per sec and $\lambda_2 = 3$ calls per sec. In this example $\bar{\gamma}(\lambda_1, \lambda_2) = 3.5$. Consider now the following load sheding strategies:

Pre-dial control for the BP and post-dial control for the AP: It follows that in the present example $r_1 = 2$, and $r_2 = 2$, whereas $s_1 = 1.75$, and $s_2 = 1.75$. Of course, we can also use $s_1 = 2$, and $s_2 = 1.5$, but it is more convenient to have the same rate at each controller [15]. Note that the AP performance is the same for both these alternatives.

Pre-dial control only: Again there are several alternatives. We can set $r_1 = r_2 = 1.75$, or $r_1 = 2$, $r_2 = 1.5$. For both these alternatives, the AP performance is the same, but we will show that the average end-to-end call setup delay is less with the first scheme, in which the BP loads are balanced.

The first strategy is adopted in AT & T's 5ESS switch. The C-DOT DSS architecture bears a very close resemblance with AT&T's 5ESS. In the 5ESS lines and trunks terminate on

Downloaded by [University of California Santa Barbara] at 08:43 16 June 2016

Downloaded by [University of California Santa Barbara] at 08:43 16 June 2016

Interface Modules (IMs) similar to the Base Modules (BMs). There is a Time-Multiplexed Switch, a Message Switch and a Central Processor (CP) similar to the Administrative Processor (AP) in the C-DoT DSS.

The 5ESS employs a pre-dial scheme for the IM and a post-dial scheme for the CP. The post-dial scheme is a rate control scheme based on occupancy measurements in the CP and is implemented at the IMs [15]. Here, the rate is varied in steps depending on the occupancy exceeding some threshold levels. The various threshold values and step sizes have been fixed in the switch operation by experimentation.

At first sight, it may seem that a post-dial control scheme as used in the 5ESS for the CP can offer good transient performance. However, the performance of a switch is ultimately dictated by the set-up delay incurred in order to place a call through the switch. This comprises delays encountered by the call at various processors, as it is setup. The overload control mechanism should aim at keeping the overall end-to-end delay of a call to a minimum, while keeping the throughput as large as possible.

In a post-dial scheme such as that of the 5ESS, though the control can react rapidly to an overload, it is clear that under sustained overload, serving calls at the BMs only to reject them later is wasteful of BM capacity. It would be good to have a mechanism that rejects calls right at their entry point to the BM, *ie,* a pre-dial control. In such a case, we can have reduced dial-tone delays. In fact, our simulation results have shown that even the transient performance of the post dial control is not superior to the controls we devise.

At variance with the adhoc techniques used in earlier and existing switches, we have given a novel treatment for overload control by formulating it as a problem of finding the amount of load to be accepted at each BM, so that average end-to-end call setup delay is minimised, while keeping the throughput as large as possible (*ie,* at $\bar{\gamma}(\lambda_1,\ldots,\lambda_N)$). As pointed out above, in the 2 BM example, the objective of maximum throughput can be achieved in several ways. If the mean delay curves of the BPs are convex, nondecreasing, then the alternative with balanced BP loads will give lower end-to-end mean delay.

Formulation and solution of the overload control problem

Here we see how the overload control problem can be formulated as an optimization problem. Recall that each call gets served at the BP and subsequently at the AP. Our objective is to minimise the delays incurred by a call at the BPs as well as at the AP, while keeping the throughput as large as possible. We shall assume that all the call rejection is done before the BPs.

N, b, and a are as defined earlier. Let

x_i = call arrival rate to BP_i

r_i = accepted call rate at BP_i

We will first assume that x_i is a load that can be safely handled by BM_i (*ie,* $x_i = \min(\lambda_i, \rho^*_{BP}/b)$). Assume the presence of a pre-dial control for all the BMs that lets in loads that are within their capacities. Thus essentially the formulation that we will now present will show how to further limit the load to protect the AP.

Suppose we model the BPs and the AP as M/M/1 Queues. We will relax this assumption below.

For our problem with N Base Modules, the average delay in the network of Figure 24 then becomes,

$$\sum_{i=1}^{N}\left(\frac{r_i b}{(\sum_{j=1}^{N} r_j)(1-br_i)}\right)+\frac{a}{1-a\sum_{i=1}^{N} r_i}$$

It is clear that when no control is exercised, all the load offered to the BM_i, *ie,* x_i is let in. Thus,

$$0 \le r_i \le x_i$$

Moreover, the total carried load at the BMs which is the total offered load to the AP in steady state can be atmost ρ^*_{AP}/a. Thus in order to maintain the maximum possible throughput, we need

$$\sum_{i=1}^{N} r_i = \min\left(\sum_{i=1}^{N} x_i \rho^*_{AP}/a\right)$$

Denote the quantity on the right hand side by Λ. Note that, with this equality constraint, the second term in the average delay expression is a constant and hence can be disregarded while minimising the delay expression.

Thus, we have the following non linear optimization problem with linear constraints.

$$\text{Minimize} \sum_{i=1}^{N} \frac{br_i}{1-br_i}$$

$$\text{subject to } 0 \le r_i \le x_i \text{ and } \sum_{i=1}^{N} r_i = \Lambda$$

Performance studies on the BP, the AP and the SSC have yielded convex increasing delay curves at each of these processors. This is in fact typical of most queuing models.

Hence, letting $d(.)$ denote the mean delay function for each BP, we conclude that $c(r) = rd(r)$ is also convex increasing, and the above problem becomes

$$\text{Minimize} \sum_{i=1}^{N} c(r_i)$$

$$\text{subject to } 0 \le r_i \le x_i \text{ and } \sum_{i=1}^{N} r_i = \Lambda \qquad (19)$$

Downloaded by [University of California Santa Barbara] at 08:43 16 June 2016

The solution of this optimisation problem is given by the following theorem.

Theorem 1: The solution $(r^*_1, r^*_2, \ldots, r^*_N)$ to the above optimisation problem (eqn 19) is such that $r^*_i = \min(x_i, r^*)$, where r^* is obtained as the solution of $\sum_{i=1}^{N} \min(x_i, r^*) = \Lambda$.

Proof: It is easily checked that the solution satisfies the Kuhn-Tucker sufficient conditions.

Observe that the form of the optimal solution naturally lends itself to a threshold control implementation. The ideal throughput of such a controller is given by

$r_i = x_i$ if $x_i < r^*$

$r_i = r^*$ if $x_i < r^*$

A threshold controller is realised as follows, and does not require knowledge of the x_i. Consider the presence of a token pool at the BM. Each call that enters the BM is accepted for service at the BM, only if the pool is not empty. The call takes away a token. Arriving calls that find the pool empty, are rejected. When the threshold is r^*, a token is added to the pool every $1/r^*$ units of time. The token pool can hold a maximum of M tokens. The ideal throughput is achieved with $M = \infty$. Thus to keep the throughput as close to ideal as possible, a large stack is desirable. This, however, leaves open the possibility of too large a number of calls being dispatched to the BM in a short period of time, *ie,* the possibility of burstiness leading to larger queuing delays at the BP queue. Thus the value of the maximum token count is a compromise between throughput and delay.

On-line determination of the optimal control level

In real time operation of the switch, the arrival rate of calls is unknown, and will vary with the time of the day. This necessitates a control algorithm that obtains r from on-line measurements. The value of r should closely track the optimal control r^* as the offered load varies.

For fixed $(\lambda_1, \ldots, \lambda_N)$, *ie,* fixed $(x_1, \ldots, x_N)$, define $f(r) = \sum_{i=1}^{N} \min(x_i, r)$. Observe that $f(r)$ is the rate of calls reaching the AP when the threshold control at each of the BMs is set to r. The problem of obtaining r^* can now be formulated as that of obtaining the projection of the root of $f(r) - \rho^*_{AP}/a = 0$ into the interval $[\rho^*_{AP}/(aN), \rho^*_{BP}/b]$ [12]. Note that "noisy" measurements of $f(r)$ can be obtained by measuring the occupancy of the AP over successive intervals. The control level is adjusted at the ends of these measurement periods.

Thus the problem becomes one of finding a projection of the zero of a certain function $f(r)$ whose noisy values can be obtained for each value of its argument r. Such a problem is well known in the control literature. The same problem was obtained in the context of post-dial control of the CP in the 5ESS switch, and well known stochastic approximation techniques have been applied to it by the first author in [15]. If y_k is the measurement of $f(r_k)$, (*ie,* at the kth step of the iteration), the algorithm yields r_{k+1} as follows

$$r_{k+1}^{-1} = \theta_{k+1} = \left[\left(\theta_k + \frac{r_k}{\sum_{i=k-W+1}^{k} r_i^2} (y_{k+1} - 1) \right) \right]_{b/\rho^*_{BP}}^{(aN)/\rho^*_{AP}} \quad (20)$$

where $[.]_{\underline{\theta}}^{\bar{\theta}}$ is the projection of the quantity in [.] into the interval $[\underline{\theta}, \bar{\theta}]$. The sequence $\{\theta_k\}$ is an intermediate quantity whose interpretation can be found in [15].

Integrated overload control for the BPs and the AP

In our analysis so far, we have concentrated our efforts on the overload control problem for the AP and hence assumed the offered loads to each of the BMs to be within their capacities. Now we relax the assumption and see how our proposed solution of rate control can incorporate the safety of the BMs. In addition to the predial controls for implementation of overload control for the AP, let us imagine an independent control at each of the BMs to protect the BM from overload Fig 25. Suppose that the BP overload controls are also implemented as threshold controls. Let

u_i = the threshold at BP_i for protecting BP_i

r_i = the threshold that is applied at each BP to protect the AP.

λ_i = rate of calls offered to BP_i.

Finally, let

$x_i = \min(u_i, \lambda_i)$

and $r_i = \min(x_i, r)$

ie, r_i is the controlled rate of calls entering BP_i.

It follows that

$$r_i = \min(\min(u_i, \lambda_i), r) = \min(\lambda_i, \min(u_i, r))$$

Letting $t_i = \min(u_i, r)$, we see that both the controls can be integrated into a single threshold control, as shown in Fig 26. This suggests that both the AP and the BM in question work on the control algorithm independently and determine the

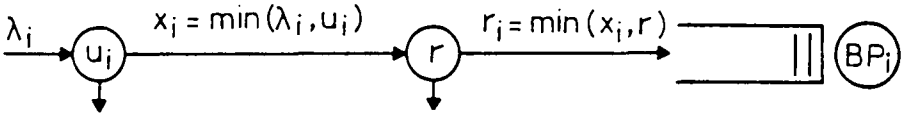


Fig 25 Schematic of independent BP and AP controls

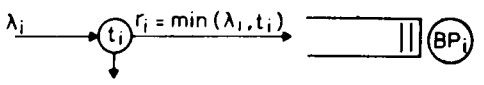


Fig 26 Schematic of integrated BP and AP controls

values r and u_i. The BM then applies a rate min (u_i , r) as the control.

Simulation experiments and results

A series of simulation experiments were done to study the proposed overload control.

- Each of the N BMs is modelled as a single server queue. The BMs are all identical with equal service speeds. Calls arrive at each BM in a Poisson process. Note that in reality, each call does not demand one continuous stretch of service but takes service at the BM intermittently. We model a call as follows: Each call generates a random number of stimuli. A stimulus requires an exponentially distributed service time of b_{stim} millisecs. Each stimulus, after its service time of b_{stim} millisecs has been expended, goes into a wait state with a probability p where $1/(1-p)$ = average number of stimuli per call. The wait state is represented by a delay node with a delay distribution that is exponential with mean t ms. Thus the mean service time of a call at the BM is $b = \frac{b_{stim}}{1-p}$.

- The Administrative Processor (AP) is modelled as a single server /M/1 queue. Calls departing from the queues of the various BMs enter this queue. Each call requires a millisecs at the AP.

- We assume that each BM is offered a load all of which requires service at the AP also. Note that this assumption is not strictly true, since a BM is subjected to load due to both originations and terminations. Only originations eventually require service at the AP. Terminating calls at a BM require service only at the BM. Further work is needed to incorporate termination traffic into our model.

The numerical data used are as follows :

N = 30; ρ^*_{AP} = 0.7; ρ^*_{BP} = 0.7; Maximum value for the threshold counter M = 30; a = 10ms; b_{stim} = 20ms; t = 500ms p = 0.9 Measurement and control update interval = 5 seconds

Several experiments were performed to study various threshold adjustment algorithms, and the one displayed above (eqn (20)) was found to have the best transient and steady state performance [12]. We report here the results of an experiment that was done to study the effect of the location of the control points, (pre-dial or post-dial or both) and to study their relative performance with respect to the steady state and transient behaviour. The loads on the BMs were changed after every 3 minutes to study the adaptiveness of the control strategies.

For the first 3 minutes, the loads on the BMs were $\lambda_1 \ldots \lambda_{20}$ = 3.5, $\lambda_{21} \ldots \lambda_{30}$ = 1. The correct level of control is 3.0. For the next 3 minutes, the loads were $\lambda_1 \ldots \lambda_{30}$ = 3.0. The correct level of control is 2.33. For the last 3 minutes of the simulation, the loads were $\lambda_1 \ldots \lambda_{30}$ = 2.0. The correct level of control is 4.5.

Three alternatives were studied

Pre-dial rate control for the BM and a post dial rate control for the AP: This simulation was done to evaluate the performance of the post-dial scheme used in the 5ESS. Figure 27 shows the performance. Let this scheme be called Scheme I.

Pre-dial rate control for the BM and a pre dial plus a post dial rate control for the AP: This control strategy was tested to see the advantage in having both a post-dial mechanism that may have the advantage of a better transient behaviour and a pre-dial mechanism that has better steady state behaviour. Under sustained over load, the post-dial conrol will not have any effect since the pre-dial conrol would already have limited the loads to satisfy the AP requirements. Figure 28 shows the results. We shall call this Scheme II.

Pre-dial rate control for the BM as well as the AP: Figure 29 shows the results. This is Scheme III. We also obtained the mean dial-tone delay (this is taken to be the mean time for serving the first stimulus in the call model described above), and the mean delay in the AP for each experiment. These are shown in Table 2. It is seen that Scheme I gives the worst performance in terms of average delays at the BP and the AP. Scheme II gives a performance close to that of Scheme III, but is slightly inferior. We now compare the schemes I and III. The strategy used in the 5ESS namely a pre-dial control and an independent post-dial control for the AP gives BP delays higher than that of the scheme that uses only a single pre-dial

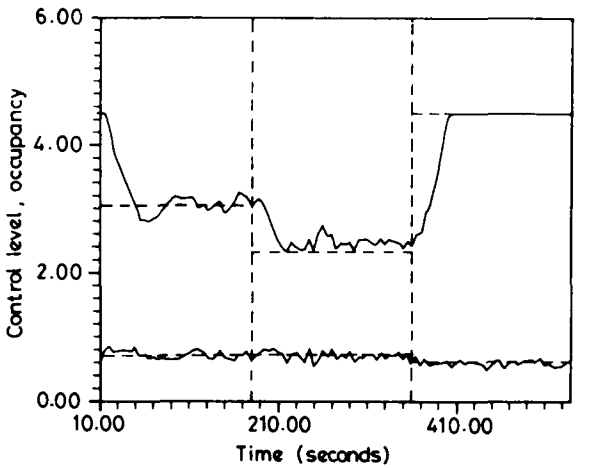


Fig 27 AP control level (upper plot) and AP occupancy (lower plot) vs time. Pre-dial control for BPs, and post-dial control for the AP. Mean dial-tone delay = 45.58 ms; mean delay at the AP = 29.04 ms

Downloaded by [University of California Santa Barbara] at 08:43 16 June 2016

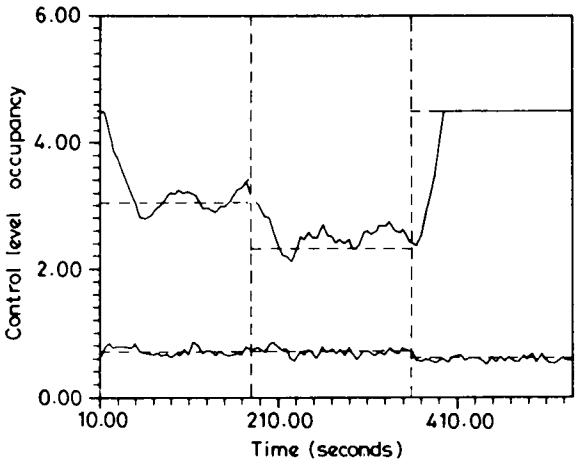


Fig 28 AP Control level (upper plot) and AP occupancy (lower plot) vs time. Pre-dial control for BPs, pre-dial and post-dial control for the AP. Mean dial-tone delay = 40.38 ms; mean delay at the AP = 22.67 ms

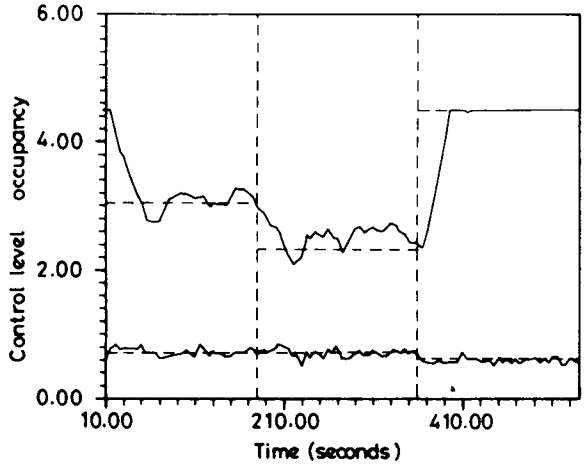


Fig 29 AP Control level (upper plot) and AP occupancy (lower plot) vs time. Pre-dial control only. Mean dial-tone delay = 39.77 ms; mean delay at the AP = 21.6 ms

TABLE 2 Mean Ddal-tone delay and mean delay in AP for the three overload control schemes

Control scheme	Mean dial-tone delay (ms)	Mean delay in AP (ms)
I	45.6	29.0
II	40.4	22.7
III	39.8	21.6

control for the BPs as well as the AP. This is of course expected since in Scheme I the BPs are unnecessarily handling load that is later rejected by the post-dial control. Scheme I, however, also gives a slightly higher delay at the AP. This is quite unexpected because a post dial scheme should react to a transient condition faster. The reason for the poor performance can be attributed to the finite convergence time of the algorithm in responding to the change in load. If we could have a hypothetical situation in which the AP could immediately detect an overload, then Scheme I would indeed give lower delays at the AP.

CONCLUSION

We developed a detailed simulation model for the BP, including customer behaviour and detailed process interactions. Using this model, the capacity of the BP was found to be 6600 BHCA at 63% occupancy. We developed an improved overload control that maintains throughput upto 30% overload. We showed how the service discipline in the SCP can be modified to increase the BP capacity to 8700 BHCA, at an occupancy of 82%.

Our BP overload control has been implemented by C-DoT, and has shown the expected performance in lab measurements.

We developed a methodology for determining the capacity of the AP and the SSC. We first identified the performance requirements to be met at each of these processors. We then developed queuing models to obtain the performance measures for the SSC and AP. Analysis was done to determine the Laplace Transforms of the performance measures which were then numerically inverted to yield the probability distributions of the performance measures as a function of load. From the analysis performed for SSC, we infer that the SSC can safely handle loads corresponding upto 85-90% processor occupancy. The operational requirement of 70% maximum occupancy yields a capacity of 130,000 BHCA for the SSC. From the performance analysis of the AP, we conclude that the AP can handle a capacity load of 165,000 BHCA corresponding to 60% occupancy while satisfying all performance requirements.

Measurements of occupancies of the AP and the SSC, conducted at C-DoT, exhibited a non-linear variation at moderate to high loads. By a simple mathematical argument, we have deduced the reason for this reported nonlinearity, and have also pointed out the correct methodology for the determination of processor occupancy.

We next studied the design of an overload control for the entire switch. We formulate the problem as one that minimises the delay involved in setup of a call through the switch while keeping throughput as high as possible. This yielded a non-linear minimisation problem and a pre-dial rate control was guessed as the optimal solution. The solution was proved to satisfy Kuhn-Tucker sufficiency conditions for the minimisation problem.

Downloaded by [University of California Santa Barbara] at 08:43 16 June 2016

We then developed a control strategy that obviates the need for several independent controls working at the various processors. The control is simply a pre-dial rate control at the Base Modules. We showed how such a control can be adaptively implemented and used simulation to compare its performance with some natural alternatives. Our proposed control is found to be the best in steady state and transient performance.

FUTURE WORK

Work remains to be done to use the methodology we propose in this paper to develop a comprehensive overload control strategy, that protects all the processor modules, and accounts for all the types of traffic that modules are subjected to. In particular, we have not accounted for termination traffic in the BPs; further our model does not incorporate the SSC. Our methodology establishes a conceptual framework, and it should not be difficult to incorporate these other factors.

ACKNOWLEDGEMENT

We are grateful to several C-DoT engineers for answering patiently our many questions about the hardware and software architecture of the switch, and for providing us with various measurements. In particular, we wish to thank Venkatrajendran, Divisional Manager, Software, and the members of the Performance Engineering Group, Srinivasachar, Naseem Meriam, and Venugopal.

REFERENCES

1. F P Kelly, Blocking probabilities in large circuit switched networks, *Advances in Applied Probability,* vol 18, pp 473-505, June 1986.
2. D Y Burman, J P Lehoczky & Y Lim, Insensitivity of blocking probabilities in a circuit-switching network, *Journal of Applied Probability,* vol 21, pp 850-859, Dec 1984.
3. L Kleinrock, *Queueing Systems*, vol I and II, John Wiley.
4. R W Wolff, *Stochastic modelling and the theory of queues,* Prentice Hall, 1989.
5. G Hebuterne, *Traffic flow in switching systems*, Artech House, 1987
6. Mischa Schwartz, *Telecommunication networks: protocols, modelling and analysis,* Addison Wesley, 1987.
7. A Tanenbaum, *Computer networks*, IInd, Prentice Hall, 1988.
8. E C Russel, *Building simulation models with SIMSCRIPT II.5,* CACI Publication, Los Angeles, 1983.
9. B Hareesh Kumar, Performance analysis and optimisation of the C-DoT digital switching system, *Master of Engineering Project Report*, Dept of ECE, IISc, Bangalore, January 1990.
10. J S Kaufman & Anurag Kumar, Traffic overload control in a fully distributed switching environment, in *Teletraffic Sciences for New Cost Effective Systems, Networks, and Services, Proceedings of the ITC-12,* M Bonatti (ed), North Holland, 1989.
11. B T Doshi & E Lipper, Comparison of service disciplines in queuing systems with delay dependent customer behaviour, *Applied Probability and Computer Science, The Interface,* vol II, R L Disney & T J Ott, Birkhauser (ed), 1982, pp 269-301.
12. B Madhavi, Performance analysis of the SSC and the AP and design of the system level overload controls in the C-DoT Digital Switching System, *Master of Engineering Project Report,* Dept of ECE, IISc, Bangalore, January 1991.
13. D L Jagerman, An inversion technique for the Laplace transform, *Bell System Technical Journal,* vol 61, pp 1995-2002, Oct 1982.
14. S W Fuhrmann & R B Cooper, Stochastic decomposition in an M/G/1 queue with generalised server vacations, *Operations Research,* vol 33, 1985.
15. Anurag Kumar, Adaptive overload control of the central processor in a distributed system with a star topology, *IEEE Transactions on Computers*, vol 38, pp 1502-1512, November 1989.
16. M Bazaraa & C M Shetty, *Nonlinear programming: theory and algorithms,* Wiley, New York, 1979.

Downloaded by [University of California Santa Barbara] at 08:43 16 June 2016