OVERLOAD CONTROL IN A HIERARCHICAL SWITCHING SYSTEM

David MANFIELD, Bob DENIS, Kalyan BASU, Guy ROULEAU

BNR
Ottawa, Canada

ABSTRACT

Large digital switching systems have distributed call processing implemented in a hierarchical architecture with high-level call processing handled by a central processing facility, and low-level functions handled by a large number of peripheral processors. In a large digital switch without overload controls the loss of throughput when the real-time attempt capacity is exceeded is catastrophic, due to the extremely steep load-service relations at the point of switch capacity. In this paper the requirements for overload controls in a distributed switch are investigated through the use of a novel, distributed, call-based simulator which is modeled on the actual switch architecture. The performance of a successful set of overload controls is demonstrated, and rationalized in terms of the differing functional requirements for overload controls at each level of the call processing hierarchy.

1. INTRODUCTION

Most modern digital switching systems have distributed processing of telephone calls implemented through a hierarchal switch architecture. High-level call processing functions requiring centralized resources are handled in a central processor (e.g., network path selection, service circuit allocation, translation, routing). Low-level call processing functions are handled by a large number of peripheral units, each one in control of a number of lines or trunks. The low-level call control functions include idle line scan, DP digit collection, line supervision, tones and ringing. The two levels of the control hierarchy communicate by means of messages to synchronize the concurrent real-time call processing. Both the peripheral controller (PC) and the central control (CC) maintain state machines for a particular call.

The switch real-time processing capacity determines the maximum attempt load and throughput that the switch can handle. The determination of this capacity is a complex problem in itself, and we do not address it here. Suffice it to say that the real-time capacity of all the switch processing elements needs to be considered. In the work reported here it is assumed that it is the CC-real-time which is the limiting system resource, because all calls are funneled through the CC from all the PC.

From the engineering point of view, the dial tone delay characteristic of the CC is a very steep function of the attempt load when the switch is operating near the point of capacity. This steep load-service relation is almost purely a result of the large switch size and not of the software architecture. The large switch size makes it particularly susceptible to overloads, due to the destructive interaction between the subscriber abandon behavior and the sudden transition to high dial tone delay which occurs just beyond switch capacity. Rapid degradation in throughput characterized by low call completion rates results. This effect has already been described in the literature, e.g., [1]. What distinguishes large switches is the speed at which throughput drops beyond the switch capacity limit.

In comparison with the overload controls developed for analog SPC systems [1-5], the controls we discuss here in relation to distributed digital SPC switches are significantly different. Analog SPC switches historically have a monolithic, single-processor control. This processor has direct control over the peripheral line scanning, and therefore overload controls can be implemented by switch-wide counting mechanisms to regulate the rate of accepted new origins, based on feedback of some direct real-time indicator. In contrast to this, a distributed digital switch does not have direct control over the periphery from the central control. The challenge is to construct controls for all the call processing elements in the control hierarchy which effectively cooperate to produce globally optimal overload response.

In this paper we address the requirements and problems related to the design of a distributed overload control mechanism for a large digital local (Class 5) office. The contribution of this paper is the definition of a successful set of controls, and their performance evaluation through the use of a novel quasi-distributed overload simulator which is modeled on the actual switch architecture. In section 2 the general requirements for a distributed overload control are discussed. In Section 3 some details of the switch operation are given and in Section 4 the design details of the distributed simulator are given. The performance results and discussion are contained in Section 5.
OVERLOAD CONTROL REQUIREMENTS

While there may be instances where it is possible to exploit certain features of a particular system architecture to implement overload controls, there are nonetheless implementation principles which remain invariant. We focus on the principles as they pertain to distributed switching systems.

The primary requirement is for separate overload controls at both the PC and CC levels of call control. Although the nature of the controls at each level is different, they must work together to protect the global switch performance. This goal of cooperativeness is not necessarily easy to achieve, given the speed of response required, and the possibility of only some peripherals being overloaded.

The CC controls must protect the CC real-time, which is the primary central resource. This must be done by limiting the rate at which new work enters the CC and ultimately leads to the discarding of calls. The key requirements for the CC controls is that they consume minimum real-time to execute. Comparison of some alternatives is done in Section 6. Certain switch architectures [6, 7] have an inherent advantage in the priority queue structure supported by the CC scheduler, in that origination work is kept separate and hence directly controllable under overload. In general, however, there is an implicit requirement that once an origination event is processed by the CC, the call should be guaranteed for completion, excepting, of course, in cases of speech path congestion in the network, peripherals or outgoing trunks where treatment is required.

The role of the peripheral overload controls is quite different, and must be complementary to the CC controls. The PC is not a global resource, and for the purposes of our work here we assume that PC real time is not an issue. The requirement for the PC is to dynamically throttle its load to the CC while at the same time preserving the integrity of its lines which may have had calls subject to CC overload control action (discard, or long delay). If a call is discarded or suffers long delay in the CC, the line must not be prevented from making subsequent attempts i.e., the line should not be left "high and dry". The grade of service offered by the switch under overload will not be good, but this is unavoidable regardless of architecture. There is however a key requirement that the subscriber who goes off-hook and waits long enough should eventually receive dial tone.

Looking at the switch as a whole, the requirements for the global controls are to maintain switch throughput during short-term load peaks by smoothing the load without loss of calls (though it is possible that some calls will experience delay), and to maintain continuity of service during sustained overloads when normal grades of service cannot be met. In a distributed switch there is always the possibility of only parts of the system being overloaded, in which case the controls should not be active unless the integrity of central resources is threatened. This means that spare resources should be automatically directed at the overloaded parts of the system up to the point where the global response begins to deteriorate.

SYSTEM DESCRIPTION

3.1 Overview

The hardware and software details of a particular switch architecture are very complex, and many of the details are not relevant to the study of overload performance. Therefore we give here only a partial representation to illustrate the principles of hierarchical control and to allow us to concentrate at the system level on the operational characteristics of the components whose performance have a direct impact on the overload behavior. The system overview concentrating on the real-time functional blocks is depicted in Figure 3.1.

The subscriber loop hardware interface presents the true state of the line. The PC is made up of a group of separate functional processors which communicate via messages. Line state changes and DP signaling information are detected by the loop interface circuitry which is scanned by the Signalling Processor (SP), whose functions are line signaling and supervision. Signalling information and call events are passed to the Main Processor (MP) which maintains the individual call state machines and implements the peripheral call processing logic under overall control from the CC. The CC maintains the central call state machines responsible for all the high-level call processing such as receiver allocation, digit translation and routing. Both the PC(MP) and the CC maintain a set of priority scheduling queues for the execution of call-processing and non-call-processing tasks.

The key elements of the overload controls are in the MP in the periphery and the CC, corresponding to the major elements in the call processing hierarchy. The queuing structure for processing line call originations is depicted in Figure 3.2. The SP periodically scans all idle lines at a nominal rate of once every 200 ms and reports switch-hook state changes to the MP via messages placed in scan report queue buffers in common memory. The MP checks the report queue periodically for incoming messages and when it is able, processes these messages according to the local state machine (seizing whatever resources are necessary) and sends off a message to the CC via the switch messaging system which is not explicitly depicted in Figure 3.2. Messaging delays between the PC and CC are very short.
3.2 Central Control

The central control real-time is the primary limiting system resource governing the attempt capacity. The important features of the CC call processing with respect to overload performance are the following.

Firstly, call progress messages are processed with higher priority than call origination messages. This ensures that calls in set-up and disconnect phases are protected because this work is done in preference to the processing of origination events which represent new work in the system \[6, 7\]. This has been demonstrated both by simulation and by field studies to be an inherently stable arrangement for call processing.

Secondly, to avoid long CC origination queue delays which would otherwise result under overload conditions, there is a CC mechanism for discarding originations corresponding to abandoned calls. This can be done before any execution of the call processing code, based on a delay threshold in the origination queue, at virtually zero real-time cost when the origination is taken off the head of the origination queue. The indirect effect of this control is to limit the length and hence the delays in the CC origination queue. The design goal is to prevent wasting real-time on ineffective subscriber attempts.

3.3 Peripheral Controls

The SP scans each line periodically and off-hook events result in an origination message being written into a buffer on the scan report queue. If no buffer is available, then the state change is ignored and will be picked up again the next time that the line is scanned (if the line is still off-hook). If the line abandons before it is scanned then the attempt is never seen by the system. The SP then waits for the MP response to the origination.

The scan report queue is checked at closely-spaced intervals by the MP for incoming messages. When an origination message is processed, the line state information is updated and a message prepared for sending to the CC. At the same time a timer is started to signal a timeout if no CC response to this message is received. While the timer is active, the MP sends no more origination phase messages to the CC, representing a one-at-a-time flow control of events from lines in the origination phase. The CC response is recognized by the terminal number contained in each call processing message, and if the response is for the terminal which last sent a message to the CC, then the MP timer is reset to allow the MP to send further messages to the CC if there are any.

The peripheral controls are characterized by two parameters, namely the number of scan report buffers, and the timeout associated with the one-at-a-time flow control for origination phase messaging.

There is one more critical component of the peripheral controls which is related to the maintenance of line integrity. Knowing that it is possible for the CC to discard calls under the action of its own overload controls, it is necessary for the PC to recover lines that have had calls discarded so that further attempts can be made by them. In normal operation, the PC depends upon receiving a CC response for every message that it sends and, under overload, such a response may not be forthcoming, intentionally or not. Further, it is necessary to guarantee that the patient subscriber eventually receives dial tone regardless of the degree of overload. To accomplish these dual purposes, the PC has the facility whereby when a timeout for the CC response to an origination phase message occurs, the line state is idled to allow a system-generated reorigination. This aspect of the implementation is critical for preserving line integrity between the PC and CC and is very important in the robustness of the global overload controls.

4. SIMULATOR DESCRIPTION

4.1 Overview

The principal challenge for an overload control simulator is one of time scale. It is necessary to both model the CC scheduling operations which take place on a time scale of microseconds, and it is also necessary to model the subscriber call traffic process which is on the order of an average call holding time (100-200 seconds). For overload performance investigation a call-based simulator is required, because the relationship between successive subcalls of a single call \[8\] needs to be preserved in order to model the process of call abandon and reattempts which depends on the instantaneous switch response, and which is at the heart of the system overload performance. Sub-call simulation is not possible because the causal link between subscriber events and switch delays is broken.

The switch simulator is modular and has a hierarchical structure based directly on the architecture of the switching system. The structure is depicted in Figure 4.1 and has three major modular components for simulating respectively the central control, the line controllers and the trunk controllers. Although the objective is to study line traffic overloads, the trunk traffic must be simulated in order to provide a realistic traffic environment.
4.2 CC Simulator

The CC simulator itself resembles a subcall simulator in that it is driven by messages from the peripheral simulators, representing the various call events. The peripheral simulators for line and trunk peripherals are call-based (at least for the set-up phases). The peripheral simulators generate subscriber events, do the peripheral event processing, and process the CC responses. The subscriber behavior is integrated into the peripheral simulators. In a more clean, but not necessarily more efficient, design it may be desirable to have the subscriber models separate.

The call-based simulator preserves the timing information between subcalls (events) of the same call. The subcall sequence for a line call is depicted in Figure 4.2. Purely from the point of view of simulation run time and storage requirements it is not possible to explicitly simulate the entire length of each call. To do so would involve keeping track of the order of at least twenty thousand simultaneously active calls, most of which would be in the conversation phase. It is much simpler to simulate the disconnect subcalls as a stream of independent events, governed only by a current count of the number of calls in the conversation phase. This reduces the simulation warm-up time and storage requirements by an order of magnitude. Subcall sequences are only preserved during the call set-up phases.

The coarseness of each of these clocks represents the relative accuracy required to acceptably model the traffic processes in each of the simulation modules. The CC subcall simulator therefore needs to have only one more event type added to its event list, namely a "peripheral processing" event which results in the line simulator being invoked. Periodically, this event also results in the trunk and disconnect simulator being invoked. This approach limits the simulation horizon of the CC simulator, and the number of future peripheral events to one, excluding incoming call processing messages. Huge amounts of processing time are saved this way, because it avoids the need to sort long event lists repeatedly.

This finite simulation horizon method has some implications for the peripheral call processing. It is not acceptable to have peripheral events generated in batches at the peripheral clock epochs, resulting in a periodic batch input to the CC simulator. Rather, at a peripheral clock epoch, the events generated are spread over the succeeding clock interval according to a Poisson process. This is easy to do because the order statistics of a uniformly-distributed random variable are exponentially distributed.

Each call in the set-up phase is characterized by a unique sequence number, and a list of call attributes which define the state of the call. The principal attributes are:

- call Type (line-to-line, etc.)
- call disposition (completed, uncompleted, treatment, etc.)
- DP/DT (lines) or DP/MF (trunks)
- phase of call (origination, dialing, etc.)

In addition the peripheral simulator maintains, via the call attribute vectors, three state machines for each call while in the origination
phase. These are respectively the actual line state (on-hook, off-hook), the state of the origination event, and the state of the abandon event. Since the call processing is distributed, each of the CC and PC processors may have a different view of the line state, and the actual line state may be different again. Each active call is checked at the epochs of the peripheral clock to test for call progressions (e.g., abandon), depending on its current state. If the state is the change, the peripheral state machines are updated and (possibly) a message is generated to be sent to the CC and is allocated a time in the next clock interval using the method described previously.

4.4 Subscriber Behaviour Model

Subscriber response to switch delay is characterized by abandoned calls and successive reattempts. When the dial tone delay is protracted there is a marked rise in the number of calls which abandon before dial tone, and calls where dialing commences before dial tone, resulting in a partial-dial abandon [1]. From the CC point of view, a partial-dial abandon due to dial tone delay looks the same as an abandon before dial tone, since in each case it receives the same messages, namely an origination followed by an abandon. (A digits message only goes to the CC when a full set has been collected.) The only difference between the two types of abandon is that the partial-dial abandon occurs a little later than the false-start abandon.

The subscriber behavior model for line calls was based on the following parameters:

- Mean time to abandon due to DTD
- Probability of reattempt after abandon
- Mean wait before reattempt

The characterization for incoming (trunk) calls was based on the following parameters:

- Mean wait for Start-to-Dial signal
- Probability of reattempt
  - 1.0 First reattempt
  - 0.0 Subsequent reattempts

These numbers are based in part on results reported in [7, 9, 10].

When considering subscriber behavior, recall that there is a residual level of abandoned calls even when the switch dial tone delay is so small as to be unnoticeable. These abandons are due to subscriber action independent of switch response and are modeled differently. In the framework of the peripheral simulator presented here, this residual level of abandoned calls was modeled by a separate class of call disposition (uncompleted) in the call attribute vectors.

5. RESULTS AND DISCUSSION

The results shown in this section are some of the final results coming from the investigation of a number of different alternatives for the switch overload controls. The objective here is not only to demonstrate the effectiveness of the final design, via the simulation vehicle outlined in Section 4, but also to look at the reasons for its success as compared to other design alternatives.

The overload performance for a large switch without overload control is qualitatively presented in [1] and we do not reproduce it here, except to note the almost vertical drop in throughput when the point of switch capacity is exceeded. This behavior is characteristic of any large switch.

Turning now to the introduction of overload controls, the major results are depicted in Figure 5.1, where normalized switch throughput is plotted as a function of the normalized switch offered load (nominal). The normalized offered load of 1.0 corresponds to the switch CC real-time attempt capacity, which depends primarily on the call mix. (For the overload studies discussed here a typical Class 5 office call mix was used, but the results are in no way peculiar to the call mix used.) The parameter for the family of throughput curves given in Fig. 5.1 is the number of message buffers in the PC scan report queue (recall Fig. 3.2). When the repeat queue is allowed to be long, say 15, then the
overload performance is very poor and is indistinguishable from the switch performance without any overload controls at all. As the number of report buffers is reduced, so the overload performance improves, until with a single buffer we reach near-optimal throughput performance which is stable and sustainable regardless of the degree of overload.

The scan report queue functions solely as a buffer between the SP idle-line scanning and the MP peripheral call processing. If delays in this queue are large then the probability of call abandon before the origination can be processed by the MP is very high. This means that either an abandon message soon follows, or the origination is processed by the CC only to find the line state on-hook when the CC response reaches the MP. This results in much wasted time in the MP and the CC. Under overload the report queue is almost always full, and is emptied FIFO by the MP at a rate which is largely determined by the MP timeout, since the CC response will be slow. For the sake of illustration, if the report queue is 15 long, and the timeout is 1 second, an origination has a wait of the order of 15 seconds to be processed and sent to the CC, and in this time the call has almost surely been abandoned. By choosing the report queue to be the minimal size of one, the PC delays are kept short so that when an origination is processed it has a good chance of getting dial tone before abandon.

This does not mean that subscribers get short dial tone delay, since the effect of the control is to keep the excess call load outside the system at the line scan level. Many attempts will not get dial tone before abandon, but this is unavoidable. Although dial tone delay is long with overload controls, it is still shorter than the delay without controls, as can be seen in Figure 5.2.

The size of the report queue alone is not sufficient to give rise to the system stability under overload. The stability results from the peripheral reorigination facility. When the CC slows down under a CC overload, the MPs begin to time out waiting for responses to origination phase messages, leading to the generation of system reoriginations. These reoriginations have the effect of tying up the MP–CC messaging via the MP timeout, and throttling the load to the CC. In other words, if an origination ties up the MP for one second, a single reorigination means that the MP is tied up a total of two seconds for the same call, thus reducing the effective MP attempt rate. Just enough reoriginations are generated to balance the CC capacity, constituting an automatic negative feedback control between CC and PCs triggered by the CC origination delay. In this way the switch throughput is maintained independently of the magnitude of the overload.

The secondary PC/MP parameter is the value of the timeout for CC response to an origination phase message. The impact of the timeout value is depicted in Figure 5.3, where the switch throughput is plotted against MP timeout, for a fixed nominal normalized offered load of 1.15 (15% overload). Throughput is plotted both with and without overload controls. Without controls there is a point at which the timeout alone is sufficient to throttle the load enough to de-load

Figure 5.2 Probability Dial Tone Delay > 3 Sec. vs Offered Load

Figure 5.3 Throughput vs PC Timeout
the CC and restore throughput. The location of this point depends on the switch configuration and call mix. Below the critical timeout value, CC delays are very high and the call completion rate very low. With the overload controls in place the picture changes dramatically, and we see that the throughput does not vary much with timeout value. This is because the negative-feedback overload control is self-tuning and settles to a near-ideal operating point. In short, when the timeout is chosen to be short, more system reoriginations are generated to throttle the NP. When the timeout is long, fewer reoriginations are generated to achieve virtually the same operating point.

There is an important conclusion to be drawn from the above arguments, and it is that regardless of the number of PCs terminated on the switch and the traffic offered by each PC, regardless of the call mix, regardless of the office real-time capacity, the CC throughput is stable under overload. This is true because the controls are inherently adaptive. It means that there are no overload control parameters that need to be "tuned" for each switch individually, which can be seen not to be the case for overload control mechanisms in some other switching systems.

A side benefit of the controls is that by protecting the CC from line overloads, the switch response times for incoming trunk calls can be maintained, given that the incoming load itself is not presenting an overload to the switch. Trunk overload controls are a separate issue and we do not discuss them further here.

There has been recent work reported on the use of LIFO service strategies for peripheral origination queues as an overload control measure [1, 11]. The principle behind this idea is that most overload should be reported first because it has the best chance of success. This strategy was tried in the overload simulator for the PC report queue, but was not successful. While the throughput response was significantly better than that without controls, the throughput did not remain stable under overload and dropped off with increasing load. In any case, for the successfully implemented controls with a single report buffer, the queue service discipline (FIFO or LIFO) becomes irrelevant.

6. CONCLUSIONS

The system features and simulation testing of the overload performance for a distributed, hierarchical switching system have been discussed. The key features which lead to good overload performance have been identified as:

- Priority processing of call progress work in the Central Control
- The strict limitation on scan report queue length in the line peripherals
- The automatic system reorigination mechanism

These factors together lead to a set of cooperative overload controls whose stability is independent of office configuration and call mix. The overload controls have already been introduced into offices in the field, and although no office has yet been subject to a full-scale overload, the controls have demonstrated positive field behaviour under some limited abnormal traffic conditions.

The switch overload simulator which was used as the testing and validation tool was based on a novel quasi-distributed architecture for maximum run-time efficiency. The major features of the simulator were:

- Separate CC and PC clocks reflecting the different time scales required for central and peripheral processing
- The finite-horizon simulation used for the peripheral controllers to limit the CC simulation event list to a small size.

The overload simulator is being evolved to study a wider class of system performance problems. In particular it is being extended to look at trunk traffic overloads, the overload performance of new peripherals being designed for the switch, and the switch response to abnormal maintenance activity in the face of carrier failure.

REFERENCES


5.1B-4-7