Considering Control Theory in Resource Management Scenarios

A Djafari Marbini and Lionel Sacks

Department of Electronic and electrical Engineering, University College London

Abstract: Control theory has been used for many years to analyse dynamical systems and design controllers for them. Computer networks are an example of dynamic systems. Currently resource management and scheduling in networks have been using open loop mechanisms, completely ignoring the feedback generated by the systems. Feedback control can be used to meet QoS requirements. This paper will be looking at the impact of using a closed loop control system, in addition to the usual scheduling mechanisms.

1 Introduction.

The rapid growth of the communication technologies (e.g. Internet, mobile, telephony) has increased network complexity, while connecting millions of computers. This has given rise to heterogeneity both at a moment in time and through time. Feedback is generated due to the relationships between users and users, and users and networks. All this is the making of an extremely complex and dynamic system. The complex nature of traffic makes it very difficult for it to be modelled. This is the direct consequence of heterogeneity both in the type of services and diversity in networking technologies and different policies administered by the networks. And yet modelling traffic, and networks in general is the essence of understanding network characteristics, and effects of proposed changes to network operations [1].

It would be interesting and quite possibly advantageous to use traditional control theory mechanisms to control this dynamic and complex system

2. The problem.

There are different types of approaches in dealing with resource management. The author is interested in the use of control theory and developing a controller for use in resource allocation scenarios. The controller will be used to allocate resources in environments that traffic could exhibit complex behaviour. The controller should aim to control the behaviour of a complex traffic, but should not exhibit complexity itself.

A PID controller is an example of a controller which has been used successfully in dynamic systems. Its success in other types of dynamic systems can be used as a starting point to building a controller used in scheduling mechanisms. In control theory terms the scheduling system is dynamic, i.e. the output of the system not only depends on the current tasks, but also on tasks submitted previously that remain in the system. An important feature of the PID controller is that, it does not need a precise analytical model of the system that is being controlled [1]. This makes it ideal for use in scheduling. A mapping of control theory methodology to scheduling is needed. In a typical feedback control problem the following issues are considered:

1. Control Objective - In the case of the scheduling mechanisms the control objective is to keep the task miss ratio to a minimum, while achieving high utilization and throughput. A miss ratio of 0% is not ideal as it would severely under utilize a system. These are the typical requirements of scheduling in soft real-time systems. The miss ratio can be used as the controlled variables.

2. Manipulated Variable – The total CPU utilization requested by all accepted tasks in the system can be the manipulated variable. The total CPU required can be manipulated by accepting or rejecting more tasks.

3. Enabling Technologies – Actuators used to manipulate the CPU utilization.

3. Traffic Types

To model a scheduling system with feedback control mechanisms, initially one needs to identify the tasks arrival rate and the workload statistics. For example in the model that is presented in section 4, which is a simplified version of the model presented by C. Lu [2], the tasks arrival rate is periodic, and the workload in this case the task CPU requirements, follow uniform distribution.

Work load						
	Distribution	Periodic	Uniform	Gaussian	Exponential	Pareto
Arrival Rate	Periodic		*			
	Uniform					
	Exponential					
	Gaussian					
	Pareto					

Figure 1. Different statistical combinations between arrival rate and workload

When a scheduling system is then developed real statistic e.g. multicast data traffic needs to be tested on the system to see if the scheduling system will hold up. Different combination of distribution needs to be tested, so that appropriate type of scheduling system can be adapted for a particular type or of traffic. As a starting point in the quest for building a controller used in resource management scenarios one of the simplest statistical combination between arrival rate and workload has been chosen in this paper and similar results to that of model presented by C. Lu has been reproduced.

4. Model

The model used in this paper is a single server receiving tasks from 20 different sources. Each source has a period, which equals to the task deadline. The workload of the tasks is described in terms of execution time, ET_i. Each task has a worst case execution time ($WCET_i$) and a best case execution time distribution ($BCET_i$). Uniform distributions between particular values are used to generate $BCET_i$, $WCET_i$, and period. An estimated execution time (EET_i) and an average execution time (AET_i) is given by the following equations.

$$EET_i = (WCET_i + BCET_i) \times 0.5 \qquad (1)$$

$$AET_i = EET_i \times etf \qquad (2)$$

etf is the execution time factor, which can be tuned to estimate the accuracy of the estimation. The larger the etf the more pessimistic the estimation will be. In this particular experiment the etf was in the range of 0.4 t0 1.6. The actual average execution time gets generated from a uniform variable in the interval [AET_i, $WCET_i$] or [$BCET_i$, AET_i] depending on the outcome of a random Bernoulli trail with probability (AET_i - $BCET_i$)/($WCET_i$ - $BCET_i$).

The model was built in Matlab as illustrated in figure 2. The tasks arrive at the beginning of each experiment to overload the system. The miss ratio of the accepted tasks was periodically monitored and sent to PID. The PID calculates the control signal using equation 3. The control signal is then fed to Admission Controller (AC) and on the basis of the control signal; AC decided whether or not to accept any more incoming tasks.

$$\Delta CPU(t) = C_p error(t) + C_I \sum_{IW} error(t) + C_D \frac{error(t) - error(t - DW)}{DW} \quad (3)$$

Where, IW is the time units that will considered in the integral term.
DW is the time units that will considered in the derivative term.
C_p, C_I, and C_D are the PID parameters.

$$error(t) = SetMissRatio - MissRatio(t)$$

Equation 3 is an approximation of the PID equation[1].

An earliest deadline first (EDF) scheduling mechanism is adopted in the admitted tasks. So the tasks are served in order of the deadline. EDF is an example of dynamic scheduling.

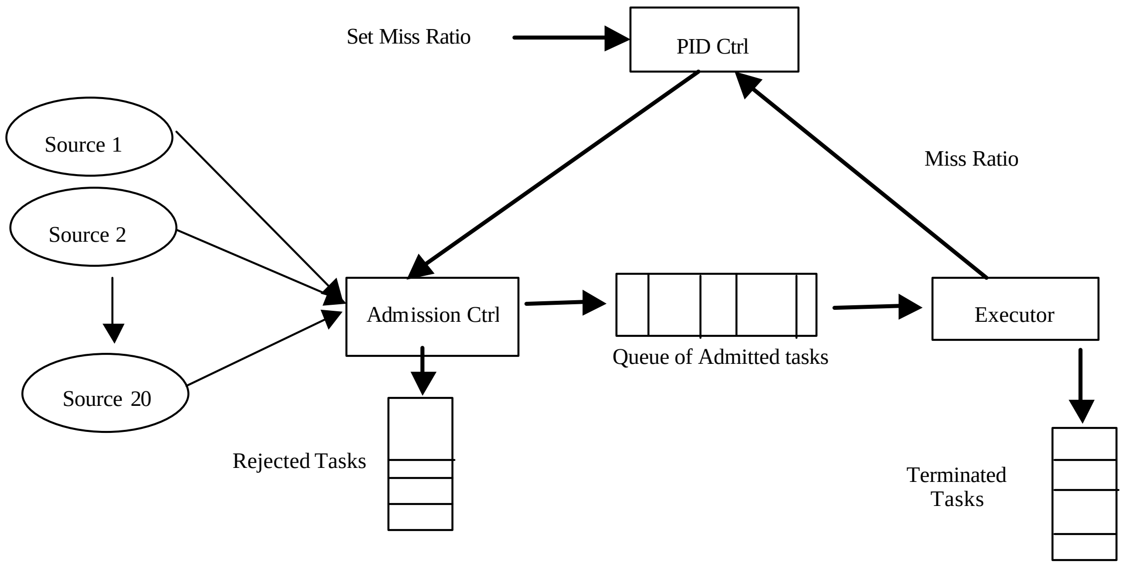


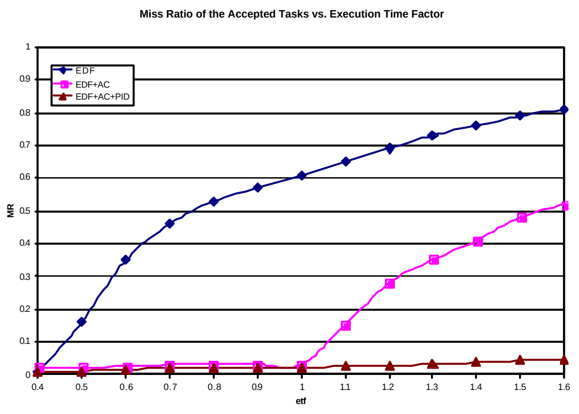
Figure 2. The scheduling model.

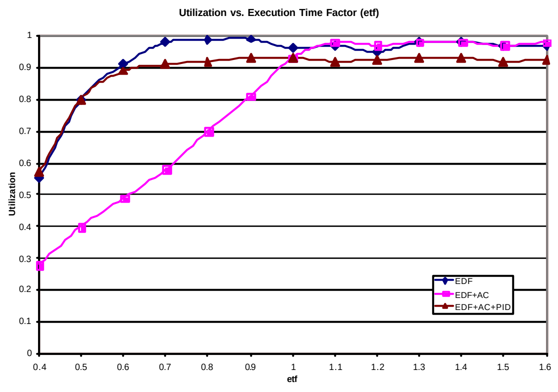
5. Results

The results presented in figures 3 and 4 are similar to those in [2]. However in their model each task had 2 different service levels. Each service level had different execution time. An extra controller called the Service Level Controller (SLC) was used to choose between the different service levels of a task and this in addition to admission controlle r was used to utilise CPU.

In figure 3 one can see that the miss ratio of the admitted tasks, when only an EDF scheduling mechanism is used increases with etf. When EDF+AC is operating, the miss ratio expectedly improves, but it drastically increases again when etf >1, i.e. when average execution time is larger than its estimation. This suggests that this type of scheduling mechanism can only be useful when a more accurate estimation of the tasks execution times is available to the system. When PID is adopted the system performance drastically improves with less than 10% miss ratio, and this does not change with the accuracy of the execution time estimate.

[1] PID control formula: $control(t) = C_P Error(t) + C_I \int Error(t)dt + C_D \frac{dError(t)}{dt}$





As expected since the in EDF scheduling, all the tasks are admitted the utilisation is very high in compression to when EDF and AC mechanism are running concurrently. The utilisation of EDF+AC mechanism only improves when average execution time is larger than its estimation (etf > 1). The EDF+AC+PID keeps to a high level of utilisation relative to the other two mechanisms.

6. Conclusion and Future Work

The use of PID controller drastically decreases the miss ratio of the accepted tasks; additionally it presents a high CPU utilisation. Consequently the QoS has been improved. This gives an initial positive feedback, for the case of using PID controllers or in fact other types of controllers in scheduling mechanisms. However, the type of traffic statistics used to produce these results is infrequent and is of the simplest case scenarios. Therefore it is important to verify the findings with other traffic statistical combinations shown in figure1. The author is planning to use tasks with Gaussian arrival distribution to test the PID controller.

References.

[1]Sally Floyd, Vern Paxson. Difficulties in Simulating the Internet. IEEE/ACM Transactions on Networking, February 2001.

[2] Chenyang Lu, John A. Stankovic, Gang Tao, and Sang H. Son, Design and Evaluation of a Feedback Control EDF Scheduling Algorithm, 20th IEEE Real-Time Systems Symposium (RTSS 1999), Phoenix, AZ, December 1999

[2]Jacqeline Wilkie, Michael Johnson, and Reza Katebi. Control engineering – an introductory course. ISBN 0-333-77129-X