A New Two-Phase Approach for Petri Net Based Modeling of Scheduling Problems

Reggie Davidrajuh
University of Stavanger
Stavanger, Norway
reggie.davidrajuh@uis.no

Abstract. This paper presents a new two-phase approach for Petri Net based modeling of scheduling problems. Though Petri Nets have been used as heuristic approach for modeling scheduling problems, literature study reveals two major difficulties: 1) the large size of Petri Net models, and 2) the inability to differentiate workstations. In this paper, these two difficulties are avoided by a two-phase approach known as Activity-Oriented Petri Nets (AOPN). General Purpose Petri Net Simulator (GPenSIM) is a new Petri Net simulator that implements AOPN on MATLAB platform. This paper introduces AOPN and GPenSIM in a tutorial style, working through an example on job scheduling in grid computing. This example shows the usability of the AOPN approach for the modeling of scheduling problems and the easiness of GPenSIM for coding and simulation.

Keywords: Scheduling problems, Activity-Oriented Petri Net (AOPN), GPenSIM, Petri Nets, grid computing.

1 Introduction

Scheduling is a process to manage time and costs effectively when a sequence of tasks are to be performed using a set of workstations; the sequence of tasks have a precedence order, meaning some tasks must have been completed before a particular task can start [1].

Scheduling is an important issue in all branches of engineering. In production engineering, scheduling can minimize the production time and costs, by optimal use of the resources such as man and machines; assembly line balancing problems (ALBP) is a special class of scheduling problems in production engineering [2].

Literature study reveals a large number of works that use Petri Nets for scheduling problems; literature also reveals the huge size of resulting Petri Net models and the difficulties in differentiating workstations as two main problems associated with the use of Petri Nets [3; 4; 5]. This problem is especially true when modeling resource-scheduling problems, where a large number of resources is usually involved [5]. However, the two problems are solved using advanced techniques: for example, Jeng and DiCesare [6] uses control nets and Wu and Zhou [7] uses Resource-oriented Petri Nets to overcome these difficulties.
The aim of this paper is to introduce Activity-Oriented Petri Nets (AOPN) as a simple yet effective technique to model scheduling problems; this paper is written in a tutorial style to achieve this aim, taking job scheduling in grid computing as an example. As the goal of this paper is to show how simple and easy it is to use AOPN for scheduling problems, discussion is kept to a minimum about the various advanced techniques and complex approaches for scheduling problems, inclusive advanced Petri Nets based approaches.

Fig. 1. Grid computing environment [8]

2 A Scheduling Problem in Grid Computing

This section presents a scheduling problem in grid computing environment.

Figure 1 shows a grid where the broker plays a central role in accepting jobs (multi-task jobs comprised of multiple tasks) from the local clients and delegating these tasks to the local workers [8]. Figure 1 also shows that the broker is a node in the ring of many brokers; thus, the broker may also accept tasks from other brokers and also route local tasks to external workers via the brokers in the ring.

Let us assume that the broker accepts a job from a local client; the job consists of 19 tasks as shown in Figure 2. The tasks are shown as nodes with labels $n_1$ to $n_{19}$. The diagram shown in Figure 2 is known as the precedence graph, as it shows the number of tasks involved in the job, and the sequential relationship between the tasks.

Table 1 shows the types of the tasks, based on the required software and hardware to perform the task (requirements). Table 2 lists the tasks $n_1$ to $n_{19}$, along with their type and the expected processing time of the respective task, if it was to be run on a standard desktop system. The processing times are given as stochastic timing, with Gaussian (normal) distribution with mean and standard deviation pair. Table 3 shows the types of the eight local workers based on their software and hardware configuration. The broker can only assign a task with specific requirements to a worker with the matching configuration; e.g. task $n_1$ can be assigned to workers $w_1$ to $w_5$, whereas task $n_6$ cannot be assigned to any local worker as the local workers are not equipped with enough main memory (minimum 40GB); thus, $n_6$ must be routed to an external worker. Worker $w_8$ cannot be assigned to perform any task of this job, as it possesses too little memory.