# SPA for Performance Modelling

Lab 1

9th April 2013

## 1 Upgrading a PC LAN

The problem we consider is to determine the mean waiting time for data packets at a PC connected to a local area network, operating as a token ring. Such a network uses a transmission medium that supports no more than one transmission at any given time. To resolve conflicts, a token is passed round the network from one node to another in round robin order. A node has control of the medium, i.e. it can transmit, only whilst it holds the token. In a PC LAN every PC corresponds to a node on the network. Other nodes on the network might be peripheral devices such as printers or faxes but for the purposes of this study we make no distinction and assume that all nodes are PCs. There are currently four PCs (or similar devices) connected to the LAN in a small office, but the company has recently recruited two new employees, each of whom will have a PC. Our task is to find out how the delay experienced by data packets at each PC will be affected if another two PCs are added.

Each PC can only store one data packet waiting for transmission at a time, so at each visit of the token there is either one packet waiting or no packet waiting. The average rate at which each PC generates data packets for transmission is known to be $\lambda$ (see Figure 1. We also know the mean duration, $d$, of a data packet transmission, and the mean time, $m$, taken for the token to pass from one PC to the next. It is assumed that if another data packet is generated, whilst the PC is transmitting, this second data packet must wait for the next visit of the token before it can be transmitted. In other words, each PC can transmit at most one data packet per visit of the token.

### 1.1 Modelling the system

The first stage in developing a model of the system in PEPA is to determine the components of the system and the actions which they can undertake.

Here it seems clear that one type of component should be used to represent the PCs. The components representing the four/six PCs with have essentially the same behaviour. But since token visits the nodes in order we will need to distinguish the components.

We will need another component to represent the medium. The medium can be represented solely by the token.

The second stage is to decide on the **actions** that we want/need for each component.

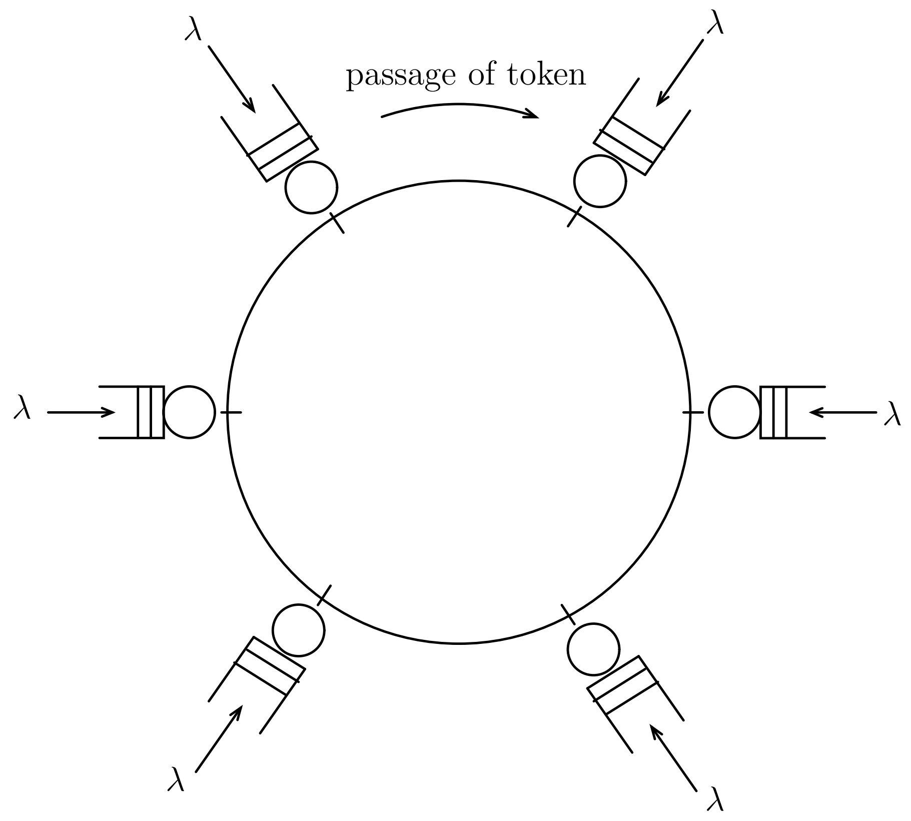

Figure 1: A six-node token ring network

It is always a good idea to test your reasoning at an early stage. In this case the most sensible approach is to consider a very simple version of the problem, for example, a network with just two PCs and use the one-step navigator in the Eclipse Plug-in Tool.

Construct PEPA models of the system consisting of a LAN with 4 nodes and 6 nodes and compare their performance. You will find ready constructed models on the webpage if you get stuck but it will be more informative to try to do it for yourself.

The following values should assigned to parameters of the model in both cases (the unit of time is assumed to be milliseconds),

$$\lambda = 0.01 \qquad d = 10 \qquad \mu = 0.1 \qquad m = 1.0 \qquad \omega = 1.0$$

Use the PEPA Eclipse Plug-in to calculate the average *waiting time of data packets* at any PC in the network. Since all the PCs are statistically identical, we can arbitrary choose one as representative.

## 2 Modelling ACPI with PEPA

An organisation running a large number of desktop computers is committed to achieving continued reduction of its environmental impact. To this end, it has implemented software to automatically put unused computers into a sleep state according to the Advanced Configuration and Power Interface (ACPI) specification. The ACPI specification describes six sleep states of a computer system (S0, ..., S5) in terms of a high-level view of the hardware consisting of a CPU, RAM, a hard disk, and a power supply; and the software consisting of the operating system.

| **State** | **Description** | **Probability** |
|---|---|---|
| S0/Working | System is on. The CPU is fully up and running; power conservation is on a per-device basis. | |
| S1 Sleep | System appears off. The CPU is stopped; RAM is refreshed; the system is running in a low power mode. | From S0 the system goes to S1 30% of the time |
| S2 Sleep | System appears off. The CPU has no power; RAM is refreshed; the system is in a lower power mode than S1. | From S0 the system goes to S2 30% of the time |
| S3 Sleep (Standby) | System appears off. The CPU has no power; RAM is in slow refresh; the power supply is in a reduced power mode. This mode is also referred to as 'Save To RAM'. | From S0 the system goes to S3 20% of the time |
| S4 Hibernate | System appears off. The hardware is completely off, but system memory has been saved as a temporary file onto the hard disk. This mode is also referred to as 'Save To Disk'. | From S0 the system goes to S4 10% of the time |
| S5/Off | System is off. The hardware is completely off, the operating system has shut down; nothing has been saved. Requires a complete reboot to return to the Working state. | From S0 the system goes to S5 10% of the time |

Models of the CPU, power supply, RAM, Disk and operating system have been developed as shown in Figure 2. You will find these process definitions in a file `ACPI.pepa` on the course webpage (`http://homepages.inf.ed.ac.uk/jeh/venice2013/ACPI.pepa`). The activities $s1\_sleep$, …, $s5\_sleep$ put the computer into its sleep states. The activities $s1\_wake$, …, $s5\_wake$ cause it to wake up.

Develop a PEPA component which can ensure that the components which make up the computer system can be put into the above sleep states and successfully awoken from the sleep states, according to the probability distribution given. Assume that the power saving action occurs after 100 minutes and on average machines are woken again after 50 minutes.

$$
\begin{aligned}
CPU\_running &\stackrel{\mathit{def}}{=} (stop\_cpu, r_{c_1}).CPU\_stopped \\
&\quad + (power\_down\_cpu, r_{c_2}).CPU\_powered\_down \\
CPU\_stopped &\stackrel{\mathit{def}}{=} (start\_cpu, r_{c_3}).CPU\_running \\
CPU\_powered\_down &\stackrel{\mathit{def}}{=} (power\_up\_cpu, r_{c_4}).CPU\_running \\
\\
PowerSupply\_on &\stackrel{\mathit{def}}{=} (low\_power\_mode, r_{p_1}).PowerSupply\_low \\
&\quad + (lower\_power\_mode, r_{p_2}).PowerSupply\_lower \\
PowerSupply\_low &\stackrel{\mathit{def}}{=} (full\_power\_mode, r_{p_3}).PowerSupply\_on \\
PowerSupply\_lower &\stackrel{\mathit{def}}{=} (full\_power\_mode, r_{p_4}).PowerSupply\_on \\
\\
RAM\_on &\stackrel{\mathit{def}}{=} (slow\_refresh\_ram, r_{r_1}).RAM\_slow \\
&\quad + (power\_down\_ram, r_{r_2}).RAM\_off \\
RAM\_slow &\stackrel{\mathit{def}}{=} (fast\_refresh\_ram, r_{r_3}).RAM\_on \\
RAM\_off &\stackrel{\mathit{def}}{=} (power\_up\_ram, r_{r_4}).RAM\_on \\
\\
Disk\_on &\stackrel{\mathit{def}}{=} (save\_to\_disk, r_{d_1}).Disk\_on \\
&\quad + (restore\_from\_disk, r_{d_2}).Disk\_on \\
&\quad + (power\_down\_disk, r_{d_3}).Disk\_off \\
Disk\_off &\stackrel{\mathit{def}}{=} (power\_up\_disk, r_{d_4}).Disk\_on \\
\\
OS\_on &\stackrel{\mathit{def}}{=} (shut\_down\_os, r_{o_1}).OS\_off \\
OS\_off &\stackrel{\mathit{def}}{=} (reboot, r_{o_2}).OS\_on
\end{aligned}
$$

Figure 2: PEPA component definitions for ACPI