Lecture 27
Chapter # 8
PETRI NET-BASED PERFORMANCE MODELING

INTRODUCTION

• Petri nets (PNs) provide a *graphical tool* as well as a *notational method* for the formal specification of systems.
Actually, PNs have a specific place in computer systems performance assessment ranging somewhere between the analytic queueing theory & the computer simulation.

• The systems PNs model tends to include more than simply an arrival rate and a service rate.

• Two important aspects in modeling the performance of computer systems
(1) contention for resources and (2) synchronization between various concurrent activities.
• The former aspect can be adequately represented by Queuing Network Modeling, but the later cannot.
Here using petri nets models, we can actually try to cover both these important aspects. Also, the petri nets are used for modeling both hardware & software systems. & they are used for designing & modeling systems in which concurrent processing considerations & dynamic dependency occurs. It is also commonly named as 'graphical come mathematical model'.

• Petri nets have long been used for modeling synchronization behavior of concurrent systems but not as completely as simulations.
• They act more like simulations in that they allow the modeler to examine single entities within the system, as well as their movement and effect on the state of entire system.
• Petri nets were first introduced in 1966 to describe concurrent systems.
• This initial introduction was followed by continual improvements
for example,
  • the addition of timing to transitions,
  • priority to transitions,
  • types to tokens,
  • and colors depicting conditions on places and tokens.

Initially petri nets were quite simple, they were simply used to describe concurrent systems but with the passage of time & with the continuous improvement & advancements these points were added to the basic petri nets.

GRAPHICAL REPRESENTATION

Petri Nets have been followed by the developments of software tools to aid in the modeling & analysis of systems using the petri net concept. So, we can use Petri Nets to represent computer systems & they provide a mean to actually abstract the basic elements of the system & its information flow using only four fundamental concepts.

- Petri nets are usually represented graphically according to the following conventions:
- Places are represented by circles,
- transitions by bars,
- input function by arcs directed from places to transitions,
- output function by arcs directed from transitions to places, and
- markings by small filled circles called tokens.

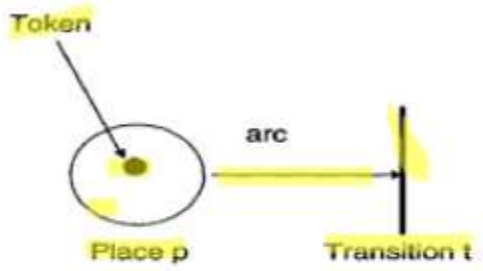

Fig 1: Basic Petri net components

Places are used to represent the possible systems components & their state for example a Disk drive could be represented using a place as for a program or other resource.
& transitions are used to describe an event that may result in a different system state. For example, the action of reading an item from a disk drive or the action of writing an item to a disk drive could be modeled as a separate transition.
& the arcs actually represent the relationship that exits between the transitions & places you can think of the arc as let's suppose providing a path for the activation of a transition.
& tokens are used to define the state of the Petri Networks. So, token in the basic petri nets models are the non-descriptive markers & they are stored in places & are simply used in defining the petri nets marking. & the marking of a petri network place by the placement of a token can actually be viewed as the statement of the condition of the place.

Example

Fig 2 illustrates a simple Petri net with only one place and one transition.

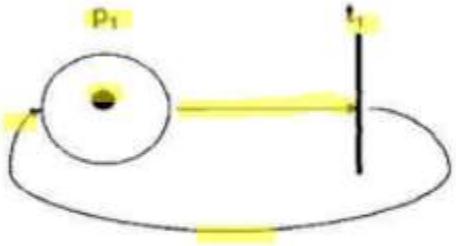

Fig 2: Example perpetual motion Petri net

- The former arc is input arc, while the later is an output arc.
- The placement of a token represents the active marking of the Petri net state.

If any place got a token it means that particular petri network work has active.

- The Petri net shown in Fig 2 represents a net that will continue to cycle forever.

There's no stopping in this particular network.

REPRESENTATION USING SET NOTATION

Using this notation, we can describe a PN as a 5-tuple, M= (P, T, I, O, MP)
• where P is the set of places, P = {P_1, P_2, ..., P_n} drawn as circles (n circles)
• T is the set of transitions, T= {t_1, t_2, ..., t_m} drawn as bars (m bars)
• I □ P x T is the input function, drawn as arcs from places to transitions
• O □ T x P is the output function, and drawn as arcs from transitions to places
• MP □ P x Z^+ is a marking of the net; tokens
• Here, Z^+ denotes the set of all nonnegative integers, it means that we can have any number of tokens indifferent places.
• I represent a bag of sets of input functions for all transitions,
• I = {I_{t1}, I_{t2}, ..., I_{tm}}, mapping places to transitions,
• O represents a bag of sets of output functions for all transitions,
• O = {O_{t1}, O_{t2}, ..., O_{tm}}, mapping transitions to places; and
• MP represents the marking of places with tokens.
• The initial marking is referred to as MP_o.
• MP_o is represented as an ordered tuple of magnitude n, where n represents the number of places in our Petri net.
• Each place will have either no tokens or some integer number of tokens.

For example, the Petri net graph depicted in Fig 3 can be described using the previous notation as:

• M = (P, T, I, O, MP)
• P = {p1, p2, p3, p4, p5}
• T = {t1, t2, t3, t4}
• I(t1) = {p1}
• I(t2) = {p2, p3, p5}
• I(t3) = {p3}
• I(t4) = {p4}
• O(t1) = {p2, p3, p5}
• O(t2) = {p5}
• O(t3) = {p4}
• O(t4) = {p2, p3}

We don't have any token in any of the places.

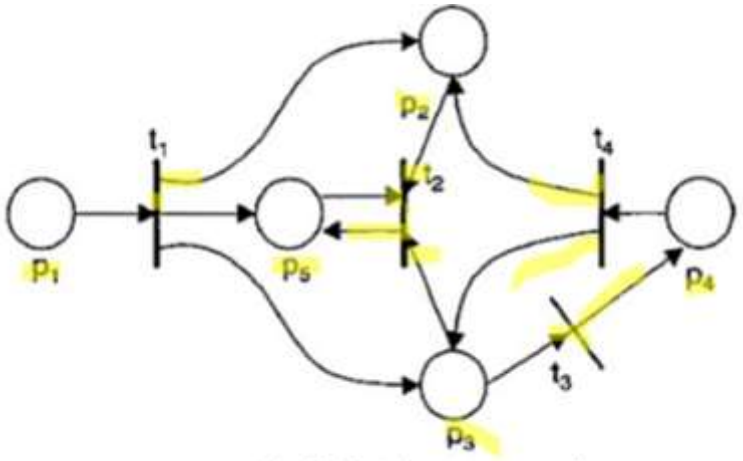

Fig 3: Petri net example

Dynamic Behavior of Petri-Nets

• The dynamic behavior of a Petri net is described by the sequence of transition firings.
• The firing rules are as follows: Let *t* be a transition with incoming arcs from places ip_1, ..., ip_K and outgoing arcs to places op_1, ..., op_L for some K, L > 1.
• Then t can fire if and only if each of the K input places contains at least one token.
• As a result of firing, one token will be removed from each of the K input places, and one token will be added to each one of L output places.

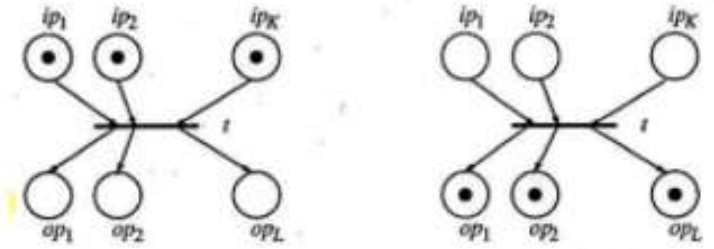

Fig 4: A Petri net (a) before firing and (b) after firing

DUAL OF A PETRI-NET

• The *dual* of a Petri net: transitions changed to places and places changed to transitions.

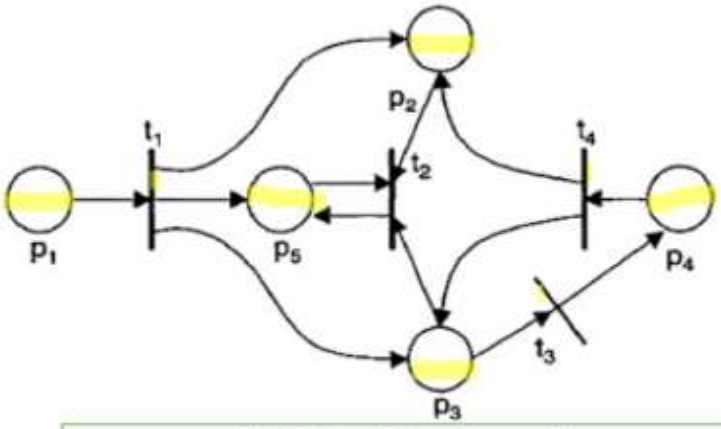

Fig 3: Petri Net Example

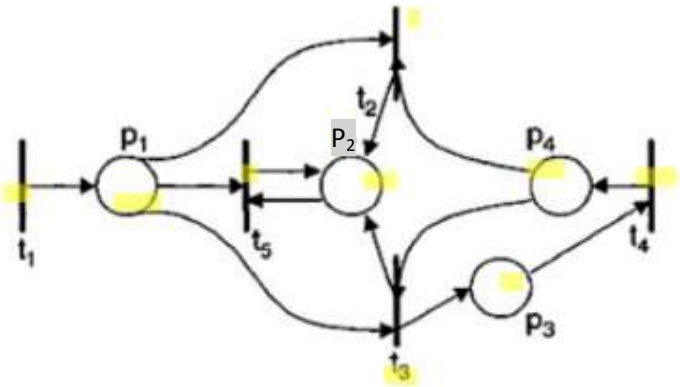

Fig 5: Dual of Petri Net from Fig 3

Remember that the input & output functions will remain same while making dual of the petri networks we will not be changing positions of the arcs.