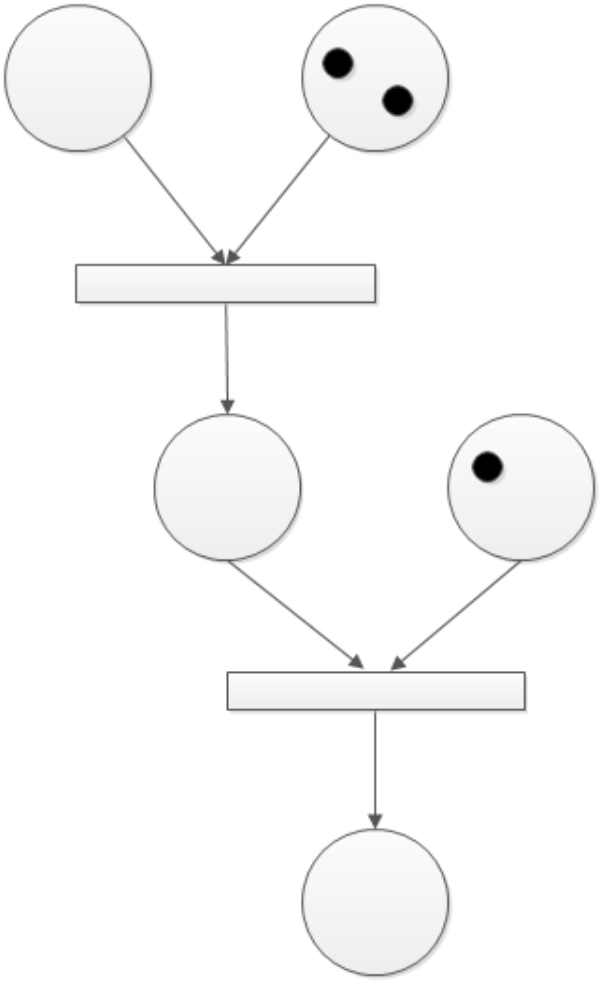
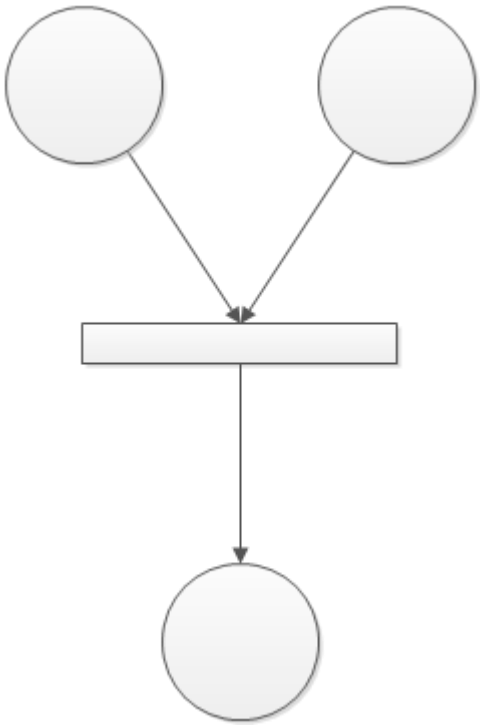
Petri Net

Presentation, discussion and application *Fabio Angeletti*



Introduction

A Petri net is commonly known as a Place/Transition (or briefly P/T) net and it's a mathematical modeling language used often to describe distributed systems. A directed bipartite graph depicts a Petri net in which places (aka conditions) are represented with circles, instead bars are used for represent transitions (aka events that can happen). The directed arcs depict which transitions occurs are pre/post conditions for which places (also the vice versa applies). Petri nets are usually defined via a graphical notation (for stepwise processes) that is really simple to understand and write, however powerful and precise. On the other hand they are an important extension to the finite state machines, can represent in a compact way general concepts such as synchronism, asynchronous behavior, concurrent operations, resources sharing, etc.



Finite state machines can't represent system with infinite states but finite places, Petri net can. Another difference between finite state machine and Petri net is in the concept of system state and state transition, in the first both concept embrace the whole system, in the latter both concepts can be thought as an aggregation of partial blocks, each one influencing a part of the system. That's why Petri net are prone to represent asynchronous system in a natural way.

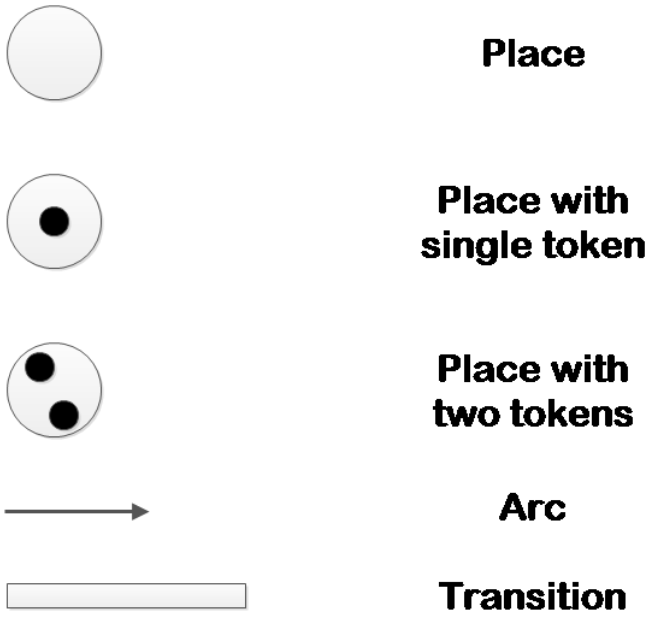
Basics and terminology

As written above, a Petri net consists of places, transitions and arcs. An arc connects a place and a transition (or vice versa), never two places or two transitions. The place from which an arc runs to a transition is called input place, if the arc instead runs from a transition to a places, the last is called output place. The token is another entity that should be present in a Petri net, any distribution of tokens over the places is a configuration of the net that is called marking. A transition can fire if there are sufficient tokens in all its input places so that the arc can crossed (we'll discuss about it later); when it fires, all tokens in its input places are consumed, and then other tokens are re-created in its output places according to the arc's cost. The fire event is non-interruptible step, atomic. An execution of a Petri net is nondeterministic (unless an execution policy is defined), so if there are multiple transitions that are enabled (i.e. they can fire) in the same time, any one of them may fire. Petri net is good to model a distributed system's concurrent behavior cause of this nondeterministic approach to the fire mechanism.

Petri net is a state-transition system that extends the class of elementary net.

- A net is defined as a triple $N = (P, T, F)$ where
    - P and T are disjoint finite sets of places (P) and transitions (T)
    - $F \in (P \times T) \cup (T \times P)$ is a set of arcs (for flow relations)
- Given a net $N = (P, T, F)$ a configuration is a set C so that $C \in P$
- An elementary net is a net in the form $EN = (N, C)$ where:
    - $N = (P, T, F)$ is a net
    - C is such that $C \in P$ is a configuration
- A Petri net is a net in the form $PN = (N, M, W)$ which extends the elementary net so that:
    - $N = (P, T, F)$ is a net
    - M is the initial marking representing the initial distribution of the tokens
    - W is the arc weight mapping

In the diagram of a Petri net, places are conventionally depicted with circles, transitions with long narrow rectangles and arcs as one-way arrows that show connections of places to transitions or transitions to places. If the diagram were of an elementary net, then those places in a configuration would be conventionally depicted as circles, where each circle encompasses a single dot called a token. The place circles may encompass more than one token to show the number of times a place appears in a configuration. The configuration of tokens distributed over an entire Petri net diagram is called a marking. Different markings can depict different configuration and, in some cases, different system states.



Syntax and semantics

A Petri net graph is 3-tuple (S, T, W) where:

- S is a finite set of places
- T is a finite set of transitions
- S and T are disjoint, so no element of S can belong also to T (and vice versa)
- $W = (S \times T) \cup (T \times S) \to \aleph$ is a multiset of arcs (i.e. it assigns to each arc a non-negative integer arc multiplicity, note that no arc can connect to places or two transitions)

The flow relation is the set of arcs $F = \{(x, y) \mid W(x, y) > 0\}$. Often a Petri net can't have any arc multiplicity higher than 1. Likely, a Petri net can be defined with F instead of W, so the Petri net graph is a bipartite multigraph $(S \cup T, F)$ with node partitions S and T.

The preset of a transition t is the set of its input places ${}^\circ t = \{s \in S | W(s, t) \to 0\}$; its postset is the set of its output places $t^\circ = \{s \in S | W(t, s) \to 0\}$. The definitions of preset and postset for places are analogous.

A marking is a multiset of its places, defined as a mapping $M: S \to \aleph$. The marking assigns to each place a number of tokens.

To summarize a Petri net is finally a 4-tuple (S, T, W, M_0) where:

- (S, T, W) is a Petri net graph
- M_0 is the initial marking

Firing a transition t in marking M consumes $W(s, t)$ tokens from each of its input places s, and produces $W(t, s)$ tokens in each of its output places s

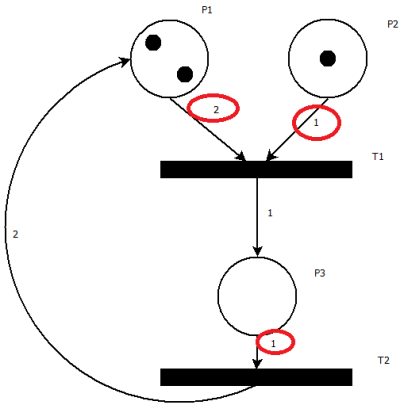
A transition is enabled (so that can fire) in M if there are enough tokens in its input places for the consumption to be possible. We are generally interested in what may happen when transitions may continually fire in arbitrary order. For a Petri net $N = (S, T, W, M_0)$ we are interested in the firings that can be performed starting from the initial marking M_0, it's the set of reachable markings, and represents the state space of the net.

Representation with vectors and matrices

There is a mathematical representation of a Petri net, used often for automatic analysis or to verify the some properties of the system. It is composed by the following elements:

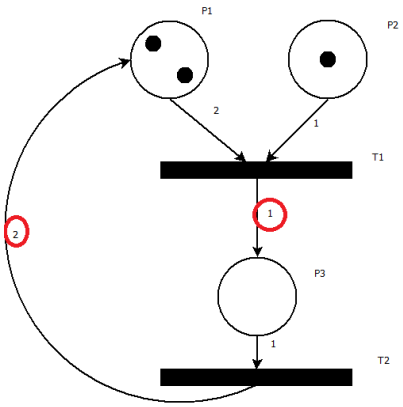
a. Input matrix
b. Output matrix
c. Effect matrix
d. Marking vector
e. Clicks sequence
f. Occurrences vector

The input matrix (I) has a p number of rows (each for each place in the system) and t columns (each for each transition in the net). Its element in position (k,j) has a value equal to weight of the arc that connects the place k to the transition j, if this arc exists, otherwise is equal to 0.



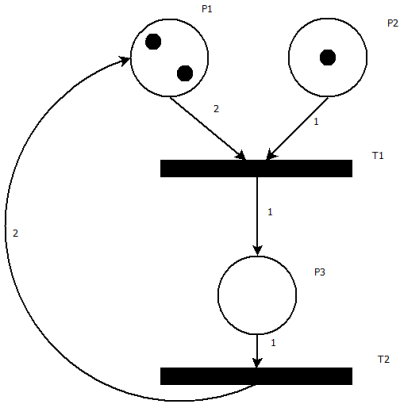
$$I = \begin{bmatrix} * & T1 & T2 \\ P1 & 2 & 0 \\ P2 & 1 & 0 \\ P3 & 0 & 1 \end{bmatrix}$$

The output matrix (O) has a p number of rows (each for each place in the system) and t columns (each for each transition in the net). Its element in position (k,j) has a value equal to weight of the arc that connects the transition k to the place j, if this arc exists, otherwise is equal to 0.



$$O = \begin{bmatrix} * & T1 & T2 \\ P1 & 0 & 2 \\ P2 & 0 & 0 \\ P3 & 1 & 0 \end{bmatrix}$$

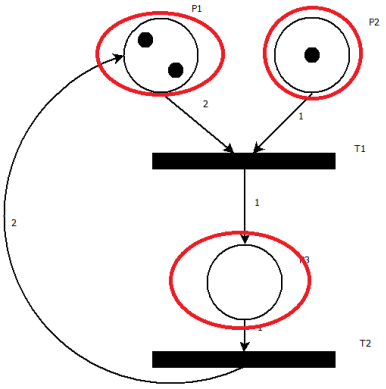
The effect matrix (E) has a p number of rows (each for each place in the system) and t columns (each for each transition in the net). It is defined as the difference between the output matrix and the input matrix.



$$E = O - I = \begin{bmatrix} 0 & 2 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} - \begin{bmatrix} 2 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} -2 & 2 \\ -1 & 0 \\ 1 & -1 \end{bmatrix}$$

This matrix E does not contain all the information that is contained in individual matrices I and O. In fact, it is immediate to note that if for some value of the indices i and j occurs that O (i, j) = I (i, j), or if the i-th place was connected with the transition j-th by two arcs of opposite direction and the same weight, would be E(i, j) = 0. Thus, such a situation is not distinct from the one in which the place i and j the transition were not connected. Similar considerations apply if the arcs connecting a place and transition have opposite direction and different weight. The structure consists of a place and a transition connected by two arcs of direction the opposite is called self-ring. The networks without self-loops, (i.e. devoid of places that are simultaneously input and output to the same transition) are those networks as well. For them, arrays of input and output have non-zero elements in positions mutually exclusive. For them, therefore, the use of the effect matrix is equivalent to the use of the individual matrices input and output.

Given a marking M, the marking vector m is a column vector of dimension p (each for each place in the net). An element in position p has a value equal to the number of token in the place p. It contains only no negative values.



$$m = \begin{bmatrix} P1 & 2 \\ P2 & 1 \\ P3 & 0 \end{bmatrix}$$

With the definition of the marking vector and effect matrix, the concepts of enable and fire of a transition can be reformulated in a very simple and intuitive way. If we highlight the t columns in the matrices I, O and E writing them as $I = [I_1\ I_2 \dots\ I_T\ ]\ \ O = [O_1\ O_2 \dots\ O_T\ ]\ \ E = [E_1\ E_2 \dots\ E_T\ ]$ then the

condition for enabling the i-th transition in the marking M becomes simply $M \geq I_i$ (i.e. the fact that in M there are, in the preset places for the i-th transition, at least as many tokens as the weight of the arc between each of the firsts and the seconds). Moreover, the click of the transition from the i-th marking M produces a new marking M' which is easily computed as $M' = M - I_i + O_i = M + E_i$

An enabled click sequence in a marking M_0 is a sequence of transitions t_i, in which t_1 is enabled in M_0 and the fire of t_i leads to a marking M' in which t_{i+1} is enabled.

We observe that any sequence of transitions is not in general a sequence of clicks. In fact it is said that given a sequence of transitions isn't sure that you can't find a sequence of markings enabling the transitions of given sequence. To emphasize the fact that exists or not such a sequence of markings and avoid confusion, we'll say when necessary, that a sequence of clicks is or is not permissible.

The occurrences vector s, associated with a sequence of clicks S, is a column vector of size t whose generic component s_i is equal to the number of occurrences of transition t_i the in the sequence S.

Thanks to definitions and observations made previously is possible to reach a very compact formulation of the evolution of a Petri net, similar to any dynamic system. For such a representation is given the name of equation of state, as it is used to calculate the "next state" (i.e. marking) of a network known the previous marking and the event (the click of a transition) occurred. Suppose that M_0 is the current marking of a given network with effect matrix E, and suppose that it is also possible to apply a sequence of clicks S, with occurrences vector s. Let M_1 be the marking reached after the application of sequence S. It can be easily observed that: $M_1 = M_0 + Es$

The reachability set $R(N, M_0)$ of a net N with initial marking M_0 is the smallest marking set that comprehends only markings from which, with the fire of a transition, we go to another marking belonging to the set. The reachability graph of a net N with initial marking M_0 is the graph in which nodes are associated to the element of $R(N, M_0)$ and the edges are associated to transitions that cross from a marking to another in $R(N, M_0)$.

Evolution

A transition is enabled if all places of its preset contain a number of tokens at least equal to the weight of the arc that connects them to the transition. The firing of a transition causes the removal of any upstream (i.e. in its preset) and the addition to every place downstream (i.e. in its postset) of a number of tokens equal to the weight of the arcs that connect to these places. The marking of all places which are neither input nor output to the transition remains unchanged.

Along the evolution of the network, the fire of transitions causes a "flow" of tokens. However, to make good use of the networks in order to describe the physical systems, it is not good to think that the token "pass through" the transition. Is much more correct to think that the tokens removed from presets places disappear and then some tokens in the postsets places will be created (which in fact may even be in a different number).

The firing rule is not sufficient to fully determine the evolution of a network because, in a generic marking, it may happen that more than one transitions are enabled to fire (and of course if you choose to make certain of these triggers or some other, future developments the network are not the same). In standard P/T networks is adopted for the determination of the transition from triggering the following rule:

Consider a P/T network with current marking M and let S be the set of transitions enabled in M: only a transition of S is chosen, at random, for the shot. The criterion of choice is completely non-deterministic and guarantees the respect of the locality of the evolution of the system, i.e. independence of events. Once an enabled transition fires, to decide what will be the future transition empowered to fire you have to implement a new assessment of the network, as the marking created by the click of the previous transition may have enabled new transitions and have disabled some of those enabled previously.

The question of how to choose the transition to trigger if there be more than one enabled is still the subject of debate and there are different conventions. The results that I will show are coherent with the rule set above, but there is at least an alternative (which is very common in programs for the simulation of Petri nets). This alternative differs from the rule just stated mainly because if in the enabled transitions set that there are some transitions that aren’t in conflict, these fire all. As for those in conflict, there are many different conventions: a random, priority, and so on. This rule choice is widespread because it is very intuitive, so it is good and know that many products for the simulation adopt it.

Relations between transitions

Sequence

Two transitions t1 and t2 are called in sequence and t1 precedes t2 in a given marking M when, with t1 enabled and t2 not enabled, the firing of t1 enables t2

Conflict

Two transitions t1 and t2 are in structural conflict if and only if they have at least one input place in common. However, this is not enough to decide whether two transitions are really in conflict with each other. Two transitions t1 and t2 are said to be in effective conflict in marking M if they are in structural conflict, if both are enabled in M and the number of tokens contained in their presets are not sufficient to meet all the weights of the arcs that connect them to the two transitions. The structural conflict depends on the topology of the network, the effective conflict also by the current marking. Note that the structural conflict does not imply that the actual conflict can occur

Concurrence

Two transitions t1 and t2 are in structural concurrency with each other when they do not share any input place, i.e. the firing of one transition does not disable the other. The concept effective concurrency is introduced, a situation that occurs only during the evolution of the network. Two transitions t1 and t2 are said to be in effective concurrency in the marking M if both are enabled in M. Note that the structural concurrency implies that can also occur the effective one.

Fundamental properties

Markings properties

Reachability

A marking M_1 is said to be reachable from a given marking M if there exists at least one sequence of transitions such that by firing from M is possible to obtain the marking M_1. The set of markings reachable from marking M is indicated with [M>. Then, to indicate that the marking M' is reachable from M, we'll write $M' \in$[M>

Liveness

A marking M is said "alive" if and only if, chosen any transition t of the network, from M is possible to reach a marking (denoted by M') in which t is enabled; in other words, a marking M is "alive" if and only if, by evolving the network from it, there aren't transitions that can't be enabled. According to the notation just introduced, finally, write that the marking M is alive if and only if $\forall t \in T \; \exists M' \in$[M> such that t is enabled in M'.

Places properties

k-boundedness

A place of a network is said to k-bounded if, on any reachable marking of network, the value of its marking never exceeds a predetermined value k (i.e. if the place will never contain more than k tokens).

Transitions properties

Liveness

Let M_0 the initial marking of a Petri net, its transition t is said "alive" if and only if $\forall M \in [M_0>$ $\exists M' \in$[M> such that t is enabled in M', or if and only if, starting from any marking reachable from the initial one, it is possible to reach another marking where the transition is enabled.

Net properties

Reversibility

A Petri net with initial marking M_0 is said to be reversible if for each marking $M \in [M_0>$, M_0 is reachable from M. In other words, a network is reversible if it is always possible to bring it back in the initial marking.

Home state

A state (i.e. a marking) M_i of a Petri net is said to be a home state of the network if, for each marking if $M \in [M_0>$, M_i is reachable from M. In other words, a marking is a home state if it is possible to go from any other marking that the network can achieve.

k-boundedness

A Petri net is said "k-bounded" if all its places are k-bounded. A limited network can't contain an infinite number of tokens, and thus has a finite number of marking possible. So a k-bounded network is equivalent to a finite state machine, and it is in this sense that finite state machines are a special case of Petri net.

Liveness

A Petri net is "alive" if and only if all the markings reachable from initial marking are "alive". A Petri net is "alive" if and only if all its transitions are "alive".

A transition t is said "alive" if from any reachable marking of the net is possible to reach another in which the transition t is enabled. Clearly, if the transition t fires (as it is enabled), it will reach a marking M' of network which is reachable from the initial one by definition; therefore M' belong in the definition of "alive" transition. Thus the "alive" transition must be able to fire again from M', for example reaching another marking M'' that enables it again, and so on. It follows that in an "alive" network all transitions can fire infinite times whatever is the marking reachable from the initial marking. The liveness is therefore a very strong condition. It is in fact not possible to establish necessary and sufficient conditions for the liveness for general-purpose networks.

Applications

There are many available tools to simulate, validate and analyze Petri nets. Some of these are developed by Universities and distributed as freeware. We will now use QPME, a tool designed specifically to model and simulate queuing Petri networks. It is based on Eclipse environment and offers a very good support to analysis, bottleneck identification and graph based results. This tool also supports colored Petri net so that different kind of traffic can be treated in a scenario.

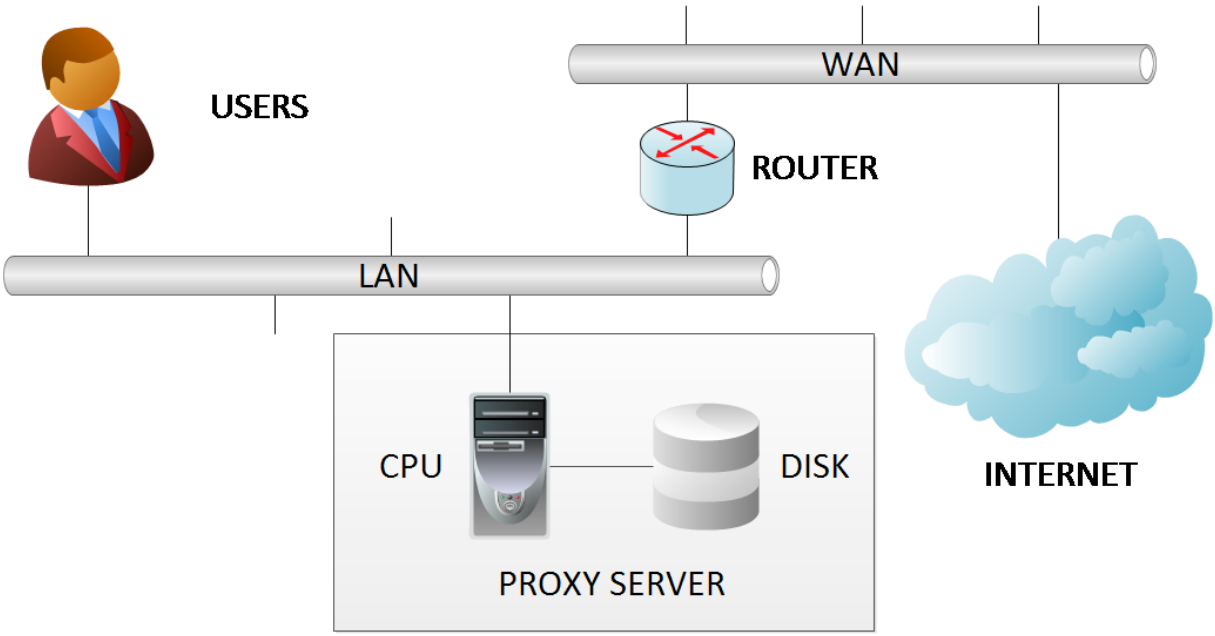
Provided as example, next follows a capacity planning problem that will be solved using the mentioned tool.

Exercise text

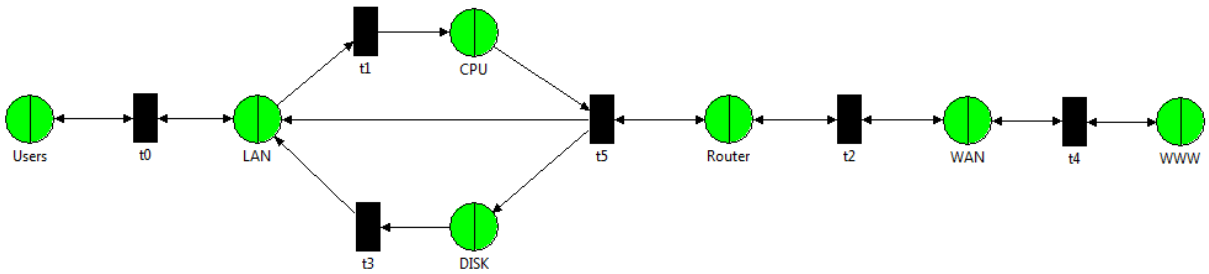
A Web site handles 100 users connected to a 1 Gbps Ethernet that on turn is connected to Internet through a router having a latency of 10 microsec/packet. The connection is established through a link that guarantees a bandwidth of 10 Mbps. Inside the LAN, there is a proxy server composed of a CPU and a disk: the CPU service demand is 0.1 msec in case of hit on the proxy server, 0.2 msec in case of miss; the disk service time is 10 msec/KB on read access. The percentage of users accessing the Web site is 10% and their access rate is 0.2 req/sec when they are on think-time state. The clients request two types of documents: the first one of 2048 bytes (80% of requests) and the second one of 100 KB (remaining 20% of requests). The dimension of the HTTP request is 400 bytes. The Internet RTT is 200 msec. The transfer rate from the remote server is 100 KB/sec on average.

Evaluate the average response time experienced by each user and identify the bottleneck, hypothesizing that the probability of hit on the proxy server is 60 %.

Network model for the given exercise looks like this, it helps understand the whole system but lacks in information about how to model in a queuing network, for example is not clearly a closed network, such it is.



A brief view on QPME model is instead more particular and precise, it let understand the closed network topology, and is possible to follow data flow with a small effort.



To make the simulation work properly, we need to calculate basic data, such as service time, demand time and max throughput for each queue given each data flow

- HTTP REQUEST

  It represents the request that a user makes to the system, the data size is given in the exercise text and is 400 bytes

$$NSegments = 1$$

$$NDatagrams = \frac{20 + 400}{1480} = 1$$

$$Ovhd_{ETH} = 20 + 20 + 18 = 58\ bytes$$

$$S_{LAN}^{REQ} = \frac{(400 + 58) * 8}{10^9} = 3.664\ \mu s$$

$$S_{WAN}^{REQ} = \frac{(400 + 58) * 8}{10^7} = 366.4\ \mu s$$

$$S_{ROUTER}^{REQ} = 10\ \mu s$$

- BIG DOCUMENT REPLY

  It represents the service time of the big document's reply to a request made to the system. Given the transfer of 100Kbytes of data, as written in the exercise, we have

$$NSegments = 2$$

$$NDatagrams = \frac{20 + 102400}{1480} = 70$$

$$Ovhd_{ETH} = 20 + 70 * (20 + 18) = 2680\ bytes$$

$$S_{LAN}^{BIGDOC} = \frac{(102400 + 2680) * 8}{10^9} = 840.64\ \mu s$$

$$S_{WAN}^{BIGDOC} = \frac{(102400 + 2680) * 8}{10^7} = 84.064\ ms$$

$$S_{ROUTER}^{BIGDOC} = 10 * 70 = 700\ \mu s$$

$$S_{DISK}^{BIGDOC} = \frac{102400}{102400} = 1\ s$$

- SMALL DOCUMENT REPLY

  It represents the service time of the small document's reply to a request made to the system. Given the transfer of 2Kbytes of data, as written in the exercise, we have

$$NSegments = 1$$

$$NDatagrams = \frac{20 + 2048}{1480} = 2$$

$$Ovhd_{ETH} = 20 + 2 * (20 + 18) = 96\ bytes$$

$$S_{LAN}^{SMALLDOC} = \frac{(2048 + 96) * 8}{10^9} = 17.152\ \mu s$$

$$S_{WAN}^{SMALLDOC} = \frac{(2048 + 96) * 8}{10^7} = 1.7152\ ms$$

$$S_{ROUTER}^{SMALLDOC} = 10 * 2 = 20\ \mu s$$

$$S_{DISK}^{SMALLDOC} = \frac{2048}{102400} = 20\ ms$$

- CONNECTION HANDLING

  HTTP application protocol relies on TCP transport protocol, it offers reliable connection thanks to the three way handshaking. This process requires the exchange of some packets so also this must be considered

$$NSegments = 1$$

$$NDatagrams = 1$$

$$Ovhd_{ETH} = 20 + 20 + 18 = 58\ bytes$$

$$S_{LAN}^{HANDLING} = \frac{58 * 8}{10^9} = 0.464\ \mu s$$

$$S_{WAN}^{HANDLING} = \frac{58 * 8}{10^7} = 46.4\ \mu s$$

$$S_{ROUTER}^{HANDLING} = 10\ \mu s$$

Now we calculate demand times and after the throughput

$$D_{CPU}^{HIT} = 0.1\ ms$$

$$D_{CPU}^{MISS} = 0.2\ ms$$

$$D_{DISK}^{SMALLDOC} = \frac{2048}{102400} = 20\ ms$$

$$D_{DISK}^{BIGDOC} = \frac{102400}{102400} = 1\ s$$

$$D_{INTERNET}^{SMALLDOC} = \left(\frac{7}{2}RTT + DownloadTime\right) = \left(\frac{7}{2} * 0.2 + \frac{2048}{102400}\right) = 0.72\ s$$

$$D_{INTERNET}^{BIGDOC} = \left(\frac{7}{2}RTT + DownloadTime\right) = \left(\frac{7}{2} * 0.2 + \frac{102400}{102400}\right) = 1.7\ s$$

Other values are exactly the same of the corrispective service times, connection management is almost ignored in the local system due to the very little service times, but RTT is foundamental in INTERNET environment so it is used.

About the throughput, it is the inverse ratio of the demand time

$$T_{LAN}^{REQ} = 272925\ per\ second$$

$$T_{LAN}^{BIGDOC} = 1189\ per\ second$$

$$T_{LAN}^{SMALLDOC} = 58302\ per\ second$$

$$T_{WAN}^{REQ} = 2729.25\ per\ second$$

$$T_{WAN}^{BIGDOC} = 11.89\ per\ second$$

$$T_{WAN}^{SMALLDOC} = 583.02\ per\ second$$

$$T_{CLIENTS} = 0.2\ each\ user\ per\ second$$

$$T_{CPU}^{HIT} = 10000\ per\ second$$

$$T_{CPU}^{MISS} = 5000\ per\ second$$

$$T_{DISK}^{BIGDOC} = 1\ per\ second$$

$$T_{DISK}^{SMALLDOC} = 50\ per\ second$$

$$T_{ROUTER}^{REQ} = 100000\ per\ second$$

$$T_{ROUTER}^{BIGDOC} = 1428\ per\ second$$

$$T_{ROUTER}^{SMALLDOC} = 50000\ per\ second$$

$$T_{INTERNET}^{BIGDOC} = 0.588\ per\ second$$

$$T_{INTERNET}^{SMALLDOC} = 1.388\ per\ second$$

These simulation results represent the average system behavior during 1’000’000 seconds. Such long time is used to ensure no transient influences. The numbers equal to zero is due to maximum accuracy of 10^{-3}

Queue	Mean Total Token Population	Total Departure Throughput	Total Arrival Throughput	Mean Token Residence Time	Queue Utilization
Clients (queue)	7,017	1,948	1,948	3,603	0,972
Local_LAN (queue)	0	3,895	3,895	0	0
Proxy_CPU (queue)	0	1,948	1,948	0	0
Proxy_DISK (queue)	0,598	1,17	1,17	0,512	0,254
Remote_LAN (queue)	0,016	1,556	1,556	0,011	0,014
Router (queue)	0	1,556	1,556	0	0
WWW (queue)	2,367	0,778	0,778	3,042	0,712

It’s clear that system usage is under 100%, driven by limited user requests. Outside the local system the bottleneck is due to Internet, inside is limited by the proxy disk.

The average response time seen by a user entering the system is given by clients “Mean Token Residence Time” of the table above, about 3.6 seconds

The simulation file is hosted here

References

- "Petri net" on Wikipedia (http://en.wikipedia.org/wiki/Petri_net)
- Platform Independent Petri net Editor 2 (PIPE2 - http://pipe2.sourceforge.net/index.html)
- Notes from Politecnico di Milano – sede di Cremona – Automazione Industriale (http://www.cremona.polimi.it/dispense/Automazione%20Industriale)
- "Petri Nets: Properties, Analysis and Applications" by Tadao Murata – IEEE
- "Modeling Distributed Real-Time Systems using Adaptive Petri Nets" by O. Baldellon, J.C. Fabre, M. Roy
- "Petri net modeling and simulation of a distributed computing system" by J.M. Tjensvold
- "Petri Nets" by Y. Narahari
- "Comparison of Petri Net and Finite State Machine Discrete Event Control of Distributed Surveillance Networks" by M. Zhu, R.R. Brooks
- CPNtools (http://cpntools.org/)
- Time Petri Net Analyzer – TINA (http://projects.laas.fr/tina//)
- QPME (**Q**ueueing **P**etri net **M**odeling **E**nvironment) - http://descartes.ipd.kit.edu/projects/qpme/