Far East Journal of Experimental and Theoretical Artificial Intelligence
Volume 4, Number 1, 2009, Pages 33-43

PERFORMANCE MODELING IN COMPARISON OF SWITCH NETWORK AND BLUETOOTH TECHNOLOGY BY USING MARKOV ALGORITHM AND QUEUEING PETRI NETS WITH THE SECURITY OF STEGANOGRAPHY

V. B. KIRUBANAND and S. PALANIAMMAL

Department of MCA
VLB Janakiammal College of Arts and Science, India

Department of Science and Humanities
VLB Janakiammal College of Engineering and Technology, India

Abstract

The main theme of this paper is to find the performance of the Switch network with the Bluetooth technology using the Queueing Petri Net model and the Markov algorithm with the security of steganography. This paper mainly focuses on comparison of switch and Bluetooth technologies in terms of service rate and arrival rate. In Markov algorithm by using this model $M/M(1, b)/1$ bulk service rule, the performance of client-server model has been studied and the results are obtained. When comparing the service rates from the Switch network and the Bluetooth technology, it has been found that the service rate from the Bluetooth technology is very efficient for implementation. From the values obtained from the Bluetooth technology, the values from other wireless technologies can be calculated and implemented in future for better performance. QPNs facilitate the integration of both hardware and software aspects of the system behavior in the improved model. This lends itself very well to modeling distributed component-based systems such as modern e-business applications (New order, Change order, Order status, Customer status). Simultaneous

---

Keywords and phrases: bluetooth, switch, queueing Petri nets, Markov algorithm, steganography.

Communicated by Shun-Feng Su

Received February 25, 2009; Revised August 4, 2009

resource possession, synchronization, blocking and contentions for software resources can easily be modeled by adding the wireless technologies in the systems and by using QPNs. The purpose of steganography is to hide the information, those can be transferred from one system to another through the Bluetooth technology with security measures. QPNs are very powerful as a performance analysis and prediction tool. By demonstrating the power of QPNs as a modeling paradigm in further fore coming technologies we hope to motivate further research in this area.

1. Introduction

Switch

A network switch is a small hardware device that joins multiple computers together within one Local Area Network (LAN). Technically, network switches operate at layer two (Data Link Layer) of the OSI model.

Network switches appear nearly identical to network hubs, but a switch generally contains more intelligence (and a slightly higher price tag) than a hub. Unlike hubs, network switches are capable of inspecting data packets as they are received, determining the source and destination device of each packet, and forwarding them appropriately. By delivering messages only to the connected device intended, a network switch conserves network bandwidth and offers generally better performance than a hub.

As with hubs, Ethernet implementations of network switches are the most common. Mainstream Ethernet network switches support either 10/100 Mbps Fast Ethernet or Gigabit Ethernet (10/100/1000) standards.

Different models of network switches support differing numbers of connected devices. Most consumer-grade network switches provide either four or eight connections for Ethernet devices. Switches can be connected to each other, a so-called daisy chaining method to add progressively larger number of devices to a LAN [http://compnetworking.about.com].

Bluetooth technology

Bluetooth is an open wireless protocol for exchanging data over short distances from fixed and mobile devices, creating Personal Area Networks (PANs). It was originally conceived as a wireless alternative to RS232 data cables. It can connect several devices, overcoming problems of synchronization. The implication is that Bluetooth does the same with communications protocols, uniting them into one universal standard. Bluetooth uses a radio technology called *frequency-hopping spread spectrum*, which chops up the data being sent and transmits chunks of it on up to 79 frequencies. In its basic mode, the modulation is Gaussian Frequency-Shift Keying (GFSK). It can achieve a gross data rate of 1Mb/s. Bluetooth provides a way to connect and exchange information between devices such as mobile phones, telephones, laptops, personal computers, printers, Global Positioning System (GPS) receivers, digital cameras, and video game consoles through a secure, globally unlicensed Industrial, Scientific and Medical (ISM) 2.4GHz short-range radio frequency bandwidth. The Bluetooth specifications are developed and licensed by the Bluetooth Special Interest Group (SIG) [www.bluetooth.com]. The Bluetooth SIG consists of companies in the areas of telecommunication, computing, networking, and consumer electronics.

The main objective of these models is to find the following with the usage of Markov algorithm:

- To find the waiting time.
- To find the busy period.

Steganography

Steganography is the art and science of hiding messages. The word steganography is derived from the Greek words “steganos” and “graphein”, which mean “covered” and “writing”. Steganography, therefore, is covered writing. Historical steganography involved techniques such as disappearing ink or microdots. Modern steganography involves hiding data in computer files. It is fairly easy to hide a secret message in a graphic file without obviously altering the visible appearance of that file [www.tech-faq.com].

2. Queueing Networks

A network consists of a set of interconnected queues. Each queue represents a service station, which serves requests (also called *jobs*) sent by customers. A service station consists of one or more servers and a waiting area which holds requests waiting to be served. When a request arrives at a service station, its service begins immediately if a free server is available. Otherwise, the request is forced to wait in the waiting area. Service is done as per the general bulk service rule introduced by Neuts [9] with $a = 1$. As per this rule immediately after the completion of the service, if the server finds a unit present, it start its service; if it finds one or more but atmost *b*, it takes them all in a batch and if it finds more than *b*, it takes in the batch for service *b* units, while others wait. The batch takes a minimum of one unit and a maximum of *b* units [7]. The time between successive request arrivals is called *interarrival time*. Each request demands a certain amount of service, which is specified by the length of time a server is occupied serving it, i.e., the service time. The delay is the amount of time, the request waits in the waiting area before its service begins. The response time is the total amount of time, the request spends at the service station, i.e., the sum of the delay and the service time.

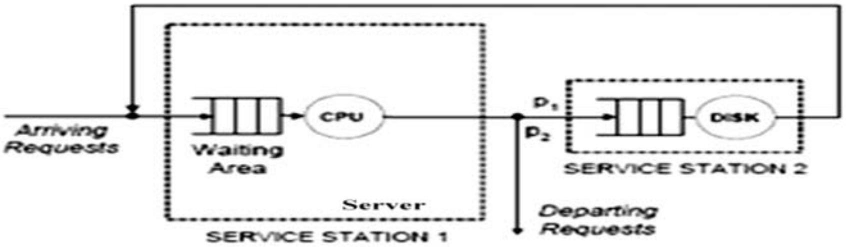

Figure 1. A basic queueing network.

Figure 1 shows a basic network with two queues, i.e., service stations. Arriving requests first visit service station 1, which has one server (representing CPU). After requests are served by the server, they move to service station 2 (representing a disk device) with probability p1 or leave the network with probability p2. Requests completing service at station 2 return back to station 1. The interconnection of queues in a network is described by the paths requests may take which are specified by routing probabilities. A request might visit a service station multiple times while it circulates through the network. The total amount of service time required, over all visits to the station, is called *service demand* of the request at the station. Requests are usually grouped into classes with all requests in the same class having the same service demands. The algorithm which determines the order in which requests are served at a service station is called *scheduling strategy* (or *scheduling/discipline*) [3]. Some typical scheduling strategies are:

- FCFS (First-Come-First-Served): Requests are served in the order in which they arrive. This strategy is typically used for queues representing I/O devices.

- LCFS (Last-Come-First-Served): The request that arrived last is served next.

- PS (Processor-Sharing): All requests are assumed to be served simultaneously with the server speed being equally divided among them. This strategy is typically used for modeling CPUs.

- IS (Infinite-Server): There is an ample number of servers so that no queue ever forms. Service stations with IS scheduling strategy are often called *delay resources* or *delay servers*.

3. Petri Nets

Petri Nets were introduced in 1962 by Carl Adam Petri. An ordinary Petri Net (also called *Place-Transition Net*) is a bipartite directed graph composed of places, drawn as circles, and transitions, drawn as bars. A formal definition [2] is given below:

Definition (PN). An ordinary Petri Net (PN) is a 5-tuple,

$$PN = (P, T, I^-, I^+, M_0),$$

where

1. P is a finite and non-empty set of places,
2. T is a finite and non-empty set of transitions,
3. $P \bigcap T = \varnothing$,
4. $I^-, I^+ : P \times T \to N_0$ are called *backward* and *forward incidence functions,* respectively,
5. $M_0 : P \to N_0$ is called *initial marking*.

The incidence functions I^- and I^+ specify the interconnection between places and transitions. If $I^+(p, t) > 0,$ an arc leads from place p to transition t and place p is called an *input place* of the transition. If $I^+(p, t) > 0,$ an arc leads from transition t to place p and place p is called an *output place* of the transition. The incidence functions assign natural numbers to arcs, which we call *weights* of the arcs. When each input place of transition t contains at least as many tokens as the weight of the arc connecting it to t, the transition is said to be *enabled*. An enabled transition may fire, in which case it destroys tokens from its input places and creates tokens in its output places. The amounts of tokens destroyed and created are specified by the arc weights. The initial arrangement of tokens in the net (called *marking*) is given by the function M_0 which specifies how many tokens are contained in each place. When a transition fires, the marking may change. Figure 2 illustrates this using a basic Petri Net with 4 places and 2 transitions. The Petri Net is shown before and after firing of transition t_1, which destroys one token from place p_1 and creates one token in place p_2.

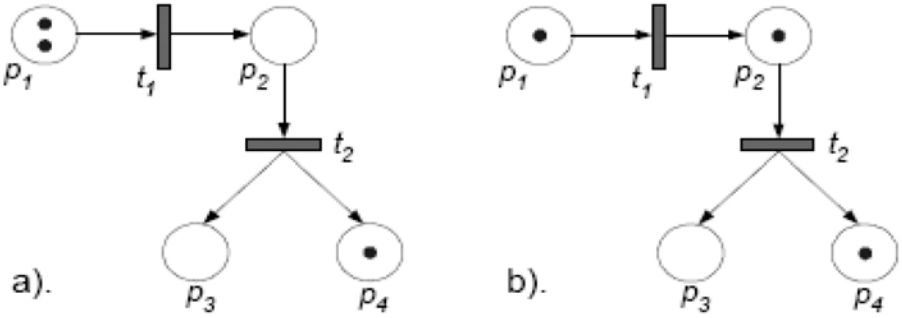

Figure 2. An ordinary Petri Net before and after firing transition t_1.

4. Basic Queueing Petri Nets

The main idea in the creation of the QPN formalism was to add and timing aspects to the places of Colored Generalized Stochastic Petri Nets [2]. This is done by allowing queues (service stations) to be integrated into places of CGSPNs. A place of a CGSPN which has an integrated queue is called a *place* and consists of two components, the queue and a depository for tokens which have completed their service at the queue. This is depicted in Figure 3.

The behavior of the net is as follows: tokens, when fired into a place by any of its input transitions, are inserted into the queue according to the queue's scheduling strategy. Tokens in the queue are not available for output transitions of the place. After completion of its service, a token is immediately moved to the depository, where it becomes available for output transitions of the place. This type of place is called a *timed place*. Tokens in immediate places can be viewed as being served immediately. Scheduling in such places has priority over scheduling/service in timed places and firing of timed transitions.

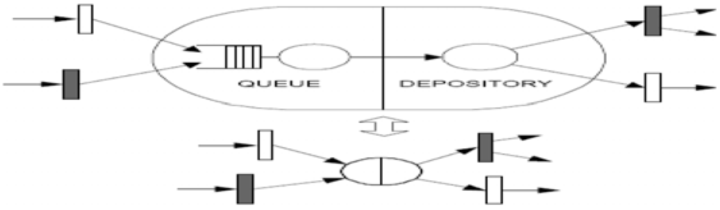

Figure 3. A queueing place and its shorthand notation.

An enabled timed transition fires after an exponentially distributed delay according to a race policy. Enabled immediate transitions fire according to relative firing frequencies and their firing has priority over that of timed transitions. We now give a formal definition of a QPN and then present an example of a QPN model.

Definition (QPN). A Queueing Petri Net (QPN) is a 8-tuple $QPN = (P, T, C, I^-, I^+; M_0, Q, W)$, where

1. $CPN = (P, T, C, I^-, I^+; M_0)$ is the underlying Colored Petri Net.

2. $Q = (q_1, ..., q_{|P|})$ is an array whose entry q_i, denotes the description of a queue taking all colors of $C(P)$ into consideration, if p_i is a timed place or equals the keyword untimed, if p_i is an untimed place.

3. $W = (w_1, ..., w_{|T|})$ is an array of functions whose entry w_i is defined on $C(t_i)$ and $\forall c \in C(t_i) : w_i(c)$ is the description of a probability distribution function specifying the firing delay due to color $c \in C(t_i)$, if transition t_i is a timed transition or is a weight specifying the relative firing frequency due to color $c \in C(t_i)$, if transition t_i is an immediate one.

5. Hierarchical Queueing Petri Nets

As already mentioned, the main hurdle to the quantitative analysis of QPNs is the fact that most analysis techniques available are based on Markov Chains and are therefore susceptible to the state space explosion problem. More specifically, as one increases the number of queues and tokens in a QPN, the size of the state space of the underlying Markov Chain grows exponentially and quickly exceeds the capacity of today’s computers. This imposes a limit on the size and complexity of the models that are analyzable. An attempt to alleviate this problem was the introduction of Hierarchically-Combined Queueing Petri Nets (HQPNs) [1]. The main idea is to allow hierarchical model specification and then exploit the hierarchical structure for efficient numerical analysis. This type of analysis is termed structured analysis and it allows models to be solved, which are about an order of magnitude larger than those analyzable with conventional techniques.

HQPNs are a natural generalization of the original QPN formalism. In HQPNs, a place may contain a whole QPN instead of a single queue. Such a place is called a *subnet place* and is depicted in Figure 4. A subnet place might contain an ordinary QPN or again a HQPN allowing multiple levels of nesting. For simplicity, in this paper, we restrict ourselves to two-level hierarchies. We use the term High-Level QPN (HLQPN) to refer to the upper level of the HQPN and the term Low-Level QPN (LLQPN) to refer to a subnet of the HLQPN.

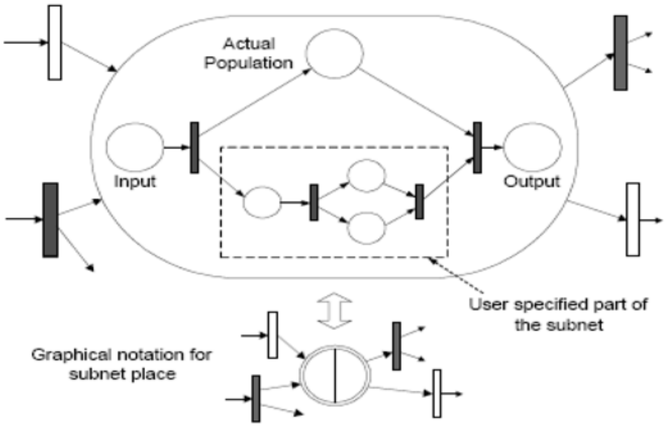
Actual Population
Input
Output
User specified part of the subnet
Graphical notation for subnet place

Figure 4. A subnet place and its shorthand notation.

Every subnet of a HQPN has a dedicated input and output places, which are ordinary places of a colored Petri Net. Tokens being inserted into a subnet place after a transition firing are added to the input place of the corresponding HQPN subnet. The semantics of the output place of a subnet place is similar to the semantics of the depository of a place: tokens in the output place are available for the output transitions of the subnet place. Tokens contained in all other places of the HQPN subnet are not available for the output transitions of the subnet place. Every HQPN subnet also contains actual – population place, which is used to keep track of the total number of tokens fired into the subnet place.

6. Markov Algorithm

A Markov algorithm is a string rewriting system that uses grammar-like rules to operate on strings of symbols. Markov algorithms have been shown to be turing-complete, which means that they are suitable as a general model of computation and can represent any mathematical expression from its simple notation [www.knowledgerush.com].

7. Bulk Service Rules

There could be a number of policies or rules according to which batches for bulk service may be formed. The following are the types of bulk service policies or rules frequently discussed in the literature.

Bailey [9] and Downton [9] considered that units are served in batches of not more than *b* (say). If, immediately after the completion of a service, the server finds more than *b* units waiting, he takes a batch of *b* units for service while others wait; if he finds *r* units $(0 \leq r \leq b)$, he takes all he *r* units in a batch for service [7]. Bloemena [9], Jaiswal [9] and Neuts [9] considered the same rule with the restriction that $r! = 0\ (1 \leq r \leq b)$, i.e., the service facility stops until *a* unit arrives. This rule will be called the *usual bulk service rule*, while bailey's rule will be its modified type (it will be also called *bulk service rule* with intermittently available server). Jaiswal points out that the distribution of the queue length for the modified rule can be obtained from that of the usual rule [7].

The rule with a fixed batch size *k* has been considered by Fabens [9], Takacs [9] and others. In this case, the server waits until there are *k* units, and serves all the *k* units in a batch. If there are more than *k* waiting when the server becomes free, he takes a batch *k* for service, while others wait [7].

Neuts [9] considered the general bulk service rule: if, immediately after the completion of a service, the server finds less than *a* units present, he waits until there are *a* units, whereupon he takes the batch of *a* units for service; if he finds *a* or more but atmost *b*, he takes them all in a batch and if he finds more than *b*, he takes in the batch for service *b* units, while others wait. The batch takes a minimum of *a* units and a maximum of *b* units [7]. This rule will be called *general bulk service rule* as rules under (1) and (2) can be taken as particular cases.

Bhat [9] considered the rule that the number taken in a batch is a random variable *y*. The corresponding Markov model is denoted by $M/M^{\wedge}(y)/1$ [7].

There are actual situations where one rule seems to be better suited or to be more appropriate than the others.

Performance measures

(i) Probability that the server is idle

$$p_{0,0} = \frac{(1-r)}{1-r+(\lambda/\mu)} = \frac{(1-r)^2}{(1-r)^2 + r(1-r^b)}.$$

(ii) Probability that the server is busy and *n* units in the system

$$p_{1,n} = \frac{(1-r)(1-r^b)}{(1-r)^2 + (1-r^b)} r^{n+1}, \quad n = 0, 1, 2, \ldots.$$

(iii) Waiting Time Density

$$V(t) = \frac{\lambda p_{0,0}}{((1-r))} [(1-r^b)\exp\{-\mu(1-r^b)t\}].$$

(iv) Expected Waiting Time Density

$$E(T) = \frac{r}{\mu[(1-r)^b + r(1-r^b)]}.$$

(v) Expected Busy Period

$$\frac{E(B)}{E(I)} = \frac{1-p_{0,0}}{p_{0,0}}.$$

From the above formulas we are calculating the Performance of Client-Server model.

λ (Arrival rate) - Is the number of request send by the client machine to the server machine [7].

μ (Service rate) - Is the response provided by the server machine to the client machine [7].

A comparative study of the wired technology with the wireless technology has been done with the help of the Markovian model $M/M(1, b)/1$ for expected waiting time density [i.e., $E(T)$] and expected busy period [i.e., $E(B)$].

Comparative study of waiting time and busy period in Switch network and Bluetooth network

Particulars	Switch (in sec)	Bluetooth (in sec)
λ-Arrival Rate	1/3	1/3
μ-Service Rate	1/2.32	1/1.8
$E(T)$-Expected waiting time	0.8818	0.6841
$E(B)$-Expected busy period	3.3200	2.571

The probability that the server is idle and the probability that the server is busy are same for both of the networks.

Conclusion

Thus this research concludes that the performance of the client server architecture is better in the Bluetooth technology when compared to Switch network and we hope to motivate further research in this area.

References

[1] F. Bause, P. Buchholz and P. Kemper, Hierarchically combined queueing Petri Nets, Proc. 11th Int. Conf. on Analysis and Optimizations of Systems, Discrete Event Systems, Sophia-Antipolis, France, 1994.

[2] F. Bause and F. Kritzinger, Stochastic Petri Nets - An Introduction to the Theory, Vieweg Verlag, 2002.

[3] G. Bolch, S. Greiner, H. de Meer and K. S. Trivedi, Queueing Networks and Markov Chains - Modelling and Performance Evaluation with Computer Science Applications, Wiley, New York, 1998.

[4] http://compnetworking.about.com.

[5] http://www.knowledgerush.com/kr/encyclopedia/markov_algorithm/.

[6] S. Kounev and A. Buchmann, Performance Modelling of Distributed e-business Applications using Queueing Petri Nets, 2003, pp. 143-155.

[7] J. Medhi, Stochastic Processes, 2nd ed., New Age International Publishers, 1994.

[8] www.bluetooth.com.

[9] www.tech-faq.com.