Modeling and performance analysis of QoS data

Dariusz Strzciwilk and Wlodzimierz M. Zuberek

Department of Applied Informatics
Warsaw University of Life Sciences
02-776 Warsaw, Poland

Department of Computer Science
Memorial University
St. John’s NL, Canada A1B 3X5

Abstract. Performance analysis of communication systems that support quality of service is discussed in this paper. The developed models take into account multiservice network architecture, i.e., the models support transmission of data belonging to different classes of traffic. Simple priority queueing models with three levels of priorities are used for illustration of properties of such systems, especially when the traffic becomes intensive. Timed Petri nets are used for modeling the communication systems and discrete–event simulation provides performance characteristics of developed models. The paper shows that timed Petri nets offer a convenient, flexible and effective approach to modeling and performance analysis of computer networks.

Keywords: quality of service, QoS data, Petri nets, timed Petri nets, priority queueing systems, discrete–event simulation.

1 Introduction

QoS (Quality of Service) is one of the most important aspects of the design and maintenance of current as well as next generation computer networks [19, 11, 17]. In order to satisfy the qualitative and quantitative network requirements, the design of computer networks cannot be sepa-erated from the network infrastructure and its properties; the design process must take into account network typologies, nodes, structure, and must determine the resource capacities needed. It must also provide appropriate strategies for management and handling of network traffic. Moreover, it is necessary to determine the capacity of interfaces and queueing mechanisms that have been used. Mechanisms and methods for management of queues of packets in the buffers of intermediary nodes to reduce the negative effects of overloading can easily be found in the literature [5].

Quality of service can be considered as the ability of a network to deliver the service in a better way or to provide special service to a group of users or applications, at the expense of other users and applications. Guaranteeing adequate quality of service is of particular importance for real-time applications such as voice transmission - VoIP (Voice over IP) and video IPTV (Internet Protocol Television) [18, 21] as these services are particularly sensitive to delay and require guaranteed bandwidth. For this reason, real–time voice and video traffic use higher priority and may need better treatment in the network than other services and applications, such as email, FTP or HTTP. QoS is a mechanism which uses different priorities for the data of different classes. QoS allows the use of, inter alia, priority queues to provide guaranteed bandwidth or to compensate delay variation in networks [13, 20, 1]. Some aspects of effective management of the QoS data are of particular interest not only in wired networks, but also in wireless networks, which are more limited in their resources than in the case of wired networks [10]. In this context, the need to optimally design the system and to use network resources in the best possible way is of special importance.

The aim of this work is to use timed Petri nets [24] to evaluate the performance of PQS (Priority Queueing Systems) and QoS schemes developed for computer networks. The results may be helpful in the design and effective use and management of telecommunication networks. Moreover, they can provide some insight into the complex behavior of computer networks, distributed systems, multiprocessor systems, telephone switching systems and mobile phone networks.

1This version of the paper is insignificantly different from the original one. The paper has been revised and this revision resulted in a number of straightforward corrections, additions and deletions.
2 QoS architectures

Research conducted by the IETF (Internet Engineering Task Force) on QoS architectures in IP (Internet Protocol) networks resulted in two architectures for the categorization of traffic into classes with different levels of quality of services. The first architecture is known as the IntServ (Integrated Services), described in RFC 1633 [4], while the second architecture is DiffServ (Differentiated Services), described in RFC 2475 [3]. These architectures allow to extend the default BE (Best Effort) model [7], currently used on the Internet. The expanding area of application of IP networks makes it essential to guarantee quality of services provided by the IP protocol, especially for real-time applications, multimedia, online gaming, etc. Class services implemented in the IntServ model enable support for any type of application for which a guaranteed network service (Guaranteed Service) is intended. The guaranteed services are associated with reservation of resources, and this creates problems of scalability in backbone networks. A large signaling overhead is needed to carry information about the parameters of the reservations and about dedicated CPU resources for handling packets from a specific stream. DiffServ architecture eliminates the scalability issue of the IntServ model. The differentiated services model provides the most extensive and appealing approach to QoS support in current IP networks. The classification of data streams takes place on the edge of the network, and within the network streams are assigned to DiffServ classes. The packet’s class is stored in a six bit DSCP (Differentiated Services Code Point) field located in the IP header (see Fig. 1). The value of the DS field in the DSCP DS Code Point information is a label used to classify packets in the DiffServ architecture. DS field is divided into smaller fields, which are responsible for determining the appropriate traffic as EF (Expedited Forwarding) or AF (Assured Forwarding); the last 2-bit fields are unused. For a 6-bit DSCP field, 64 classes of traffic can be defined. In the IPv4 headers, the TOS (Type of Service) field consists of 2 fields: 3-bit precedence field and 4-bit TOS field. The first field indicates the superiority of traffic and the second defines the special requirements eg. a small delay.

![Figure 1: Comparison of the DS field and Precedence/TOS](image)

Diff-Serv model does not require any special signaling protocol. In this model, the network traffic is divided into classes of traffic and each class may be assigned to a different level of service, for example, VoIP traffic is assigned to a high-priority class, while sending eg. e-mail is treated in accordance with the BE model. The required quality of service can thus be provided by DiffServ Services. DSCP code values also affect the queue discipline as well as rules of packet discarding in the case of buffer overflows. Edge routers also use a more advanced mechanisms to provide the desired service. As indicated in [6], if the capacity of any system is increased in order to meet the demands of users, user requests will also increase and will utilize the increased capacity of the system. It means that the mere increase in the capacity of the transmission channel will not significantly improve the quality of transmission without the use of additional mechanisms to ensure the desired service. Elements controlling the traffic can be implemented properly on border routers or on internal routers (see Fig. 2).
Figure 2: The differences in the functions of edge and internal routers

### 3 Priority queueing systems and QoS

Priority queueing systems constitute a large class of queueing systems in which the incoming requests are to be handled differently depending upon their importance. Efficient transmission of QoS traffic in such systems is obtained by appropriate queueing. Simple queueing as FIFO (First In First Out) preserves the order of the incoming packets, while other queueing disciplines can use more complex rules for processing the incoming packets. During congestion, packets can be selectively discarded in accordance with their probability of rejection. Typical routers use the first-come first-served rules on each interface output (see Fig. 3).

As long as the queue is not empty, the server (or “Processor” in Fig.3) forwards packets from the queue by selecting a package which is waiting (in the queue) for the longest time. One of major drawbacks of such a solution is that packets of higher priority do not receive any recognition and better service. This can be eliminated by using priority queueing which maintains multiple queues (see Fig. 4) for packets of different priorities. When a packet is needed for transmission, first the highest priority queue is checked, and if it is nonempty, the packet is selected from this queue. If the highest priority queue is empty, the next queue (in order of priorities) is checked, and so on. Consequently, the lowest priority packets can be forwarded only when all higher priority queues are empty.

Figure 3: FIFO queueing mechanisms

A priority queue can provide service of better quality service than the FIFO queue because higher priority traffic, such as VoIP or multimedia, can reach the destination faster. However, the disadvantage of this solution is the fact that higher priority traffic can block the lower priority queues. If there is continuous flow in a high-priority queue, the packet in the lower-priority queue will never have a chance to be processed. No bandwidth guarantees are possible in (pure) priority queueing.

More advanced quality–oriented queueing provides service on the “fair basis”, i.e., all (nonempty) queues are equally entitled to receive service (fair queueing), and, in the case of congestion, there is a weight associated with each queue that determines the fraction of the bandwidth allocated to this queue (weighted fair queueing).
4 The model and its performance

Several versions of Petri nets have been used as models of systems which exhibit concurrent and parallel activities. Stochastic Petri nets [2] and timed Petri nets [24] have many similarities but deal with temporal properties of models in a different way, so sometimes the similarities may be misleading.

The basic model used here is known as an inhibitor Petri net $\mathcal{N}$, which is a bipartite directed graph [12]:

$$\mathcal{N} = (P, T, A, H),$$

where $P$ and $T$ are two disjoint sets of vertices, $P \cap T = \emptyset$, called places and transitions, respectively, $A$ is a set of directed arcs connecting places with transitions and transitions with places, $A \subseteq P \times T \cup T \times P$, and $H$ is a set of inhibitor arcs connecting places with transitions, $H \subset P \times T$. Normally, $A \cap H = \emptyset$.

A place is shared if it is connected to more than one transition. A shared place $p$ is free-choice if the sets of places connected by directed as well as inhibitor arcs to all transitions sharing $p$ are identical. A net is free-choice if all its shared places are free-choice. Only free-choice nets are used in this paper.

$\mathcal{N}$ describes the static structure of a Petri net model. The dynamic aspects are represented by tokens, which are assigned to places, and which can change this assignment under some conditions. Marked Petri net, $\mathcal{M}$, is usually defined as a pair [14]:

$$\mathcal{M} = (\mathcal{N}, m_0),$$

where $\mathcal{N} = (P, T, A, H)$ and $m_0$ is the initial marking function, $m_0 : P \to \{0, 1, 2, ...\}$, which describes the initial distribution of tokens over the places of $\mathcal{N}$.

If $m$ is a marking in a net $\mathcal{N}$, $m : P \to \{0, 1, ...\}$, then a transition $t \in T$ is enabled by $m$ if all its input places are marked by $m$ and all its inhibitor places are unmarked by $m$:

$$t \text{ is enabled by } m \iff \forall (p, t) \in A : m(p) > 0 \land \forall (p, t) \in H : m(p) = 0.$$

Each transition $t$ enabled by a marking $m$ can occur (or fire) removing a single token from each of its input places and adding a single token to each of its output places. This creates a new marking $m'$ that is directly reachable from $m$ by occuring (or firing) $t$, $m \xrightarrow{t} m'$:

$$m \xrightarrow{t} m' \iff \forall p \in P : m'(p) = \begin{cases} m(p) - 1, & \text{if } (p, t) \in A \land (t, p) \notin A, \\ m(p) + 1, & \text{if } (t, p) \in A \land (p, t) \notin A, \\ m(p), & \text{otherwise}. \end{cases}$$

For performance analysis of Petri net models, temporal characteristics of occurring transitions must be taken into account. This can be done in several ways assigning occurrence timed to places, or to transitions, or even to arcs of the Petri net models. In timed Petri nets [24], occurrence times are associated with transitions,
and transition occurrences are real-time events, i.e., tokens are removed from input places at the beginning of the occurrence period, and they are deposited to the output places at the end of this period. All occurrences of enabled transitions are initiated in the same instants of time in which the transitions become enabled (although some enabled transitions cannot initiate their occurrences). If, during the occurrence period of a transition, the transition becomes enabled again, a new, independent occurrence can be initiated, which will overlap with the other occurrence(s). There is no limit on the number of simultaneous occurrences of the same transition (sometimes this is called infinite occurrence semantics). Similarly, if a transition is enabled “several times” (i.e., it remains enabled after initiating an occurrence), it may start several independent occurrences in the same time instant.

More formally, a free-choice timed Petri net $T$ is a triple:

$$T = (M, c, f),$$

where $M$ is a marked net, $c$ is a choice function which assigns free-choice probabilities to free-choice classes of transitions, $c : T \rightarrow [0, 1]$, and $f$ is a timing function which assigns an (average) occurrence time to each transition of the net, $f : T \rightarrow \mathbb{R}^+$, where $\mathbb{R}^+$ is the set of nonnegative real numbers.

The occurrence times of transitions can be either deterministic or stochastic (i.e., described by some probability distribution function); in the first case, the corresponding timed nets are referred to as D–timed nets [23], in the second, for the (negative) exponential distribution of firing times, the nets are called M–timed nets (Markovian nets) [22]. In both cases, the concepts of state and state transitions have been formally defined and used in the derivation of different performance characteristics of the model. In simulation applications, other distributions can also be used, for example, the uniform distribution (U–timed nets) is sometimes a convenient choice. Moreover, different distributions can be associated with different transitions in the same model providing flexibility that is used in simulation examples that follow.

In timed nets, the occurrence times of some transitions may be equal to zero, which means that such occurrences are instantaneous; all such transitions are called immediate (while the others are called timed). Since the immediate transitions have no tangible effects on the (timed) behavior of the model, it is convenient to ‘split’ the set of transitions into two parts, the set of immediate and the set of timed transitions, and to first perform all occurrences of the (enabled) immediate transitions, and then (still in the same time instant), when no more immediate transitions are enabled, to start the occurrences of (enabled) timed transitions. It should be noted that such a convention effectively introduces the priority of immediate transitions over the timed ones, so the conflicts of immediate and timed transitions are not allowed in timed nets. Detailed characterization of the behavior or timed nets with immediate and timed transitions is given in [24].

Fig.5 shows a Petri net model of the priority queueing (Fig.1) with a common source (left) and with independent sources for each class of traffic (right).

**Figure 5:** Petri net models of priority queueing with “integrated” source (left) and “independent” sources (right).

For the ‘integrated’ model, place $p_0$ with transition $t_0$ model the source of packets (with exponentially distributed inter-arrival times, controlled by the occurrence time $f(t_0)$, the average inter-arrival time. Place $p_{01}$ is shared by transitions $t_{01}$, $t_{02}$ and $t_{03}$ which constitute a free-choice class, with choice probabilities $c(t_{01})$, $c(t_{02})$
and $c(t_{03})$ such that $c(t_{01}) + c(t_{02}) + c(t_{03}) = 1$. In some of the performance studies these three probabilities are equal, in others they are very different to show the effects of one class of traffic on the performance of other classes.

The three places, $p_1$, $p_2$ and $p_3$, are queues for class–1, class–2 and class–3 packets, respectively. It is assumed that all queues have infinite capacities. It is also assumed that class–1 (place $p_1$) has the highest priority and class–3 (place $p_3$), the lowest priority. Transitions $t_1$, $t_2$ and $t_3$ model the transmission channel which can be used by class–1 packets ($t_1$), class–2 packets ($t_2$) or class–3 packets ($t_3$). All three transitions are timed, and the occurrence times model the transmission times of packets (in this case, a deterministic time equal to 1 time unit; for simplicity, the same for all packets). Place $p_0$, shared by these three transitions, is marked by a single token (i.e., $m_0(p_0) = 1$) which guarantees that only one transition can be occurring at each moment of time. The inhibitor arc $(p_1, t_2)$ does not allow $t_2$ to occur if there is a class-1 packet waiting for transmission. Similarly, inhibitor arcs $(p_1, t_3)$ and $(p_2, t_3)$ allow $t_3$ to occur only if there are no class–1 or class–2 packets waiting for transmission.

For the ‘independent’ model (Fig.5 right), the common source of packets, $t_0$, is replaced by three independent sources, $t_{01}$, $t_{02}$ and $t_{03}$, so any arrival rate can be easily changed without affecting the others. Fig.6 shows the average waiting times for classes 1, 2 and 3, as functions of traffic intensity (for the ‘integrated’ model) when the distribution of packets among the classes is equal (left) and when it is very unequal (right).

![Figure 6: Average waiting times for ‘integrated’ source](image)

When the traffic intensity approaches 1, the most affected class is class–3, the lowest priority class. Because of priorities, it can be assumed that the traffic intensity $\rho$ is composed of three equal parts (Fig.6 left) with the lowest values of traffic intensity corresponding to class–1 (the highest priority), the middle part of traffic intensity corresponding to class–2, and the highest values of traffic intensity corresponding to class–3 (the lowest priority). In the right part of Fig.6, class–2 is also affected by high values of traffic intensity as it corresponds to the second highest 5% of traffic intensity.

Fig.7 shows average waiting times as functions of traffic intensity in one class only (the ‘independent’ model). It should be noted that the traffic in class–3 (Fig.7 right) has practically no influence on the waiting times of classes 1 and 2.

Fig.8 shows the utilization of the transmission channel for class–1, class–2 and class–3 packets as functions of traffic intensity in class–1 (the ‘independent’ model).

Fig.8 shows some “blocking effects” created by priority queueing. For the case $\rho_2 = \rho_3 = 0.25$ (Fig.8 left), as the traffic intensity $\rho_1$ increases, first the utilization of class 3 decreases to zero (which means that class 3 receives less and less service), and then class 2 is similarly blocked. For traffic intensity close to 1, there is practically no transmission of packets of class 2 and 3.

It should be noted that, in Fig.8 left, for $\rho_1 \geq 0.5$, class 3 becomes nonstationary, and for $\rho_1 \geq 0.75$, so is class 2. Similar effects can be observed in Fig.8 right.
Figure 7: Average waiting times as functions of traffic intensity $\rho$ for $\rho_2 = \rho_3 = 0.25$ (left) and as functions of $\rho_3$ for $\rho_1 = \rho_2 = 0.25$ (right).

Figure 8: Channel utilizations as functions of traffic intensity $\rho_1$ for $\rho_2 = \rho_3 = 0.25$ (left) and $\rho_2 = 0.5, \rho_3 = 0.25$ (right).

5 Concluding remarks

Petri nets are used increasingly often for modeling and performance analysis of computer networks [15]. However, the models can easily become rather complex, with hundreds of elements, and then efficient analysis cannot be performed without the use of flexible and robust software tools. Many such tools are available, but only a few can be used for realistic applications.

Also, hierarchical modeling, in which some parts of the model are represented at a very fine level of detail while other are quite general, can easily be supported by Petri nets. Moreover, when the properties of detailed models are known, they can be incorporated into more general models, increasing their accuracy.

There are many other aspects of computer networks than can (and most likely will) be studied in the future. One of them is the influence of the distribution function of the inter-arrival times on the performance of the system and the traffic shaping techniques, another is a more realistic representation of packet transmission times which should take packet lengths and packet length distributions into account.

References

(BTH), Karlskrona, Sweden 2011.


