On Short Packets First: A delay-oriented prioritization policy

G. Papastergiou[1], C. Georgiou, L. Mamatas and V. Tsaoussidis

Department of Electrical and Computer Engineering
Democritus University of Thrace
12 Vas. Sofias Str.
67100, Xanthi, Greece

ABSTRACT

We propose a new scheduling discipline that allows for efficient interoperation of dedicated systems (such as sensor and voice networks) with the Internet. More precisely, our approach provides delay guarantees to applications that do not contribute to congestion because of their tiny packet sizes and low transmission rates. In addition, our approach uses a second level of prioritization that conditionally favors, in terms of delay, applications with slightly longer packets as well. Non-Congestive Queuing Plus (NCQ+) promotes applications that require comparatively small service times, as long as their total service times cause insignificant delays to other packets in the queue. Therefore, we prioritize packets, and in turn corresponding flows, according to their impact on total delay. We evaluate NCQ+ using ns-2 based experiments.

Keywords: Service Differentiation, NCQ+, QoS, LIBS

1. INTRODUCTION

Congestive applications may monopolize communication resources and cause major queuing delays to applications that have minor contribution to congestion. Such applications do not really cause significant delays, raising naturally the issue of whether they deserve a prioritized service or not. For example, a sensor-generated packet could experience almost-

[1] Corresponding author. Tel.: 0030 694262013.
E-mail address: gpapaste@ee.duth.gr (G. Papastergiou).

zero delay had it been favored by a prioritized scheduling scheme, at an almost-zero cost to other congestive flows [16].

The typical best-effort service does not treat fairly applications that do not contribute to congestion, but have to suffer delays caused by other applications. For example, in emergent situations (e.g., an earthquake) a single message from a body-sensor with location information of someone that needs help, although it causes zero delays, it may be lost or severely delayed by congestive applications that monopolize communication resources.

Departing from such situation, we introduce a new service philosophy for packets, called Less Impact Better Service (LIBS) [15-17]. In general, LIBS differentiates service among packets based on the delay they cause and as long as the cumulative impact of prioritization is minor. Since delay is highly correlated to resource allocation, in the context of delay-sensitive applications, efficient resource allocation needs to be characterized by the delay suffered by each flow with regard to the delay they cause. The latter is also implicitly associated with the delay that users tolerate. That is, LIBS service allocation paradigm, provides a fairer, delay-wise, service, and increased user satisfaction for applications that utilize small packets and rates. Other applications do not suffer significant extra delays.

In [15-17] we proposed a novel scheduling discipline, namely Non-Congestive Queuing (NCQ) that realizes the LIBS philosophy and which provides a single service prioritization level for applications that have minor contribution to congestion. In this work, we introduce a new packet scheduling mechanism, namely Non-Congestive Queuing Plus (NCQ+) which extends NCQ towards a more sophisticated service differentiation. NCQ+, as NCQ, is a system-oriented service discipline based on LIBS philosophy, which targets in satisfying more users.

In this context, we need to elaborate on the statistical properties of congestive and non-congestive applications. Non-congestive applications are not determined solely by the size of

packets they use, but by the impact on total system delay. Thus, non-congestive applications are characterized by the small, percentage-wise, volume of transmitted data. Typical sensor applications transmitting data through the Internet and utilizing tiny packets at small rates can be considered as non-congestive. Furthermore, VoIP packets may conditionally satisfy the non-congestive property as long as their impact on the other application is statistically insignificant.

NCQ+ guarantees that applications transmitting tiny volume of data, such as typical sensor applications, and thus do not contribute to congestion, will suffer zero delays in the Internet. Moreover, other non-bandwidth-demanding applications, such as VoIP applications, which may have minor impact on the total system delay will benefit from a conditional prioritization service, as long as the extra delay of congestive applications is not significant and the guarantees for non-congestive applications are not violated. Thus, NCQ+ allocates fairly buffer resources to non-congestive applications through the packet scheduler, while traditional Internet applications, including FTP, will experience statistically insignificant extra delays.

Buffer resource allocation is controlled by two thresholds, namely $ncqthresh_1$ and $ncqthresh_2$. We consider that the former is fixed and defines, percentage-wise, the buffer space available for service prioritization. Thus, $ncqthresh_1$ points to the level of buffering that corresponds to the maximum amount of delay that can be considered statistically insignificant to the congestive flows. $Ncqthresh_2$ operates within this level aiming at service differentiation among non-congestive packets only. In this context, it further classifies tiny and small packets (e.g., sensor and voice packets respectively) according to their impact on the total system delay. Tiny packets, and thus corresponding applications, receive *Guaranteed Service* (GS) as long as their traffic volume is below $ncqthresh_1$, while small packets receive prioritization by utilizing the available space between the two thresholds (*Prioritized Service*, PS). Thus,

typical congestive applications receive a *Best-Effort Service* (BES), by exploiting the remaining buffer space. Although NCQ+ deals with only two distinct classes of non-congestive packets, it can be easily extended to work with more classes as well.

Nowadays, Internet telephony is steadily gaining popularity, while mobile VoIP (i.e., over Wi-Fi or WiMax) becomes an important service. Additionally, an increasing number of autonomous sensor networks interoperate with the Internet for sensor data delivery and monitoring. The varying delays and loss characteristics of the Internet have a significant impact on the performance of such applications. Thus, typical sensor and VoIP applications require certainly differentiated, yet somewhat distinctive service.

Classic differentiation schemes require identification of applications/flows or alternatively, a verifiable packet marking technology. In order to avoid the cost of packet preparation for differentiated services, we take advantage of two distinctive properties of non-congestive applications: the small size and small data volume of their packets. The key idea of Non-Congestive Queuing Plus (NCQ+) that we discuss here departs from the operational dynamics of gateways: they may service small packets instantly. Non-congestive flows do not cause significant delays and hence should not suffer from delays. Due to its simplicity, the proposed strategy is easily deployable and practically useful; with a minor modification effort, user service can have major improvement. NCQ+ does not require any modification at the transport protocol or packet marking; a minor modification of the gateway's software is sufficient.

Although our approach sounds straightforward, the system properties and design details reveal interesting dynamics. We show that NCQ+ improves real-time communication capability of sensor devices and conditionally enhance VoIP calls quality, without any significant service degradation for congestive flows. The minor occasional cost on congestive applications is not perceivable by their users.

The structure of the paper is the following: In section 2 we discuss the related work. In section 3 we provide the pseudo-code of NCQ and NCQ+ and present the basic assumptions and fundamental concepts. In section 4 we present and justify our evaluation plan and metrics. Next, we present the results, analysis and justification. In section 5 we summarize our conclusions.

2. RELATED WORK

Recent approaches to service differentiation rely on overlay architectures and services. They span across a spectrum of architectures and protocols that support call admission control that inevitably wastes resources and impacts other concurrent applications, or application marking along with priority scheduling that results on application-oriented instead of packet-oriented services.

Internet service differentiation is application-specific and naturally oriented either by some explicit and strict flow characteristics or by some application class. Even in the latter case, associating application types with service classes introduces inevitably some 'classification' cost, relevant with the granularity of classification scale; and requires a rather sophisticated implementation, ranging from packet marking, to shaping, scheduling and dropping schemes. Beyond that, application-based network services inherently involve predetermined requests that the network - one way or another - needs to satisfy, leading it to a prescribed behavior that may not permit maximum system performance. Thus, application-based services are occasionally associated with limited operational flexibility for the network, which in turn may lead to degraded system performance.

Perhaps network engineering would have been different had the pressing demand of application requirements been ignored. For example, a natural principle to lead the design of network services (and consequently the service differentiation policy) could have been the

network ability to function, the number of users serviced better without damaging the rest, or the service offered on the basis of the cost to other applications. For instance, it is not unreasonable to service first applications that require minimal time for service; in that case the gain for such applications can be significant, while the cost for the other applications may be small.

A similar, system-oriented scheduling concept has been studied in operating systems, where some schedulers select processes based on their completion time, rather than the time they started (shortest job first). Such a service alone may lead to starvation in case the rate of small processes is sufficient to keep the processor busy; processes demanding more time for completion could never get their turn. However, due to the cost of context switch, the lack of precision in estimating cost-per-process and the limited concurrent presence of processes, this domain had limited scheduling flexibility; our service differentiation scheme guarantees better service for non-congestive data only as far as the service of congestive applications is not degraded. In support of this goal, [1] confirms that the average delay for the system tends to be reduced when customers with short service times are given high priority.

A lot has been done in the networking community aiming at controlling traffic based on specific application requirements. According to the *intserv* approach [11], functionality of network components needs to allow for guarantees through signaling and reservation. In turn, *intserv* requires per-flow state information, which limits scalability. Other approaches require overlay architectures along with protocol modifications, which realize the corresponding relation between application classes and packet marks. Inevitably, class-based approaches (such as *diffserv* [10], [2], CBT [20] or [19]) overcome the limitation of state information but limit similar traffic to the same QoS class, introducing a 'classification' error. For example, DCBT with ChIPS [4] extends CBT by providing dynamic thresholds and lower jitter for multimedia traffic. However, it assumes that all multimedia traffic require the same QoS. In

[9], the authors introduced the Alternative Best Effort mechanism (ABE). ABE improves performance of delay-sensitive traffic but uses only two possible traffic classifications: delay- and throughput-sensitive. Delay-sensitive applications sacrifice throughput, and vice versa. Traffic Sensitive QoS mechanism (TSQ) [6] allows applications to indicate via marking their preferable delay / throughput sensitivity at packet-level. However, their approach does not imply service per-packet. That is, although applications may mark all or selected packets only, the basis for marking is not the current network state, which is unknown[1] by the application, but rather the specific properties of a packet. The dynamics of such service interactions imply application-level QoS indeed. Application level QoS, has therefore two undesirable properties: first, it requires a prescription-based network operation, which confines network flexibility to reach an optimal operating point, system-wide, and second, it introduces a 'classification' error. Note that in an effort to cancel the 'classification' error one has to introduce more detailed categories and classes; that is, to trade it with processing overhead.

LIBS, however, can be integrated into application-oriented QoS strategies and produce a hierarchical service dynamic: service-oriented prioritization, with application-specific requests therein. For example, a favorably-marked large packet will not be serviced first if the following packet will have zero impact to its service. Therefore, different priorities can be assigned via packet marking to the different non-congestive or congestive applications.

In references [15-17], we introduced Non-Congestive Queuing (NCQ) and showed that NCQ satisfies more users with diverse demands on delay and throughput. However, NCQ deals only with one aggregating class of non-congestive applications (e.g., both sensor and VoIP packets receive the same priority). Here, we introduce NCQ+, which is designed to provide different levels of priority to selected non-congestive applications. For example,

[1] even if it is measured, the precision and granularity of measurements are dubious

NCQ+ can provide guarantees to sensor applications, while favoring conditionally VoIP calls. Other service strategies for delay-sensitive traffic include Low Latency Queuing (LLQ) [5]. LLQ is a mechanism that provides a strict Priority Queue (PQ) to Class-Based Weighted Fair Queuing (CBWFQ) [21]. With LLQ, delay-sensitive data (such as voice and sensor traffic) is dequeued and serviced first. In addition, such type of traffic can be favored by a resource reservation protocol (such as RSVP [22]), which requests, collectively, a certain amount of bandwidth and latency at every RSVP-enabled network hop.

3. LIBS-BASED POLICIES

In this section, we discuss two alternative LIBS-based policies: the Non-Congestive Queuing (NCQ) and the Non-Congestive Queuing Plus (NCQ+).

3.1 Non-Congestive Queuing (NCQ)

Non-Congestive Queuing (NCQ) deals with only one class of non-congestive packets and assumes that prioritized non-congestive traffic cannot exceed a predetermined threshold, called *ncqthresh*, which represents the upper limit of permitted prioritized service. The threshold typically reflects a service percentage for prioritization. However, this percentage corresponds to the number of packets; not the occupied buffer space. Indeed, since service prioritization applies for small packets only, the queue size that corresponds to the prioritized packets, percentage-wise, is much smaller.

Although the perspective of NCQ is more general, initially, NCQ deals with two classes of packets: small packets[1], and long packets, which typical Internet applications use for data transfers (i.e., 1040 bytes - in our case). NCQ uses priority queuing to implement priority service. That is, within the same buffer, each packet is checked for its length, contrasted to the

[1] In our experiments, this class includes both sensor and VoIP packets.

current state of prioritized service rate and gets priority whenever it satisfies two conditions: (i) length is below a maximum value[1] for small-identified packets, namely *size_thresh*, and (ii) prioritized service rate is below *ncqthresh*.

In Appendix A we attempt to approach numerically the impact of NCQ priority on congestive traffic for any given proportion of traffic classes [17].

At this stage of our work, we exclude ACKs from prioritization even though they have a small size. This tactic is expected to increase transmission rate; how far this can happen (considering also the delayed-ACK scheme which is widely deployed) and how far it can impact congestion control, is an open issue. We note that NCQ may occasionally favor a part of a non-congestive data flow. However, a possible slight increase[2] of the re-ordered packets is counterbalanced by the high number of packet drops that are avoided due to the prioritization. Figure 1 shows the pseudocode of NCQ.

```
for every received packet
begin
   count received packets
   if (packet_length < size_thresh) and
       favored_packets / received_packets < ncqthresh)
   then
      packet receives high priority
      count favored packets
   else
      packet receives normal priority
   end
end
```

Figure 1. NCQ pseudocode

3.2 Non-Congestive Queuing Plus (NCQ+)

3.2.1 FUNDAMENTAL CONCEPTS

Here, we attempt to extend NCQ towards a more sophisticated service prioritization scheme that considers more non-congestive applications. That is, we differentiate non-

[1] We experimentally set this value to 150 bytes.
[2] It is not significant in our experiments.

congestive applications further and produce two application classes: those which utilize tiny packets (e.g., typical sensor data); and those which utilize small packets (e.g., typical VoIP applications). In the context of insignificant impact on other applications, Non-Congestive Queuing Plus (NCQ+) provides Guaranteed Service (GS) to applications utilizing tiny packets at low data rates, while other non-congestive applications receive conditionally Prioritized Service (PS).

Tiny packets do not cause statistically important delays, so it is reasonable to be promoted. Additionally, applications with small packets at low data rates have a minor contribution to congestion and can be prioritized[1] as long as their cumulative impact on total system delay is insignificant. Therefore, prioritization should be bounded by a service threshold similar to *ncqthresh*, namely $ncqthresh_1$. Inevitably, since the impact is limited, service guarantee is not always possible; and since resource allocation favors selected applications only, application service is better than simply differentiated. We call this type of service: "better-guaranteed".

NCQ+ assumes two distinct classes of non-congestive packets: tiny packets (e.g., sensor-generated packets, 40 bytes in our experiment) and small packets (e.g., VoIP packets, 140 bytes in our experiment). However, NCQ+ can be easily extended to cover more classes as well. In particular, it introduces a second level of prioritization by using a second traffic threshold, namely $ncqthresh_2$. The first level of prioritization is enabled when a single condition is satisfied (i.e., when the total prioritized service rate is below $ncqthresh_1$), while the second level requires an additional condition to be satisfied: prioritized service rate for the second class must be below $ncqthresh_2$. NCQ+ applies the former to applications that utilize tiny packets at low data rates and the latter to other non-congestive applications. Figure 2 (a) illustrates the allocated buffer resources for prioritized service and the relation between the two thresholds. $Ncqthresh_1$ is fixed and defines, percentage-wise, the allocated buffer space

[1] as with NCQ scheduling discipline

for service prioritization. That is, it points the maximum prioritized service rate of non-congestive packets that causes statistically insignificant amount of delay to congestive applications. $Ncqthresh_2$ limits the prioritized service rate of small packets to a level that does not confine the allocated resources to tiny packets.

We define $X_{tiny}(t)$ as the percentage of the incoming tiny packets at time t. To simplify the following analysis we always assume that $X_{tiny}(t) < ncqthresh_1$. That is, there are always prioritization buffer resources for providing Guaranteed Services to tiny packets. In this context, in order for NCQ+ to provide GS, the following equation should hold:

$$ncqthresh_2(t) \le ncqthresh_1 - [X_{tiny}(t) + e(t)] \qquad (1)$$

where $e(t)$ is a safety margin for provided GS and is discussed in next section. Obviously, $ncqthresh_1$ is the maximum value of $ncqthresh_2$. Given that the demand for guaranteed prioritized service by applications utilizing tiny packets may vary, $ncqthresh_2$ should be able to adapt accordingly. We discuss this process in the following section.

We distinguish packets based solely on their length. Thus, we define two size thresholds, namely $size\_thresh_1$ and $size\_thresh_2$. Packets with length lower than $size\_thresh_1$ are identified as tiny, while the size for a small-identified packet ranges from $size\_thresh_1$ to $size\_thresh_2$. To summarize, a tiny packet is favored if the total prioritized service rate for non-congestive traffic is below $ncqthresh_1$, while the prioritization for a small packet is confined to $ncqthresh_2$.

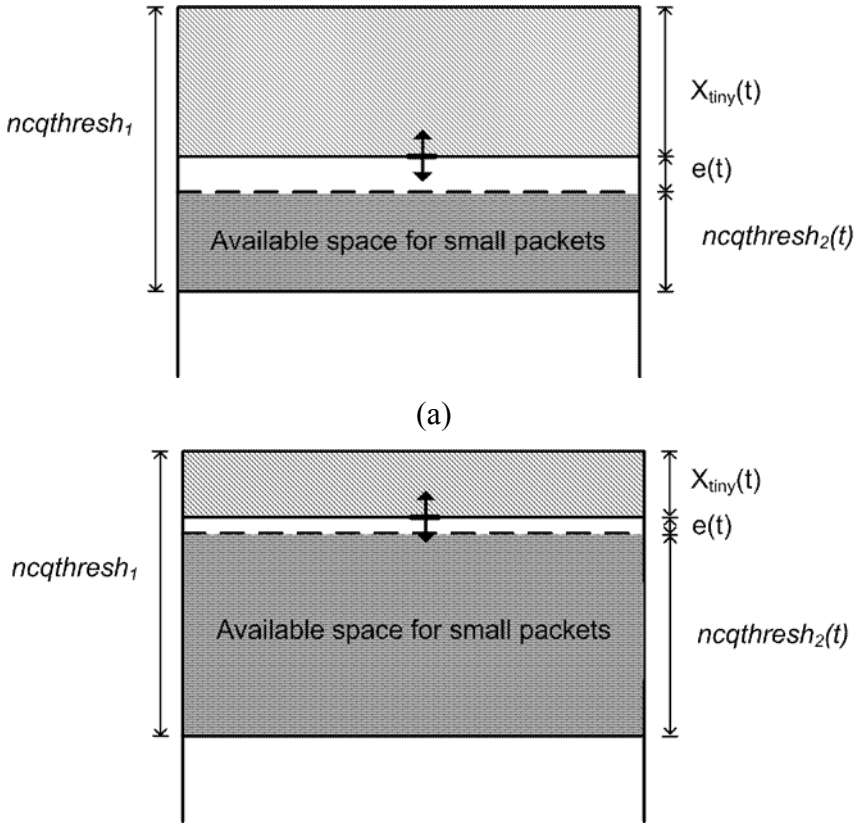


Figure 2. (a) Allocated buffer resources for service prioritization and (b) *ncqthresh*$_2$ adaptation

3.2.2 PRIORITIZED SERVICE

NCQ+ provides Guaranteed Service to tiny packets, when the allocated prioritization buffer space to this class of traffic is at least equal to the percentage $X_{tiny}(t)$ of the incoming tiny packets. However, NCQ+ leaves a safety margin $e(t)$ so as to satisfy a potential increase of $X_{tiny}(t)$. The safety margin $e(t)$ is proportional to $X_{tiny}(t)$, with a proportional coefficient α. Thus:

$$e(t) = \alpha \cdot X_{tiny}(t),\ 0 < \alpha < 1 \qquad (2)$$

From the above equation it comes that:

$$ncqthresh_2(t) = ncqthresh_1 - (1+\alpha) \cdot X_{tiny}(t) \qquad (3)$$

where $(1+\alpha)\cdot X_{tiny}(t)$ defines the allocated resources to applications utilizing tiny packets.

As mentioned earlier, *ncqthresh*$_2$ adapts to the current state of the prioritized service demand by applications transmitting tiny packets. This adaptation is triggered by either of two events: (a) the non-prioritization of a tiny packet; and (b) the non-prioritization of a small packet. When any of these events occurs, *ncqthresh*$_2$ is recalculated using Equation 3. The first event is triggered only when the percentage of tiny packets X_{tiny} plus the percentage of favored small packets Y_{small} exceeds *ncqthresh*$_1$. Thus, the following equation holds:

$$X_{tiny}(t+t_1)+Y_{small}(t+t_1)\geq ncqthresh_1 \qquad (4)$$

where $t+t_1$ is the time that at which a tiny packet is not favored. For the favor rate $Y_{small}(t)$ of small packets the following equations hold:

$$Y_{small}(t)=X_{small}(t) \text{ when } X_{small}(t)\leq ncqthresh_2(t)\ \forall\ t\in(0,\infty) \qquad (5)$$

and

$$\max\{Y_{small}(t)\}=ncqthresh_2(t) \qquad (6)$$

Since prioritization of small packets is constrained by two prioritization levels, the second event applies either to the case where the total prioritization service for non-congestive packets exceeds *ncqthresh*$_1$ or to the case where the percentage of small packets becomes higher than the *ncqthresh*$_2$. The first one is similar to the non-prioritization case described before and leads to a decrease in *ncqthresh*$_2$. In the second case, and the most important one, the scope of the recalculation of *ncqthresh*$_2$ is dual. As X_{tiny} decreases and thus the allocated resources to applications utilizing tiny packets are too generous, *ncqthresh*$_2$ increases smoothly. That is, each time a small packet is not favored due the restriction of *ncqthresh*$_2$, this threshold is recalculated in order to investigate potential available resources. We note that the safety margin $e(t)$ is preserved. Figure 2b illustrates the above process. However, *ncqthresh*$_2$ also decreases whenever X_{tiny} has increased and belongs to the interval

$(X_{tiny}(t), X_{tiny}(t)+e(t))$. The following equations hold for the event of a non-prioritized small packet:

$$X_{tiny}(t+t_2)+Y_{small}(t+t_2) \geq ncqthresh_1 \quad (7)$$

or

$$Y_{small}(t+t_2) \geq ncqthresh_2(t) \quad (8)$$

where $t+t_2$ is the time that at which a tiny packet is not favored. Figure 3 shows the pseudo-code for NCQ+.

```
for every received packet
begin
  count received packets
  if (packet_length ≤ size_thresh1) and
     (total_prioritized_service < ncqthresh1)
  then
    packet receives high priority
    count favored tiny packets
  else
     ncqthresh2 = ncqthresh1 - k*tiny_packets_favor_rate
     packet receives normal priority
  end
  if (size_thresh1 < packet_length ≤ size_thresh2) and
     (small_packets_favor rate < ncqthresh2) and
           (total_prioritized_service < ncqthresh1)
  then
     packet receives high priority
        count favored small packets
  else
        ncqthresh2 = ncqthresh - k*tiny_packets_favor_rate
        packet receives normal priority
  end
end
```

Figure 3. NCQ+ Pseudo-code

4. EVALUATION METHODOLOGY

We have implemented our evaluation plan on the ns-2 network simulator [18]. We simulate VoIP traffic based on the following assumptions: During a conversation, speakers alternate between activity and idle periods. Taking into consideration the ON and OFF periods [3], as well as the heavy-tailed characteristics and self-similarity of VoIP traffic [8]

we use Pareto distribution for modeling the call holding times. We configure Pareto with a mean rate to correspond to transmission rate of 64kbps and the shape parameter is set to 1.5. In accordance with [3], we distribute ON and OFF periods with means 1.0s and 1.35s respectively. We simulate VoIP streams of 64kbps (following the widely-used ITU-T G.711 coding standard [7]) and we set packet sizes at 140 bytes (carries 15ms G.711-encoded speech plus a 20-byte packet header. We simulate the sensor flows by sending periodically packets of 40 bytes (20 bytes of sensor data plus a 20-byte packet header). The interval between two consecutive sensor transmissions is set to 50ms.

Non-congestive traffic (i.e, VoIP and sensor traffic) is transferred by UDP packets while congestive FTP traffic is carried by TCP. The TCP version we use is TCP NewReno and all implemented mechanisms are based on DropTail. In our evaluation, we focus on the impact of our proposal (i.e, NCQ+) and thus we use NCQ and DropTail as reference points. We investigate different scenarios, with different proportions of FTP, sensor and VoIP flows. We demonstrate that NCQ+:

i) is scalable

ii) favors non-congestive applications with insignificant impact on the performance of congestive flows.

iii) provides better guaranteed service to sensor-based applications.

iv) favors VoIP applications when resources are available.

4.1 Evaluation metrics

We measure application performance using *Goodput* defined as:

$$Goodput = \frac{Original\_Data}{Time} \tag{9}$$

where $Original\_Data$ is the number of bytes delivered to the high-level protocol at the receiver (i.e, excluding retransmitted packets and overhead), and $Time$ the amount of time required for the corresponding data delivery. We use the $Average\_Goodput$ in order to measure the efficiency per flow. The $Average\_Goodput$ for n flows is defined as:

$$Average\_Goodput = \frac{\sum_{i=1}^{n}(Goodput_i)}{n} \quad (10)$$

where $Goodput_i$ is the $Goodput$ of the i^{th} flow and n the flows number.

Since sensor applications are sensitive to packet losses, we additionally measure the efficiency of sensor applications based on the packet loss rate (PLR) experienced by each flow. PLR is the ratio of the number of lost packets over the number of packets sent by the application. Thus, we define $Average\_PLR$ as:

$$Average\_PLR = \frac{\sum_{i=1}^{n} PLR_i}{n} \quad (11)$$

where PLR_i is the packet loss rate of i^{th} flow and n the flows number.

We characterize the quality of voice communication using the R-Factor, which is included in the E-Model [12, 14] (an ITU-proposed analytic model of voice quality). R-Factor captures voice quality and ranges from 100 to 0, representing best and worst quality, respectively. R-Factor is also associated with the Mean Opinion Score (MOS). MOS is the arithmetic average of opinions where ”excellent” quality is represented by 5, ”good” by 4, ”fair” by 3, ”poor” by 2 and ”bad” by 1. R-Factor incorporates several different parameters, such as echo, background noise, signal loss, codec impairments and others. In [13], authors simplified E-Model to transport-level measurable quantities and resulted in a more suitable R-Factor formula. Based on the above, we define R-Factor as:

$$R = \alpha - \beta_1 d - \beta_2 (d - \beta_3) H(d - \beta_3) - \gamma_1 - \gamma_2 \ln(1 + \gamma_3 e) \quad (12)$$

where $\alpha = 94.2$, $\beta_1 = 0.024\text{ms}^{-1}$, $\beta_2 = 0.11\text{ms}^{-1}$, $\beta_3 = 177.3\text{ms}$, d expresses the mouth-to-ear delay and e the packet loss rate. For the G.711 codec, $\gamma_1 = 0$, $\gamma_2 = 30$, $\gamma_3 = 15$.

The R-Factor is related to the MOS through the following set of expressions:

For $R < 0$: MOS = 1

For $R > 100$: MOS = 4.5

For $0 < R < 100$: $\text{MOS} = 1 + 0.035 \cdot R + (R \cdot (R-60) \cdot (100-R) \cdot 7 \cdot 10^{-6})$

For reference purposes, we give the relation of R-Factor with MOS in Table 1.

Table 1. R-Factor, Quality ratings and MOS

R-Factor	Quality of voice rating	MOS
$90 < R < 100$	Best	4.35 – 4.5
$80 < R < 90$	High	4.02 – 4.34
$70 < R < 80$	Medium	3.60 – 4.03
$60 < R < 70$	Low	3.10 – 3.60
$50 < R < 60$	Poor	2.58 – 3.10

4.2 Evaluation scenarios

We conduct experiments with mixed traffic, including packets from VoIP and sensor applications. We use a complex topology (Figure 4), which incorporates multiple bottlenecks, cross traffic and wireless links. The router R1 is the bottleneck for the competing non-congestive VoIP and congestive FTP traffic, while router R2 is another bottleneck for non-congestive VoIP and sensor traffic, and congestive FTP traffic. To simulate mobile VoIP users ns-2 error models were inserted into the access links of VoIP source and sink nodes. We used the Bernoulli model in order to simulate wireless link errors, with packer error rate adjusted to 0.01. The bandwidth of each access link is 1 Mbps, while the propagation delay link varies deterministically from 8 to 12 ms. Queue size for all scenarios is 100 packets, $ncqthresh_1$ is set to 0.05, and each experiment's duration is 60s. We use three distinct

scenarios in order to evaluate the behavior of NCQ+ in different environments. Parameter α of NCQ+ algorithm is always equal to 0.1. We use NCQ and DropTail as reference points.

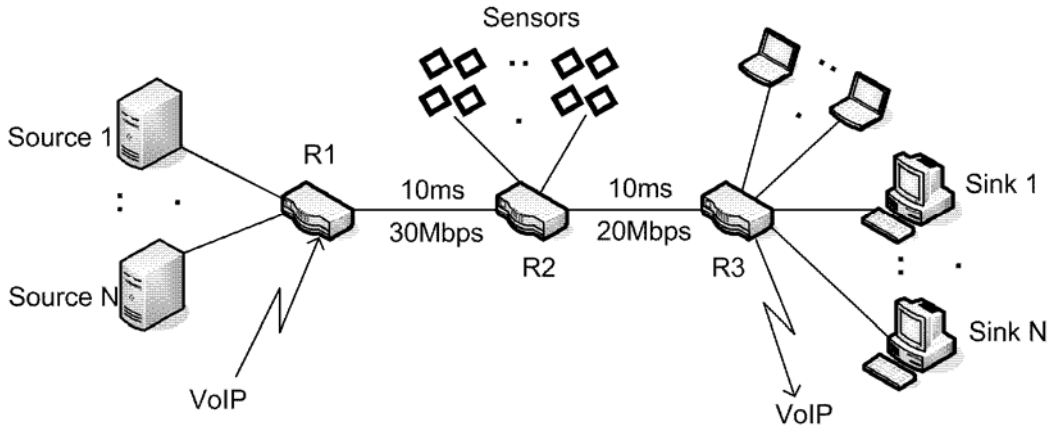


Figure 4. Network Topology

We describe the three scenarios below:

- **Scenario 1 - Scalability of NCQ+:** In the first scenario, we vary the total number of flows from 80 to 200, while maintaining a constant percentage of non-congestive flows. Thus, the 10% of the flows are VoIP calls and sensors, and the remaining 90% congestive FTP flows. We use an equal number of VoIP and sensor flows. This scenario allows us to draw conclusions on the scalability of the proposed mechanisms.

- **Scenario 2 – Impact of the VoIP load:** In this scenario, we capture the impact of NCQ+ on the performance gain of sensor and VoIP applications, compared to NCQ and DropTail, considering different levels of VoIP traffic. We set total number of flows to 120, while sensor flows are the 5% of the total (i.e, 6). The number of VoIP calls is adjusted from 2.5% to 15% of total flows (i.e., from 3 to 18 calls). Scenario 2 matches well the case of a natural disaster (e.g., an earthquake), at its early stages, where the number of mobile VoIP calls is expected to increase while less users will

continue to download files. If this is the case, critical sensor data (e.g., temperature sensors) should be, as much as possible, unaffected by this change.

- **Scenario 3 – Impact of the sensor-generated traffic:** In the last scenario, similarly to the previous one, we set the number of VoIP flows to 5% of a total of 120 flows. We range the number of sensors from 2.5% to 15% and we evaluate the impact of NCQ+ on the performance gains of sensor and VoIP applications. Going back to the previous example, scenario 3 fits well to the next stages of a natural disaster. FTP users will remain decreased and the temporary increase of VoIP will have been canceled, while more sensors may be activated for monitoring purposes.

In order to test the adaptation of $ncqthresh_2$, we generate conditions of dynamic contention in router R_2. More precisely, the incoming rate of sensor packets is changed dynamically throughout each experiment. Each sensor flow starts and stops transmitting data according to a deterministic pattern which leads to alternating periods of increasing, constant and decreasing sensor data rate. Eventually, each sensor flow transmits data for the half of the duration of each experiment (i.e., 30s).

4.3 Experimental Results

Scenario 1 – Scalability of NCQ+

As we can see in Figure 5 the $Average\_Goodput$ of FTP flows has almost no difference for a different total number of flows. Although non-congestive traffic is favored by both NCQ or NCQ+ (Figures 6 - 9), the impact on the congestive flows is insignificant. In Figure 6, we

observe the impact of NCQ and NCQ+ on the $Average\_Goodput$ of VoIP flows. Due to the non-congestive nature of the VoIP packets, VoIP calls are favored significantly by both mechanisms. VoIP flows benefit more when NCQ is applied, having a maximum gain of about 22% (in case of 200 flows) compared to NCQ+. This is a presumable result since NCQ+ guarantees prioritized service to sensor flows and confines the resources available to the VoIP. However, VoIP traffic is significantly favored by NCQ+ compared to DropTail (e.g., about 20% gain in case of 160 flows). As we can see from Figure 7, when NCQ+ is deployed, the $Average\_Goodput$ of sensor applications becomes equal to their data transmission rate. This happens because $ncqthresh_2$ fluctuates in order to give priority to every sensor packet. NCQ also improves the performance of sensor applications, however no service guarantees are provided.

Similar to the above observations both NCQ and NCQ+ significantly improve voice quality (Figure 9), as more calls are rated better. Again, VoIP flows benefit more when NCQ is deployed. In Figure 8 we illustrate the $Average\_PLR$ of sensor applications. Although NCQ reduces the packet loss rate experienced by sensor applications, it leads to a considerable large number of losses compared to NCQ+. When NCQ+ is deployed, sensor packets are unaffected by the state of the buffers and no packets are lost.

Comparing NCQ+ to NCQ, there is a tradeoff between the performance gain of VoIP applications and that of sensor ones. However, performance degradation of one class of traffic does not lead to an equivalent gain at the performance of sensor traffic. Total system gain can be improved. For example in the case of 120 flows, NCQ+, compared to NCQ, increases the $Average\_Goodput$ of sensor applications by 15.9%, while $Average\_Goodput$ of VoIP applications decreases by 8.4%.

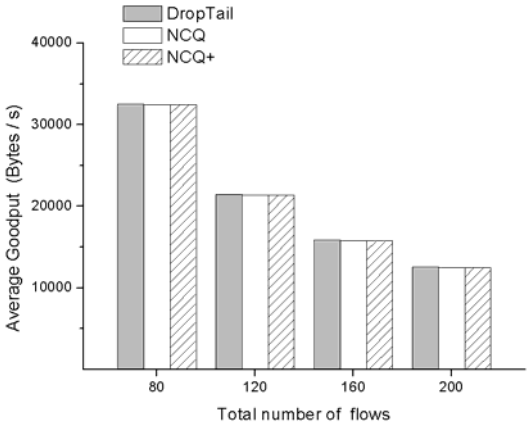


Figure 5. Average Goodput of FTP flows

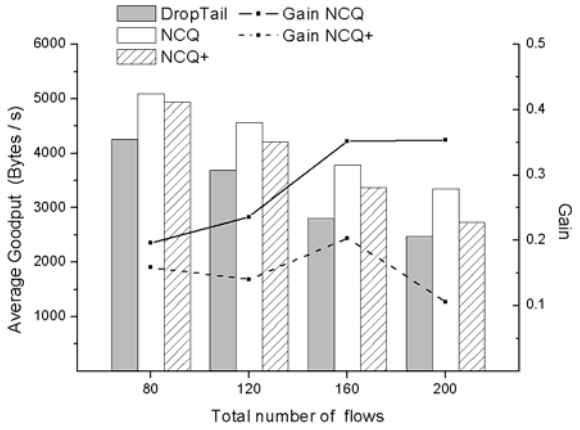


Figure 6. Average Goodput of VoIP flows

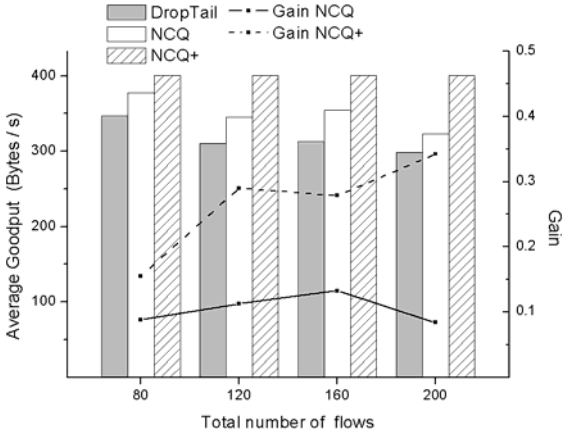


Figure 7. Average Goodput of sensor flows

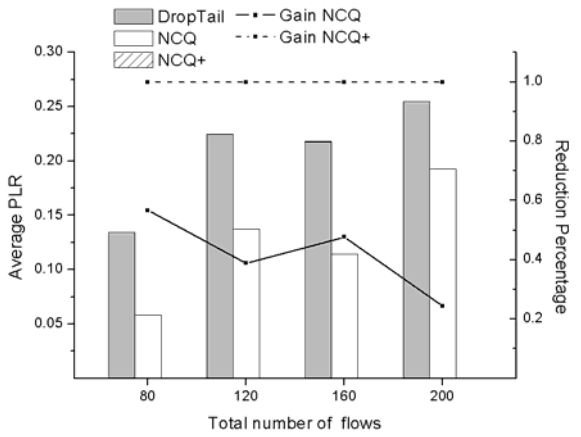


Figure 8. Average PLR of sensor flows

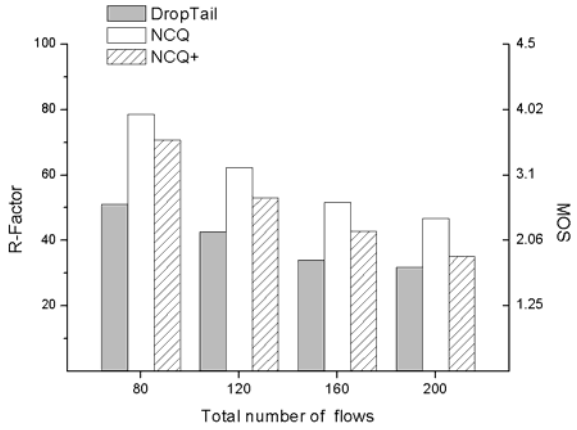


Figure 9. R-Factor

Scenario 2 – Impact of the VoIP load

In this scenario, we adjust the number of VoIP calls, while maintaining a constant number of sensor applications. As we can see from Figures 10 - 13 , non-congestive flows have significant improvement with both NCQ and NCQ+, in terms of *Goodput* , voice quality and *PLR* . While the number of VoIP flows increases, their performance improvement is decreased due to the limited number of available resources (e.g., 24,6% and 9,9% improvement in *Goodput* with NCQ+ for 2.5% and 15% VoIP flows, respectively). However, performance gains remain significant. In all cases NCQ improves VoIP performance more than NCQ+. The adaptation of $ncqthresh_2$ to the incoming rate of sensor

packets guarantees best *Goodput* performance (Figure 11) and zero packet losses (Figure 12) to sensor flows, while improves VoIP performance as much as possible (Figures 10 and 13). Sensor traffic prioritization is always unaffected by prioritized service demand of VoIP applications.

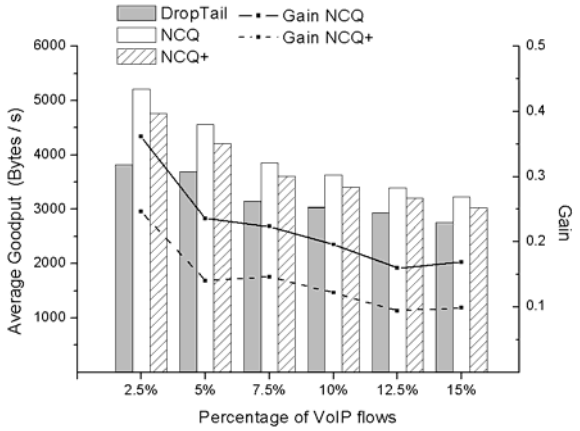


Figure 10. Average Goodput of VoIP flows

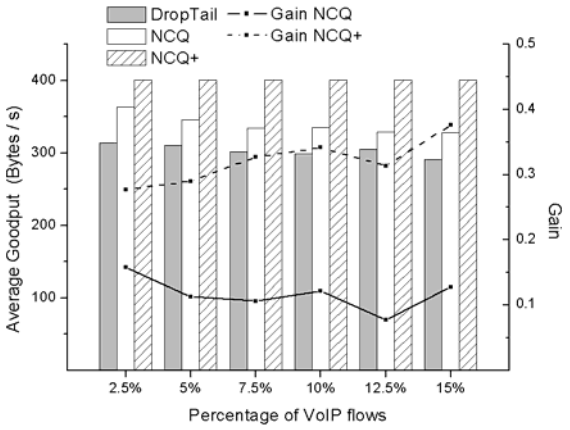


Figure 11. Average Goodput of sensor flows

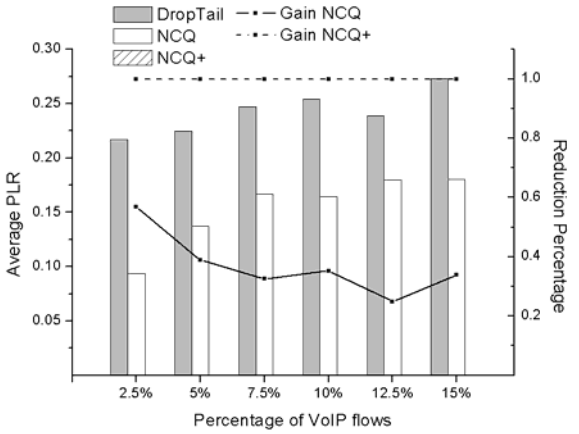


Figure 12. Average PLR of sensor flows

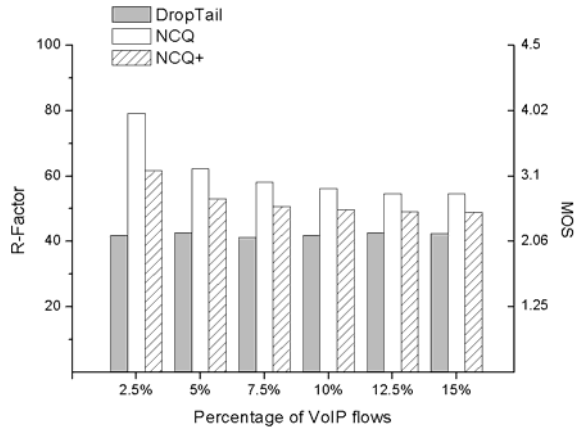


Figure 13. R-Factor

Scenario 3 – Impact of the sensor-generated traffic

In Figure 15 we can see that as soon as the percentage of sensor flows is low (up to 7.5%) the *Average _ Goodput* of sensor packets is maximized with NCQ+. The rate of sensor packets arriving at router R_2 does never exceeds the $ncqthresh_1$ and sensor flows receive Guaranteed Service. Further increase reduces *Goodput* gains and increases the *Average _ PLR* (Figure 16). NCQ+ achieves always a major improvement with a minimum gain of 10.6% when the percentage of sensor flows reaches 15%. Moreover, an increase in the percentage of sensor-generated traffic limits the benefits in *Goodput* (Figure 14) and quality (Figure 17) of voice calls. NCQ+ allocates as much resources as possible to sensor

applications, at the cost of a considerable less improvement on the performance of VoIP calls compared to NCQ. Thus, for sensor flows higher than 7.5% of the total, there is a non-equivalent gain on the performance of sensor applications. For example, for 18 sensor flows (i.e., 15% of the total), NCQ+, compared to NCQ, achieves a 4.7% improvement in *Average_Goodput* of sensor applications, while the *Average_Goodput* of VoIP calls is deteriorated by 10.2%. Same observations apply also to Figures 16 and 17.

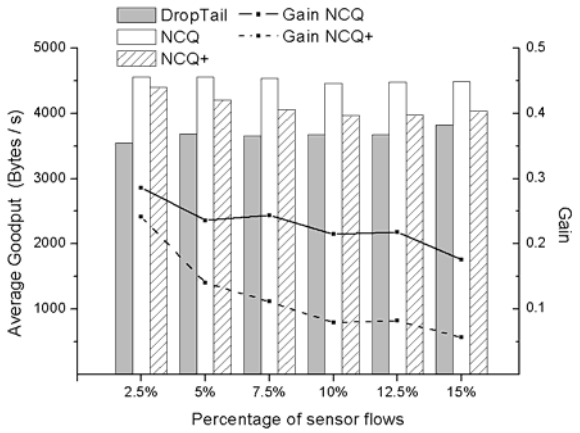


Figure 14. Average Goodput of VoIP flows

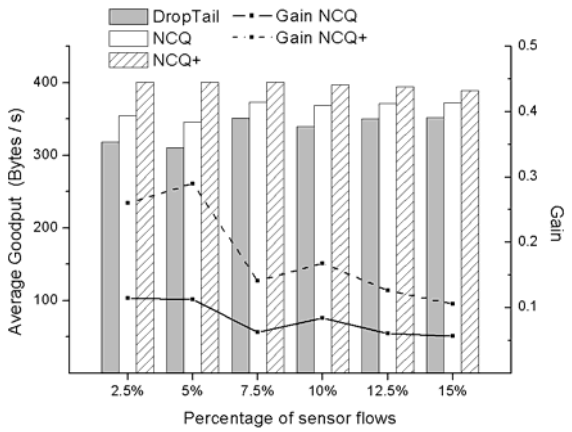


Figure 15. Average Goodput of sensor flows

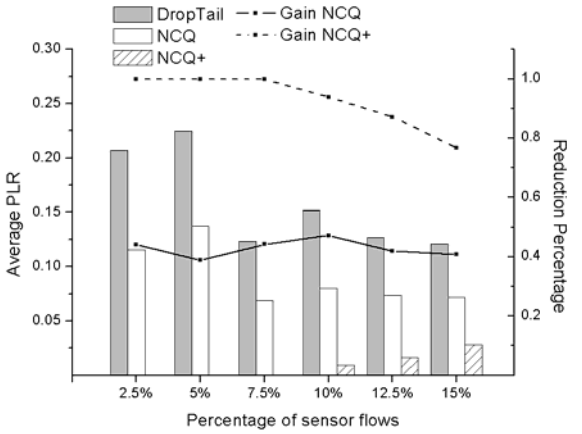


Figure 16. Average PLR of sensor flows

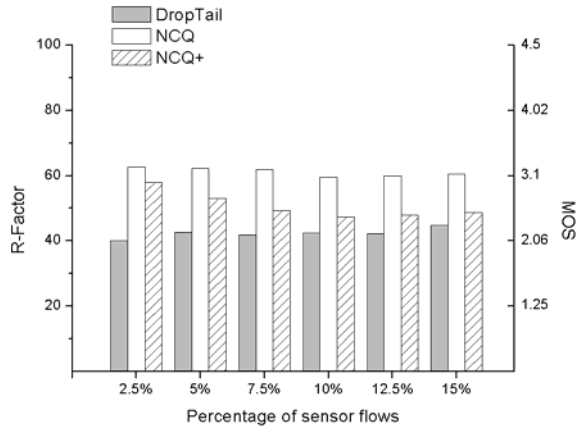


Figure 17. R-Factor

5. CONCLUSIONS

We proposed a new scheme for service differentiation based on a system-oriented approach which realizes the LIBS philosophy. Our scheme, however, does not conflict with existing application-oriented service differentiation technologies, such as marking. NCQ+ differentiates the service provided to sensor and VoIP applications, without damaging traditional internet applications. We demonstrated that NCQ+ can be adjusted to provide Guaranteed Services, in terms of delay, to sensor applications and conditionally improve the performance of VoIP applications, as long as Guaranteed Services are not violated and the

impact on the performance of the other flows is insignificant. Whenever prioritization buffer resources are not sufficient, NCQ+ provides "better-guaranteed" services to sensor applications. Further analytical and experimental results with more sophisticated mechanisms are underway.

APPENDIX A: NUMERICAL ANALYSIS OF THE IMPACT OF NCQ

We attempt to approach numerically the impact of NCQ priority on congestive traffic for any given proportion of traffic classes. We assume two classes of traffic (the non-congestive and congestive) that are formed by a large number of flows. We assume that all packets arriving at the bottleneck queue follow a Poisson distribution. Class 1 has priority over class 2. We use a non-preemptive head-of-line priority system per class. Class 1 has smaller packets (so, average service-time too) and lower packet-arrival rate ($\lambda_1 < \lambda_2$). We summarize our notation in Table I.

Table 1. Notation table

Symbol	Description
λ_1	Arrival rate of class 1
λ_2	Arrival rate of class 2
T_{S1}	Average service-time of class 1
T_{S2}	Average service-time of class 2
$\lambda = \lambda_1 + \lambda_2$	Total arrival rate
$u_1 = \lambda_1 \cdot T_{S1}$	Utilization of class 1
$u_2 = \lambda_1 \cdot T_{S1} + \lambda_2 \cdot T_{S2}$	Cumulative utilization
T_{Q1}	Average queuing delay for class 1
T_{Q2}	Average queuing delay for class 2
T_Q	Average queuing delay

We define the following:

Waiting Time: Waiting time represents the amount of time a packet waits for service in the queue.

Service Time: Service time represents the amount of actual service time required by a packet and is proportional to its size.

Time-in-System: Time-in-system equals to the Waiting Time plus Service Time (in our case is the same as Queuing Delay).

The packet-departure rate equals to the service distribution, because we are using a single server.

In the three different cases of prioritization below, we calculate the average queuing delay for each class and for the system:

1. Class 1 has full priority over class 2.
2. The two classes have the same priority (scheduling without priority).
3. Class 1 has priority over class 2 but only when less than the *ncqthresh* percentage of the total traffic is prioritized.

Case 1: Priority Scheduling: We calculate the average waiting time for each of the two classes as:

$$T_{W1} = \frac{\lambda_1 \cdot T_{S1}^2 + \lambda_2 \cdot T_{S2}^2}{2 \cdot (1 - u_1)} \quad (1)$$

$$T_{W2} = \frac{\lambda_1 \cdot T_{S1}^2 + \lambda_2 \cdot T_{S2}^2}{2 \cdot (1 - u_1) \cdot (1 - u_2)} \quad (2)$$

Consequently, the total average waiting time equals to the average of T_{W1}, T_{W2} weighted by the arrival rate for each class:

$$T_W = \frac{\lambda_1}{\lambda} \cdot T_{W1} + \frac{\lambda_2}{\lambda} \cdot T_{W2} \quad (3)$$

We calculate the queuing delay for each class and estimate the total average time-in-system:

$$T_{Q1} = T_{W1} + T_{S1} \tag{4}$$

$$T_{Q2} = T_{W2} + T_{S2} \tag{5}$$

$$T_Q = \frac{\lambda_1}{\lambda} \cdot T_{Q1} + \frac{\lambda_2}{\lambda} \cdot T_{Q2} \tag{6}$$

Case 2: Non-Priority Scheduling: Without a priority queue, the two classes (non-congestive and congestive) have the same average waiting time. In such case, the network utilization of the system is:

$$u_{np} = u_1 = u_2 = \lambda_1 \cdot T_{S1} + \lambda_2 \cdot T_{S2} \tag{7}$$

The service time:

$$T_{Snp} = \frac{\lambda_1}{\lambda} \cdot T_{S1} + \frac{\lambda_2}{\lambda} \cdot T_{S2} \tag{8}$$

The average waiting time:

$$T_{W1np} = T_{W2np} = \frac{u_{np} \cdot T_{Snp}}{2 \cdot (1 - u_{np})} \tag{9}$$

The average time-in-system:

$$T_{Q1np} = T_{W1np} + T_{S1} \tag{10}$$

$$T_{Q2np} = T_{W2np} + T_{S2} \tag{11}$$

From (10) and (11), we get:

$$T_{Qnp} = \frac{\lambda_1}{\lambda} \cdot T_{Q1np} + \frac{\lambda_2}{\lambda} \cdot T_{Q2np} \tag{12}$$

Using the equations (4), (5), (6), (10), (11), (12), we calculated the average queuing delays for each class as well as for the system, for two different percentages of non-congestive traffic (Figures 1(a)-1(d)).

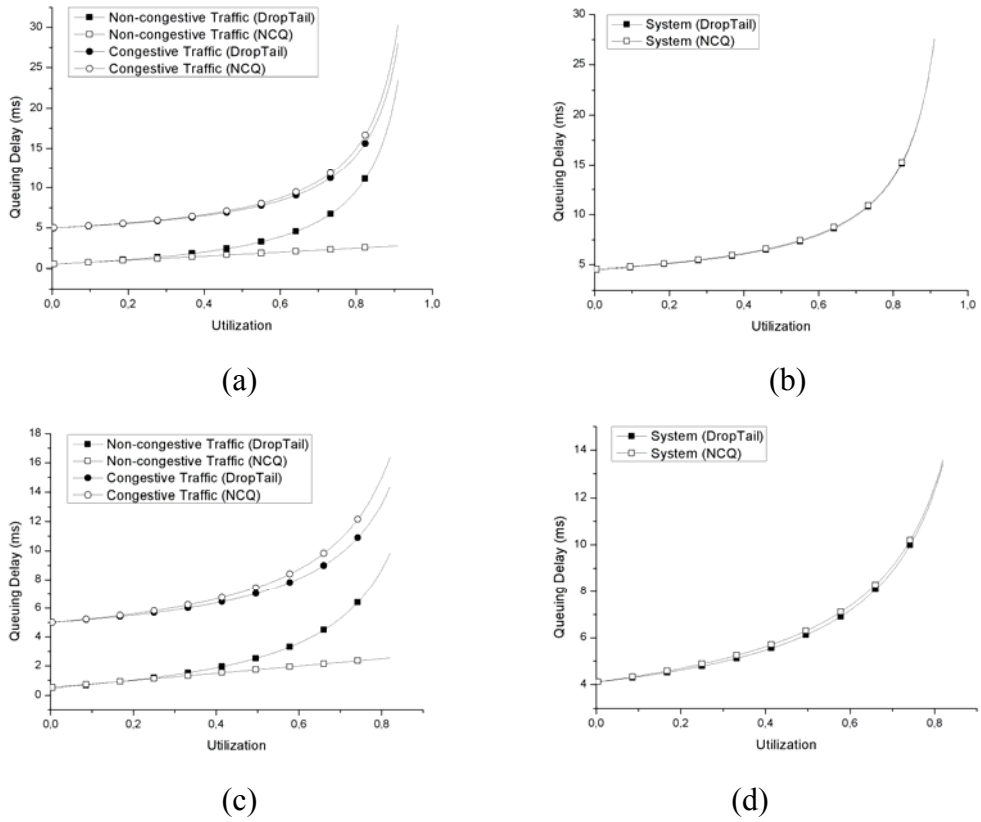


Figure 1. (a) and (c): Average Queuing Delay of Congestive and Non-Congestive Traffic (10% and 20% of the packets are Non-Congestive, respectively), (b) and (d): System's Average Queuing Delay (10% and 20% of packets are Non-Congestive)

In Figures 1(a), 1(b) the 10% of arriving packets form the non-congestive traffic (class 1) and the 90% the congestive (class 2). Their service times are 0.5ms and 5ms, respectively. For high utilizations (exceeding 0.6), there is a slight increase in the queuing delay of the congestive traffic for a significant improvement in the average delay of the non-congestive (Figure 1(a)). When we increase the rate of the non-congestive packets to 20% and for high

utilizations (exceeding 0.3), the impact of the prioritization on the congestive traffic appears significant (Figure 1(c)). For both percentages of the non-congestive traffic (10 and 20%), the average queuing delay of the system remains statistically the same (Figures 1(b), 1(d)).

Case 3: Priority Scheduling with ncqthresh: In the following analysis, we assume that only a portion of the non-congestive traffic is favored. More precisely, *ncqthresh* represents the percentage of the total traffic that can be favored without any statistically important impact on the congestive traffic and corresponds to the $\frac{\lambda_1}{\lambda} \cdot ncqthresh$ percentage of the non-congestive traffic, which we call *kthresh*. The two priority classes (1 and 2) consist of the *kthresh* percentage of non-congestive traffic and the (1-*kthresh*) percentage of the non-congestive traffic plus the congestive traffic, respectively.

We calculate the arrival rates and service times for each class:

$$\lambda_1' = kthresh \cdot \lambda_1 = \frac{\lambda_1^2}{\lambda} \cdot ncqthresh$$

$$\lambda_2' = (1 - kthresh) \cdot \lambda_1 + \lambda_2 = (1 - \frac{\lambda_1}{\lambda} \cdot ncqthresh) \cdot \lambda_1 + \lambda_2$$

$$\lambda' = \lambda_1' + \lambda_2'$$

$$T_{S1}' = T_{S1}$$

$$T_{S2}' = (1 - kthresh) \cdot \frac{\lambda_1}{\lambda_2'} \cdot T_{S1} + \frac{\lambda_2}{\lambda_2'} \cdot T_{S2} =$$

$$T_{S2}' = (1 - \frac{\lambda_1}{\lambda} \cdot ncqthresh) \cdot \frac{\lambda_1}{\lambda_2'} \cdot T_{S1} + \frac{\lambda_2}{\lambda_2'} \cdot T_{S2}$$

$$u_1' = \lambda_1' \cdot T_{S1}'$$

$$u_2' = \lambda_1' \cdot T_{S1}' + \lambda_2' \cdot T_{S2}'$$

The average waiting time for each of the two traffic classes becomes:

$$T'_{W1} = \frac{\lambda'_1 \cdot T'^{\,2}_{S1} + \lambda'_2 \cdot T'^{\,2}_{S2}}{2 \cdot (1 - u'_1)} \tag{13}$$

$$T'_{W2} = \frac{\lambda'_1 \cdot T'^{\,2}_{S1} + \lambda'_2 \cdot T'^{\,2}_{S2}}{2 \cdot (1 - u'_1) \cdot (1 - u'_2)} \tag{14}$$

We calculate the waiting times of each class using the weighted average of the waiting times T'_{W1} and T'_{W2}:

$$T_{W1} = \frac{kthresh \cdot \lambda_1}{\lambda_1} T'_{W1} + \frac{(1 - kthresh) \cdot \lambda_1}{\lambda_1} T'_{W2}$$

$$= kthresh \cdot T'_{W1} + (1 - kthresh) \cdot T'_{W2} =$$

$$\frac{\lambda_1}{\lambda} \cdot ncqthresh \cdot T'_{W1} + (1 - \frac{\lambda_1}{\lambda} \cdot ncqthresh) \cdot T'_{W2} \tag{15}$$

$$T_{W2} = T'_{W2} \tag{16}$$

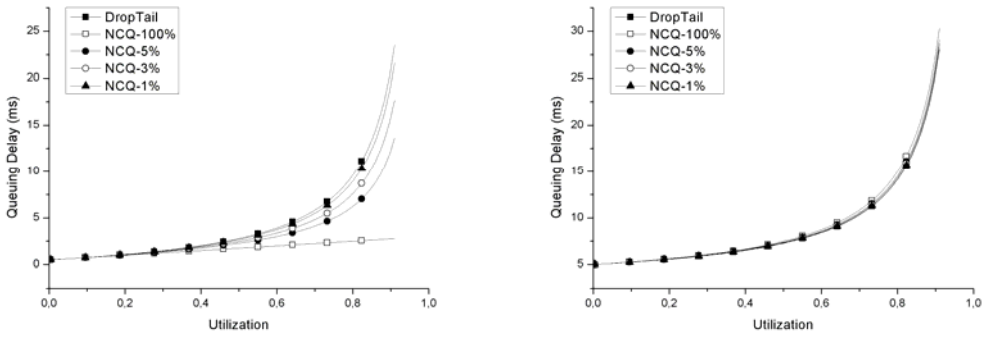
$$T_{Q1} = T_{W1} + T_{S1} \tag{17}$$

$$T_{Q2} = T_{W2} + T_{S2} \tag{18}$$

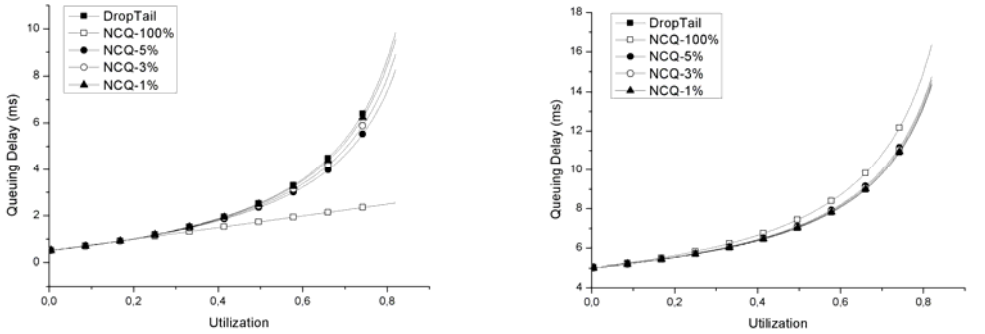
For the system:

$$T_Q = \frac{\lambda_1}{\lambda} \cdot T_{Q1} + \frac{\lambda_2}{\lambda} \cdot T_{Q2} \tag{19}$$

For a low *ncqthresh* value (e.g., 0.01 - 0.05), the impact of the prioritization on the average queuing delay of the congestive traffic is almost zero (Figures 2(b), 2(d)). Actually, *ncqthresh* bounds the prioritization of the non-congestive traffic to a limit that is not harmful to the bandwidth exploitation of the congestive applications. In utilizations below 0.23%, the average queuing delay of the non-congestive traffic remains statistically the same. As the network utilization builds up, there are significant gains for the non-congestive applications in terms of delay and increase more for higher values of *ncqthresh* (Figures 2(a), 2(c)).



(a) (b)



(c) (d)

Figure 2. (a) and (c): Average Queuing Delay of the Non-Congestive Traffic (10% and 20% of the packets are Non-Congestive, respectively), (b) and (d) Average Queuing Delay of the Congestive Traffic (10% and 20% of the packets are Non-Congestive, respectively)

REFERENCES

[1] D. Bertsekas and R. Gallager, Data Networks, Prentice-Hall, Englewood, Cliffs, New Jersey, 1987 (2nd Ed. 1991).

[2] R. Bless, K. Nichols, and K. Wehrle, A Lower Effort Per-Domain Behavior (PDB) for Differentiated Services, RFC3086, 2003.

[3] P. Brady, A Statistical Analysis of On-Off Patterns in 16 Conversations, The Bell System Technical Journal, 47 (1968) 73-91.

[4] J. Chung, and M. Claypool, Dynamic-CBT and Chips Router Support for Improved Multimedia Performance on the Internet, in: Proceedings of the Eighth ACM International Conference on Multimedia, October 2000.

[5] Cisco, Low Latency Queuing, http://www.cisco.com/en/US/docs/ios/12_0t/12_0t7/feature/guide/pqcbwfq.html.

[6] M. Claypool, R. Kinicki, and A. Kumar, Traffic Sensitive Active Queue Management, in: Proceedings of the 8th IEEE Global Internet Symposium, Miami, Florida, March 2005.

[7] R. Cole and J. Rosenluth, Voice over IP Performance Monitoring, ACM SIGCOMM Computer Communications Review, 31 (2) (2001) 9-24.

[8] T. D. Dang, B. Sonkoly, and S. Molnar, Fractal Analysis and Modelling of VoIP Traffic, in: Proceedings of 11th International Telecommunications Network Strategy and Planning Symposium, June 2004.

[9] P. Hurley, J. Boudec, P. Thiran, and M. Kara, ABE: Providing a Low-Delay Service within Best-Effort, IEEE Network, 15 (3) (2001) 60–69.

[10] IETF, Differentiated Services Charter, http://www.ietf.org/html.charters/OLD/diffservcharter.html, 2003.

[11] IETF, Integrated Services Charter, http://www.ietf.org/html.charters/OLD/intservcharter.html, 2000.

[12] ITU-T Recommendation G.107, The E-Model, a Computational Model for Use in Transmission Planning, December 1998.

[13] ITU-T Recommendation G.711, Pulse Code Modulation (PCM) of Voice Frequencies, November 1988.

[14] ITU-T Recommendation G.113, General Characteristics of General Telephone Connections and Telephone Circuits – Transmission Impairments, February 1996.

[15] L. Mamatas, and V. Tsaoussidis, A New Approach to Service Differentiation: Non-Congestive Queuing, in: Proceedings of CONWIN 2005, July 2005.

[16] L. Mamatas, and V. Tsaoussidis, Differentiating Services for Sensor Internetworking, in: Proceedings of Med-Hoc-Net 2007, June 1007.

[17] L. Mamatas, and V. Tsaoussidis, Differentiating Services with Non-Congestive Queuing (NCQ), Technical Report: TR-DUTH-EE-2006-11.

[18] S. McCanne and S. Floyd, NS-2 Network Simulator, 1997.

[19] W. Noureddine, and F. Tobagi, Improving the Performance of Interactive TCP Applications Using Service Differentiation, Computer Networks, 40 (1) (2000) 19–43.

[20] M. Parris, K. Jeffay, and F. Smith, Lightweight Active Router-Queue Management for Multimedia Networking, in: Proceedings of SPIE Conference on Multimedia Computing and Networking, January 1999.

[21] H. Zhang, Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks, Proceedings of the IEEE, 83 (10) (1995) 1374–1396.

[22] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala, RSVP: A New Resource Reservation Protocol, IEEE Communications Magazine, 40 (5) (2002) 116–127.