A delay-oriented prioritization policy based on non-congestive queuing

G. Papastergiou[1, *, †], C. Georgiou[1], L. Mamatas[2] and V. Tsaoussidis[1]

[1]*Department of Electrical and Computer Engineering, Democritus University of Thrace, Xanthi, Greece*
[2]*Department of Technology Management, University of Macedonia, Thessaloniki, Greece*

SUMMARY

In this paper, we depart from our previously proposed Non-Congestive Queuing (NCQ) mechanism and 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 noticeably 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. Based on the NCQ concept, 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. Results show that whenever prioritization buffer resources are sufficient, NCQ+ can provide non-detectable delays to sensor applications and conditionally improve the performance of VoIP applications, while maintaining the impact on the performance of the other flows insignificant.

KEY WORDS: service differentiation; NCQ+; QoS; LIBS; voice over IP

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-zero delay had it been favored by a prioritized scheduling scheme, at an almost-zero cost to other congestive flows [1].

The typical Best-Effort Service (BES) does not treat applications that do not contribute to congestion fairly, since they 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 needing help, although it causes almost-zero delays, may be lost or severely delayed by congestive applications that monopolize communication resources.

Departing from such situation, we introduce a new service practice for packets, called less impact better service (LIBS) [1–3]. 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

*Correspondence to: G. Papastergiou, Department of Electrical and Computer Engineering, Democritus University of Thrace, Xanthi 67100, Greece.
†E-mail: gpapaste@ee.duth.gr

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 [1–3] we proposed a novel scheduling discipline, namely Non-Congestive Queuing (NCQ) that realizes the LIBS practice 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 toward a more sophisticated service differentiation. NCQ+, as NCQ, is a system-oriented service discipline based on the LIBS service practice, which targets at 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 the 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.

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.

NCQ+ guarantees that applications transmitting tiny volume of data, such as typical sensor applications, and thus do not contribute to congestion, will suffer almost-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 buffer resources fairly 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 $\mathrm{ncqthresh}_1$ and $\mathrm{ncqthresh}_2$. We consider that the former is fixed and defines, percentage-wise, the buffer space available for service prioritization. Thus, $\mathrm{ncqthresh}_1$ points to the level of buffering that corresponds to the maximum amount of delay that can be considered statistically insignificant to the congestive flows. $\mathrm{Ncqthresh}_2$ 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 *No-Delay Service* (NDS) as long as their traffic volume is below $\mathrm{ncqthresh}_1$, whereas small packets receive prioritization by utilizing the available space between the two thresholds (*Prioritized Service*, PS). Thus, typical congestive applications receive a *BES*, by exploiting the remaining buffer space. All types of service are implicitly expressed in terms of queuing delay and packet dropping rate. That is, applications receiving NDS will experience zero queuing delays and dropping rates, whereas those receiving PS will experience conditionally better queuing delays and dropping rates compared with congestive flows. Although NCQ+ deals with only two distinct classes of non-congestive packets, it can be easily extended to work with more classes as well.

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 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. Owing 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 as follows: In Section 2, we discuss the related work. In Section 3, we provide the pseudocode 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 in 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 applications that require minimal time to service first; 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, Bertsekas and Gallager [4] confirm 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 [5], 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* [6, 7], CBT [8] or [9]) overcome the limitation of state information but limit similar traffic to the same QoS class, introducing a 'classification' error. For example, DCBT with ChIPS [10] extends CBT by providing dynamic thresholds and lower jitter for multimedia traffic. However, it assumes that all multimedia traffic requires the same QoS. In [11], 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) [12] allows applications to indicate via marking their preferable delay/throughput sensitivity at the 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[‡] 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 hierarchical service dynamics: 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 on its service. Therefore, different priorities can be assigned via packet marking to the different non-congestive or congestive applications.

In References [1–3], we introduced NCQ and showed that it 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, NCQ+ can provide delay guarantees to sensor applications, while favoring conditionally VoIP calls. Other service strategies for delay-sensitive traffic include Low Latency Queuing (LLQ) [13]. LLQ is a mechanism that provides a strict Priority Queue (PQ) to Class-Based Weighted Fair Queuing (CBWFQ) [14]. 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 [15]), 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 NCQ and the NCQ+.

3.1. Non-Congestive Queuing (*NCQ*)

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 to 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,[§] 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,

[‡]Even if it is measured, the precision and granularity of measurements are dubious.
[§]In our experiments, this class includes both sensor and VoIP packets.

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

Figure 1. NCQ pseudocode.

within the same buffer, each packet is checked for its length, contrasted to the current state of prioritized service rate and gets priority whenever it satisfies two conditions: (i) length is below a maximum value[¶] for small-identified packets, namely size_thresh, and (ii) prioritized service rate is below ncqthresh.

NCQ does not include any mechanism to differentiate between heavy and light traffic flows where all packets are small sized. A typical congestive application could be intentionally transformed into a small-packet, high-rate application. Since ncqthresh and packet length confine the amount of gain, the transformation will cause that much overhead that the penalty of transformation will be greater than the gain. In [3], we calculate numerically the actual impact of such transformation. In Appendix A, we approach numerically the impact of NCQ priority on congestive traffic for any given proportion of traffic classes [3].

At this stage of our work, we exclude ACKs from prioritization although 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[‖] 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.

3.2. Non-Congestive Queuing Plus (*NCQ+*)

Here, we attempt to extend NCQ toward a more sophisticated service prioritization scheme that considers more non-congestive applications. In the following sections, we discuss the fundamental concepts of NCQ+ and how service prioritization is achieved.

3.2.1. Fundamental concepts. NCQ+ differentiates non-congestive applications further and produces two application classes: those that 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, NCQ+ provides NDS to applications utilizing tiny packets at low data rates, while other non-congestive applications receive conditionally PS.

Tiny packets do not cause statistically important delays, and hence could be promoted. Additionally, applications with small packets at low data rates have a minor contribution to congestion and can be prioritized[**] 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: 'less-delay'.

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

[¶]We experimentally set this value to 150 bytes.
[‖]It is not significant in our experiments.
[**]As with NCQ scheduling discipline.

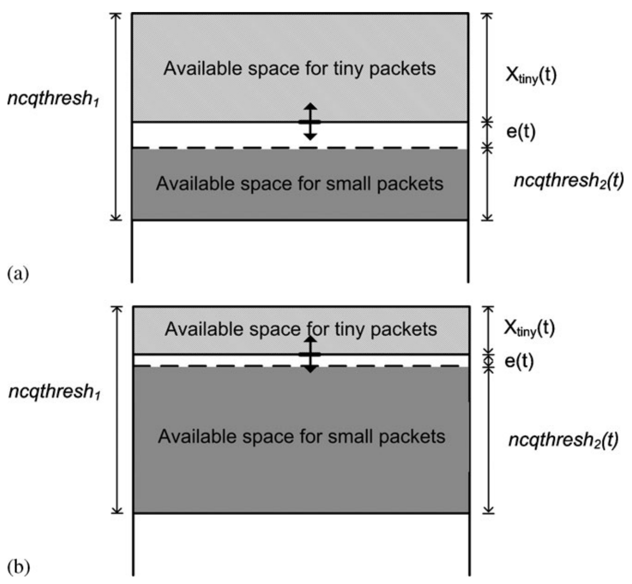
Available space for tiny packets
ncqthresh1
Xtiny(t)
e(t)
Available space for small packets
ncqthresh2(t)
(a)
Available space for tiny packets
Xtiny(t)
e(t)
ncqthresh1
Available space for small packets
ncqthresh2(t)
(b)

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

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$_1$), whereas 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 for service prioritization. That is, it sets 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 $X_{\text{tiny}}(t)$ as the percentage of the incoming tiny packets at time t. To simplify the following analysis, we always assume that $X_{\text{tiny}}(t) < \text{ncqthresh}_1$. That is, there are always prioritization buffer resources for providing NDS to tiny packets. In this context, in order for NCQ+ to provide NDS, the following equation should hold:

$$\text{ncqthresh}_2(t) \leqslant \text{ncqthresh}_1 - [X_{\text{tiny}}(t) + e(t)] \tag{1}$$

where $e(t)$ is a safety margin for provided NDS and is discussed in next section. Obviously, ncqthresh$_1$ is the maximum value of ncqthresh$_2$. Given that the demand for NDS 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$_2$.

3.2.2. Prioritized service.

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

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

From the above equation we have:

$$\text{ncqthresh}_2(t)=\text{ncqthresh}_1-(1+\alpha)\cdot X_{\text{tiny}}(t) \tag{3}$$

where $(1+\alpha)\cdot X_{\text{tiny}}(t)$ 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 occur, 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:

$$X_{\text{tiny}}(t+t_1)+Y_{\text{small}}(t+t_1)\geqslant\text{ncqthresh}_1 \tag{4}$$

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

$$Y_{\text{small}}(t)=X_{\text{small}}(t) \quad \text{when } X_{\text{small}}(t)\leqslant\text{ncqthresh}_2(t) \quad \forall\ t\in(0,\infty) \tag{5}$$

and

$$\max\{Y_{\text{small}}(t)\}=\text{ncqthresh}_2(t) \tag{6}$$

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 2(b) illustrates the above process. However, ncqthresh$_2$ also decreases whenever X_{tiny} has increased and belongs to the interval $(X_{\text{tiny}}(t), X_{\text{tiny}}(t)+e(t))$. The following equations hold for the event of a non-prioritized small packet:

$$X_{\text{tiny}}(t+t_2)+Y_{\text{small}}(t+t_2)\geqslant\text{ncqthresh}_1 \tag{7}$$

or

$$Y_{\text{small}}(t+t_2)\geqslant\text{ncqthresh}_2(t) \tag{8}$$

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

4. EVALUATION METHODOLOGY

A major issue we need to address prior to implementing our evaluation plan is to determine the framework for a fair comparison of NCQ mechanisms and services. In this context, we need to highlight that NCQ builds solely on the DropTail queue management without requiring any further marking, header modifications or architectural arrangements (unlike several diffserv approaches for example). Therefore, in order to evaluate the particular impact of the NCQ mechanisms, a fair

```
for every received packet
begin
    count received packets
    if (packet_length ≤ $size\_thresh_1$) and
       (total_prioritized_service < $ncqthresh_1$)
    then
        packet receives high priority
        count favored tiny packets
    else
        $ncqthresh_2$ = $ncqthresh_1$ – k*tiny_packets_favor_rate
        packet receives normal priority
    end
    if ($size\_thresh_1$ < packet_length ≤ $size\_thresh_2$) and
       (small_packets_favor_rate < $ncqthresh_2$) and
       (total_prioritized_service < $ncqthresh_1$)
    then
        packet receives high priority
        count favored small packets
    else
        $ncqthresh_2$ = $ncqthresh_1$ – k*tiny_packets_favor_rate
        packet receives normal priority
    end
end
```

Figure 3. NCQ+ pseudocode.

point of comparison is the DropTail itself; that is, the queuing mechanism that lacks only the NCQ feature we proposed.

A broader comparison of the service impact of different approaches could also be possible; we note however two major issues that justify why we avoid this attempt presently. First, different approaches incorporate different requirements (e.g. packet marking), architectural constraints (e.g. ingress–egress nodes, etc.) or overhead costs (e.g. header extensions, processing, etc.), which in turn call for a broader evaluation scenario which is out of the scope of this paper.[††] Second, NCQ+ can complement existing mechanisms and hence a comparison with complementing mechanisms may not necessarily make sense. Consider for example Weighted Fair Queuing (WFQ) [16]. Each queue that is assigned a particular weight may still utilize the service improvements of NCQ+. Therefore, a fair model of comparison in that particular case would be to compare not NCQ+ with WFQ, but instead WFQ with WFQ with NCQ+ incorporated.

We have implemented our evaluation plan on the ns-2 network simulator [17]. We simulated VoIP traffic based on the following assumptions: During a conversation, call participants alternate between activity and idle periods. Taking into consideration the ON and OFF periods [18], as well as the heavy-tailed characteristics and self-similarity of VoIP traffic [19] we use Pareto distribution for modeling the call holding times. We configure Pareto with a mean rate to correspond to transmission rate of 64 kbps and the shape parameter is set to 1.5. In accordance with [18], we distribute ON and OFF periods with means 1.0 and 1.35 s, respectively. We simulate VoIP streams of 64 kbps (following the widely used ITU-T G.711 coding standard [20]) and we set packet sizes at 140 bytes (carries 15 ms 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 50 ms.

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 NDS to sensor-based applications,
(iv) favors VoIP applications when resources are available.

[††]In this paper, we are interested in evaluating the particular impact of NCQ+ alone.

4.1. Evaluation metrics

We measure application performance using Goodput defined as

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

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

$$\text{Average\_Goodput} = \frac{\sum_{i=1}^{n} (\text{Goodput}_i)}{n} \tag{10}$$

where Goodput_i is the Goodput of the ith flow and n the number of flows.

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

$$\text{Average\_PLR} = \frac{\sum_{i=1}^{n} \text{PLR}_i}{n} \tag{11}$$

where PLR_i is the PLR of the ith flow and n the number of flows.

We characterize the quality of voice communication using the R-Factor, which is included in the E-Model [21, 22] (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 [23], 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

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

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 PLR. For the G.711 codec, $\gamma_1 = 0$, $\gamma_2 = 30$ and $\gamma_3 = 15$.

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 I.

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

Table I. 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

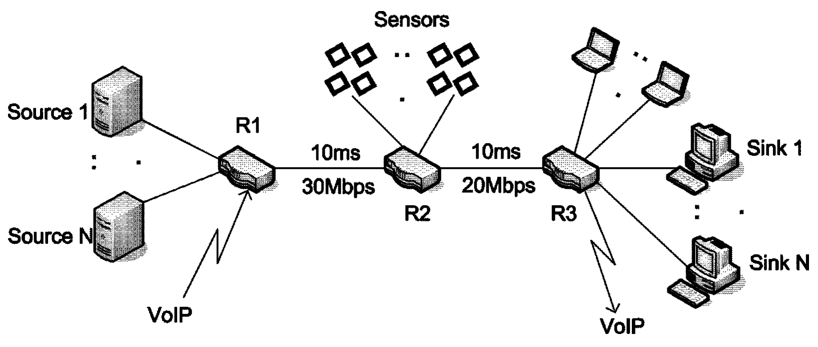

Figure 4. Network topology.

and wireless links. The router R1 is the bottleneck for the competing non-congestive VoIP and congestive FTP traffic, whereas 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 packet error rate adjusted to 0.01. That is, we simulate only random errors, not correlated burst errors, that occur on wireless links, and this also justifies our low packet error rate. The bandwidth of each access link is 1 Mbps, while the link propagation delay varies deterministically from 8 to 12 ms. Queue size for all scenarios is 100 packets, each experiment's duration is 60 s and $\text{ncqthresh}_\text{l}$ is set to 0.05.

The value of $\text{ncqthresh}_\text{l}$ was selected based on two criteria. The first criterion was to guarantee minimal impact on non-prioritized service classes. Results of [3] indicated that 0.05 statistically does not damage applications that utilize NCQ services. The second criterion was to determine a 'service space' appropriate and sufficient to allow for service differentiation. This requires a meaningful value for $\text{ncqthresh}_\text{l}$. In this context, service threshold of 0.01 could not allow for any significant service impact. In conclusion, we determined experimentally that 0.05 was a meaningful choice. By the same token, we set parameter α equal to 0.1. Although this is not a restrictive choice and can be further investigated, this 10% safety margin was in all experiments adequate to cope with occasional increases in the tiny packets arrival rate.

We use three distinct scenarios in order to evaluate the behavior of NCQ+ in different environments. We use NCQ and DropTail as reference points. 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, 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 with NCQ and DropTail, considering different levels of VoIP traffic. We set total number of flows to 120, while sensor flows are 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, similar 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 be canceled, while more sensors may be activated for monitoring purposes.

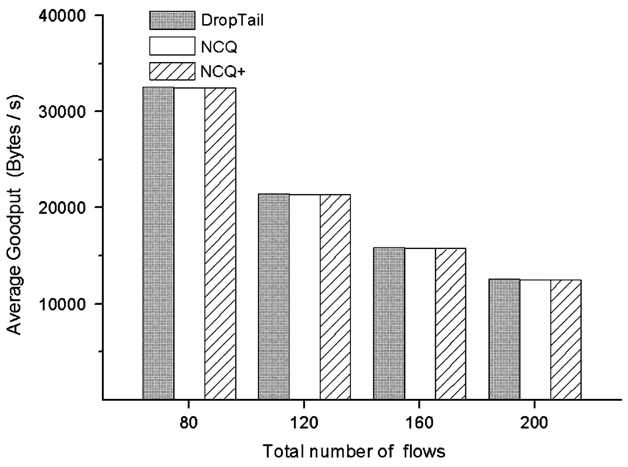

Figure 5. Average Goodput of FTP flows.

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 that leads to alternating periods of increasing, constant and decreasing sensor data rate. Eventually, each sensor flow transmits data for half of the duration of each experiment (i.e. 30 s).

4.3. *Experimental results*

4.3.1. 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. Owing 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 with 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 with 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 PLR experienced by sensor applications, it leads to a considerable large number of losses compared with NCQ+. When NCQ+ is deployed, sensor packets are unaffected by the state of the buffers and no packets are lost.

Comparing NCQ+ with 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 with NCQ increases the Average_Goodput of sensor applications by about 16%, while Average_Goodput of VoIP applications decreases by about 8%.

4.3.2. 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,

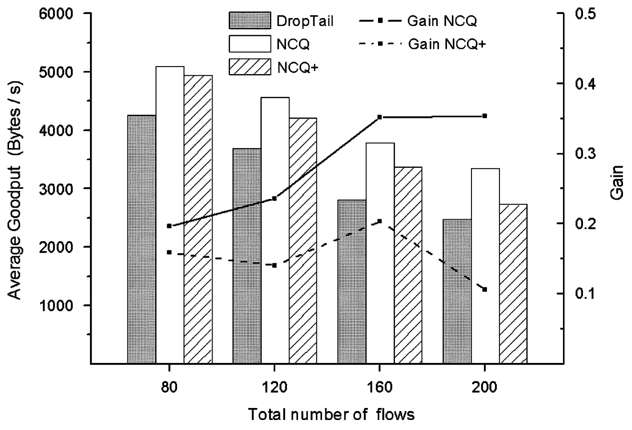

Figure 6. Average Goodput of VoIP flows.

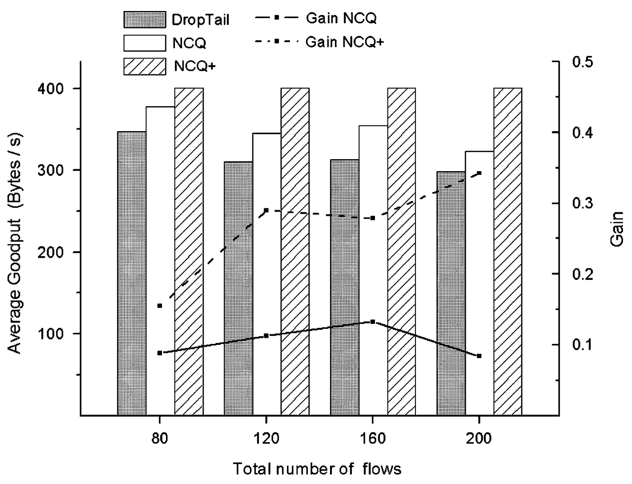

Figure 7. Average Goodput of sensor flows.

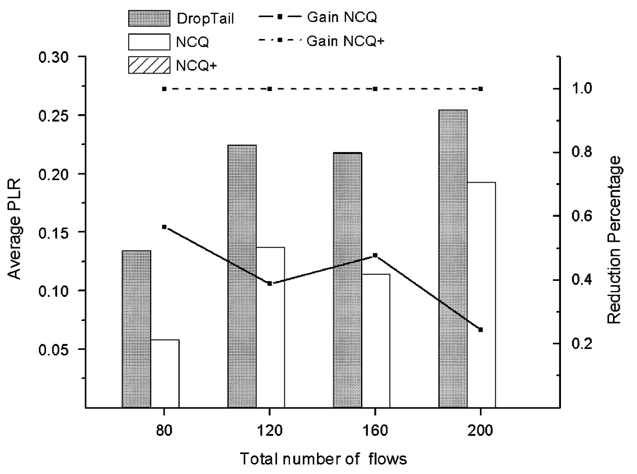

Figure 8. Average PLR of sensor flows.

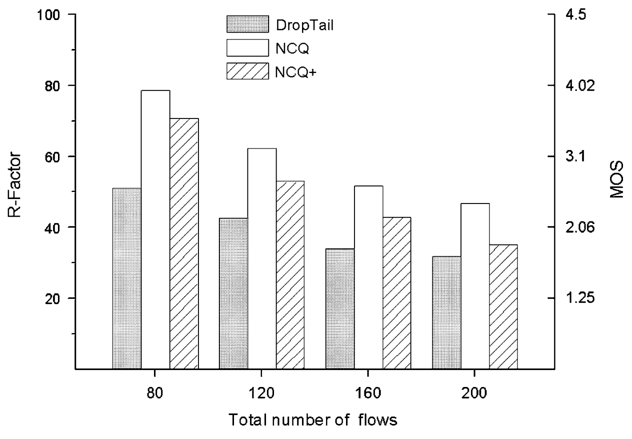

Figure 9. R-Factor.

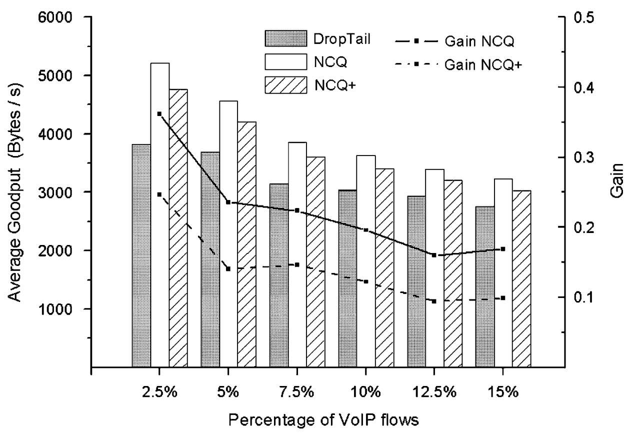

Figure 10. Average Goodput of VoIP flows.

non-congestive flows have significant improvement in 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. about 25and 10% 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 the best Goodput performance (Figure 11) and zero packet losses (Figure 12) to sensor flows, while improving VoIP performance as much as possible (Figures 10 and 13). Sensor traffic prioritization is always unaffected by PS demand of VoIP applications.

4.3.3. 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 never exceeds the ncqthresh_1 and sensor flows receive NDS. Further increase reduces Goodput gains and increases the Average_PLR (Figure 16). NCQ+ achieves always a major improvement with a minimum gain of about 11% 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 with

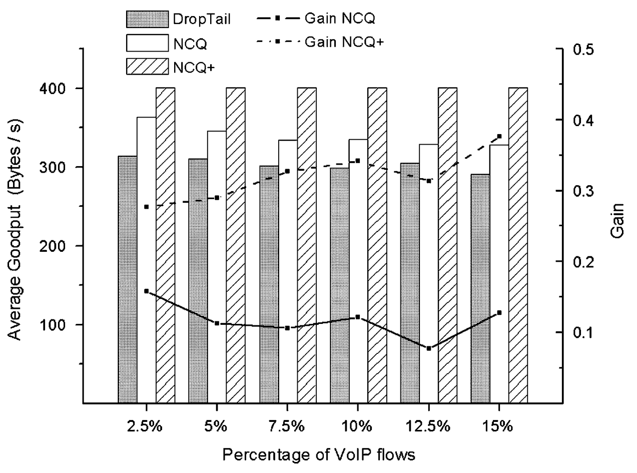

Figure 11. Average Goodput of sensor flows.

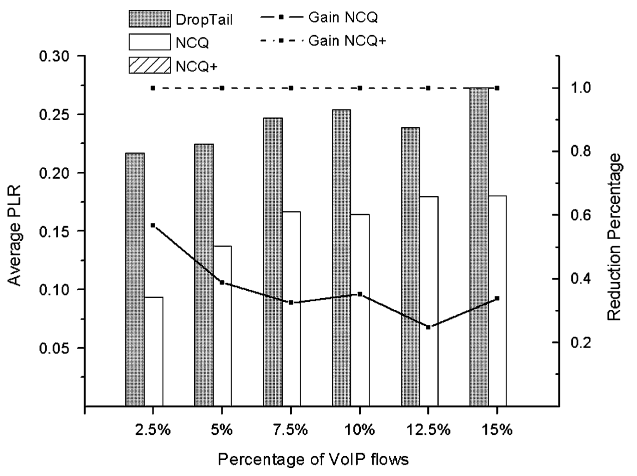

Figure 12. Average PLR of sensor flows.

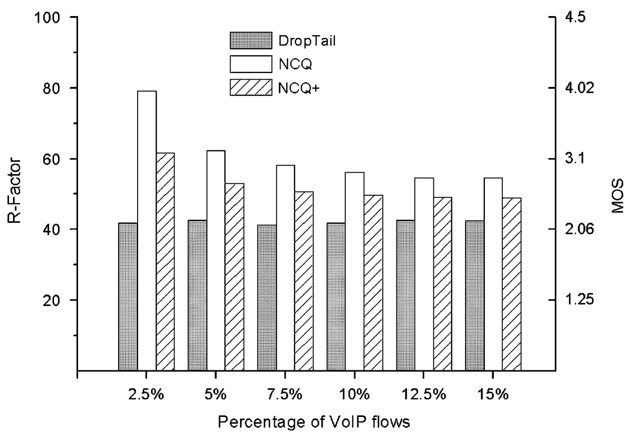

Figure 13. R-Factor.

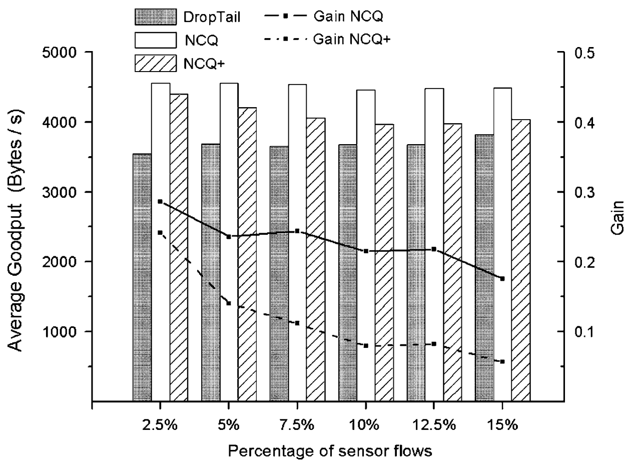

Figure 14. Average Goodput of VoIP flows.

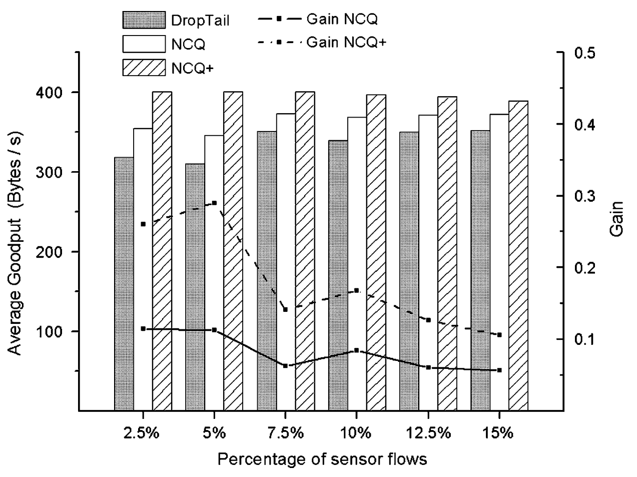

Figure 15. Average Goodput of sensor flows.

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 with NCQ, achieves about 5% improvement in Average_Goodput of sensor applications, while the Average_Goodput of VoIP calls is deteriorated by about 10%. Same observations apply also to Figures 16 and 17.

5. CONCLUSIONS

We proposed a new scheme for service differentiation based on a system-oriented approach that realizes the LIBS practice. Our scheme, however, does not conflict with the 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 NDS to sensor applications and conditionally improve the performance of VoIP applications, as long as NDSs are not violated and the impact on the performance of the other flows is insignificant. Whenever prioritization buffer resources are

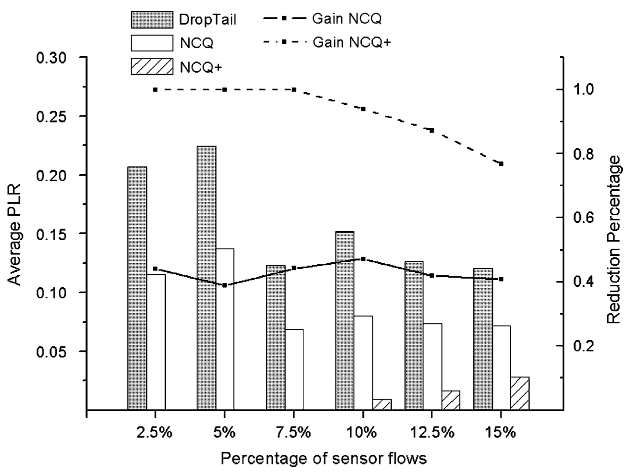

Figure 16. Average PLR of sensor flows.

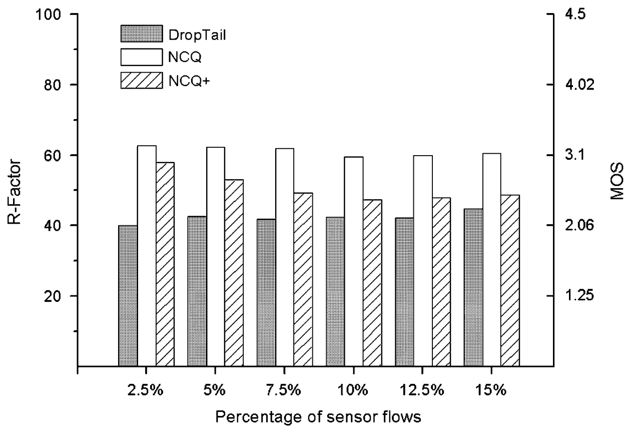

Figure 17. R-Factor.

not sufficient, NCQ+ provides ‘less-delay’ services to sensor applications. Further analytical and experimental results with more sophisticated mechanisms are underway.

APPENDIX A: NUMERICAL ANALYSIS ON 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 (hence, average service-time too) and lower packet-arrival rate ($\lambda_1 < \lambda_2$). We summarize our notation in Table A1.

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.

Table A1. 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

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:

$$T_{W1} = \frac{\lambda_1 \cdot T_{S1}^2 + \lambda_2 \cdot T_{S2}^2}{2 \cdot (1 - u_1)} \tag{A1}$$

$$T_{W2} = \frac{\lambda_1 \cdot T_{S1}^2 + \lambda_2 \cdot T_{S2}^2}{2 \cdot (1 - u_1) \cdot (1 - u_2)} \tag{A2}$$

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

$$T_W = \frac{\lambda_1}{\lambda} \cdot T_{W1} + \frac{\lambda_2}{\lambda} \cdot T_{W2} \tag{A3}$$

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

$$T_{Q1} = T_{W1} + T_{S1} \tag{A4}$$

$$T_{Q2} = T_{W2} + T_{S2} \tag{A5}$$

$$T_Q = \frac{\lambda_1}{\lambda} \cdot T_{Q1} + \frac{\lambda_2}{\lambda} \cdot T_{Q2} \tag{A6}$$

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

$$u_{np} = u_1 = u_2 = \lambda_1 \cdot T_{S1} + \lambda_2 \cdot T_{S2} \tag{A7}$$

The service-time:

$$T_{Snp} = \frac{\lambda_1}{\lambda} \cdot T_{S1} + \frac{\lambda_2}{\lambda} \cdot T_{S2} \tag{A8}$$

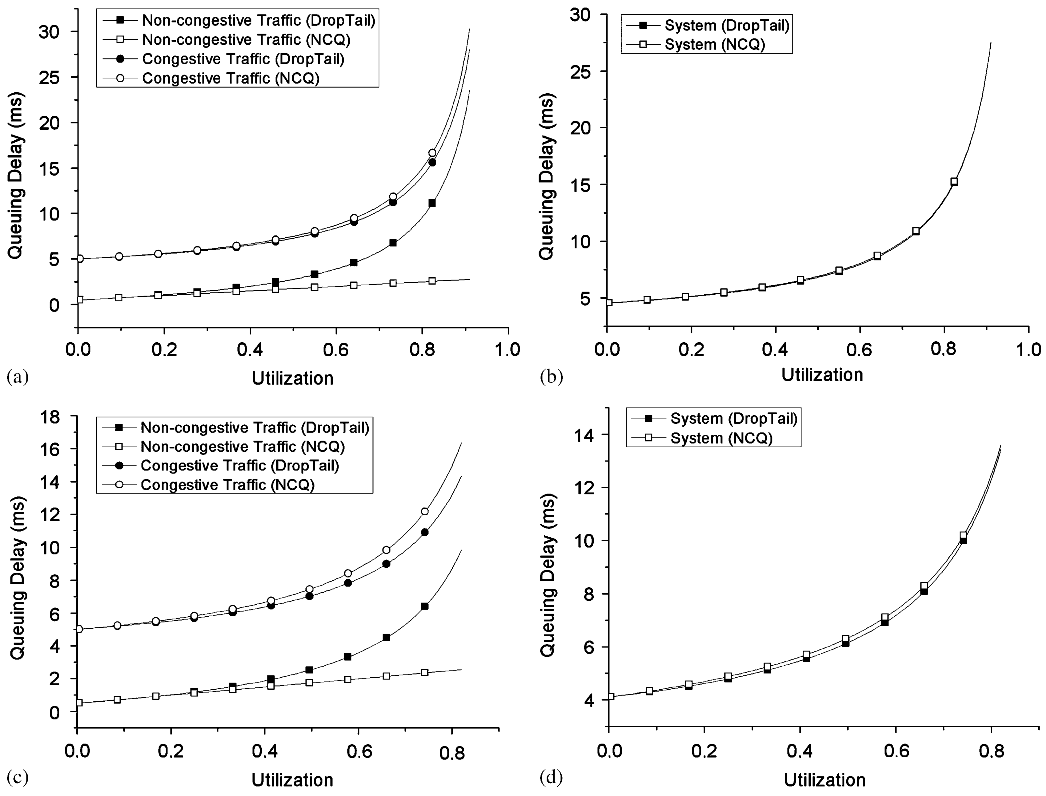

Figure A1. (a) and (c): average queuing delay of congestive and non-congestive traffic (10 and 20% of the packets are non-congestive, respectively) and (b) and (d): system's average queuing delay (10 and 20% of packets are non-congestive).

The average waiting time:

$$T_{\mathrm{W1np}} = T_{\mathrm{W2np}} = \frac{u_{\mathrm{np}} \cdot T_{\mathrm{Snp}}}{2 \cdot (1 - u_{\mathrm{np}})} \tag{A9}$$

The average time-in-system:

$$T_{\mathrm{Q1np}} = T_{\mathrm{W1np}} + T_{\mathrm{S1}} \tag{A10}$$

$$T_{\mathrm{Q2np}} = T_{\mathrm{W2np}} + T_{\mathrm{S2}} \tag{A11}$$

From (A10) and (A11), we get:

$$T_{\mathrm{Qnp}} = \frac{\lambda_1}{\lambda} \cdot T_{\mathrm{Q1np}} + \frac{\lambda_2}{\lambda} \cdot T_{\mathrm{Q2np}} \tag{A12}$$

Using Equations (A4), (A5), (A6), (A10), (A11), (A12), we calculated the average queuing delays for each class as well as for the system, for two different percentages of non-congestive traffic (Figures A1(a)–(d)).

In Figure A1(a), (b) the 10% of arriving packets form the non-congestive traffic (class 1) and the 90% the congestive (class 2). Their service times are 0.5 and 5 ms, 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 A1(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 A1(c)).

For both percentages of the non-congestive traffic (10 and 20%), the average queuing delay of the system remains statistically the same (Figures A1(b), (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 $(\lambda_1/\lambda)\cdot$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:

$$\lambda'_1 = \text{kthresh}\cdot\lambda_1 = \frac{\lambda_1^2}{\lambda}\cdot\text{ncqthresh}$$

$$\lambda'_2 = (1-\text{kthresh})\cdot\lambda_1+\lambda_2 = \left(1-\frac{\lambda_1}{\lambda}\cdot\text{ncqthresh}\right)\cdot\lambda_1+\lambda_2$$

$$\lambda' = \lambda'_1+\lambda'_2$$

$$T'_{\text{S1}} = T_{\text{S1}}$$

$$T'_{\text{S2}} = (1-\text{kthresh})\cdot\frac{\lambda_1}{\lambda'_2}\cdot T_{\text{S1}}+\frac{\lambda_2}{\lambda'_2}\cdot T_{\text{S2}}$$

$$T'_{\text{S2}} = \left(1-\frac{\lambda_1}{\lambda}\cdot\text{ncqthresh}\right)\cdot\frac{\lambda_1}{\lambda'_2}\cdot T_{\text{S1}}+\frac{\lambda_2}{\lambda'_2}\cdot T_{\text{S2}}$$

$$u'_1 = \lambda'_1\cdot T'_{\text{S1}}$$

$$u'_2 = \lambda'_1\cdot T'_{\text{S1}}+\lambda'_2\cdot T'_{\text{S2}}$$

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

$$T'_{\text{W1}} = \frac{\lambda'_1\cdot T'^2_{\text{S1}}+\lambda'_2\cdot T'^2_{\text{S2}}}{2\cdot(1-u'_1)} \tag{A13}$$

$$T'_{\text{W2}} = \frac{\lambda'_1\cdot T'^2_{\text{S1}}+\lambda'_2\cdot T'^2_{\text{S2}}}{2\cdot(1-u'_1)\cdot(1-u'_2)} \tag{A14}$$

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

$$\begin{aligned} T_{\text{W1}} &= \frac{\text{kthresh}\cdot\lambda_1}{\lambda_1}T'_{\text{W1}}+\frac{(1-\text{kthresh})\cdot\lambda_1}{\lambda_1}T'_{\text{W2}} \\ &= \text{kthresh}\cdot T'_{\text{W1}}+(1-\text{kthresh})\cdot T'_{\text{W2}} \\ &= \frac{\lambda_1}{\lambda}\cdot\text{ncqthresh}\cdot T'_{\text{W1}}+\left(1-\frac{\lambda_1}{\lambda}\cdot\text{ncqthresh}\right)\cdot T'_{\text{W2}} \end{aligned} \tag{A15}$$

$$T_{\text{W2}} = T'_{\text{W2}} \tag{A16}$$

$$T_{\text{Q1}} = T_{\text{W1}}+T_{\text{S1}} \tag{A17}$$

$$T_{\text{Q2}} = T_{\text{W2}}+T_{\text{S2}} \tag{A18}$$

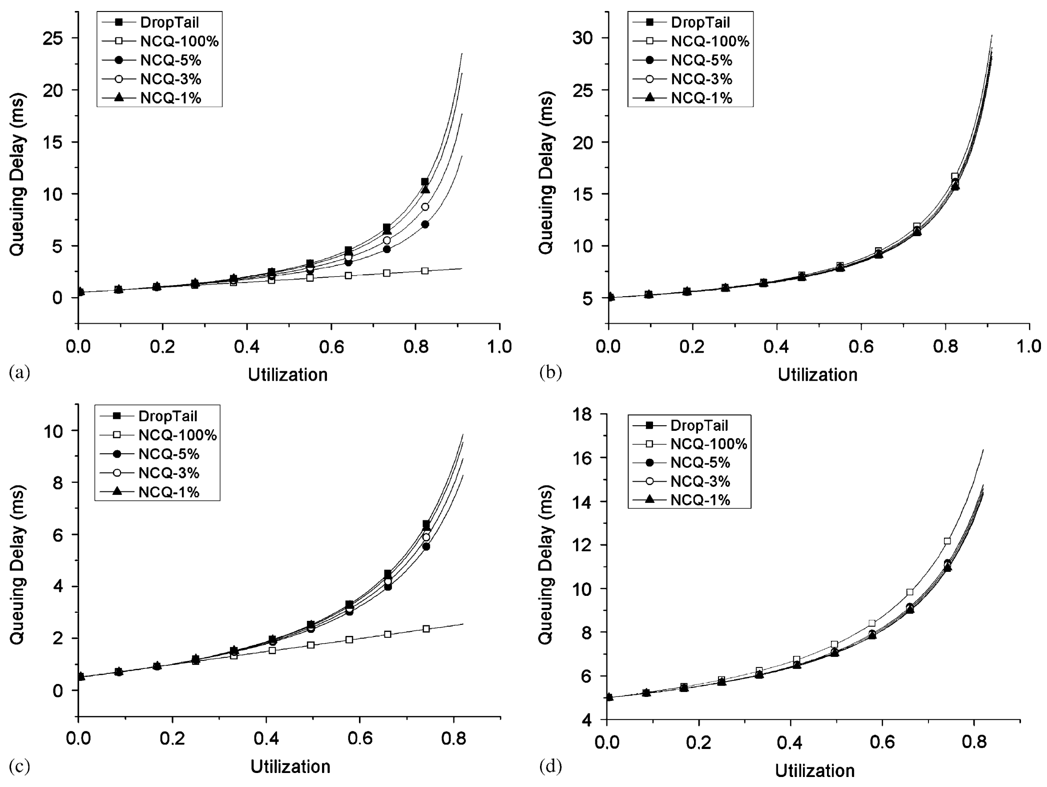

Figure A2. (a) and (c): average queuing delay of the non-congestive traffic (10 and 20% of the packets are non-congestive, respectively) and (b) and (d) average queuing delay of the congestive traffic (10 and 20% of the packets are non-congestive, respectively).

For the system:

$$T_Q = \frac{\lambda_1}{\lambda} \cdot T_{Q1} + \frac{\lambda_2}{\lambda} \cdot T_{Q2} \tag{A19}$$

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 (Figure A2(b), (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 (Figure A2(a), (c)).

REFERENCES

1. Mamatas L, Tsaoussidis V. Differentiating services for sensor internetworking. *Proceedings of Med-Hoc-Net 2007*, Corfu, Greece, June 2007.
2. Mamatas L, Tsaoussidis V. A new approach to service differentiation: non-congestive queuing. *Proceedings of CONWIN 2005*, Budapest, Hungary, July 2005.
3. Mamatas L, Tsaoussidis V. Differentiating services with non-congestive queuing (NCQ). *IEEE Transactions on Computers* 2008; **58**(5):591–604.
4. Bertsekas D, Gallager R. *Data Networks* (2nd edn, 1991). Prentice-Hall: Englewood Cliffs, NJ, 1987.
5. Braden R, Clark D, Shenker S. Integrated services for the internet architecture: an overview, *RFC 1633*, June 1994.
6. Blake S, Black D, Carlson M, Davies E, Wang Z, Weiss W. An architecture for differentiated services, *RFC 2475*, December 1998.

7. Bless R, Nichols K, Wehrle K. A lower effort per-domain behavior (PDB) for differentiated services. *RFC 3086*, 2003.
8. Parris M, Jeffay K, Smith F. Lightweight active router-queue management for multimedia networking. *Proceedings of SPIE Conference on Multimedia Computing and Networking*, San Jose, CA, U.S.A., January 1999.
9. Noureddine W, Tobagi F. Improving the performance of interactive TCP applications using service differentiation. *Computer Networks* 2000; **40**(1):19–43.
10. Chung J, Claypool M. Dynamic-CBT and chips router support for improved multimedia performance on the Internet. *Proceedings of the Eighth ACM International Conference on Multimedia*, Los Angeles, CA, U.S.A., October 2000.
11. Hurley P, Boudec J, Thiran P, Kara M. ABE: providing a low-delay service within best-effort. *IEEE Network* 2001; **15**(3):60–69.
12. Claypool M, Kinicki R, Kumar A. Traffic sensitive active queue management. *Proceedings of the Eighth IEEE Global Internet Symposium*, Miami, FL, March 2005.
13. Cisco. Low latency queuing. Available from: http://www.cisco.com/en/US/docs/ios/12_0t/12_0t7/feature/guide/pqcbwfq.html.
14. Zhang H. Service disciplines for guaranteed performance service in packet-switching networks. *Proceedings of the IEEE* 1995; **83**(10):1374–1396.
15. Zhang L, Deering S, Estrin D, Shenker S, Zappala D. RSVP: a new resource reservation protocol. *IEEE Communications Magazine* 2002; **40**(5):116–127.
16. Demers A, Keshav S, Shenker S. Analysis and simulation of a fair queuing algorithm. *Journal of Internetworking Research and Experience* 1990; 3–26.
17. McCanne S, Floyd S. NS-2 network simulator. Available from: http://www.isi.edu/nsnam/ns.
18. Brady P. A statistical analysis of on-off patterns in 16 conversations. *The Bell System Technical Journal* 1968; **47**:73–91.
19. Dang TD, Sonkoly B, Molnar S. Fractal analysis and modelling of VoIP traffic. *Proceedings of 11th International Telecommunications Network Strategy and Planning Symposium*, Vienna, Austria, June 2004.
20. Cole R, Rosenluth J. Voice over IP performance monitoring. *ACM SIGCOMM Computer Communications Review* 2001; **31**(2):9–24.
21. ITU-T Recommendation G.107. The E-model, a computational model for use in transmission planning, December 1998.
22. ITU-T Recommendation G.113. General characteristics of general telephone connections and telephone circuits—transmission impairments, February 1996.
23. ITU-T Recommendation G.711. Pulse code modulation (PCM) of voice frequencies, November 1988.

AUTHORS' BIOGRAPHIES

Giorgos Papastergiou received a diploma in Electrical and Computer Engineering from Democritus University of Thrace, Greece in 2005 and an MSc degree in Informatics from Aristotle University of Thessaloniki, Greece in 2007. He is currently working toward the PhD degree in the Department of Electrical and Computer Engineering, Democritus University of Thrace, Greece. His current research interests are in the area of transport protocols for deep space communications and Delay-Tolerant Networking.

Chryssa Georgiou received a diploma in Electrical and Computer Engineering from Democritus University of Thrace, Greece in 2005 and an MSc in Systems Engineering & Management from Democritus University of Thrace, Greece in 2008. She has worked with SingularLogic Group, Greece as an Oracle software developer from 2007 to 2008. She is currently working with Cosmote, Greece in the field of transmission network engineering.

Lefteris Mamatas is a Lecturer in the Department of Technology Management at the University of Macedonia. He received his PhD in June 2008 from the Democritus University of Thrace. He has long experience in protocol engineering (e.g., energy efficient transport/network layer protocols, protocols for DTN & User-Provided Networks etc), experimental & theoretical analysis on performance evaluation and test-bed implementation. He was a founder of Araneous Internet Services (an IT company situated in North Greece). In 2009, he chaired the IEEE Workshop on the Emergence of Delay/Disruption-Tolerant Networks (E-DTN). He published 10 journal papers, 17 conference papers and 1 book. More information can be found at: http://users.uom.gr/∼emamatas.

Vassilis Tsaoussidis received a BSc in Applied Mathematics from Aristotle University, Greece; a Diploma in Statistics and Computer Science from the Hellenic Institute of Statistics; and a PhD in Computer Networks from Humboldt University, Berlin, Germany (1995). Vassilis held faculty positions in Rutgers University, New Brunswick, SUNY Stony Brook and Northeastern University, Boston. In May 2003, he joined the Department of Electrical and Computer Engineering of Democritus University, Greece. His research interests lie in the area of transport/network protocols, i.e., their design aspects and performance evaluation. Vassilis is editor in chief for the Journal of Internet Engineering and editor for the journals IEEE Transactions in Mobile Computing, Computer Networks, Wireless Communications and Mobile Computing, Mobile Multimedia and Parallel Emergent and Distributed Systems. He participated in several Technical Program Committees in his area of expertise, such as INFOCOM, NETWORKING, GLOBECOM, ICCCN, ISCC, EWCN, WLN, and several others.