# Differentiating Services with Non-Congestive Queuing (NCQ)

Lefteris Mamatas, *Member, IEEE,* Vassilis Tsaoussidis, *Senior Member, IEEE*

***Abstract*— We discuss a new packet service paradigm, called 'Less Impact Better Service' (LIBS), which is realized through a novel queuing discipline, called 'Non-Congestive Queuing' (NCQ). NCQ prioritizes small packets when conditions permit, and utilizes service thresholds to confine the delay impact of prioritization on congestive applications. We show that LIBS and NCQ satisfy more users with diverse demands on delay and throughput. We obtained both analytical and simulation results, which are very promising. Diversity is simulated by using FTP, sensor and VoIP traffic.**

***Index Terms*— QoS, Service Differentiation, Packet Scheduling**

## I. Introduction

Typical scheduling paradigms of packet networks do not match well the requirements of non-congestive applications, which transmit minor data volumes but suffer, however, major queuing delays. 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 may experience almost-zero delay favored by a prioritized scheduling scheme, at an almost-zero cost to other congestive flows. In the same context, a voice application, although it generates periodic traffic at small rates, experiences delays comparable with other applications that transfer large data volumes.

Typical service paradigms assume resource demand exceeding resource supply, thus focusing on bandwidth sharing among flows. Other service paradigms incorporate a proportional service scheme, focusing on bandwidth allocation in proportion to the demand. However, both perspectives lack a delay-oriented service discipline. Delay-oriented services have been traditionally managed based on delay requirements of some applications, which are eventually reflected in the prioritization during scheduling. Thus, service disciplines are primarily application-oriented and have the inherent property to satisfy some applications more, rather than satisfying more applications.

We exploit here a system-oriented service discipline, which targets in satisfying more users. We depart from two main observations:

1) Non-congestive flows do not cause significant delays and hence should not suffer from delays. We call this service discipline 'Less Impact Better Service' (LIBS).
2) As it is with Internet voice where the users tolerate more delays with Internet than with dedicated telephone lines, users of long and congestive applications such as FTP, tolerate more delays than the users of small-data sensor applications.

In this context, bandwidth alone could not have the central role; instead, efficient distribution of resources needs to be characterized by the delay suffered by each flow in relation to the delay they cause. The latter is occasionally associated with the delay that users tolerate. Our service approach promotes small-sized packets at small rates, which define 'non-congestive' traffic. To avoid starvation and also significant delay impact on congestive traffic, non-congestive traffic is confined by corresponding service thresholds. Hence, we analyze the behavior of systems where non-congestive traffic has controlled prioritization without affecting congestive traffic. From a user perspective, applications that utilize small data packets and rates (and are also intolerant to long delays) are satisfied while other applications suffer almost-zero extra delays.

The key idea of Non-Congestive Queuing (NCQ) [27], [28] that we discuss here, departs from the operational dynamics of gateways: they may service small packets instantly. Non-congestive packets do not cause significant delays and hence should not suffer from delays. Although our approach sounds straightforward, the system properties and design details reveal interesting dynamics. The simplicity of NCQ algorithm reduces implementation and deployment effort. NCQ does not require any modification at the transport protocol or packet marking; a minor modification of the gateway's software is sufficient.

Our primary assumption is that some applications, such as sensors or even VoIP generate packets in the form of non-congestive traffic. Typically, they transmit periodically small packets. For example, in [12] the authors claim that both the packet size and data rate in many typical sensor networks are very small. In [40] the authors assume a maximum sensor packet size of 44 bytes while in [37] it is considered that typical packet sizes in a sensor network are 32 bytes, 64 bytes, 96 bytes, and 128 bytes.

Sensor and voice data have strict requirements in delay and the service received by the network cannot be judged on the basis of achieved throughput. This observation calls for a new metric for application fairness as well, which relies mainly on the delay rather than on throughput. We introduce the *Application Satisfaction Index (ASI)*, which captures the delay fair share per application on the basis of the delay impact of each application on others. Thus, ASI reflects how fairly applications receive service, delay-wise, under diverse delay expectations and impact.

The proposed service paradigm impacts other performance measures as well, such as energy expenditure, which is very significant indeed for energy-limited sensors or VoIP mobile devices. The gains in energy are achieved through reduction of the communication and hence application time and their importance varies depending on the device itself,

the communication pattern, the network contention, etc. We show that NCQ improves energy efficiency of sensor devices and other non-congestive applications, without damaging the dynamics of multiple-flow equilibrium and without causing any statistically important *Goodput* losses to the congestive flows.

In section II we discuss the related work. In sections III and IV, we provide the pseudo-code of NCQ and present the basic assumptions, fundamental concepts and projected outcome, based on analytical methods. Furthermore, we explore the impact of different application strategies regarding the size of the transmitting packets, assuming that a LIBS-based approach is deployed. In section V we discuss the experimental evaluation plan and metrics, along with a justification for the plan. Furthermore, we present the results, analysis and justification and demonstrate the impact of service thresholds on system performance. In section VI we discuss several open issues and address reasonable concerns. In section VII we summarize our conclusions.

## II. Related Work & Discussion

Service differentiation in computer networks is a topic of research, mainly focused on supporting application requirements for delay and bandwidth. Differentiation is mainly an operation that deals with reallocating bandwidth and controlling delay for the benefit of strict-service-requiring applications and at the cost of more flexible applications. Therefore, service differentiation is based on the principle 'get better service - if you need better service'.

Internet service differentiation has not been designed on the basis of theoretical research; rather, it was driven by the need for supporting real-time multimedia applications over the Internet. Such applications do have strict bandwidth and delay requirements; the flows that generate can describe their specifications in broad or detailed terms, and the network can plan for guaranteed service or otherwise for somewhat better service. Two basic approaches have gained acceptance: According to diffserv [3], the inherent properties of packet-switched Internet are masked with a number of gentle mechanisms, naturally matching Internet's structure, which - one way or another - shape traffic at some functionally-enhanced nodes. Alternatively, with intserv [4] the architecture itself can be redesigned to allow for guarantees through signaling and reservation.

Differentiation in both the aforementioned cases 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 requires a rather sophisticated implementation, ranging from packet marking, to shaping, scheduling and dropping schemes. 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. It is not unnatural 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 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 to congestive applications is not degraded. Thus, only a limited amount of non-congestive data should be able to benefit from our differentiating scheme.

According to [2], the average delay for the system tends to be reduced when customers with short service times are given high priority. The authors give examples from everyday life such as special checkout counters for customers with few items or the waiting lines at copying machines, where people often give priority to others who need to make just a few copies.

A lot has been done in the networking community aiming at controlling traffic based on its characteristics. Controlling is implemented either through scheduling or through dropping policies mainly aiming at penalizing high - bandwidth - demanding flows rather than favoring low - bandwidth - demanding flows. In [14] Floyd and Fall introduced mechanisms based on the identification of high-bandwidth flows from the drop-history of RED [15]. The RED-PD algorithm (RED with Preferential Dropping) [26] uses per-flow preferential dropping mechanisms. Two other approaches that use per-flow preferential dropping with FIFO scheduling are Core-Stateless Fair queuing (CSFQ) [38] and Flow Random Early Detection (FRED) [23]. CSFQ marks packets with an estimate of their current sending rate. The router uses this information in conjunction with the flow's fair share estimation in order to decide whether a packet needs to be dropped. FRED does maintain a state although only for the flows which have packets in the queue. The flows with many buffered packets are having an increased dropping probability.

The CHOKe mechanism [34] matches every incoming packet against a random packet in the queue. If they belong to the same flow, both packets are dropped. Otherwise, the incoming packet is admitted with a certain probability. Authors of [33] extended the CSFQ and CHOKe approaches. The Stochastic Fair Blue (SFB) [13] uses multiple levels of hashing in order to identify high-bandwidth flows. Anjum and Tassiulas proposed in [1] a mechanism that drops packets based on the buffer occupancy of the flow while ERUF [36] uses source quench to have undeliverable packets dropped at the edge routers. On the other hand, SRED [32] caches the recent flows in order to determine the high-bandwidth flows.

In [18], the authors introduced the Alternative Best Effort mechanism (ABE). ABE improves performance of delay-sensitive traffic but uses only two possible traffic classifica-

tions: delay- and throughput-sensitive. Delay-sensitive applications sacrifice throughput, and vice versa. Traffic Sensitive QoS mechanism (TSQ) [7] 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 several undesirable properties:

- The markings are predetermined and cannot correspond to various possible packet sizes. They inherently produce a 'quantization' error.
- Marks correspond to priorities and do not take into account the impact of prioritization. For example, a favorably-marked packet will be serviced first even if the following packet will have zero impact to its service.
- Marks are application-level service requests; the system can find an optimal operating point for multiplexed applications only by introducing system-oriented criteria.

We note that packet marking and NCQ are not competitive technologies. The former can complement the latter through a second level of prioritization. For example, different priorities can be assigned via packet marking to the different non-congestive or congestive applications.

## III. Our approach: Non-Congestive Queuing

We assume different classes of packets according to their size. NCQ is incorporated into routers to differentiate services according the impact of each traffic class on delay. For example, a class with smaller packets and sending rate receives better service than one with large packets or high sending rate. A natural question therefore is what if small-packet rate reaches levels, which delay significantly long-packet transmission. We complement the differentiating scheme with a service threshold: The favored 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.

On the other hand, a typical application could be intentionally transformed into a small-packet, high-rate application. Since the *ncqthresh* and the packet length confine the amount of gain, the transformation should cause that much overhead and extended communication time that naturally the penalty of transformation will be greater than the gain. In section IV-B, we calculate numerically the impact of such transformation.

Although the perspective of NCQ is more general, initially, we only deal with two classes of packets: small packets (up to 130 bytes[2]) and long packets that typical Internet applications use for data transfers. The threshold of 130 bytes follows the assumption that typical sensor applications usually range from 32 to 128 bytes [37]. NCQ uses non-preemptive priority queuing to implement priority service. That is, 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 to 130 bytes and (ii) prioritized service rate is below *ncqthresh*.

The algorithm below shows the pseudo-code for NCQ:

```
for every received packet
begin
        count received packets
        (congestive and non-congestive)
        if (packetLength<130)
         and
          (favored_packets /
           received_packets < ncqthresh)
        then
         packet receives high priority
         count favored packets
        else
          packet receives normal priority
        end
end
```

In this stage of our work, we do not favor ACKs even though they have a small size. We note that NCQ may occasionally favor a part of a non-congestive data flow. However, a possible slight increase[3] of the re-ordered packets is counterbalanced by the high number of packet drops that are avoided due to the prioritization.

## IV. Analysis

### A. Impact of NCQ

Initially, 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[4]. 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.

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.

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

[2]For longer packets, we experimentally determined that the cost of prioritization for the congestive flows starts to be significant.

[3]It is not significant in our experiments.

[4]It is widely adopted (such as [16], [30], [22], [10]) that the packet arrival process for highly multiplexed environments tends to a Poisson Distribution.

| Symbol | Description |
|---|---|
| $\lambda_1$ | Arrival rate of class 1 |
| $\lambda_2$ | 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 |
| $u_1 = \lambda_1 T_{S1}$ | Utilization of class 1 |
| $u_2 = \lambda_1 T_{S1} + \lambda_2 T_{S2}$ | Cumulative utilization |
| $T_{Q1}$ | Average queuing delay for class 1 |
| $T_{Q2}$ | Average queuing delay for class 2 |
| $T_Q$ | Average queuing delay |

TABLE I
NOTATION TABLE

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:

$$T_{W1} = \frac{\lambda_1 T_{s1}^2 + \lambda_2 T_{s2}^2}{2(1-u_1)} \tag{1}$$

$$T_{W2} = \frac{\lambda_1 T_{s1}^2 + \lambda_2 T_{s2}^2}{2(1-u_1)(1-u_2)} \tag{2}$$

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

$$T_W = \frac{\lambda_1}{\lambda} T_{W1} + \frac{\lambda_2}{\lambda} T_{W2} \tag{3}$$

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

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

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

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

*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:

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

The service time:

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

The average waiting time:

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

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:

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

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)).

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 $\frac{\lambda_1}{\lambda} * ncqthresh$ 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:

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

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

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

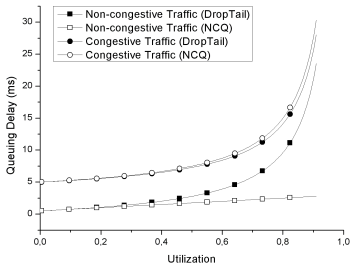
$$T_{S1}' = T_{S1}$$

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

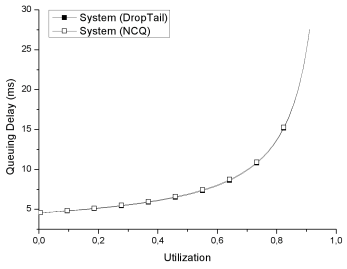
$$T_{S2}' = (1 - \frac{\lambda_1}{\lambda} * ncqthresh) * \frac{\lambda_1}{\lambda_2'} * T_{S1} + \frac{\lambda_2}{\lambda_2'} * T_{S2}$$

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

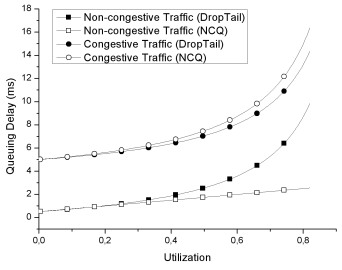
$$u_2' = \lambda_1' T_{S1}' + \lambda_2' T_{S2}'$$



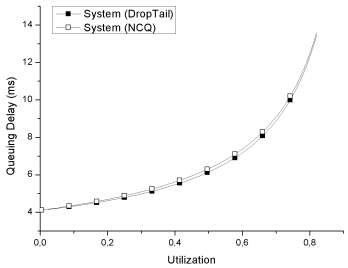

(a) Average Queuing Delay of Congestive and Non-congestive Traffic (10% of the packets are Non-congestive)




(b) System's Average Queuing Delay (10% of the packets are Non-congestive)




(c) Average Queuing Delay of Congestive and Non-congestive Traffic (20% of the packets are Non-congestive)




(d) Average Queuing Delay of Congestive and Non-congestive Traffic (20% of the packets are Non-congestive)

Fig. 1. Numerical Results

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

$$T'_{W1} = \frac{\lambda'_1 T'^2_{s1} + \lambda'_2 T'^2_{s2}}{2(1-u'_1)} \tag{13}$$

$$T'_{W2} = \frac{\lambda'_1 T'^2_{s1} + \lambda'_2 T'^2_{s2}}{2(1-u'_1)(1-u'_2)} \tag{14}$$

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

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

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

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

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

For the system:

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

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 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)).

### *B. Design Strategies of Applications*

Assuming that a LIBS-based mechanism is deployed, an application may follow alternative strategies in order to have performance gains. Here, we explore analytically such strategies. We assign a specific task (i.e., to transfer 5MB) to each of the traffic classes and calculate the service time of each task.

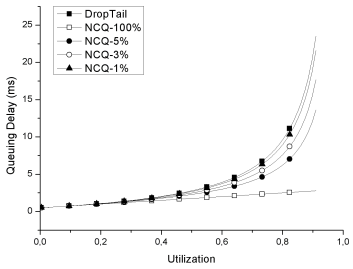
The required number of transmitted packets for a specific class $n$ ($NP_n$):

$$NP_n = \frac{Total\_Task\_Data_n}{Packet\_Size_n - Overhead\_Per\_Packet} \tag{20}$$

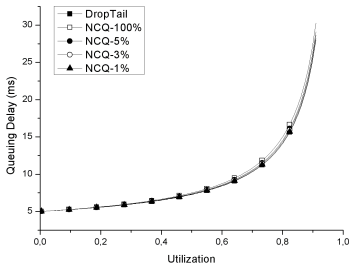
The service time for a specific class $n$ ($ST_n$):

$$ST_n = T_{Q1} * NP_n \tag{21}$$

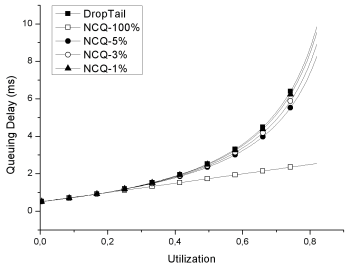
The following analysis is based on the equations (20), (21). We assume that each packet has 40 bytes overhead.



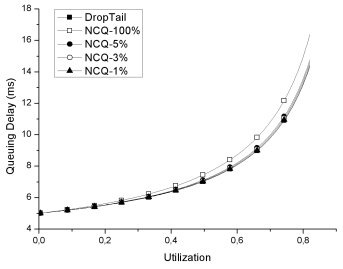

(a) Average Queuing Delay of the Non-congestive Traffic (10% of the packets are Non-congestive)




(b) Average Queuing Delay of the Congestive Traffic (10% of the packets are Non-congestive)




(c) Average Queuing Delay of the Non-congestive Traffic (20% of the packets are Non-congestive)




(d) Average Queuing Delay of the Congestive Traffic (20% of the packets are Non-congestive)

Fig. 2. Numerical Results

*1) FTP Application:* We assume two different traffic classes which consist of large packets, in order to approach numerically a similar behavior to bulk-data transfer (FTP). Each class has 1KB packets. The 10% of the arriving packets form the first class and the 90% the second one. An application designer may decide to use smaller packets in order to have performance gains due to the LIBS-based mechanism. Is this effective? We show that such applications should, in general, avoid splitting their packets into smaller ones.

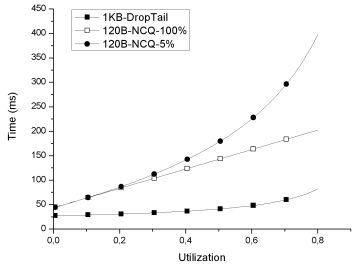



Fig. 3. Numerical Results of FTP Application

In Figure 3, we depict the total service time required for the first traffic class to transfer 5 MB, in three distinct cases:

1) The FTP application uses 1KB packets.
2) The FTP application uses 120-byte packets, the NCQ mechanism has been deployed but there is no limitation like *ncqthresh*.
3) The FTP application uses small packets (120-byte) and the NCQ mechanism has been deployed with an *ncqthresh* with value 0.05.

According to Figure 3, there is a significant performance decrease in terms of total service time for the NCQ mechanism without an *ncqthresh* limitation (e.g., 240% decrease for utilization 0.60). In the case of an *ncqthresh* with the value of 0.05, the performance decrease reaches 371% (for 0.60 utilization). Although the small packets are favored from the NCQ mechanism, the increased overhead due to the higher number of required packets overcomes these gains.

*2) Congestive Multimedia Application:* In this scenario, we used two traffic classes: 500-byte and 1KB packets, respectively. We assume that the first class follows the traffic pattern of a Congestive Multimedia Application (uses packets with average-size). The 10% of the arriving packets form the first class and the 90% the second one. Although in this case we have less extra overhead, the performance gains of the NCQ mechanism are still inadequate to overcome this drawback (Figure 4). For utilization 0.60, the performance decrease in case of NCQ without *ncqthresh* is 143% and in case of NCQ with *ncqthresh* 0.05 is 253%.

*3) Non-congestive Multimedia Application:* In this scenario, we approach numerically a non-congestive multimedia application. The first class represents the multimedia application and forms the 10% of the traffic and the second class the FTP application. The multimedia application has a packet size of 140 bytes. In this scenario, we investigate whether it is worth for this application to change its packet size from 140 to 120 bytes.

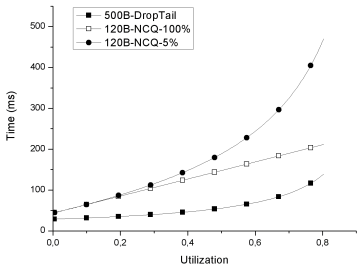



Fig. 4. Numerical Results of Congestive Multimedia Application

For both versions of NCQ we have a significant decrease in the average queuing delay (see Figure 5) but is inadequate for a significant improvement in terms of task completion time due to the increased overhead (Figure 6). As we can see from Figure 6, for the NCQ without *ncqthresh* we have an improvement in terms of total service time for higher utilizations than 0.48 (e.g., 21% for utilization 0.60). In the case of NCQ with *ncqthresh* 0.05 we have a performance decrease for all utilizations (e.g., 19% for utilization 0.60). Consequently, in special cases such applications may adjust their packets in order to have performance gains.

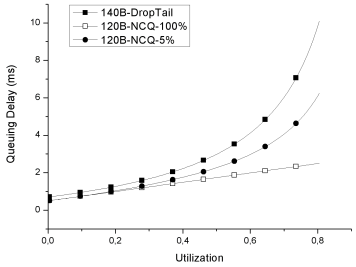



Fig. 5. Queuing Delay for Non-Congestive Multimedia Application

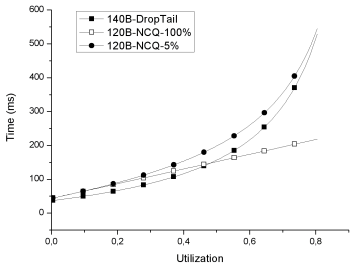



Fig. 6. Task Completion Time for Non-Congestive Multimedia Application

To summarize, the policy to adjust the size of the transmitted packets to the NCQ threshold is very often inefficient. However, there are cases that an application can be favored from the NCQ mechanism, if it adopts smaller packets. For example, a non-congestive application may slightly reduce its packet size in order to have performance gains from a LIBS-based packet scheduling algorithm.

Furthermore, in case a malicious application tries to monopolize communication by transmitting small packets at high data rates, the extra introduced overhead is not counterbalanced by the gains. A Denial of Service attack (DoS) that uses aggressive flows consisting of small packets, in the worst case, would disable the extra prioritization of NCQ. However, if the attack uses large packets, many non-congestive applications would not suffer from the exhaustion of the resources.

## V. Evaluation

### A. Evaluation Methodology

We have implemented our evaluation plan on the ns-2 network simulator [31]. We attempt to address five specific matters:

1) To show by simulation that numerical results do not lack any important parameter (Scenario 1: Impact of NCQ on FTP applications).
2) To show the impact of NCQ on sensor-based applications (Scenario 2: Internetworking with Sensors).
3) To evaluate the applicability of NCQ mechanism for VoIP traffic (Scenario 3: Impact of NCQ on VoIP traffic).
4) To explore whether NCQ can be incrementally deployed (Scenario 4: Incremental deployment of NCQ).

We discuss each scenario along with the corresponding results.

### B. Evaluation Results

*1) Scenario 1: Impact of NCQ on FTP applications:* In this scenario, we use a simple dumbbell topology as shown in Figure 7.

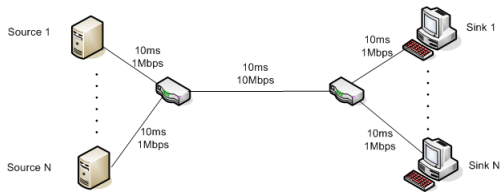



Fig. 7. Simulation topology

We measure:

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

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 $Time$ is the amount of time required for the corresponding data delivery. We used the *Average Goodput* in order to measure the efficiency per flow. The *Average Goodput* for *n* flows is defined as:

$$AverageGoodput = \frac{\sum_{i=1}^{n}(Goodput_i)}{n}$$

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

Additionally, we measure application efficiency and user satisfaction based on the *worst* and *average task completion*

*time* as well as on the *number of completed tasks*. Every congestive FTP flow transmits 1MB data and every non-congestive FTP flow 10KB. We assume that a task is completed with the successful transmission of the carried data of the corresponding flow.

Furthermore, we experiment with different traffic thresholds and traffic class proportions, to demonstrate the overall system behavior when the NCQ parameters change. We carried out the same experiment with different values of *ncqthresh* (i.e., 0.01, 0.03 and 0.05) in order to evaluate the impact of traffic threshold (*ncqthresh*). As we can see from the following results, in average, the value of 0.05 is a good choice.

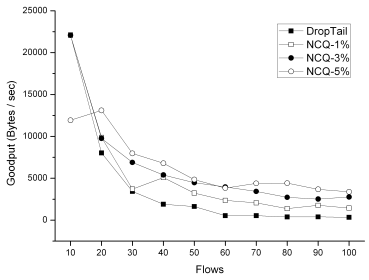
At this point, we adjust the number of non-congestive flows to the 10% of the total flows. As we can see in Figure 8(a), the non-congestive flows achieve significant performance gains (e.g., 10.43 times - in the case of 80 flows and *ncqthresh* 5%) in terms of *Goodput*. The impact of the prioritization on the congestive flows is not only insignificant but we also notice occasionally a slight performance improvement in terms of *Goodput* for the congestive flows (e.g., 7.6% improvement - in the case of 100 flows and *ncqthresh* 5%) and the overall system (Figures 8(b), 8(c)). This is not unreasonable: the impact of timeouts caused by short packets is more significant for non-congestive flows compared with the impact of long packets; i.e., regardless of the packet length, timeout is the same and extending total time for a small retransmission degrades system throughput. It is very interesting to note that the NCQ has a positive impact also in the system's average / worst task completion time (Figures 9(a), 9(b)). According to Figure 9(b), all tasks are completed up to 24.8 seconds sooner (i.e., in the case of 90 flows and *ncqthresh* 3%).

Next, we varied the traffic proportion of congestive and non-congestive flows. As the rate of the non-congestive flows increases (i.e., the number of the non-congestive packets increases), the benefit for non-congestive packets is gradually decreasing. This behavior is not symptomatic: the *ncqthresh* value confines the priority service to guarantee small impact on congestive flows. For example, we can see in Figure 10(a) (rate of non-congestive packets 20%) that the NCQ algorithm favors now a smaller portion of the non-congestive packets. However, the impact on the average task completion time of the non-congestive flows remains significant (Figure 10(b)).

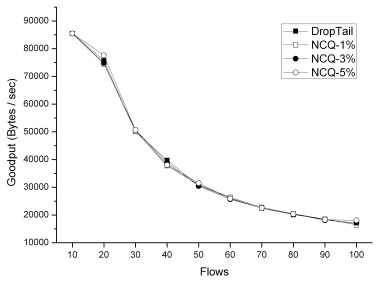
Additionally, In figure 11 we depict the number of completed tasks at different time instances for both congestive and non-congestive flows. In this example, we consider the case of the 50 flows. We show that, on average, more applications finish earlier. For example, at the 30.5th second 7 flows are finished in the case of DropTail and 10 flows in the case of the NCQ-1%.

We adjusted the task of the FTP flows to be 100KB, assuming short-lived flows (e.g., like a typical web-page). As we can see from the figures 12(a), 12(b) the non-congestive flows have a significant improvement in terms of goodput without noticable impact on the goodput of the short-lived FTP flows.

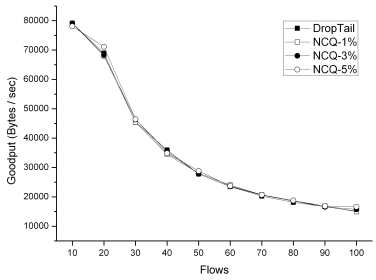
*2) Scenario 2: Internetworking with Sensors:* For a more realistic scenario of internetworked sensor applications, we integrated NRL's Sensor Network Extension [11] into ns-2. Our scenario consists of a sensor network (a grid of 25 wireless sensors) and a simple dumbbell wired topology (see Figure 13). The sensor application transfers periodically data to a wired node (the data collector). The sensor application notifies the data-collector about the behavior of a moving phenomenon. The simulated phenomenon has a pulse period of 0.01 seconds. The sensor-generated data and the congestive FTP flows coexist in the same link and cross the same NCQ-enabled gateway. The number of FTP flows ranges from 10 to 100. We used the TwoRayGround radio-propagation model and the AODV [35] routing protocol. We measured *Goodput* on both congestive (FTP) and sensor-related non-congestive traffic.




(a) Average Goodput of Non-Congestive Flows

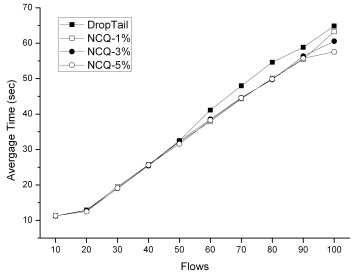



(b) Average Goodput of Congestive Flows

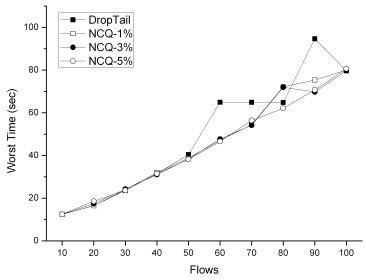



(c) Average System Goodput

Fig. 8. Impact of NCQ on Goodput

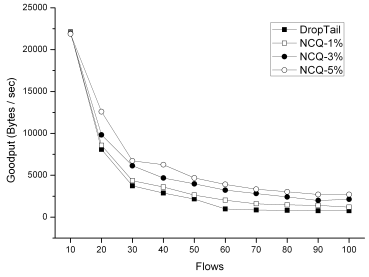



(a) Average Task Completion Time in the System

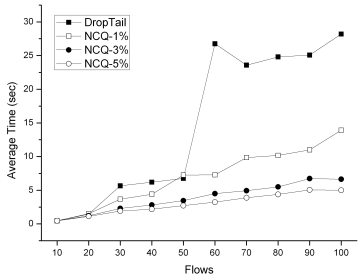



(b) Worst Task Completion Time in the System

Fig. 9. Impact of NCQ on FTP applications




(a) Average Goodput of Non-Congestive Flows (20% of the packets are non-congestive packets)




(b) Average Time of Non-Congestive Flows (20% of the packets are non-congestive packets)

Fig. 10. Impact of NCQ on Non-congestive applications

We evaluate the energy-efficiency of NCQ using the *Energy Potential (EP)* [24] index:

$$EP = 1 - (a\frac{Throughput - Goodput}{Throughput_{max}}$$

$$+b\frac{Throughput_{max} - Throughput}{Throughput_{max}})$$

The *EP* index takes into account the difference of achieved *Throughput* from *maximum Throughput* ($Throughput_{max}$) for the given channel conditions along with the difference of *Goodput* from *Throughput*, attempting to locate the *Goodput* as a point within a line that starts from 0 and ends at $Throughput_{max}$.

In order to measure fairness in the context of LIBS, we introduce *Application Satisfaction Index (ASI)*. *ASI* is defined as:

$$ASI = 1 - \frac{\left|\sum_{i=1}^{n} Delay_i - \frac{Data_i}{TotalData} Delay_{max}\right|}{nDelay_{max}}$$

Where, *n* is either the number of active nodes or the number of different traffic classes; $Data_i$ the total transmitted data of the $i_{th}$ node to the receiver application; *TotalData* the total transmitted data of all nodes; $Delay_i$ the average queuing delay of the $i_{th}$ node; and $Delay_{max}$ the maximum queuing delay of the system. *ASI* ranges from 0 to 1.

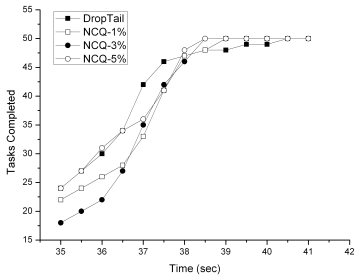
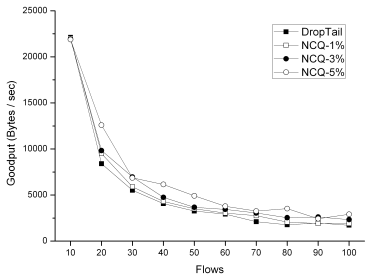


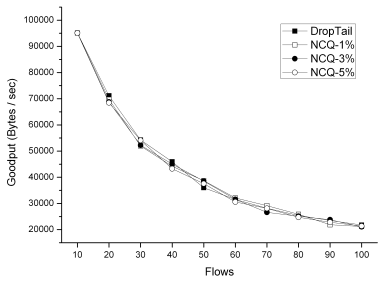

Fig. 11. Tasks Completed (50 flows)

Unlike other fairness indices (such as [6], [29], [39]), *ASI* captures the deviation of the actual delay and the expected delay per flow. Note, however, that expected delay is determined by the factor $\frac{Data_i}{TotalData}$. In this context, *ASI* represents fairness of the *LIBS* architecture, since the expected delay per packet (and in turn, per flow) grows in proportion to the volume of their transmitted packets.

In Figure 14(a) we depict the *Goodput* performance of the congestive flows and in Figures 14(b), 14(c), 15(a), 15(b) we demonstrate the *Goodput*, the *Energy Potential* and the *Application Satisfaction Index* for both overall system and Sensor Applications. We argue that the congestive FTP flows are not suffering from any important performance loss (see Figure




(a) Average Goodput of Non-Congestive Flows




(b) Average Goodput of Short FTP Flows

Fig. 12. Impact of NCQ on Non-congestive Applications & Short FTP Flows

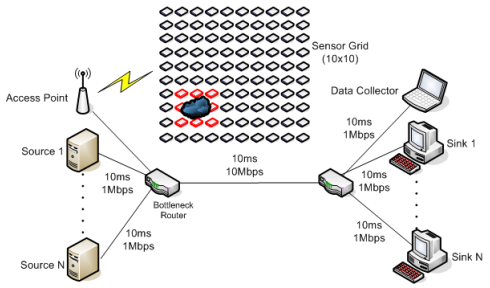
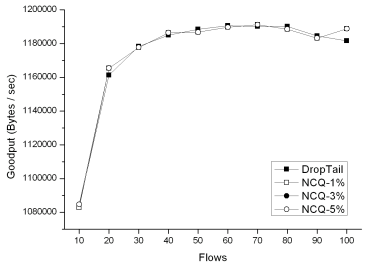
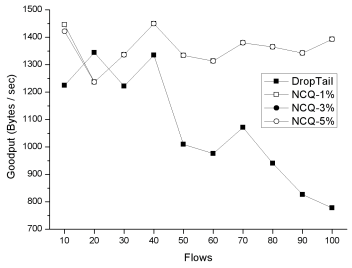


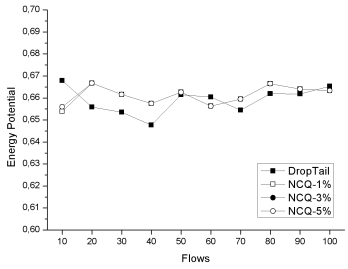

Fig. 13. Simulation topology




(a) Goodput of FTP Flows




(b) Goodput of Sensor Applications




(c) Energy Potential (Non-Congestive Flows)

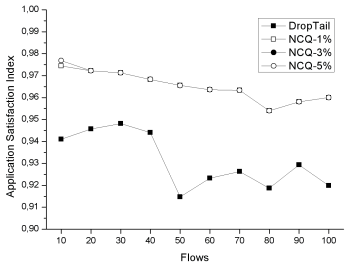
Fig. 14. Impact of NCQ on Goodput/Energy Efficiency for Sensor Internetworking

14(a)), while the performance gains for the non-congestive flows are significant (up to 92%) in terms of *Goodput* as well as in terms of energy efficiency (see Figure 14(c)). Additionally, the system appears fair according to *ASI* (Figures 15(a), 15(b)). We note that, due to *Goodput* increase of sensor applications, fairness performance may also be captured by the traditional index of fairness.

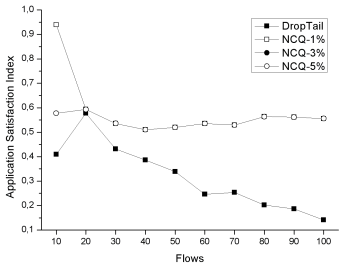
As we demonstrate in Figures 14(a), 14(b), 14(c), 15(a), 15(b), the value of *ncqthresh* does not impact the results when the system has more than 20 flows since, from that point onwards, the percentage of non-congestive traffic is less than 1%. So, every packet generated by the sensor applications is prioritized.

*3) Scenario 3: Impact of NCQ on VoIP traffic:* 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 [5], as well as the heavy-tailed characteristics and self-similarity of VoIP traffic [9], we use the *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 [5], we distribute the ON and OFF periods with means of 1.0s and 1.35s, respectively. We simulate VoIP streams of 64kbps (following the widely-used ITU-T G.711 [21] coding standard) and we set packet sizes at 120 bytes (i.e., each packet carries 10ms G.711-encoded speech and a 40-byte packet header).

In the following results, we characterize the quality of voice communication using the R-Factor, which is included in the E-Model [19], [20] (an ITU-proposed analytic model of voice




(a) Application Satisfaction Index for the System




(b) Application Satisfaction Index of Sensor Applications

Fig. 15. Impact of NCQ on Fairness for Sensor Internetworking

quality). R-Factor captures voice quality and ranges from 100 to 0, representing best and worst quality, respectively. R-Factor is also associated to the familiar estimated Mean Opinion Score (MOS). MOS equals to the arithmetic average of opinions where "excellent" quality is given a score of 5, "good" a 4, "fair" a 3, "poor" a 2 and "bad" a 1. R-Factor incorporates several different parameters, such as echo, background noise, signal loss, codec impairments and others. In [8], the 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:

$$R = \alpha - \beta_1 d - \beta_2 (d - \beta_3) H(d - \beta_3) - \gamma_1 - \gamma_2 ln(1 + \gamma_3 e) \quad (22)$$

where $\alpha = 94.2$, $\beta_1 = 0.024ms^{-1}$, $\beta_2 = 0.11ms^{-1}$, $\beta_3 = 177.3ms$, $d$ 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** $0 < R < 100$**:** $MOS = 1 + 0.035R + 7x10^{-6}(100 - R)$

For reference, we give the relation of R-Factor to MOS according to Table II.

In our scenario, we introduce extra delay of 10ms because of the G.711 voice encoding (according to [8]). We use the simple dumbbell topology of Figure 7 and adjust the *ncqthresh* to 0.01. We range the number of congestive flows from 10 to 100.

| R-Factor | Quality of voice rating | MOS |
|---|---|---|
| $90 < R < 100$ | Best | 4.34 - 4.5 |
| $80 < R < 90$ | High | 4.03 - 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 |

TABLE II

R-FACTOR, QUALITY RATINGS AND MOS

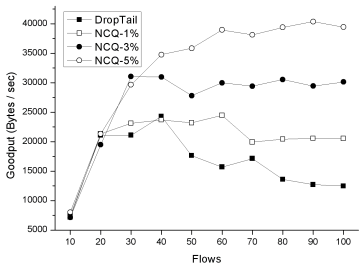
According to Figure 16(a), the VoIP traffic has significant performance gains in terms of *Goodput*. In addition, the NCQ mechanism is not degrading the performance of the congestive FTP flows significantly (Figure 16(b)) - especially when *ncqthresh* value is 0.01. Furthermore, fairness among VoIP applications is improved (Figure 16(c)).

According to Figures 17(a), 17(b), there is a significant improvement in voice quality, while more calls are rated better (Figures 17(a), 17(b)). For example, in case of 50 calls, all are rated "poor"; NCQ improves their rating to medium (Figure 17(b)). Practically, NCQ satisfies more up-to 50 users. For more than 80 calls, several calls are rated "poor", exhausting resources unduly.

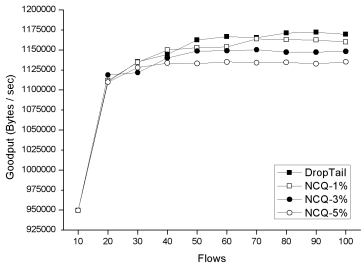
*4) Scenario 4: Incremental deployment of NCQ:* Here, we evaluate NCQ using a complex topology that incorporates multiple bottlenecks, cross and reverse traffic (Figure 18). In this scenario, we explore whether NCQ can be incrementally deployed. This analysis is important because the incremental deployment of a new network service or protocol is typically a hard problem, especially when it has to be deployed in the routers [17].

More specifically, we range the number of routers that incorporate the NCQ scheme from 0 to 4, assuming incremental deployment. The *ncqthresh* value is set to 0.03 and we have a random number of applications for both congestive and non-congestive traffic; uniformly random distributed in [1, 50] and [1, 10], respectively. The *bw* value is 10Mbps. We measure *Goodput*, *Application Satisfaction Index* and *R-Factor*. The queue size is 100 packets and the duration of the experiment is 60 sec.

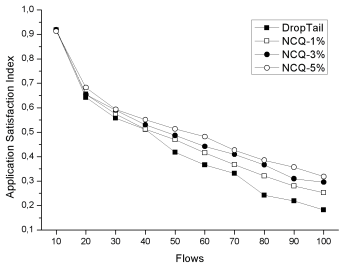
We note that this experiment is non-deterministic. Consequently, we performed 50 runs in order to have statistically accurate results. The standard deviation of our measurements is often high due to the nature of our experiment. For example, in the case of randomized contention, one flow receives almost always less throughput than 50 flows. In such results, the standard deviation for the measurements does not increase by the number of experimental runs. Each time the standard deviation of the differences between the average value and the samples is very small (e.g., 1-3%), we depict the average values and the confidence intervals (i.e., the above standard deviation). Otherwise, we analyze and interpret our results using frequency tables. More precisely, we group measurements into suitable intervals that allow us to find the most common values. Each figure that uses a frequency table, depicts the percentage of performance improvement/decrease of each measure, using




(a) Goodput of Non-Congestive VOIP Application

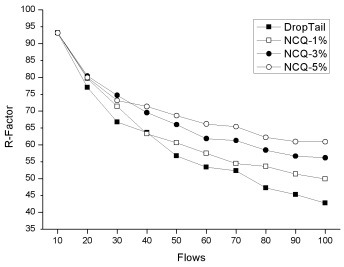



(b) Goodput of Congestive Application (FTP)

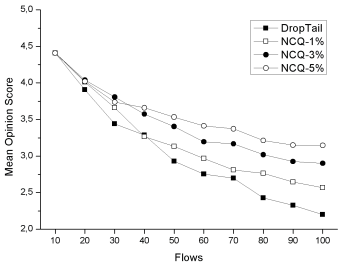



(c) Application Satisfaction Index of VoIP Application

Fig. 16. Impact of NCQ on Goodput and Fairness




(a) R-Factor of VoIP Application




(b) MOS of VoIP Application

Fig. 17. Impact of NCQ on Voice Quality

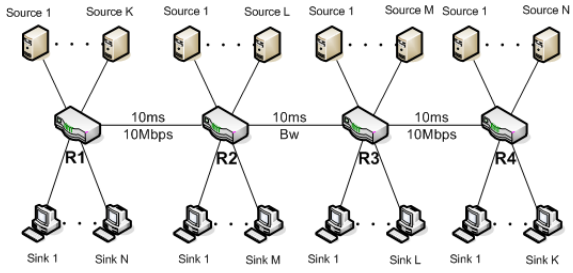



Fig. 18. Complex Network Topology

as a reference point the experimental results of a scenario using DropTail.

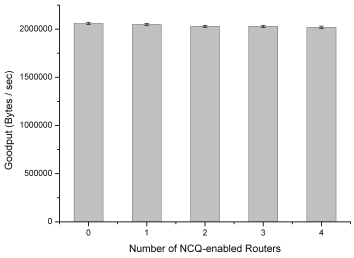
In figure 19(a), we have only a minor impact on the *Goodput* of the congestive flows for any number of NCQ-enabled routers. However, the *Goodput* of the non-congestive applications is significantly improved (Figure 19(b)). When NCQ is fully deployed (4 NCQ-enabled routers - Figure 19(b)), NCQ achieves more than 100% improvement for the 38% of the runs.

The significant improvement in terms of *Goodput* for the VoIP applications is reflected in the voice quality of each call. For example, in the case of 75% deployment (3 NCQ-enabled routers), in the majority of the runs (i.e., 88%), the VoIP flows achieve improvement that reaches 50%. For the 4% of the runs the improvement is more than 50%. The improvement in terms of *Fairness* is also significant, in the 82% of the runs we have an improvement on the *Application Satisfaction Index* (Figure 20(b)).

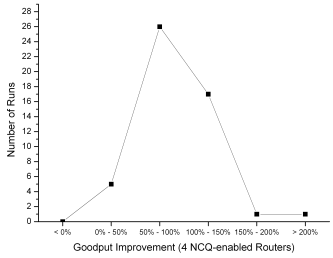
More results on the evaluation of NCQ and other similar LIBS-based mechanisms can be found in [25], including complex topologies with different congestive and non-congestive application types.

## VI. Open Issues

Although we demonstrated NCQ's high potential, we do not presently address all concerns necessary to justify its need for deployment. We are working on an extension of the algorithm to assign probabilistically priority service to small packets. The probability per packet decreases as the rate of non-congestive packet exceeds the *ncqthresh*. Probabilistic

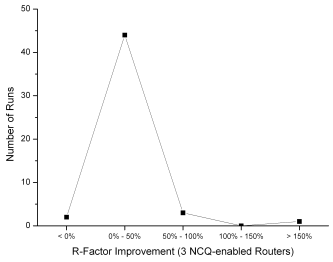



(a) Goodput of Congestive Applications

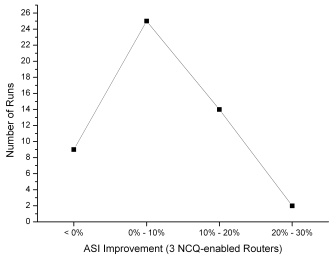



(b) Goodput Improvement of VoIP Applications (4 NCQ-enabled Routers)

Fig. 19. Performance Results in Terms of Goodput




(a) R-Factor Improvement (3 NCQ-enabled Routers)




(b) Improvement in Terms of Application Satisfaction Index (3 NCQ-enabled Routers)

Fig. 20. Voice Quality & Fairness

priority will guarantee fairness among non-congestive flows in a similar fashion to RED's probabilistic dropping [15]. An initial approach is proposed in [25]. Alternatively, the *ncqthresh* may be dynamically adjusted, based on the projected outcome.

In another front of research, ACKs[5] and control packets may benefit from the priority treatment. Control packets prioritization are expected to boost the performance of short-lived flows (mice) increasing fairness compared to long-lived flows (elephants). ACKs are 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 under further investigation.

Initially, we assume that non-congestive traffic is formed by small packets at low data rates. Although this approach has significant advantages (requires almost-zero memory state, requires minor implementation effort etc) and is suitable to demonstrate the potential of our approach, it does not cover all the cases of non-congestive traffic (e.g., the non-congestive traffic that is formed by a few long packets). A more sophisticated non-congestive traffic identification algorithm is the subject of a future work

## VII. Conclusions

We have shown that NCQ is a simple but powerful tool for service differentiation, particularly beneficial for applications that utilize small rates and short packets, such as typical sensor applications and VoIP. We discovered that a limited prioritization has a dual impact: it benefits non-congestive flows significantly and reduces contention among the congestive flows, resulting in satisfying more users; a leading design issue in today's Internet, which creates hopes for deployment success.

[5]In this paper, NCQ does not favor ACKs.

## References

[1] F. M. Anjum and L. Tassiulas. Fair Bandwidth Sharing among Adaptive and Non-Adaptive Flows in the Internet. In *IEEE InfoCom 99*, March 1999.

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

[3] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Services RFC 2475, 1998.

[4] R. Braden, D. Clark, S. Shenker, et al. Integrated Services in the Internet Architecture: an Overview, 1994.

[5] P. Brady. A Statistical Analysis of On-Off Patterns in 16 Conversations. *The Bell System Technical Journal*, 47:73–91, Jan 1968.

[6] D. Chiu and R. Jain. Analysis of the Increase/Decrease Algorithms for Congestion Avoidance in Computer Networks. *Journal of Computer Networks and ISDN*, 17(1):1–14, June 1989.

[7] 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. IEEE.

[8] R. Cole and J. Rosenbluth. Voice over IP Performance Monitoring. *ACM SIGCOMM Computer Communication Review*, 31(2):9–24, April 2001.

[9] T. D. Dang, B. Sonkoly, and S. Molnar. Fractal Analysis and Modeling of VoIP Traffic. In *Proceedings of 11th International Telecommunications Network Strategy and Planning Symposium*, Austria, June 2004. NETWORKS 2004.

[10] G. de Veciana, T. Konstantopoulos, and T.-J. Lee. Stability and Performance Analysis of Networks Supporting Elastic Services. *IEEE/ACM Transactions on Networking*, 9(1):2–14, 2001.

[11] I. Downard. Simulating Sensor Networks in NS-2. Technical Report NRL/FR/5522–04-10073, Naval Research Laboratory, Washington, D.C., May 2004.

[12] J. Elson and D. Estrin. Random, Ephemeral Transaction Identifiers in dynamic sensor networks. In *21st International Conference on Distributed Computing Systems, 2001*, pages 459–468, Apr 2001.

[13] W. Feng, D. Kandlur, D. Saha, and K. G. Shin. BLUE: A New Class of Active Queue Management Algorithms. Technical Report CSE-TR-387-99, University of Michigan, April 1999.

[14] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the Internet. *IEEE/ACM Transactions on Networking*, 7(4):458–472, 1999.

[15] S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, August 1993.

[16] S. Fred, T. Bonald, A. Proutiere, G. Régnié, and J. Roberts. Statistical Bandwidth Sharing: A Study of Congestion at Flow Level. *Proceedings of the 2001 SIGCOMM conference*, 31(4):111–122, 2001.

[17] X. He, C. Papadopoulos, and P. Radoslavov. A Framework for Incremental Deployment Strategies for Router-assisted Services. *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE*, 2:1488– 1498, 2003.

[18] P. Hurley, J. Boudec, P. Thiran, and M. Kara. ABE: Providing a low-delay service within best-effort. *IEEE Network*, 15(3):60–69, 2001.

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

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

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

[22] A. Kherani and A. Kumar. Performance Analysis of TCP with Non-persistent Sessions. In *Workshop on Modeling of Flow and Congestion Control*, Paris, France, September 2000. Ecole Normale Superieure.

[23] D. Lin and R. Morris. Dynamics of Random Early Detection. In *SIGCOMM '97*, pages 127–137, Cannes, France, september 1997.

[24] L.Mamatas and V.Tsaoussidis. Transport Protocol Behavior and Energy-Saving Potential. In *Sixth International Workshop on Wireless Local Networks (WLN 2006), Tampa, Florida*, November 2006.

[25] L.Mamatas and V.Tsaoussidis. Less Impact Better Service (LIBS): A Service Paradigm for Internet Telephony. Technical Report TR-DUTH-EE-2007-16, Democritus University of Thrace, Vas. Sofias 12, Xanthi, Greece, November 2007.

[26] R. Mahajan, S. Floyd, and D. Wetherall. Controlling High-Bandwidth Flows at the Congested Router. In *9th International Conference on Network Protocols (ICNP)*, Nov 2001.

[27] L. Mamatas and V. Tsaoussidis. A new approach to Service Differentiation: Non-Congestive Queuing. In *ICST First International Workshop on Convergence of Heterogeneous Wireless Networks (CONWIN2005)*, Budapest, Hungary, July 2006. ICST.

[28] L. Mamatas and V. Tsaoussidis. Differentiating Services for Sensor Internetworking. In *IFIP Fifth Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net 2007)*, Corfu, Greece, June 2007. IFIP.

[29] M. Marsan and M. Gerla. Fairness in Local Computing Networks. In *IEEE ICC '82*, Philadelphia, June 1982. IEEE.

[30] L. Massoulié and J. Roberts. Bandwidth Sharing and Admission Control for Elastic Traffic. *Telecommunication Systems*, 15(1):185–201, 2000.

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

[32] T. J. Ott, T. V. Lakshman, and L. H. Wong. SRED: Stabilized RED. In *Proceedings of INFOCOM*, volume 3, pages 1346–1355, 1999.

[33] R. Pan, L. Breslau, B. Prabhakar, and S. Shenker. Approximate Fairness through Differential Dropping. *ACM SIGCOMM Computer Communication Review*, 33(2):23–39, April 2003.

[34] R. Pan, B. Prabhakar, and K. Psounis. CHOKe - A Stateless Queue Management Scheme for Approximating Fair Bandwidth Allocation. In *IEEE INFOCOM*, March 2000.

[35] C. Perkins, E. Belding-Royer, and S. Das. RFC3561: Ad hoc On-Demand Distance Vector (AODV) Routing. *Internet RFCs*, 2003.

[36] A. Rangarajan. Early Regulation of Unresponsive Flows. Technical Report TRCS99-26, University of California Santa Barbara, July 1999.

[37] H. Sabbineni and K. Chakrabarty. Location-aided flooding: An energy-efficient data dissemination protocol for wireless sensor networks. *IEEE Transactions on Computers*, 54(1):36–46, 2005. Member-Harshavardhan Sabbineni and Senior Member-Krishnendu Chakrabarty.

[38] I. Stoica, S. Shenker, and H. Zhang. Core-Stateless Fair Queueing: Achieving Approximately Fair Bandwidth Allocations in High Speed Networks. In *SIGCOMM 98*, pages 118–130, 1998.

[39] D. Vardalis and V. Tsaoussidis. On the Efficiency and Fairness of Congestion Control Mechanisms in Wired and Wireless Networks. *The Journal of Supercomputing, Kluwer Academic Publishers*, 2002.

[40] Y. Zhang, W. Liu, and Y. Fang. Secure localization in wireless sensor networks. In *IEEE Military Communications Conference (MILCOM)*, pages 3169–3175 Vol. 5, Oct. 2005.