Differentiating Services for Sensor Internetworking

Lefteris Mamatas and Vassilis Tsaoussidis

Abstract—**We propose NCQ (Non-Congestive Queuing) as a scheduling discipline that allows for efficient interoperation of sensor networks with the Internet. NCQ promotes conditionally small data packets, which require comparatively minor service times, as long as their total service times cause insignificant delays to other packets in the queue. Therefore, NCQ introduces a new service philosophy, which prioritizes packets, and in turn corresponding flows, according to their impact on total delay. In this context, we also introduce a corresponding index to measure fairness as a function of the deviation of expected and received delay.**

Index Terms— Sensor Networks, Ad hoc, Energy Efficiency

I. INTRODUCTION

Typical scheduling paradigms of packet networks do not match well the requirements of sensor 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. The detailed study of such cost is the focus of the present paper.

Our primary assumption is that sensor applications generate packets in form of non-congestive traffic. Typically, they transmit periodically small packets. However, other applications may fall into this category as well, if we judge solely on the basis of packet length (such as VoIP applications) [7]. Sensor data may have strict requirements in delay and the service received by the network cannot be judged on the basis of Throughput. This observation calls for a new metric for application fairness as well, which relies mainly on the delay rather than Throughput. We introduce a delay-oriented index, which we call *Application Satisfaction Index (ASI)*. *ASI* reflects how fairly receive service, delay-wise, applications with diverse demands in Throughput and delay.

Typical sensor applications require certainly differentiated, yet somewhat distinctive service. Classic differentiation schemes require identification of applications/flows or alternatively, a verifiable packet marking technology. In order to avoid the cost of packet preparation for differentiated services, we take advantage of two distinctive properties of typical sensor data:

1) *the small size of sensor packets.*
2) *the small data volume of sensor-generated data flows.*

We apply the first property to identify sensor data; we apply the second to distinguish sensor flows from real-time application flows that utilize small packets as well. The key idea of Non-Congestive Queuing (NCQ) [6, 7] 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. We call this service discipline "Less Impact Better Service" (LIBS).

The proposed service architecture impacts other performance measures as well, such as energy expenditure, which is very significant indeed for energy-limited sensors. The gains in energy are achieved through the reduction of communication and hence application time and their importance vary depending on the sensor device itself, the communication pattern, the network contention etc.

Although our approach sounds straightforward, the system properties and design details reveal interesting dynamics. The simplicity of NCQ's core 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. We show that NCQ improves energy efficiency and real-time communication capability of sensor devices and applications, respectively, without damaging the dynamics of multiple-flow equilibrium and without causing any significant *Goodput* losses to the congestive flows.

The structure of the paper is the following: In section II we discuss the related work. In section III we provide the pseudo-code of NCQ and present the basic assumptions and fundamental concepts. In section IV we approach NCQ analytically and provide numerical results. In section V we present and justify our evaluation plan and metrics. Next, we present the results, analysis and justification. In section VI we summarize our conclusions.

II. Related Work

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.

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 [3] Floyd and Fall introduced mechanisms based on the identification of high-bandwidth flows from the drop-history of RED. The RED-PD algorithm (RED with Preferential Dropping) [5] uses per-flow preferential dropping mechanisms. Two other approaches that use per-flow preferential dropping with FIFO scheduling are Core-Stateless Fair Queuing (CSFQ) [11] and Flow Random Early Detection (FRED) [4]. 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 packets 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.

An alternative way to provide service differentiation would be based on packet marking. However, marking alone has several comparative disadvantages:

- The markings are predetermined and cannot correspond to various possible packet sizes. They inherently produce a 'classification' 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 on 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.

Therefore, 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. Non-Congestive Queuing

We assume different classes of packets according to their size. NCQ is incorporated into the routers and differentiates service according the impact of the traffic class on the delay. For example, a class with small packets and low 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 to a level, which delays significantly long-packet transmission. We complement the differentiating scheme with a service threshold: Non-congestive traffic cannot exceed a predetermined *ncqthresh* amount of prioritized service. We investigate the impact of *ncqthresh* in the result section.

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 [7], we calculate numerically the actual impact of such transformation.

Although the perspective of NCQ is more general, initially, we only deal with two classes of packets: very small packets, experimentally determined to 120 bytes and long packets that typical Internet applications use for data transfers. NCQ uses priority queuing to implement priority service. That is, with in 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

120 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
  If (packetLength<120)
      and
     (favored packets /
      received packets < ncqthresh)
  Then
      packet gets high priority
      count favored packets
  Else
      packet gets normal priority
  End
End
```

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. Class 1 is formed by the non-congestive traffic while Class 2 by the congestive. We assume that all packets arriving at the bottleneck queue follow a Poisson distribution. Class 1 has priority over class 2. We use a non-preemptive head-of-line priority system per class. Class 1 has smaller packets (so, average service-time too) and lower packet-arrival rate ($\lambda_1 < \lambda_2$). We summarize our notation in Table I.

We use the following definitions:

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

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

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

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

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

1. Class 1 has full priority over Class 2.
2. The two Classes have the same priority (scheduling without priority).
3. Only the *ncqthresh* amount of the non-congestive traffic is prioritized over the congestive. Practically, Class 1 is formed from the *ncqthresh* amount of the non-congestive traffic and Class 2 from the remaining *1-ncqthresh* amount of the non-congestive plus the congestive traffic.

TABLE I
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
$u_1 = \lambda_1 T_{S1}$	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

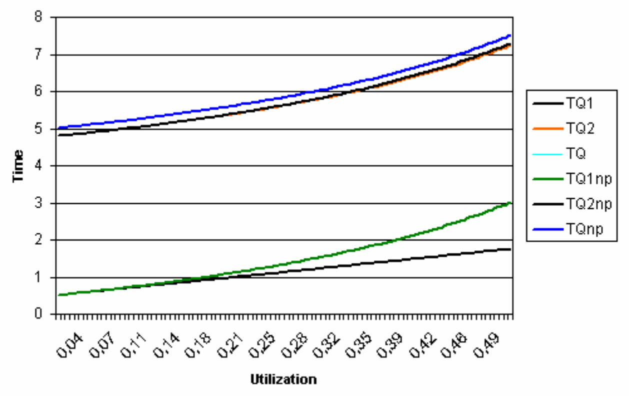

Fig. 1. Numerical Results (5% non-congestive)

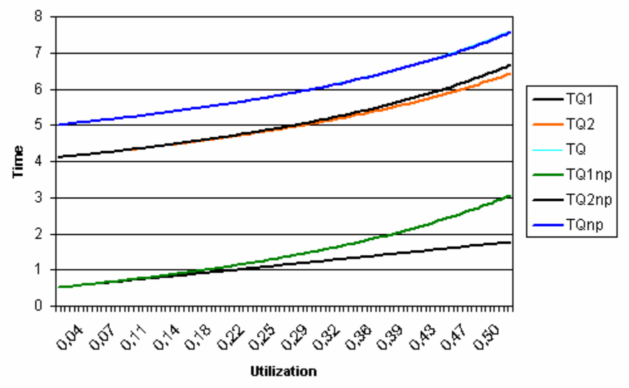

Fig. 2. Numerical Results (25% non-congestive)

Case 1: Priority Scheduling

We calculate the average waiting time for each of the two classes as:

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

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

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

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

We calculate the queuing delay for each Class and we use the weighted average in order to estimate the total average time-in-system:

$$T_{Q1} = T_{W1} + T_{S1} \quad (4)$$

$$T_{Q2} = T_{W2} + T_{S2} \quad (5)$$

$$T_Q = \frac{\lambda_1}{\lambda} T_{Q1} + \frac{\lambda_2}{\lambda} T_{Q2} \quad (6)$$

Case 2: Non-Priority Scheduling

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

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

The service time:

$$T_{Snp} = \frac{\lambda_1}{\lambda} T_{S1} + \frac{\lambda_2}{\lambda} T_{S2} \quad (8)$$

The average waiting time:

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

The average time-in-system:

$$T_{Q1np} = T_{W1np} + T_{S1} \quad (10)$$

$$T_{Q2np} = T_{W2np} + T_{S2} \quad (11)$$

$$T_{Qnp} = \frac{\lambda_1}{\lambda} T_{Q1np} + \frac{\lambda_2}{\lambda} T_{Q2np} \quad (12)$$

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, 2).

In Figure 1, the 5% of arriving packets form the non-congestive traffic (class 1) and the 95% the congestive (class 2). The service times 0.5ms and 5ms correspond to class 1 and class 2, respectively. While the average time of congestive traffic and the total average time are not affected by the use of priority queuing, the non-congestive traffic is significantly favored. When we increase the rate of non-congestive packets to 25% (see Figure 2), there is a significant impact on the congestive traffic in high utilizations (exceeding 0.3).

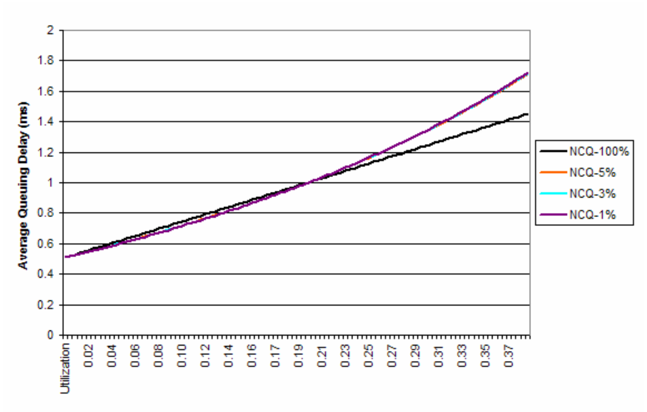

Fig. 3. Average Queuing Delay of Non-Congestive Traffic

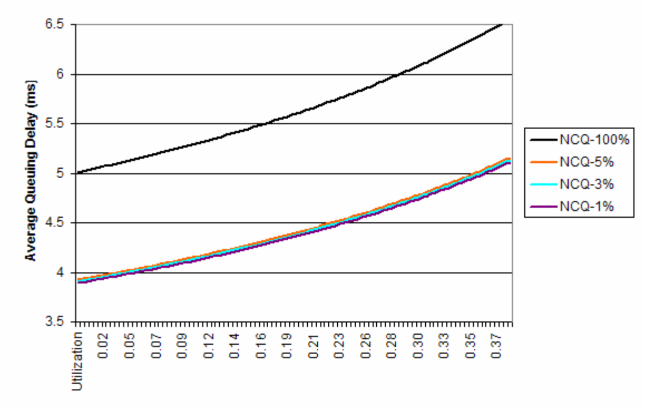

Fig. 4. Average Queuing Delay of Congestive Traffic

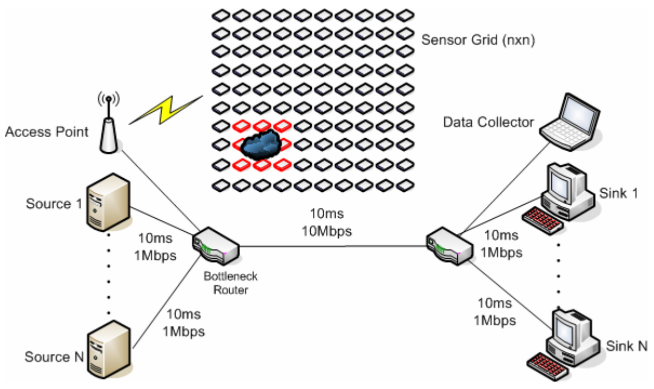

Fig. 5. Simple Topology.

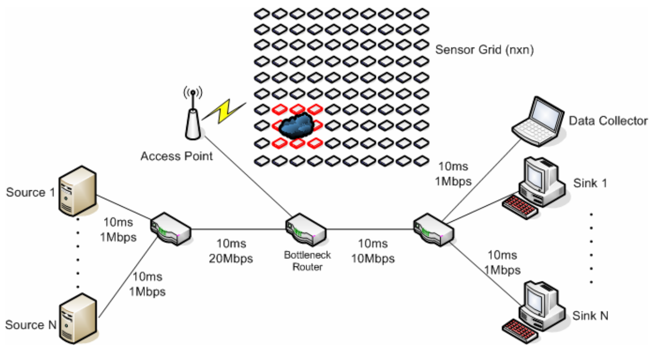

Fig. 6. Complex Topology.

Case 3: Priority Scheduling with ncqthresh

Ncqthresh represents the percentage of non-congestive traffic that can be favored without any statistically important impact on the congestive traffic. The two priority classes (1 and 2) consist of the *ncqthresh* amount of non-congestive traffic and the (1-*ncqthresh*) amount of the non-congestive traffic plus the congestive traffic, respectively. In the following analysis, we assume that only a portion (bounded by *ncqthresh*) of the non-congestive traffic is favored.

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

$$\lambda_1' = ncqthresh \cdot \lambda_1$$

$$\lambda_2' = (1 - ncqthresh) \cdot \lambda_1 + \lambda_2$$

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

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

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

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

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

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

We calculate the waiting times of each application (non-congestive and congestive) using the weighted average of the above waiting times:

$$T_{W1} = \frac{ncqthresh \cdot \lambda_1}{\lambda_1} T_{W1}' + \frac{(1 - ncqthresh) \cdot \lambda_1}{\lambda_1} T_{W2}'$$

$$= ncqthresh \cdot T_{W1}' + (1 - ncqthresh) \cdot T_{W2}'$$

$$T_{W2} = T_{W2}'$$

$$T_{Q1} = T_{W1} + T_{S1}$$

$$T_{Q2} = T_{W2} + T_{S2}$$

As we can see from Figures 3, and 4, the average queuing delay for the congestive traffic is reduced. *Ncqthresh* bounds the prioritization of the non-congestive traffic to a limit that does not allow them to reduce the bandwidth exploitation of the congestive applications. In utilizations below 0.23, the average queuing delay of the non-congestive traffic remains almost the same. In [7], we study the impact of this prioritization on applications using other packet sizes. In [6, 7] we discuss the choice of the *ncqthresh* value and complement our discussion with corresponding experiments.

V. Experimental Results

A. *Evaluation Plan*

In our experimental analysis, we used ns-2 [13] based simulations. We integrated NRL's Sensor Network Extension [2] into ns-2 in order to have more realistic scenarios of internetworked sensor applications. Our simulation scenarios consist of an ad hoc sensor network (a grid of 25 wireless sensors) and two different topologies for the main infrastructure: a simple dumbbell (Figure 5) and a more complex topology (Figure 6). In the latter topology, the wireless access point is attached on a different bottleneck router. The sensor-application transfers periodically data to a wired node (the data collector). Actually, 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* [10] routing protocol. We evaluated the performance of the proposed scheme using the *Goodput* metric on both congestive (FTP) and sensor-related non-congestive traffic:

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

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 data delivery.

We evaluated the energy-efficiency of NCQ using the Energy Potential (EP) [8] index:

$$EP = 1 - \left(a\frac{Throughput - Goodput}{Throughput_{\max}} + b\frac{Througput_{\max} - Throughput}{Throughput_{\max}}\right)$$

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:

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

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} flow; and $Delay_{max}$ the maximum queuing delay of the system. *ASI* ranges from 0 to 1.

Unlike other fairness indices (such as [1], [9], [12]), *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.

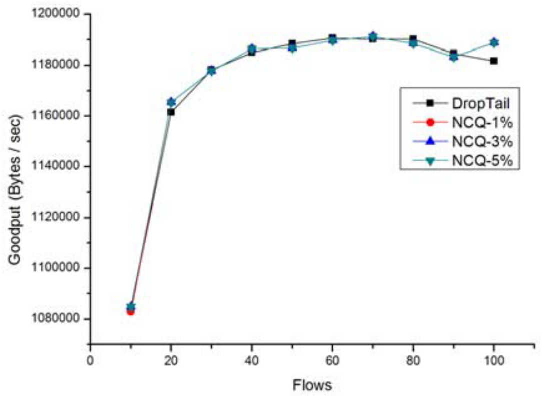

Fig. 7. Goodput of FTP Flows

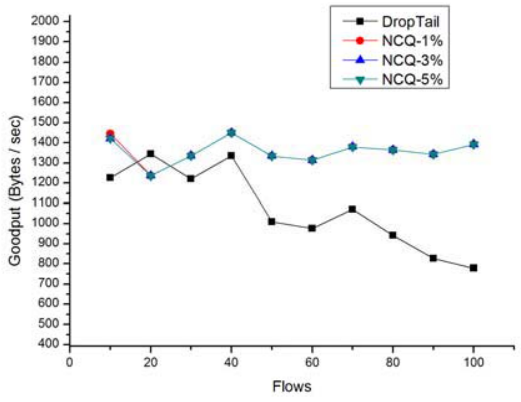

Fig. 8. Goodput of Sensor Applications

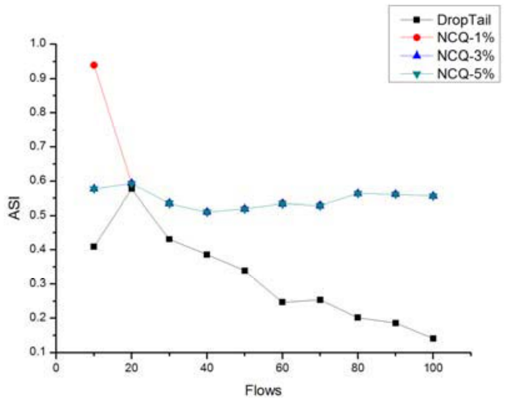

Fig. 9. ASI of Sensor Applications

B. Experimental Results

In the first scenario, we used the simple dumbbell topology of Figure 5. In Figure 7 we depict the *Goodput* performance of congestive flows and in Figures 8, 9, 10 we demonstrate the *Goodput*, *Application Satisfaction Index* and the *Energy Potential* of Sensor Applications. We can argue that the congestive FTP flows are not suffering from any important performance loss (see Figure 7), 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 11). Additionally, the system appears fair according to *ASI* (Figures 9, 10). We note that, due to *Goodput* increase of sensor applications, fairness performance may also be captured by the traditional index of fairness.

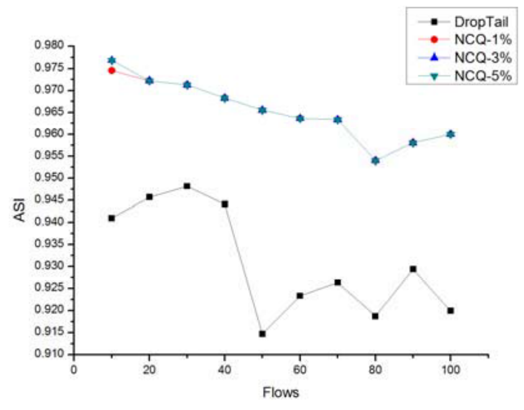

Fig. 10. System's ASI

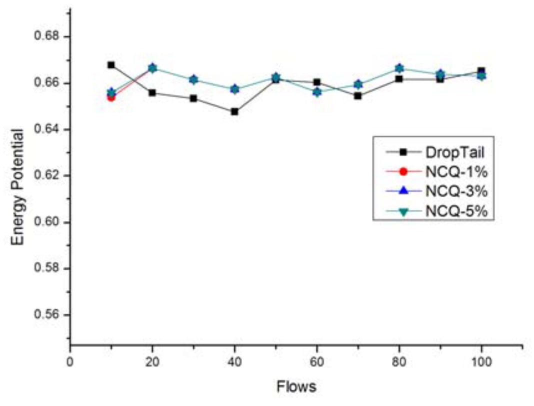

Fig. 11. Energy Potential (Non-Congestive Flows)

As we demonstrate in Figures 7, 8, 9, 10, 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.

In the second scenario, we used a more complex topology (see Figure 6) where an additional router allows for peripheral traffic and shifts the system bottleneck from the original entrance node of FTP applications.

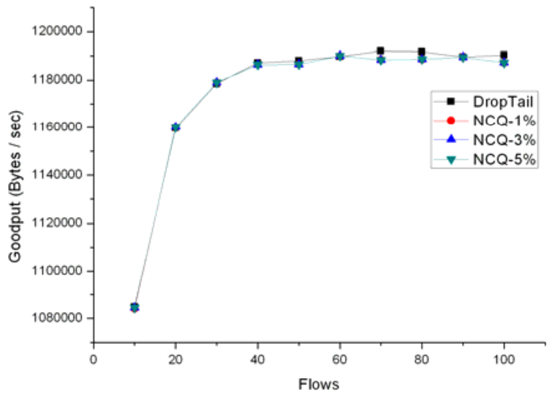

Fig. 12. Goodput of FTP Flows

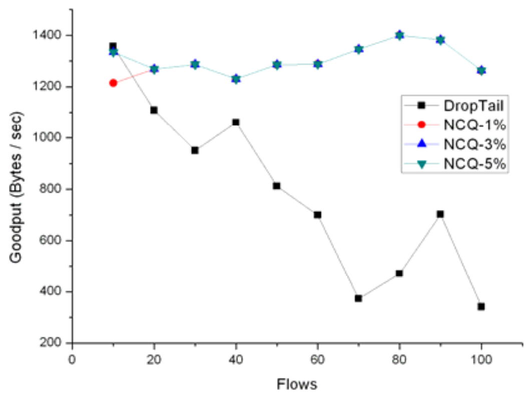

Fig. 13. Goodput of Sensor Application

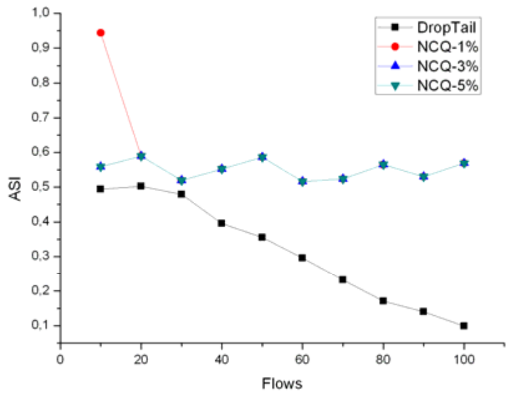

Fig. 14. ASI of Sensor Application

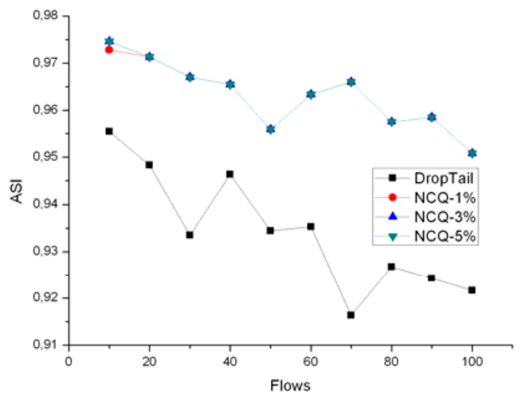

Fig. 15. System's ASI

As we can see in Figure 12, there is no statistically important impact on the *Goodput* of the congestive FTP flows. However, there is a significant improvement of sensor applications' *Goodput* (up to 260% - Figure 13). Furthermore, NCQ improved the *Application Satisfaction Index* for both sensor network and system (Figures 14, 15).

We have evaluated NCQ with several scenarios, changing the number of sensors, the type of congestive application and the simulated network topologies. Due to space limitations we do not report these results here; however, these results support our arguments further. For example, some experiments with real-time multimedia traffic can be found in [6, 7].

VI. Conclusions

We proposed a new scheme for service differentiation, based on a system-oriented approach. Our scheme, however, does not conflict with existing application-oriented service differentiation technologies, such as marking. We realized our service architecture with a scheduling algorithm (NCQ) and a corresponding fairness index (*ASI*). We demonstrated that NCQ can be adjusted to promote service for sensor applications without damaging traditional internet applications. Although our scheme is service-oriented, it has a direct impact on application performance. In simple terms, NCQ increases the amount of satisfied users within a system.

We are working on an extension of the algorithm to assign priority service to small packets probabilistically. This per-packet probability may decrease as the rate of non-congestive packet exceeds the *ncqthresh*. Probabilistic priority could guarantee fairness among non-congestive flows in a similar fashion to RED's probabilistic dropping. Alternatively, the *ncqthresh* may be dynamically adjusted, based on the projected outcome.

In another front of research, ACKs and control packets may benefit from 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 the 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.

Acknowledgment

This work was supported by the project PENED '03, funded by European Commission and the G.S.R.T. of Greek Ministry of Development.

References

[1] 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), June 1989.

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

[3] 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.

[4] D. Lin, and R. Morris, "Dynamics of Random Early Detection," *In SIGCOMM '97*, pages 127-137, Cannes, France, September 1997.

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

[6] L. Mamatas, and V. Tsaoussidis, "A New Approach to Service Differentiation: Non-Congestive Queuing", *In Proceedings of CONWIN 2005*, Budapest, Hungary.

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

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

[9] M.A. Marsan and M. Gerla. "Fairness in Local Computing Networks," *in Proceedings IEEE ICC'82*, June 1982.

[10] C.E. Perkins, E.M. Belding-Royer, and S. Das, "Ad Hoc On Demand Distance Vector (AODV) Routing." *IETF RFC 3561.*

[11] A. Rangarajan, "Early Regulation of Unresponsive Flows," *Technical Report: TRCS99-26*, July 1999.

[12] D. Vardalis, "Efficiency/Fairness Tradeoffs in Networks with Wireless Components and Transient Congestion",

The Journal of Supercomputing, Kluwer Academic Publishers, Volume 23, Issue 3, November 2002.
[13] The Network Simulator – NS-2, http://www.isi.edu/nsnam/ns.