Incentive-Compatible Differentiated Scheduling*

Martin Karsten
School of Computer Science
University of Waterloo
Waterloo, ON, Canada
mkarsten@uwaterloo.ca

Yunfeng Lin†
Department of Electrical and Computer Engineering
University of Toronto
Toronto, ON, Canada
ylin@eecg.toronto.edu

Kate Larson
School of Computer Science
University of Waterloo
Waterloo, ON, Canada
klarson@uwaterloo.ca

ABSTRACT

In this paper, we take a novel approach at service differentiation in packet-switched networks. Existing mechanisms for delay control and differentiation typically require some form of explicit resource allocation and consequently, admission control. We propose *incentive-compatible* differentiated scheduling as a basic building block for Internet service provisioning. This scheduling scheme enables policy-free approximate service differentiation without admission control, for example at Internet peering exchanges. In addition, the same building block can also facilitate precise service differentiation in combination with several forms of admission control. We present the basic design of a scheduling algorithm and discuss its implementation details and design alternatives. We prove that the algorithm has strong game-theoretic properties and present some initial simulations illustrating the effect of this scheduler on Internet traffic.

1. INTRODUCTION

Service differentiation in IP networks is a long-standing open issue. There are numerous technical proposals focusing on various aspects of the overall problem. Typical solutions comprise differentiated scheduling at the flow or traffic class level, admission control, and eventually accounting and billing. In contrast, the current Internet model is dominated by the prevalent best-effort service and the notion of "all flows are equal"-fairness implied by TCP's congestion control algorithm. Service differentiation is a generalization of this concept, but comes at the price of increased complexity.

Existing proposals for service differentiation come in two variants: with or without admission control. Absolute service guarantees inevitably require some form of admission control to reliably avoid resource overload. In the absence of admission control, systems usually resort to only offering relative service differentiation. Some proposals employ proactive resource allocation while others use network feedback for edge-based control. Edge-based control can comprise rate control and/or admission control. Certain ideas rely on cooperation among network users. In contrast, any scheme that relies on economic back-pressure also requires an accounting system in the management plane, which typically contributes significantly to the overall complexity and cost. In fact, the management complexity may pose a more serious obstacle to the deployment of service differentiation than any computational complexity of low-level functions that are targeted by most research proposals.

In this context, *first-in first-out* (FIFO) scheduling has nice characteristics, besides just being extremely simple: FIFO scheduling does not interfere with any rate allocation policy from an end- or edge-based control regime, regardless of whether admission control, feedback-based rate control, or a combination of both is being used. The control regime only needs to support a single class of service. Also, Internet transport and application protocols are typically well-equipped to cope with the varying loss and rate allocation resulting from single-class FIFO scheduling, at least to a certain extent. We denote these characteristics as the *FIFO principle*. However, FIFO scheduling does not support delay differentiation at all. Delay differentiation requires traffic classification and multiple services classes. Any kind of allocation-based or priority scheduler then requires control intervention to avoid overloading the "good" or "fast" classes. Existing schedulers that support delay differentiation typically violate the FIFO principle by requiring multi-class traffic regulation.

The goal of *incentive-compatible differentiated scheduling* (ICDS) is to adhere to the FIFO principle but to support delay differentiation. ICDS uses dynamic proportional rate allocation to avoid creating any preferable service class. Instead, each traffic source has a natural incentive to pick the service class with the most suitable delay target, depending on its burst characteristics. Thereby, ICDS is a compact and versatile building block for service differentiation.

The rest of this paper is organized as follows. In Section 2, we discuss traditional approaches to *quality of service* (QoS) and present previous work that is related to our approach. In Section 3, a system model for ICDS is presented and analyzed for its game-theoretic properties. There are several design alternatives and implementation aspects, which are briefly illustrated and discussed in Section 4. Then, in Section 5 we present simulation results to explore the actual effect of ICDS on traffic. This is followed by a discussion in Section 6 and a summary and outlook in Section 7.

2. RELATED WORK

There is plenty of related work in the area of network QoS. In this section, we only discuss archetypical examples of the various proposals. At one end of the spectrum, the IETF *Integrated Services Architecture* (IntServ) recreates the traditional telecommunications QoS model relatively closely. It comprises admission control and packet scheduling. IntServ implicitly conveys the notion of having these functions at

*This work is supported by the Natural Sciences and Engineering Research Council of Canada.

†Work done while at the University of Waterloo.

each node, although this is not strictly necessary.

The IETF *Differentiated Services Architecture* (DiffServ) has been conceived to reduce the complexity of service differentiation. In particular, it is focused on class-based traffic control to reduce the data-plane overhead compared to IntServ. Service signalling is treated as an independent problem domain and recently receives renewed attention in an IETF working group [18]. DiffServ is a rather loosely defined packet processing framework, which can be populated by many schedulers and control regimes and then results in equally many different QoS proposals.

A very simple form of service differentiation is possible by employing strict priority scheduling. In fact, this is one specific example for a relative or proportional class-based scheduler. A number of others are described in [7], for example. This kind of differentiated scheduling creates preferable traffic classes that deliver better service in all relevant parameters than other classes in the system. Consequently, it requires traffic regulation across service classes to avoid all traffic being directed to the high-quality class. Thereby, the multi-class system formed by any such a scheduler does not conform to the FIFO principle.

A different form of QoS system is created by employing admission control without any specific scheduling regime. This is often combined with traffic and load measurement for *measurement-based admission control* [3]. In a related scenario, *active queue management* schemes, such as RED [9] or one of its many successors, are used to produce load feedback that is transported to edge or end systems by means of packet marking. An example for such a system proposal is given in [4], which also includes many further references. All proposals implicitly assume a single scheduling class and limit service differentiation to the transmission rate.

Yet another proposal for efficient network QoS is given by core-stateless techniques, such as [20] and [14]. These approaches exclusively focus on reducing the data-plane overhead of packet processing, similar to DiffServ. However, there are two important shortcomings. First, core-stateless scheduling reduces the amount of state information, but still requires per-flow signalling with all nodes in a domain. However, a large fraction of the processing overhead of network signalling is spent per message transmission, regardless of the amount of state being stored [13]. Second, the nature of the distributed scheduling state results in a somewhat fragile system where a single misconfigured or erroneous node may disrupt services in a whole network domain.

In fact, operational complexity and deployment considerations may completely outweigh low-level implementation efficiency. For example, the QBone activity as part of Internet2 has abandoned the concept of premium services, citing "non-architectural, and largely non-technical obstacles" [21]. For these reasons, *non-elevated services* may be a much better fit for the Internet. Such services can be described as "deploy incrementally, with no need for policing, accounting, or significant change to operational practices" [19].

The *Best Effort Differentiated Services* (BEDS) [8] maintains fixed delay and drop ratios between two services classes. One of the service classes is intended for delay-sensitive traffic, but might incur a higher packet loss, while the other one provides lower packet loss. A similar proposal, although different in certain details is presented as *Equivalent Differentiated Services* (EDS) in [10]. In both cases, it is unclear how to configure the respective service ratios for any meaningful absolute service guarantee. A different proposal for non-elevated service differentiation is given by *Alternative Best-Effort* (ABE) [11]. A delay-oriented service class provides an absolute bound on packet latency, but not adversely affect the throughput-oriented other service class. ABE's one slight drawback is its fairly complex drop operation. In a sense, ICDS is a generalization of ABE by providing an arbitrary number of delay classes, rather than just two.

Another important consideration is that of relevance. The original reason for significant buffer capacity (resulting in noticeable delay) in routers is to provide smooth service to end systems producing bursty traffic, e.g. stemming from window-based flow control. Recently, it has been speculated that the necessary amount of buffer may be much less and may become irrelevant in terms of delay [15]. While this may be true for backbone routers [1], it is not clear whether this assumption always holds [16] and in particular, whether it holds for all parts of the network [6]. Regardless of the outcome, incentive-compatible delay and buffer differentiation is a future-proof technique that increases the flexibility to build meaningful services in packet-switched networks.

3. SCHEDULER MODEL AND ANALYSIS

3.1 Scheduler Model

When considering service differentiation in packet-switching networks, we make the following observations. Different applications and usage scenarios vary greatly in their requirements for transmission rate. Many applications can operate in a spectrum of different rates. In economic terms, such applications have a concave rate utility function. Therefore, a freely configurable transmission rate (as opposed to a few fixed rate classes) is highly desirable for a multi-service communication network. In contrast, it seems less important to support a wide variety of delay settings. In economic terms, delay utility curves are usually S-shaped, since delay requirements are based on human perception or other external factors. There is not much variety in delay requirements and consequently, a fixed number of delay classes is sufficient. On the other hand, dynamic rate allocation requires suitable buffer adjustments to achieve high utilization and meet delay targets. Inspired by these observations, ICDS is defined as a system of n service classes, with each class i having a delay target of d_i, and consists of two components.

Rate Allocation - ICDS dynamically allocates service rates in proportion to the instantaneous arrival rates. Given a link capacity C and a set of arrival rates $a_j(t)$ at time t, the service rate $r_t(i)$ for class i is computed as

$$r_i(t) = C\frac{a_i(t)}{\sum_{j=1}^{n} a_j(t)} \quad (1)$$

This (idealized) rate allocation policy is critical to maintain the FIFO principle.

Packet Discard - The packet discard component enforces the delay target by discarding those packets that cannot be forwarded in time. This assessment is non-trivial in the presence of dynamic service rate allocation.

Both components are necessary for incentive-compatibility. Various implementation alternatives, discussed in Section 4, offer different trade-offs between precision and complexity.

In summary, ICDS provides a simple and intuitive service differentiation model by offering a fixed set of delay classes

and adhering to the FIFO principle with respect to rate allocations. Besides a standalone scenario, ICDS has useful properties in combination with admission control. This is further discussed in Section 6. Therefore, ICDS is a highly flexible and reusable building block for different IP routers.

3.2 Game-theoretic Properties

To understand the game-theoretic properties of ICDS, it is necessary to map the system description of ICDS into a game-theoretic model. Each traffic source is considered as one of m *players*. Each player j is defined by its maximum delay target $d_j^* \in D = \{d_1, \ldots, d_n\}$ where D is the set of delay targets provided by n service classes of the ICDS *mechanism*. The delay target d_j^* is called the *type* of player j and it is assumed to be private information (i.e. the mechanism and the other players do not know d_j^*).

A *strategy* for player j, s_j, is a mapping $s_j : D \mapsto D$, where, given its true delay target d_j^*, the player announces some delay target $s_j(d_j^*) = d_j$ by means of choosing the corresponding ICDS service class. We place no further restrictions on the strategy of player j and, in particular, we do not assume that $s_j(d_j^*) = d_j^*$. A strategy profile, $s = (s_1, \ldots, s_m)$, is a vector specifying that each player j is playing strategy s_j. A strategy profile is also written as $s = (s_j, s_{-j})$ where $s_{-j} = (s_1, \ldots, s_{j-1}, s_{j+1}, \ldots, s_m)$.

The outcome of the game depends on the strategies chosen by the players. In particular, given a strategy profile s, we describe the resulting outcome by $o(s) = (d(s), \gamma(s))$ where $d(s) = (d_1(s), \ldots, d_m(s))$ specifies the delay $d_j(s)$ that the traffic of player j experiences, and $\gamma(s) = (\gamma_1(s), \ldots, \gamma_m(s))$ where $\gamma_j(s)$ is the drop rate experienced by player j.

The players have preferences over different possible outcomes, given their type d_j^*, which we capture using utility functions u_j. We assume (as is commonly done) that all the players are self-interested so that their utility depends only on the delay and drop-rate of their own traffic. That is: $u_j(o(s), d_j^*) = u_j((d(s), \gamma(s)), d_j^*) \equiv u_j(d_j(s), \gamma_j(s), d_j^*)$. In addition, we assume that the rate allocation policy in (1) follows the FIFO principle, such that the service rate is unaffected by the choice of service class and can be ignored here. Each player tries to choose a strategy that maximizes its utility, given that all other players are doing the same.

Game theory is concerned with finding equilibria in the space of strategy profiles. A strategy is said to be dominant, if it is a player's best strategy against any set of strategies that other players may choose. Formally, a strategy s_j^* is a dominant strategy if $u_j(o(s_j^*, s_{-j}), d_j^*) \geq u_j(o(s_j, s_{-j}), d_j^*)$ holds for all $s_j \neq s_j^*$ and arbitrary s_{-j}. A *dominant strategy equilibrium* is a strategy profile s^* where every player is using a dominant strategy. While many games do not have dominant strategy equilibria, ICDS has one. In fact, the players' dominant strategies are to reveal their true target delay to the mechanism. That is, ICDS is *strategy-proof*, which is an even stronger notion than *incentive-compatible*. The proof relies on a set of assumptions about the system and the players' utility functions:

1. If $d_j > d_k$ then $\gamma_i(d_j, s_{-i}) \leq \gamma_i(d_k, s_{-i})$
2. If $d_j(s) > d_j^*$ then $u_j(o(s), d_j^*) = 0$
3. For two strategy profiles s and s', if $d_j(s), d_j(s') \leq d_j^*$ and $\gamma_j(s) = \gamma_j(s')$ then $u_j(o(s), d_j^*) = u_j(o(s'), d_j^*)$.
4. For two strategy profiles s and s', if $d_j(s), d_j(s') \leq d_j^*$ and $\gamma_j(s) \leq \gamma_j(s')$ then $u_j(o(s), d_j^*) \geq u_j(o(s'), d_j^*)$.

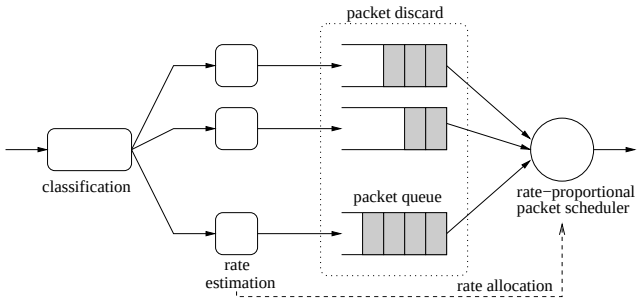

Figure 1: ICDS Implementation

Assumption 1 states that a lower delay target, enforced by a smaller buffer, results in a higher drop rate. Assumption 2 states that if the service delay is greater than the maximum target delay then the player's utility is 0. Assumption 3 states that for a fixed drop-rate, then as long as the delay is less than the maximum target delay, the player is ambivalent between the two outcomes. Assumption 4 states that as long as the delay is less than the max target delay, a player prefers to have a lower drop rate.

THEOREM 1. *ICDS is a strategy-proof mechanism.*

PROOF. (Sketch) If a player declares its target value truthfully, that is follows a strategy $s_j^*(d_j^*) = d_j^*$ when all others play s_{-j}, then its utility will be $u_j(o(d_j^*, s_{-j}), d_j^*) \geq 0$. If, instead, the player had followed strategy $s_j(d_j^*) > d_j^*$ then ICDS would put its traffic into a class with delay greater than its true target delay, resulting in $u_j(o(s_j, s_{-j}), d_j^*) = 0$ (Assumption 2) and so the player is better off declaring its true target value. If the player had followed a strategy $s_j(d_j^*) < d_j^*$ then ICDS would put it into a class with delay which is lower than if it had declared its true delay target, but with a higher drop rate (Assumption 1). From Assumptions 3 and 4, we can conclude that the player would be better off by truthfully declaring its delay target. Since player j and s_{-j} were arbitrarily chosen, we can conclude that any player is always best off truthfully revealing its true delay. That is, there is a dominant strategy equilibrium where each player reveals its target delay truthfully, or ICDS is strategy-proof. □

4. SCHEDULING ALGORITHM

A simple and straightforward implementation of ICDS is sketched in Figure 1. It consists of a rate estimation component, which controls a rate-proportional packet scheduler. Additionally, a packet discard module enforces the maximum delay target by checking the queue status in relation to the service rate and potentially discarding packets. Note that this is a first attempt at implementing ICDS. We expect that other approaches are possible.

4.1 Rate Estimation

The rate estimation algorithm used in our ICDS implementation is a modification of the *Time Sliding Window* (TSW) algorithm [5], which allows for good control over the rate estimation interval and decay of old values, independently of the different arrival rates per class. For an estimation window W, current packet size l_i, and inter-packet

spacing δ_i, the original estimation formula is

$$R_i = \frac{R_{i-1}W + l_i}{W + \delta_i}. \quad (2)$$

This formula requires a division over an arbitrary value, which is prohibitive for a high-speed packet forwarding environment. We have designed a modified version, termed *efficient TSW* (eTSW) as

$$R_i = \frac{R_{i-1}(W - \delta_i) + l_i}{W} \quad (3)$$

and have shown that it has the same properties as TSW [17]. The advantage of eTSW is that the division is fixed. If the window size W is chosen as a power of 2, then the division can be replaced by a faster shift operation. The implementation uses scaled integer representation for all computations.

Rather than estimating the absolute arrival rates, the rate estimation component in ICDS estimates the relative arrival rates, which can be used directly as the weight parameters for the rate-proportional packet scheduler. Also, relative rate estimation does not require any reference to wall-clock time and is thus cheaper to implement. It is achieved in (3) by using the total amount of arrived traffic as δ_i. This only requires a simple byte counter. Depending on the particular type of rate-proportional scheduler, the reciprocal of the relative weight may be needed to avoid a division operation when computing deadlines. This can be achieved easily in (3) by exchanging δ_i and l_i.

To spread the execution overhead of rate estimation, (3) is not computed synchronously for all classes at certain times. Instead, each packet arrival triggers an update of the rate estimate for the corresponding class. This asynchronous computation introduces a slight error in addition to the inherent estimation error, in that the sum of all estimated rates can exceed 1 temporarily. If delay targets are considered strict, this needs to be handled by an additional test before installing a new service rate.

4.2 Packet Scheduling

Since the number of classes in ICDS depends on the number of delay targets being offered and this number is rather limited, it seems feasible to pick any one of the more recent proposed rate-proportional packet schedulers. It is desirable to control burstiness and packet delays as precisely as possible without incurring large scheduling error terms. The current prototype of ICDS employs WF^{2}Q+ [2], such that experimental observations are not obfuscated by a lack of scheduling precision. It may even be worthwhile to explore the slightly more precise L-WF^{2}Q scheduler presented in [22]. In a related proposal, one of this paper's authors presents a constant-time packet scheduler with characteristics similar to WF^{2}Q+ [12].

4.3 Packet Discard

There are several alternatives for implementing the packet discard decision in ICDS. In the simplest scenario, packet deadlines are checked against the delay target whenever a class is selected for service. However, this may lead to a number of packet drops before servicing the next packet, which may not be feasible at line speed. If it is possible to limit the number of such late drops in relation to the number of packets sent, an output buffer and amortization scheme may be used to maintain efficient execution, but we have not yet investigated this.

Without such a late drop scheme, the packet discard component must assess the estimated queueing time for an arriving packet in the presence of variable rate allocation. Considering anything else than the queue and rate allocation of that packet's service class seems infeasible. Two detailed discard strategies are outlined in the next section.

4.4 Rate Allocation and Delay

A crucial aspect for the operation of ICDS is the enforcement of delay targets in the presence of variable rate allocation. We distinguish between two scenarios. If it is acceptable for a small number of packets to miss their delay targets, *loose delay* operation is used. In this mode, the sum of rate allocations may occasionally exceed 1 (cf. Section 4.1) and delay enforcement is always only based on the instantaneous backlog and rate allocation at packet arrival.

In *strict delay* mode, ICDS is enhanced by two mechanisms. First, the total amount of available service rate is kept as a state variable. Only if the new rate estimate does not exceed the sum of the existing rate allocation plus the available rate, it is used for the service class. Otherwise, the allocation remains unchanged. Second, any reduction of the rate allocation is deferred until all currently backlogged packets are serviced whereas an increase is installed immediately. This requires storing the future service rate with each packet in the packet queue and further, to keep track of the estimated queueing delay based on the sum of packet times given the respective per-packet rate allocation. Clearly, these operations result in a sub-optimal utilization of transmission resources. Also, they require to keep track of both the relative rate (to maintain the available rate) and its reciprocal, regardless of the packet scheduler. This introduces another source of inaccuracy. The effect of these measures is investigated in [17] and found to be limited.

In summary, we believe that both loose and strict delay mode are feasible operation regimes. In loose delay mode not many packets miss their deadline, while in strict delay mode the resource utilization is still very good.

5. SIMULATION RESULTS

We have run numerous simulation experiments with our ICDS prototype, both in strict and loose delay mode. We report some simple exemplary results to illustrate the effect of ICDS in loose delay mode. In a dumbbell topology, ICDS is configured with 3 service classes providing delay targets of 10ms, 30ms, and 60ms. The bottleneck link capacity is 155 Mbit/s. 3 traffic sources generate different traffic flows:

CBR - 1 UDP CBR source with 15.5 Mbit/s sending rate
TCP - 100 TCP flows with 30ms one-way propagation delay
Bursty - 32 Pareto sources with shape 1.4, on-period 50ms, and 93 Mbit/s average rate

CBR and TCP sources add a small random per-packet delay to emulate CPU overhead and avoid phase effects. This traffic mix is chosen to represent a worst-case workload with TCP data traffic, UDP traffic from streaming applications, and a large amount of erratic and bursty background traffic. In the first experiment, FIFO scheduling is used with a buffer size equivalent to 60ms. In subsequent experiments, the different traffic types are sent through different combinations of the three service classes. The observation parameters are throughput, loss, and delay. The upper half of Figure 2 depicts the evolution of the throughput with FIFO

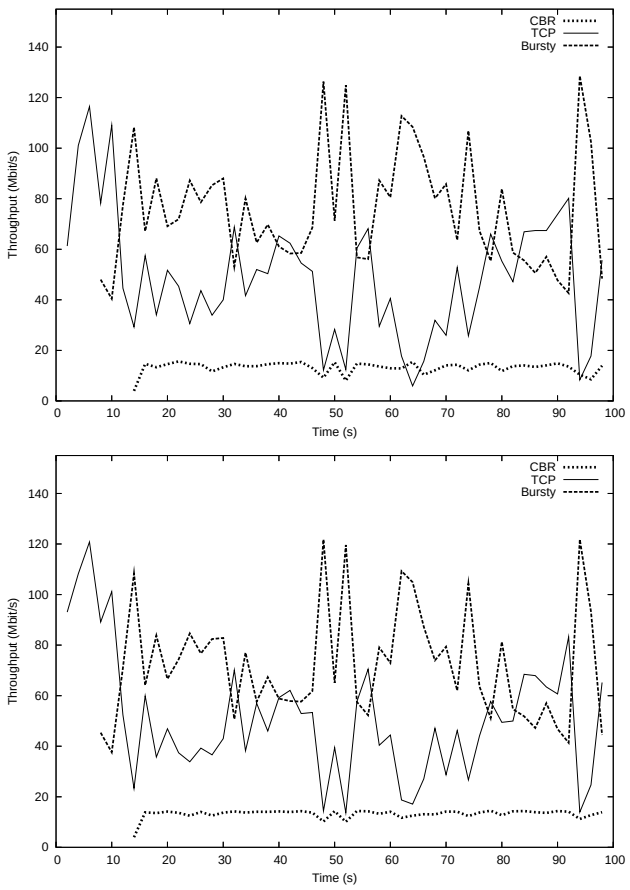

Figure 2: FIFO (top) vs. ICDS (bottom)

Table 1: Average Throughput in Mbit/sec

Scenario (CBR/TCP/Bursty)	CBR	TCP	Bursty
FIFO (60/60/60)	13.6	44.5	75.9
ICDS (10/30/60)	13.5	45.2	72.4
ICDS (10/10/60)	14.0	36.2	74.4
ICDS (10/30/30)	13.9	32.6	71.7
ICDS (10/60/60)	13.6	43.4	75.0
ICDS (10/30/10)	12.0	53.9	57.3

scheduling. This is compared with the ideal ICDS configuration where CBR traffic chooses the 10ms class, TCP chooses 30ms, and Bursty traffic chooses 60ms. It is observed that the throughput pattern does not change fundamentally.

We show the average throughput numbers from all tested configurations in Table 1. These average numbers are excluding the first 20 and last 10 seconds of the experiments. In all scenarios, the Bursty traffic spikes cause occasional short-term delay violations and drive loss rates. The relative order of loss rates from high to low is Bursty, CBR, TCP. In contrast to FIFO scheduling, the differentiated delay targets are generally met in the ICDS scenarios.

The first ICDS case 10/30/60 is the same as shown in Figure 2. No amount of buffer is really sufficient to cope with the self-similar background traffic and ICDS effectively provides some isolation between service classes. Also, TCP's RTT is reduced due to meeting the 30ms delay target. In summary, this is leads to a better rate allocation for TCP, at the expense of Bursty. The next scenario 10/10/60 simulates an attempt of the TCP traffic to get better service by sneaking into the low-delay class. It can be observed that this attempts fails, as TCP's rate average allocation is reduced. In the scenario 10/30/30 the Bursty traffic enters the 30ms class. Because the increased burstiness in the 30ms service class results in a higher loss rate, TCP congestion control results in a lower average service rate for TCP. However, Bursty traffic experiences no significant improvement, so one could describe this as a denial-of-service scenario, which will require further attention in future work (also see Section 6). The fourth scenario 10/60/60 illustrates that choosing the right service class for traffic sources with closed-loop rate control is not trivial. If the TCP traffic chooses the higher delay class at 60ms, it is not as sensitive to interference from the Bursty traffic class as before. The fifth scenario 10/30/10 shows that without the benefit from other traffic's end-to-end congestion control, there is no incentive for the Bursty traffic to enter low-delay classes. In the 10ms class, its service rate suffers significantly while harming the CBR traffic only to a moderate extent. Note that the results are very preliminary and do not account for various configuration details that affect the interaction of ICDS with feedback-controlled traffic sources such as TCP.

6. DISCUSSION

The main goal of ICDS is to provide the right incentives for traffic sources to avoid bursts or to choose the appropriate service class. Ideally, if all sources choose the proper service class, the resulting rate allocation and drop probabilities will be very close to a configuration using a single FIFO queue. However, under overload there is no fate-sharing with respect to buffering and delay, but instead each service class is treated differently. In essence, this amounts to incentive-compatible burst differentiation between different types of application traffic.

This characteristic of ICDS retains the FIFO principle and facilitates service differentiation without the need to dynamically adjust any parameter settings in core routers or peering exchanges. If at all possible, the intelligence of any network control regime should reside at edge systems with little or no control interaction with internal nodes. This is the operational model of the Internet and works well. As discussed, rate differentiation is easily possible using edge control only. ICDS extends that model with delay differentiation and facilitates two main deployment scenarios.

In *isolated* deployment, ICDS is used at specific routers along the data path that are prone to overload, but where sophisticated traffic management is infeasible. A good example for such routers are Internet peering points between network domains. In the absence of a globally coherent QoS system, ICDS will facilitate choice and delay differentiation for different applications without adverse side effects.

The alternative *domain* deployment combines ICDS with edge-based admission and rate control, for example as proposed in [4]. This creates a powerful yet simple domain QoS system that is capable of effective service differentiation, but without any control interaction with internal nodes. All other domain QoS proposals that facilitate delay differentiation either rely on static resource partitioning – detrimental to utilization – or require signalling with internal nodes.

In both deployment scenarios, ICDS inevitably amounts

to traffic aggregation. In fact, it can be regarded as a per-hop auto-aggregation scheme. At each ICDS node, traffic from different input ports is multiplexed into the same service class and treated as an aggregate at subsequent nodes. Normally, QoS guarantees for traffic aggregates require traffic shaping at multiplexing points. Otherwise, a "misbehaving" flow may distort a traffic aggregate leaving a lightly-loaded node. This situation is somewhat (but not exactly) comparable to the third ICDS scenario in the simulation experiments (cf. Section 5). At subsequent nodes, the resulting burst may lead to excessive packet drop. The problem of misbehaving flows can be addressed by using per-input packet discarding and traffic shaping. This is relatively easy with a worst-case fair scheduler that includes a traffic regulation component anyway. The traffic regulation component just needs to be changed from work-conservation to actual shaping. Given a limited number of classes and input ports, this seems entirely feasible, especially in the light of recent developments for fair rate-proportional schedulers.

7. SUMMARY AND CONCLUSIONS

In this paper, we present a novel basic idea to approach service differentiation in the Internet. We identify the FIFO principle as an important flexibility criterion and relate it to existing QoS work. We propose the ICDS mechanism as a scheduling regime that facilitates delay differentiation while adhering to the FIFO principle. ICDS is not yet a complete and operational proposal, but hopefully stimulates further work in this direction. We discuss a number of implementation details and illustrate open questions.

When making a few simple assumptions, ICDS has provably perfect game-theoretic properties, which is a very desirable characteristic in any distributed environment. We present simulation results to illustrate the effect of ICDS on actual traffic. Most simulation results confirm our assessment of ICDS and are thus encouraging for future work. However, more work is needed to assess whether the game-theoretic assumptions hold in all real-world scenarios. For example, with feedback loops in the rate control function, such as TCP's congestion control, the effective service rate is not as independent of the drop and delay properties as assumed in Section 3.2. Due to limited space, we do not address some of the relevant rate estimation details in this paper. Furthermore, it is not clear how multiplexing in a multi-hop scenario affects the validity of our simple assumptions. Finally, we discuss several deployment scenarios for ICDS and point out chances and challenges resulting from the inherent traffic aggregation taking place with ICDS.

8. REFERENCES

[1] G. Appenzeller, I. Keslassy, and N. McKeown. Sizing Router Buffers. In *Proceedings of SIGCOMM 2004*, pages 281–292, Aug. 2004.

[2] J. C. R. Bennett and H. Zhang. Hierarchical Packet Fair Queueing Algorithms. *IEEE/ACM Transactions on Networking*, 5(5):675–689, Oct. 1997.

[3] L. Breslau, S. Jamin, and S. Shenker. Comments on the Performance of Measurement-Based Admission Control Algorithms. In *Proceedings of Infocom 2000*, pages 1233–1242. IEEE, Mar. 2000.

[4] B. Briscoe, G. Corliano, P. Eardley, P. Hovell, A. Jacquet, and D. Songhurst. An Architecture for Edge-to-Edge Controlled Load Service using Distributed Measurement-Based Admission Control. Internet Draft, July 2005. Work in progress.

[5] D. Clark and W. Fang. Explicit Allocation of Best-Effort Packet Delivery Service. *IEEE/ACM Transactions on Networking*, 6(4):362–373, Aug. 1998.

[6] A. Dhamdhere, H. Jiang, and C. Dovrolis. Buffer Sizing for Congested Internet Links. In *Proceedings of Infocom 2005*. IEEE, Mar. 2005.

[7] C. Dovrolis, D. Stiliadis, and P. Ramanathan. Proportional Differentiated Services: Delay Differentiation and Packet Scheduling. *IEEE/ACM Transactions on Networking*, 10(1):12–26, Feb. 2002.

[8] V. Firoiu, X. Zhang, and Y. Guo. Best Effort Differentiated Services: Trade-off Service Differentiation for Elastic Applications. In *Proceedings of ICT 2001*. IEEE, June 2001.

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

[10] B. Gaidioz and P. Primet. EDS: A new scalable service differentiation architecture for internet. In *Proceedings of ISCC 2002*, pages 777–782. IEEE, July 2002.

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

[12] M. Karsten. A Packet Scheduler with Constant Delay and Fairness and Small Constant Execution Overhead. Apr. 2006. Proceedings of Infocom 2006.

[13] M. Karsten, J. Schmitt, and R. Steinmetz. Implementation and Evaluation of the KOM RSVP Engine. In *Proceedings of Infocom 2001*, pages 1290–1299. IEEE, Apr. 2001.

[14] J. Kaur and H. Vin. Core-Stateless Guaranteed Rate Scheduling Algorithms. In *Proceedings of Infocom 2001*, pages 1484–1492. IEEE, Apr. 2001.

[15] F. Kelly. Models for a self-managed Internet. In *Philosophical Transactions of the Royal Society A358*, pages 2335–2348, 2000.

[16] A. Kortebi, L. Muscariello, S. Oueslati, and J. Roberts. Evaluating the Number of Active Flows in a Scheduler Realizing Fair Statistical Bandwidth Sharing. *ACM SIGMETRICS Performance Evaluation Review*, 33(1):217–228, June 2005.

[17] Y. Lin. Evaluation of Incentive-Compatible Differentiated Scheduling. Master's thesis, School of Computer Science, University of Waterloo, Waterloo, ON, Canada, May 2005.

[18] Next Steps in Signaling (nsis). IETF Working Group, http://www.ietf.org/html.charters/nsis-charter.html.

[19] Qbone home page. http://qbone.internet2.edu/.

[20] I. Stoica and H. Zhang. Providing Guaranteed Services Without Per Flow Management. In *Proceedings of SIGCOMM 1999*, pages 81–94, Aug. 1999.

[21] B. Teitelbaum and S. Shalunov. Why Premium IP Service Has Not Deployed (and Probably Never Will), May 2002. Internet2 QoS Working Group.

[22] P. Valente. Exact GPS Simulation with Logarithmic Complexity, and its Application to an Optimally Fair Scheduler. In *Proceedings of SIGCOMM 2004*, pages 269–280, Aug. 2004.