LAWIN : a Latency-AWare InterNet Architecture for Latency Support on Best-Effort Networks

Katsushi Kobayashi
School of Information Science and Technology
The University of Tokyo
Bunkyo, Tokyo 113–8656, Japan
Email: ikob@acm.org

Abstract—While strict latency restrictions are imposed on network applications, current best-effort Internet architecture entirely lacks this support. In this paper, we propose a “Latency AWare InterNet” (LAWIN) architecture that supports various latency requirements while retaining the best-effort service model. In the LAWIN architecture, applications specify their desired network latency limits, or deadlines, into all packets, and routers schedule these packets according to their deadlines. To this end, we propose two earliest-deadline-first (EDF)-based packet schedulers. The first imposes the same packet loss rate on all applications regardless of the latency specified by each, and provides rough flow-rate fairness. The second scheduler imposes a biased packet loss probability. The biased scheduler also provides an efficient latency and bandwidth trading mechanism for application settings, which motivates applications to set optimal latencies in order to improve efficiency.

I. INTRODUCTION

Many network applications provide interactive experiences, such as voice over IP (VoIP), real-time games, and web-based office suites. These applications have implicit or explicit maximum delays in networks. For example, tight web page rendering thresholds as low as four seconds are required in e-commerce in order to retain customer attention. Thus, Akamai deploys more than 10,000 servers around the world to ensure low latencies [1]. In first-person-shooter games, players with low-latency connections to the game server have a significant advantage over those with large-latency connections [2]. ITU G.114 recommends that end-to-end latencies for interactive voice should be kept below 150 ms.

In spite of these strong latency requirements, latency support for existing networks is clearly insufficient. For example, bufferbloat brings typically up to 10 s latencies at the access router buffers, even when using a broadband link, such as CATV Internet [3]. The bufferbloat is caused by too large packet buffer against the bottleneck link capacity. TCP slow-start algorithm attempts to saturate the bottleneck buffer until congestion is encountered. As a result, the bloated buffer blocks all packets, including those for interactive applications, and user experience is poor.

We believe that the current simple best-effort service model of the Internet should be maintained in the future because best-effort traffic makes minimal demands in terms of economic infrastructure, relying on simple pair-wise economic relationships among ISPs, and between a user and its immediate ISP[4]. In contrast to the best-effort service, existing QoS mechanisms, Intserv and Diffserv, require more complex economic relationships and more expensive infrastructure. As a result, such QoS mechanisms have not been widely deployed yet.

The motivation of this paper is to provide a network latency support mechanism that can cooperate with the best-effort service model. Furthermore, the mechanism must support not only specific applications, like VoIP and videoconference, but also various application requirements. The key idea underlying this paper is that a “fair loss rate” will provide flow-rate fairness in best-effort networks, even with latency support. In the current Internet architecture, flow-rate fairness is achieved by TCP congestion control, which reacts to packet loss as an indication of congestion. With the most used first-come-first-served (FCFS) drop-tail scheduler, the same packet loss rate is imposed regardless of the requirements of particular applications. Thus, as long as packet loss is fair, the properties of transport protocols, including bandwidth sharing, may remain unchanged. Moreover, if the loss-dependency of the deadline can be controlled, we can control bandwidth-sharing properties, such as increasing the bandwidths of long deadline applications, and decreasing those of short ones.

In this paper, we propose a Latency-AWare InterNet (LAWIN) architecture to support various latency requirements while maintaining the best-effort service model of the Internet. In section II, we review related researches. We detail our proposed architecture in Section III, and describe two earliest-deadline first (EDF)-based packet schedulers and their implementation in Section IV. The effect of these schedulers on existing network transport protocols is presented in Section V. We discuss the deployment of LAWIN in the area in Section VI, and offer our conclusions in Section VII.

II. RELATED WORK

Several methods have been proposed for latency support in a best-effort service model, such as equivalent differentiated services, asymmetric best-effort, and rate-and-delay network services[5], [6], [7]. These methods share the idea that latency-sensitive flows cannot achieve higher bandwidth throughputs than ordinary flows by imposing higher drop rates as a tradeoff for higher scheduling priorities. However, these methods lack a broad-range latency requirement indication mechanism be-
cause they are designed to support a limited number of latency classes, generally only two.

In order to mitigate bufferbloat, new active queue management (AQM) methods have been proposed, such as CoDel and PIE[8], [9]. Both of these reduce the queueing delays by increasing the drop probability until the delay becomes shorter than that of the target. However, these approaches cannot ensure that delays are kept under a target, i.e., they control average delay, but instantaneously allow longer delays, such as TCP slow start. Moreover, the aim of these approaches is to reduce the latencies of all flows, but not to support various latency requirements.

Although this paper focuses on best-effort networks, variable latency support may contribute to closed and controlled networks, such as data-center (DC) and stock market networks. Current DC frameworks, such as MapReduce and memcache, are imposed strict job completion time requirements. However, existing systems cannot satisfy these requirements due to transport and/or network delays. For example, TCPincast collapse is caused by synchronized bursts of short-lived flows and queue build-up can be caused by coexisting large flows. pFabric simultaneously supports a number of deadlines with an EDF-based scheduler in the same manner as LAWIN, although it operates on managed DC networks rather than best-effort networks[10]. In stock markets, high-frequency trading (HFT) systems must perform with network latencies smaller than one µs[11]. Thus, latency support for packet forwarding may contribute significantly to those demanding financial applications. These applications to growth sectors might be a major thrust in the deployment of our proposed architecture.

III. LAWIN ARCHITECTURE

The goal of LAWIN, is to provide a “latency-aware” Internet that satisfies various latency requirements for any number of applications while simultaneously providing the best-effort service[12]. LAWIN adds two new components to the existing Internet: (1) per-packet deadline indications from applications, and (2) packet scheduling by routers by exploiting expiring deadline indications.

These two components interact through an appropriate header field, such as the Differentiated Services code point (DSCP) field. If DSCP is used for per-packet deadline indication, 16 or more DSCPs can be assigned because two pools of 16 DSCPs have not assigned, yet[13]. Such small space of fields is sufficient for this, while various latency supports are required. For example, 16 code points can represent a range from 10µs to 320 ms, if deadlines are expressed in the exponent of the power of two. Furthermore, even though DSCPs are not available for per-packet deadlines, other fields are possible, such as the IP option, and the flow label field in the IPv6 header.

We use per-hop latency indication, which is the lead-time in the router queue on a per-hop basis. This neither an end-to-end nor a cumulative latency indication for the following reasons. First, end-to-end latency may fluctuate if the route changes, even while applications are working. The end-to-end indication is inconsistent with these fluctuations. Thus, the latency of a new path is greater than the end-to-end indication, and the intermediate routers discard all packets.

Second, since the number of router hops along a path on the Internet is uncertain, the cumulative latency may appear to be better than the per-hop latency in terms of the control of end-to-end latencies. However, the cumulative format also causes difficulty due to its multi-hop nature. For example, if a long deadline flow passes through two congested routers, and the first congested router consumes most of the allotted latency, the second router should treat the flow as a short one because of the remaining deadline.

Furthermore, the per-hop latency indication is preferred with regard to incremental deployability. End-to-end latency indication requires that all network devices support LAWIN. By contrast, per-hop indication is efficient for partially deployed networks because it specifies the behavior of the per-hop router rather than that of the entire path. For these reasons, the deadline indication must eliminate propagation delays, and work without the support of the entire network.

IV. LATENCY-AWARE SCHEDULERS

LAWIN routers schedule packets with the deadline as a priority, in other words, the packets in the queues are sent out in order of their deadlines. This section describes two such schedulers, those that apply fair and biased packet loss rate.

A. EDF with reneging

We use the EDF with reneging (EDFR) scheduler for LAWIN. The simple EDF can provide excellent latency support on managed networks, such as DCs[14]. However, simple EDF is insufficient for the Internet, which is shared by numerous uncoordinated users. For example, if all packets are declared to have short deadlines, the deadlines of many packets will have elapsed while they are still in the scheduler queue. The stalled packets then remain at the top of the queue and block subsequent packets because EDF is not concerned about whether deadlines for packets have passed. As a result, many packets miss their deadlines under heavy load conditions[15]. By contrast, EDFR drops, or reneges, packets when their deadlines have been passed in the queue, and the stalled packets never block the queues. In EDFR, the packet loss rate improves significantly over the rate of late packets in simple EDF. For example, in an M/M/1 system with a traffic intensity, ρ, of 0.98, while more than 10% of the packets fail using simple EDF, EDFR reduces this by a factor of 40-50 times. Thus, EDFR is a better scheduler than simple EDF on best-effort services.

In order to investigate the properties of the EDFR scheduler, we conducted a simulation using M/M/1 systems. Fig. 1(A) shows the results obtained, where the simulation parameters consisted of a mean interval time, 1/λ, of 2.0, and a mean service time, 1/µ, of 1.96, and so that traffic intensity, ρ or µ/λ, of 0.98. The deadlines for packets followed a uniform distribution as: The minimum deadlines were set to 5, or \( U(5, d_{\text{max}}) \), and the mean value, \( \bar{D} \), or \((5 + d_{\text{max}})/2\), was
associated with each curve. This deadlines distribution repre-
sents that different applications require different deadlines.

In Fig. 1, the dotted line represents the dependency of the
loss rate of the entire system on the mean deadline, \( \bar{D} \), which
was theoretically derived by Kruk et al.[15]. According to
the study, the entire loss rate with EDFR does not depend
on the deadline distribution but on the mean deadline, which
decreases with increasing mean value. The dotted line passes
almost through the middle of each distribution of EDFR
(Fig. 1(A)), which agrees with theory.

All drop dependencies on the deadline distributions, shown
by solid lines, can be classified into two components: near-
flat bottoms on longer deadline areas, and steep cliffs around
excessively short deadlines. Similar bottoms and cliffs were
also obtained in our M/D/1 simulations.

The flat bottoms shows that the EDFR scheduler achieved
“fair loss rates” regardless of deadlines. This is interesting
but not surprising: when the EDFR scheduler drops a packet
because of having exceeded deadline, it does not consider
the original deadline, but the time remaining in the deadlines.
The cliff areas of the figures show that new arrivals with
shorter deadlines were blocked by the ongoing service, or
by the serializing packet. Thus, the curve of the cliff agrees
with the remaining dequeue time distribution,
\( \rho e^{-\mu t} \), which is
represented by broken lines in the figure.

The upper areas of the cliffs indicate a higher loss rate, but
this area is not critical for applications. Once the minimum
deadline that routers must support is standardized, end applica-
tions can avoid this high loss area by specifying a deadline
larger than the minimum value. If a router cannot support the
minimum deadline with the usual MTU size due to the link
speed, link layer fragmentation can be used to reduce serial-
ization delay. Therefore, we consider the discipline around the
flat bottoms of the graphs.

The constant drop property of EDFR is problematic from
the viewpoint of the entire network. If a latency-tolerant but
throughput-sensitive application specifies a longer deadline,
the end-to-end latency possibly increases. This increased la-
tency significantly degrades TCP throughputs because the
throughput of TCP is inversely proportional to the round-trip-
time (RTT)[16]. Therefore, all applications will set shorter
deadlines regardless of their requirements. Such application
behavior decreases the mean value of deadlines. As a result,
the packet loss rate of the entire network worsens. LAWIN
should provide some incentive to applications to set the longest
possible deadlines.

B. EDF with reneging later arrivals

We propose biased packet loss rates, which impose higher
loss rates on packets requiring shorter deadlines. This biased
packet loss mechanism can be realized by using an EDF with
reneging later arrivals (EDFRL) scheduler, which is a calendar
queue derivative. The calendar queue is an implementation of a
priority queue with time complexity \( O(1) \), which has a shifting
time slot array. Sub-priority schedulers are attached to each
time slot (Fig. 2)[17]. Arriving packets are allocated to time
slots that correspond to their deadlines. Furthermore, the sub-
schedulers schedule packets in a priority queue-based manner.
Thus, the packets are queued in order of their deadlines.

The EDFRL scheduler replaces priority queueing with
FCFS as sub-schedulers in the calendar queue. In EDFRL, the
arriving packets are allocated time slots in an ordinal calendar
queue manner, but are sorted in their order of arrival within
each time slot using FCFS. Let us suppose a calendar queue
with \( t_i \) time slot intervals. A packet with a \( t_i + \delta t \ (\delta t < t_i) \)
deadlines is enqueued at $T$. If, after $t_i$ time, another packet with deadline $t_{i-1} - t_i$ is enqueued, these two packets can share a time slot in the calendar queue. In EDFR, the second packet is assigned a higher priority because of its earlier deadline. On the other hand, in EDFRL, the second packet is always scheduled after the first one using the FCFS sub-scheduler. Therefore, a higher drop risk is imposed on the packets with shorter deadlines in the EDFRL scheduler.

Figs. 1(B) and 1(C) show the simulations obtained using the EDFRL scheduler with one and 10 time slot widths, respectively. The flat bottoms obtained with EDFR were inclined in EDFRL, where the slopes depended on the time slot width. Thus, latency-aware schedulers can be obtained by imposing biased and unbiased loss rates according to packet deadline.

C. Scheduler Implementation

We implemented the EDF derivative schedulers described above on Intel’s Data Plane Development Kit (DPDK) [18]. The implemented schedulers are compatible with the Ring Manager API provided by DPDK. The binary heap algorithm was used for the EDF scheduler. Furthermore, the hierarchical calendar queue was used to avoid the disadvantage of tracking the time slot array[19].

We evaluated the enqueue and dequeue processing times of $10^2$ packets on a queue pre-filled with $10^2$–$10^7$ packets. The deadline for each packet was generated from uniform random numbers. Each evaluation was performed 100 times using an HP ML350G9 server composed of two Intel E5-2600 CPUs and two 8 GB DDR4 DRAMs with 1.6 GHz clock.

![Fig. 3: Enqueue and Dequeue cost dependencies on queue size](image)

Fig. 3 shows the queueing cost results of four schedulers: simple FCFS using the Ring Manager of the original DPDK, EDF using a binary heap, EDF with a calendar queue (EDF-C), and FCFS with a calendar for EDFRL schedulers (EDFRL-C). The calendar queue comprised $2^{11}$ time slots having a two-level hierarchy with a 256-ary tree. The enqueue costs with a queue size of $10^5$ packets were 24 ns, 71 ns, 209 ns, and 113 ns per-packet using FCFS, EDF, EDF-C, and EDFRL-C, respectively (Fig.3(A)). The calendar scheduler’s enqueue costs were more than those of FCFS and EDF. The enqueue costs of all schedulers were more or less stable regardless of queue size.

With both hierarchical calendar schedulers, EDC-C and EDFR-C, the enqueue cost increased with decreasing pre-filled packets from $10^4$ to $10^5$. If the hierarchical calendar tree was sparse because of a small number of queued packets, the tree had to be updated when packets arrived. This frequent update increased enqueue costs when the number of pre-filled packets was small.

During dequeueing with $10^6$ packets, the per-packet dequeue costs were 21 ns, 583 ns 185 ns, and 76 ns using FCFS, EDF, EDF-C, and EDFRL-C, respectively (Fig.3(B)). The processing cost using FCFS and EDFRL-C remained almost unchanged regardless of queue size. By contrast, the cost using EDF increased with increasing queue size due to the deletion cost of the heap algorithm. The cost using EDF-C also increased in areas containing more than $10^5$ packets, but it was suppressed compared with the cost of EDF.

We also evaluated the queueing performances using two non-uniform memory access (NUMA) nodes, where the enqueue and dequeue procedures were processed by different CPUs in parallel. Every queueing cost for NUMA was worse than that for a single node. The performance degradation of FCFS was significant compared with those of the others because the cost of memory access was relatively large. Thus, processing cost differences between FCFS and other schedulers will decrease in real systems.

Both calendar queue based schedulers, i.e., EDF-C and EDFRL-C, performed no more than 250 ns per-packet in processing costs, or a throughput of greater than 4 Mpps. These throughputs are insufficient according to current requirements, e.g., 14 Mpps[20]. However, DPDK has bulk enqueue and dequeue APIs, which improve the performance to satisfactory levels by simultaneously treating more than one packet. For instance, per-packet queueing costs decreased to approximately 1/20 when the bulk size was 32.

It should be noted that future packet networks, e.g., 400 Gbps and 1 Tbps, will never obtain a scheduler better than FCFS because of its simplicity. Thus, scheduling algorithms whose cost is of the same order of magnitude of FCFS are believed to be easy to use.

V. EFFECTS ON EXISTING TRANSPORT

In order to evaluate the effects of deploying LAWIN schedulers over the Internet, we conducted simulations using the ns-2 simulator. The simulations focused on the impact of bandwidth sharing and packet loss on existing TCP congestion control mechanisms, i.e., NewReno and CUBIC.

Nevertheless, TCP cannot avoid delays caused by retransmission time outs (RTO) even with LAWIN. In order to maximize the benefit of LAWIN, ordinary TCP should be replaced with a new transport protocol, such as Google QUIC that can eliminate retransmission delays using forwarded error correction (FEC). On the other hand, any new Internet transport protocol, including QUIC, must include a congestion control mechanism compatible with existing TCP. Therefore, since TCP behavior can be regarded as other compatible transports ones, ordinary TCP stacks were used.

All test scenarios were based on the “Common TCP evaluation suite” to avoid possible biases caused by particular
1 ms to 10 ms. These dependences of the loss rate on slot width increased with increasing the calendar time slot width from below them for 120 ms. Furthermore, the loss rate differences the crosses in the 30 ms and 80 ms deadline test flows, and and red, higher packet loss rates were always imposed on the deadline. With the EDFRL scheduler, shown in the green EDFR imposed the same loss rates on both flows regardless of scheduler results are shown in the black makers and lines. The the deadline. In the packet loss plots (Fig. 4: Simulation topologies. (A) A 6×6 dumbbell topology that comprises two 3×3 flow groups, i.e., (T1 . . . T6) for testing, and (S1 . . . S6) for reference. All links have a capacity of 100 Mbps. The dumbbell in (B) has two pairs of nodes, (T1,T3) and (T2,T3), for test flows in addition to the 3×3 background flows among S1 to S6. An EDFR or an EDFRL scheduler is attached to the routers, R1 and R2. The bottleneck capacities are shown for 10 Mbps and 1 Gbps, and all access links have a capacity of 1 Gbps. scenarios [21]. The original ns-2 script and the traffic trace used were published by the Centre for Advanced Internet Architectures [22]. The effects of LAWIN on existing TCPs, i.e., NewReno and CUBIC, were investigated using three scenarios: (1) loss and throughput when competing TCP flows had different deadlines, (2) throughputs of long-lived TCP flows, and (3) TCP behaviors at multiple bottlenecks. For (1), where the loss and throughput were obtained when competing TCP flows with different deadlines, the simulation used the same topology and flows as the “Impact on standard TCP traffic” tests in [21](Fig. 4(A)). In this scenario, the loss and throughput were obtained between two deadlines flows: test and reference. A 100 ms deadline was set for the reference TCP flows, which was a value close to the average RTT of 102 ms. The warm-up and experimental time periods were set to 60 s and 400 s, respectively. Moderate congestion traffic and a 2% packet loss rate were generated on a legacy TCP NewReno in order to observe the loss behavior.

Fig. 5 shows the dependence of the loss and throughput on the deadline. In the packet loss plots (Fig. 5(A)), the EDFR scheduler results are shown in the black markers and lines. The EDFR imposed the same loss rates on both flows regardless of the deadline. With the EDFRL scheduler, shown in the green and red, higher packet loss rates were always imposed on shorter deadline flows, e.g., the diamond markers were above the crosses in the 30 ms and 80 ms deadline test flows, and below them for 120 ms. Furthermore, the loss rate differences increased with increasing the calendar time slot width from 1 ms to 10 ms. These dependences of the loss rate on slot width shows that more number of shorter deadlines were scheduled with lower priorities.

The bandwidth throughputs are shown in Fig. 5(B). With the EDFR scheduler, flows with shorter deadlines were always greater than those with the longer deadlines, but the differences were not very large, i.e., up to 5%. This was because the RTTs of the longer deadline flows were higher than those of the shorter ones due to the behavior of the EDFR scheduler. Thus, the throughputs were suppressed in the case of longer deadlines. On the other hand, with EDFRL, the throughputs with longer deadlines were always better than those with shorter deadlines. However, the throughputs fluctuations were not large, even when the packet loss rate gaps were quite large, e.g., 5%-10% losses with 30 ms flows and zero with 100 ms. This was because most of the traffic in the packet trace comprised short-lived flows, and most of the TCP flows did not reach a steady state. Moreover, results for CUBIC in terms of both loss and throughput were almost identical to those for NewReno.

For (2), where the throughputs of long-lived TCP flows were investigated, the simulation used the same topology and flows as the “Ramp up time” in [21], where the flows comprised two long-lived TCPs and 3×3 10% background flows Fig. 4(B)). In this scenario, the throughputs of long-lived TCPs were estimated for different competing deadlines based on the flow completion time (FCT). In the simulation, the second TCP test flow started 100 s after the first.

Fig. 6 shows the dependence of the FCT on the deadline of the flows. Using EDFR at 10 Mbps with NewReno, i.e., the plot at the top of (A), both flows had almost equal throughputs. This supports the idea “fair loss rate” achieves flow-rate fairness.

Fig. 7 shows FCT plots obtained using different deadline flows. With a shorter deadline of 60 ms, the flow always required a longer FCT than that with a longer deadline of
80 ms, even if they started first, where the duration of the flows exceeded 20 s.

Using EDFRL with 5-10 ms slots, i.e., the two plots at the bottom of Fig. 6 (A), the FCT with the shorter deadline flows in each cluster was clearly worse than that with flows with the longer deadline. The equal throughputs obtained with EDFR and the degraded shorter deadline flows were also found in the CUBIC simulations (B). Moreover, more significant throughput differences were found with EDFRL on CUBIC because its congestion control algorithm is more aggressive than that of NewReno.

These results were clearly attributable to the higher loss rates that occurred with shorter deadline flows because the throughputs of NewReno and CUBIC TCP at a steady state both followed the inverse square root of the packet loss rate [16]. This TCP property obtained with EDFRL show that LAWIN can provide an effective latency and bandwidth trading mechanism, i.e., applications that require shorter deadlines cannot obtain greater throughputs, and applications that require greater throughputs should give up shorter deadlines. In this trading mechanism, applications have to set appropriate deadlines, but not by “self-policing.”

Furthermore, the throughput and loss differences increased as the time slot widened, i.e., the two plots at the bottom of Fig. 6(A) and (B). This also shows that an appropriate loss rate bias can be obtained by adjusting the time slot width.

With a 1 Gbps bottleneck using NewReno (Fig. 6(C)), the overall FCT dependences of the deadline were not obvious due to scattered results, unlike for the 10 Mbps bottleneck. The congestion condition was less than that with the 10 Mbps bottleneck because the access link bandwidth capacity was the same as at the bottleneck. Using EDFR, longer deadline flows performed slightly worse than shorter ones due to RTT unfairness. Using EDFRL with 80-100 ms, longer deadline flows obtained slightly better throughputs than shorter ones, in contrast to those obtained using EDFR.

We believe that deployment of LAWIN is challenging because it requires updating both the end-system and the network. In order to increase the adoption rate of end systems, vendors should distribute LAWIN applications and transport vendors.
stacks as a bundled package, such as Google Chrome and SPDY[23]. Due to this distribution strategy, most major Web services, such as Akamai, Amazon Web Services (AWS), Facebook already deploy SPDY.

For networks, LAWIN should be deployed from access ISP networks rather than back-bone/core networks. Many access ISPs have direct peering with CDNs in order to reduce network delay and eliminate traffic bottlenecks caused by uplink access. As a result, most end-systems can access CDN edge servers within their own access networks. A considerable amount of Internet traffic nowadays is delivered by CDNs, approximately 15%-20% of the world’s Web traffic by Akamai[1]. Therefore, if an access ISP network and a CDN support LAWIN, such amounts of Internet traffic can be covered by LAWIN paths. Furthermore, since Web services sensitive to quality already make use of CDNs, such service providers might be interested in latency support, such as that offered by LAWIN.

LAWIN applications need to determine appropriate deadlines to ensure that efficient services are provided. Even if the condition of a network path is unknown, a reasonable deadline should be selected, such by sending a SYN packet when a TCP connection opens. These deadlines can be derived from the practical router packet buffer size on the existing Internet. For example, to maximize the TCP throughput for throughput-sensitive applications, the deadline should be set to 250 ms, which is the typical router buffer size[24]. If an application requires a VoIP level end-to-end latency, the per-hop deadline should be set to 40-65 ms, which is the buffer size recommended by DOCSIS[25]. Due to the multi-hop nature of the Internet, the end-to-end delay theoretically becomes too large to satisfy VoIP requirements. In practice, however, most Internet links are uncongested by over-provisioning; thus, DOCSIS recommends such buffer size that VoIP quality can be satisfied [26].

The LAWIN protocol can coexist with the existing Internet, but ordinary best-effort traffic is not compatible with latency-aware scheduling because it does not have any latency information. The most practical approach is for each router to determine a default deadline according to its own packet buffer size. Ordinary traffic is scheduled according to the default deadline. This approach might also be useful, if the deadline for a packet is longer than the capacity of a router scheduler.

Furthermore, the LAWIN approach may be deemed more suitable by network engineers than the existing AQM. AQM drops customer packets before the buffer full, which might conflict with an ISP engineer’s common sense that “our production service should do its best.” By contrast, LAWIN drops packets according to the customers’ preferences.

VII. CONCLUSION

In this paper, we proposed the architecture and properties of LAWIN and its schedulers, which support the latency requirements of multiple applications while maintaining the best-effort service model. Our results showed that “fair loss rate” provided approximately flow-rate fairness even with latency support, as simple best-effort traffic. LAWIN has advantages over other QoS approaches because it does not require any flow state, more than one queue, or any latency target. Indeed all it requires is a header field.

ACKNOWLEDGEMENT

This work was partially supported by JSPS KAKENHI Grant Number 26330100.
REFERENCES


