An Online Delay Efficient Multi-Class Packet Scheduler for Heterogeneous M2M Uplink Traffic

Akshay Kumar, Ahmed Abdelhadi and Charles Clancy
Hume Center, Virginia Tech
Email: {akshay2, aabdelhadi, tcc}@vt.edu

Abstract—The sensory traffic in Machine-to-Machine (M2M) communications exhibit heterogeneity in several dimensions such as packet arrival rate, delay tolerance requirements etc. However, the design of most of the existing M2M packet schedulers do not account for this intrinsic heterogeneity in sensor traffic. Therefore, motivated by this, we study the delay-performance of a heterogeneous M2M uplink channel from the sensors to the Programmable Logic Controller (PLC) in an industrial automation scenario. Besides the heterogeneity across the sensors, the data from each sensor is also heterogeneous and can be classified as either Periodic Update (PU) or Event Driven (ED) data. The PU arrivals are periodic, synchronized by the PLC and need to be processed by a prespecified firm latency deadline. On the other hand, the ED arrivals are random and have low-arrival rate without any hard service deadline. We map these contrasting arrival, service characteristics of PU and ED packets from different sensor using step and sigmoidal functions of service latency respectively. Due to the heterogeneity, the aggregated traffic at PLC forms multiple queues/classes which are then served so as to maximize a proportionally-fair system utility. Specifically the proposed multi-class scheduler gives priority to ED data from different sensors as long as the latency deadline is met for the PU data. We minimize successive PU failures using novel penalty functions and clear failed PU packets to reduce network congestion. Using extensive simulations, we compare the performance of our scheme with various state-of-the art packet schedulers and show that it outperforms other schedulers when network congestion increases. The performance gap further increases with heterogeneity in latency requirements and increase in penalty for successive PU failures.

Index Terms—M2M, Latency, Quality-of-Service, MultiClass Scheduler, Industrial Automation

I. INTRODUCTION

Automation for industrial monitoring and control processes has become commonplace these days. It can be modeled as a star-topology Machine-to-Machine (M2M) network of various process sensors communicating wirelessly to a Programmable Logic Controller (PLC) on the ‘uplink channel’ which then feeds back the output to a control device (‘downlink channel’) to ensure the desired process operation. Typically, the industrial M2M traffic has very low latency requirements (of order of few milliseconds). It can be broadly categorized as either non real-time (no deadline for task completion) or soft real-time (decreased utility if deadline not met) or firm (zero utility if deadline not met) or hard (system failure if deadline not met). For instance, the messages from a instrumented protective system or safeguarding system have hard real-time service requirements whereas preventive maintenance applications are typically served in non real-time. With increase in network size, the available computational and communication resources gets shared among a large number of sensors. This makes it harder to provide real-time service due to simultaneous access attempts from multiple sensors [1] and also wastage of wireless spectrum continuously allocated to sensors with very low transmission duty cycle [2].

Therefore in this paper, we develop an online delay-efficient packet scheduler for heterogeneous M2M uplink traffic. To begin with, we use source M2M traffic model in [3] to classify the sensor data into Periodic Update (PU) and Event Driven (ED) packets. The PU arrivals are periodic with firm latency deadline while the ED arrivals are random without hard service deadlines.

Paper Contributions: We map the contrasting delay requirements of PU and ED packets from different sensors using step and sigmoidal utility functions. We define the overall system utility as the product of (time) averaged utilities of all PU and ED traffic-classes in order to ensure proportional fairness. Our goal is to determine the optimal scheduling policy that maximizes the proposed system utility. However, due to the randomness in arrival and service times, there exists no delay-optimal online algorithms [4].

Therefore, we propose an online delay-efficient multi-class heuristic scheduler for the uplink traffic at the PLC, that gives priority to ED data as long as that does not result in a utility loss for the PU data. This results in a higher utility for the ED data. However as the size of networks increases, PU packets increasingly fail to meet their deadlines. We remove the failed PU packets from the system to reduces congestion and thus improve overall system utility. Furthermore, since successive packet drops are worse than isolated packet drops, we introduce a novel penalty function as part of PU utility function to reduce successive PU failures. Using extensive simulations, we show that this results in an overall increase in system utility as compared to other popular scheduling

1We assume firm service deadline for PU instead of hard deadline because typically the periodic data does not change much from one period to another, especially if the period is small. So if a PU packet fails to meet service deadline, the PLC can use an estimate for the PU based on the recent history, albeit with some degradation in utility.
policies such as First-Come-First-Serve (FCFS), Earliest Due Date (EDD), priority scheduling etc. Lastly, the proposed scheduler is agnostic to the wireless technology being used for the M2M uplink and is also independent of the hardware-software architecture used in practical real-time embedded systems. Also it can easily adapt to accommodate time-varying arrival and service rates for the PU and ED packets.

**Paper Outline:** The rest of the paper is organized as follows. Section II details the related work. Section III introduces the system model and defines the utility functions for PU and ED data. Then in Section IV, we formulate the utility maximization problem and in Section V describe the proposed scheduling scheme at PLC. Section VI presents simulation results. Finally Section VII draws some conclusions.

## II. RELATED WORK

A number of prior works have looked into integrating the design of real-time scheduling schemes in middle-ware based architectures so as to meet the hard deadline of real-time services. Tommaso et. al. in [5] presented a real-time service oriented architecture for industrial automation. However, these works are tangential to our line of work in the sense that our proposed scheduler is agnostic of the hardware-middleware-software architecture and maps the real-time QoS requirements of M2M traffic into utility functions that needs to be maximized.

Another line of work focuses on QoS-aware packet scheduler for M2M traffic in Long Term Evolution (LTE) network (see [6] and references therein). Most of these works use some variants of Access Grant Time Interval scheme for allocating fixed or dynamic access grants over periodic time intervals to M2M devices. Nusrat et. al. in [7] designed a packet scheduler for M2M in LTE so as to maximize the percentage of uplink packets that satisfy their individual budget limits. Unlike our work, these works design packet scheduler specific to a wireless standard such as LTE and are thus heavily influenced by the Medium Access Control architecture of LTE. Also unlike our work, they also don’t focus on the heterogeneity in packet rate, service delay requirements of traffic from different sensors.

A number of scheduling algorithms have been proposed specifically for real-time embedded systems (see [8] and references therein) that are agnostic to the application scenario, wireless technology used or hardware-software architecture. These schemes assume hybrid task sets comprising of hard periodic requests and soft real-time aperiodic requests. The goal of all these schemes is to guarantee completion of service (before deadline) for all periodic request and simultaneously aim to reduce the average response times of aperiodic requests. They are broadly classified into Fixed-priority and Dynamic priority assignments. Fixed priority schemes schedule periodic tasks using Rate-Monotonic algorithm but differ in service for aperiodic tasks. Dynamic scheduling algorithms schedule periodic tasks using EDD scheme and allow better processor utilization and enhance aperiodic responsiveness as compared to the fixed priority schemes. The drawbacks of these schemes is that they assume that the service times for each tasks are known apriori. Hence they schemes are not truly online in their present form and would require considerable modifications.

The work that is most closely related to our work is by Kumar et. al. in [9] wherein they proposed a delay-efficient heuristic packet scheduler for M2M uplink. However this work assumed the traffic originating from different sensors is identical having same arrival rate, service delay requirements etc. Also, unlike our work, their proposed scheduler does not differentiate between successive PU failures and isolated PU failures.

## III. SYSTEM MODEL

Fig 1 shows the system model illustrating the queuing process in the uplink channel (i.e., from sensors to the PLC) in an industrial automation setting. Each of the $N$ sensors transmits two types of packets to the PLC: frequent periodic updates (PU) for some process related measurement or sporadic event-driven (ED) packets triggered when the sensor measurement for a physical quantity cross its threshold. In the most general sense, the arrival rate and latency requirements of uplink traffic is different for each sensor. Therefore, we model the PU arrivals from the $i^{th}$ sensor as a deterministic periodic process with period $T_{pu}^i$ and thus rate $\lambda_{pu}^i = 1/T_{pu}^i$ while the ED arrivals is modeled as a Poisson process with rate $\lambda_{ed}^i$.

Let $s_{pu}^i$ and $s_{ed}^i$ denote the size for PU and ED packets from the $i^{th}$ sensor. The service time for PU and ED packets from the $i^{th}$ sensor are assumed to be exponential with rate $\mu_{pu}^i$ and $\mu_{ed}^i$ respectively, given by

$$\mu_{pu}^i = \mu/s_{pu}^i \text{ and } \mu_{ed}^i = \mu/s_{ed}^i.$$

where $\mu$ is the service rate at PLC per unit packet size. In general, the total time spent by a packet in the system, $T$, can
be written as sum of following components,

\[ T = T_{\text{trans}} + T_{\text{prop}} + T_{\text{cong}} + T_{\text{queue}} + T_{\text{ser}}, \]

which denote the following component delays:

- \( T_{\text{trans}} \): Transmission delay at the sensor.
- \( T_{\text{prop}} \): Propagation delay from the sensor to PLC.
- \( T_{\text{cong}} \): Congestion delay due to shared wireless channel in large-scale sensor network.
- \( T_{\text{queue}} \): Queuing delay at the PLC.
- \( T_{\text{ser}} \): Processing time for a packet at the PLC.

In this work, we ignore all the terms except the queuing delay at PLC \( T_{\text{queue}} \) and the service time \( T_{\text{ser}} \). This is because the packet size is usually quite small to ignore \( T_{\text{trans}} \) and the sensors and PLC are usually close enough on the shop floor to permit the usage of low power wireless signaling and thus ignore \( T_{\text{prop}} \). Also, we assume that the available wireless spectrum is large enough to allocate a dedicated transmission channel to each sensor; so there is no congestion delay \( T_{\text{cong}} \).

In the most general case, the PU and ED packets from \( N \) sensors form \( 2N \) separate queues/traffic-classes. Now the queuing delay for each class depends upon the scheduling policy adopted at the PLC. The scheduling policy at PLC should be chosen so as to maximally satisfy the latency constraints of PU and ED packets.

IV. Problem Formulation

We first map the latency requirements onto the utility functions for the PU and ED classes as shown in Fig. IV-B and IV-B.

A. Utility Functions

1) PU utility: As stated previously, the PU packets need to be processed within a prespecified (usually by PLC) time interval at the end of which there is no utility in serving the packet. For a PU packet from \( i \)-th sensor with latency \( l_{pu}^i \), we define the utility function as,

\[ U_{pu}^i(l_{pu}^i) = \begin{cases} 
1 & \text{if } l_{pu}^i < l_d^i \\
0 & \text{if } l_{pu}^i \geq l_d^i
\end{cases}, \tag{3} \]

where \( l_d^i \) is the latency deadline at which utility drops to 0.

We now define a penalty function so as to limit the number of consecutive PU packets that fail to meet the latency deadline.\(^2\)

Let \( r_l \) denote the run-length\(^3\) of PU drops for \( i \)-th sensor, then the corresponding penalty is,

\[ P_{pu}^i(r_l) = r_l - r_l \gamma_l, \quad r_l \geq 1 \tag{4} \]

where \( \gamma_l \geq 1 \) is a penalty function parameter denoting the degree of penalty for \( i \)-th sensor. \( \gamma_l = 1 \) implies that we do not penalize consecutive PU drops.

\(^2\)Hereafter, we use the term PU drop to indicate that PU packet fails to meet its deadline.

\(^3\)A run is a maximal length sequence of consecutive identical outcomes within a larger sequence. For example, let [SSDDDSDDDDS] denote a sequence of PU success (S) and drops (D). Then we have three runs of PU drops of lengths 1, 2 and 3.

2) ED utility: Unlike the PU packets, ED packets do not have a hard deadline, rather their utility is a strictly decreasing function of latency. We define the ED utility for \( i \)-th sensor as a sigmoidal function, as in [10], [11], given by,

\[ U_{ed}^i(l_{ed}^i) = 1 - c_i \left( \frac{1}{1 + e^{-a_i(l_{ed}^i - b_i)}} - d_i \right), \tag{5} \]

where, \( c_i = \frac{1 + e^{b_i}}{e^{-a_i} + e^{b_i}} \) and \( d_i = \frac{1}{e^{-a_i} + e^{b_i}} \). We note that \( U_{ed}^i(0) = 1 \) and \( U_{ed}^i(\infty) = 0 \). The parameter \( a_i \) is the utility roll-off factor whose value depends on the criticality of the application.

The inflection point of (5) occurs at \( l_{ed}^i = b_i \).

B. System utility function

For a given scheduling policy \( \mathcal{P} \), we first define a proportionally fair system utility function as,

\[ \mathcal{V}(\mathcal{P}) = \prod_{i=1}^{N} U_{pu}^i(P_u^i(\mathcal{P})) \cdot U_{ed}^i(P_u^i(\mathcal{P})), \tag{6} \]

where \( U_{pu}^i(\mathcal{P}) \) and \( U_{ed}^i(\mathcal{P}) \) are the average utility of PU and ED packets in the steady state given as,

\[ U_{pu}^i(\mathcal{P}) = \lim_{T_s \to \infty} \sum_{j=1}^{M_{pu}^i(T_s)} \frac{M_{pu}^i(T_s)}{M_{pu}^i(T_s)} \left( \frac{P_{pu}^i(r_l)(P_u^i(\mathcal{P}))}{M_{pu}^i(T_s)} \right), \tag{7} \]

\[ U_{ed}^i(\mathcal{P}) = \lim_{T_s \to \infty} \sum_{j=1}^{M_{ed}^i(T_s)} \frac{M_{ed}^i(T_s)}{M_{ed}^i(T_s)} \left( \frac{P_{ed}^i(r_l)(P_u^i(\mathcal{P}))}{M_{ed}^i(T_s)} \right), \tag{8} \]

where \( M_{pu}^i(T_s) \) and \( M_{ed}^i(T_s) \) are the number of PU and ED packets from \( i \)-th sensor served in time \( T_s \). \( R_i(T_s) \) denotes the number of runs of PU drops in time \( T_s \) and \( r_l \) denotes the run-length of \( j \)-th PU drop for sensor \( i \). \( l_{pu}^i \) and \( l_{ed}^i \) denote the latency of the \( j \)-th packet from \( i \)-th sensor of type PU and ED respectively. The parameters \( \beta_{pu}^i \) and \( \beta_{ed}^i \) denote the priority given to average PU and ED utility for \( i \)-th sensor.
the PU packets meet their latency deadline. We reduce latency of ED packets while ensuring that most of them are immediately preempted by the PU packet. This ensures that if the latency of any PU packet exceeds its threshold while being processed by PLC, then that ED packet gets preempted from server by the ED class with next highest priority. Thus, we reduce latency of delay-sensitive ED class without incurring very large latency for delay-tolerant packets.

C. Optimal scheduler

We now describe the utility maximization problem that needs to be solved to determine the optimal scheduling policy at PLC. It is given by,

\[ \max_{\mathcal{P}} \mathcal{V}(\mathcal{P}) = \prod_{i=1}^{N} \mathcal{U}_i^* \left( \beta_i^\text{pu}(\mathcal{P}) \right) \times \mathcal{U}_i^* \left( \beta_i^\text{ed}(\mathcal{P}) \right). \]  

(9)

If the service times of PU and ED were deterministic or known a priori, then the optimal scheme is to schedule PU jobs from sensor \( i \) so that they are completed exactly at the deadline \( \delta_i^\text{pu} \). However, since the arrival time of ED packet, the service times for PU and ED packets are random, it is not possible to determine an online optimal scheduler [4]. Therefore, we propose an online heuristic scheduler that aims to maximize the utility function and is described in the next section.

V. PROPOSED SCHEDULER

A. Service order between PU and ED classes

The utility of PU remains constant at 1 until the deadline, at which it drops to 0. Therefore, it would be best (from the perspective of ED latency) to delay the service of PU as much as possible but ensuring that we serve it before deadline. However, the randomness in PU and ED service times makes it hard to determine the optimal start of service for each PU packet in real-time. To overcome this problem, our proposed scheduler gives priority to ED packets as long as the current latency of PU packet from any sensor \( i \) is less than its preset threshold \( \delta_i^\text{pu} \), \( 0 < \delta_i^\text{pu} < \delta_i^\text{ed} \), \( \forall i = 1, 2, \ldots, N \). If the latency of any PU packet exceeds its threshold while PLC is processing any ED packet, then that ED packet gets immediately preempted by the PU packet. This ensures that we reduce latency of ED packets while ensuring that most of the PU packets meet their latency deadline.

B. Service order within PU classes

We now discuss how the proposed scheduler determines the order of service between packets of different PU classes. It is more important to meet latency deadline of PU class with high arrival period since the sensor data may have changed substantially between two consecutive arrivals. Therefore, we prioritize the service of PU classes in decreasing order of arrival time-period, i.e., PU class with highest arrival period gets highest (preemptive) priority. However, similar to previous section, the priority of PU classes comes into effect only when the packet latency exceeds its corresponding latency threshold.

C. Service order within ED classes

We now discuss how the proposed scheduler determines the order of service between packets of different ED classes. We assign higher priority to ED classes that are more delay-sensitive (i.e., have higher \( a_i \) or lower \( b_i \)). Unlike previously, the priority order among ED classes comes into effect immediately after the packet enters the system. However, if the latency of \( i \)th ED class exceeds a certain threshold \( \delta_i \), while there are packets of lower priority ED classes waiting for service, then its latency increases by the ED class with next highest priority. By definition, the threshold \( \delta_i = \infty \) for ED class with least priority. This preemption ensures that ED packets with unusually large service time do not increase latency of subsequent packets. Thus, we reduce latency of delay-sensitive ED class without incurring very large latency for delay-tolerant classes.

Now, in order to maximize the system utility for given set of system parameters, we need to determine jointly optimal value of PU and ED thresholds, \( l_i^\text{pu} \) and \( \delta_i \), \( \forall i = 1, 2, \ldots, N \). Using simulations, the optimal thresholds can be stored in a Look-Up-Table (LUT) prior to the deployment of proposed scheduler. Thus the proposed scheme can easily adapt to changing packet arrival rate, packet size, number of sensor nodes etc. by looking up and using the optimal \( l_i^\text{pu}, \delta_i \) for each case.

Another novel feature of our scheduler is that we drop PU packets from service or queue that exceed their latency deadline as there is no resultant utility from servicing such packets. However, dropping them from service will reduce congestion and thus reduce latency for packets of all the classes. The proposed scheduler is described in detail in Algorithm 1.

VI. SIMULATION RESULTS

In this section, we use Monte-Carlo simulations to evaluate the system utility performance of our scheduler against various standard scheduling policies such as FCFS, EDD. Preemptive priority scheduling. The simulation time \( T_s \) is set to a large value (40 s) to ensure a steady-state queuing behavior. \( N \) is set to 50 sensors. Although the proposed scheduler is designed for 2N PU and ED classes, the simulations for proposed scheduler becomes computationally intractable as \( N \) increases. This is because it requires us to determine 2N − 1 jointly optimal latency thresholds\(^4\). Therefore for computational tractability, we assume that half of sensory traffic (i.e., 25 sensors) can

\(^4\)The number of threshold variables is 2N − 1 rather than 2N because the optimal \( \delta_i \) for least priority ED class in \( \infty \).
be classified into one set of PU and ED class (i.e., PU1 and ED1), while the other half into PU2 and ED2 class. So, we consider a total of 4 traffic classes. We assume the arrival period per sensor for PU1 to be $T_{pu}^1 = 150$ ms, while for PU2 to be $T_{pu}^2 = 62.5$ ms. The arrival rate per sensor for ED2 is set to $\lambda_{ed}^2 = 0.004$ ms and for ED1 it is varied from 0.004–0.01 ms. PLC service rate is $\mu = 100$ bits/ms. The relative importance of all classes is set equal, i.e., $\beta_{pu}^1 = \beta_{pu}^2 = \beta_{ed}^1 = \beta_{ed}^2 = 1$, unless mentioned otherwise. The size of packets for all classes are set equal to 1. The latency deadline for PU1 is, $l_{pu}^1 = 4$ ms and for PU2 is $l_{pu}^2 = 8$ ms. The utility function parameters for ED1 are set to, $[a_1, b_1, s_1] = [1, 10, 1]$ and for ED2 are $[a_2, b_2, s_2] = [0.7, 20, 1]$ unless mentioned otherwise.

The latency threshold for ED1 is, $\gamma_1 = b_1 + (4/a_1)$ and for ED2 is $\gamma_2 = \infty$. For each simulation scenario, the latency threshold for PU1 and PU2 class are determined by jointly maximizing system utility over $0 \leq l_{pu}^1 \leq l_{pu}^{d1}$ and $0 \leq l_{pu}^2 \leq l_{pu}^{d2}$.

### A. Impact of latency-heterogeneity across sensors

In this section, we study the impact of increasing heterogeneity in latency requirements of various sensors on the performance of different schedulers. Fig. 4 shows plot of system utility for a nearly homogeneous system with $T_{pu}^1 = T_{pu}^2 = 88$ ms, $[l_{pu}^{d1}, l_{pu}^{d2}] = [7.8, 8]$ ms, $[a_1, a_2] = [0.65, 7]$ ms and $[b_1, b_2] = [19, 20]$ ms. Then, we increase the heterogeneity in system by using the default latency parameters; the resulting system utility is shown in Fig. 5. We note that the performance gap between the proposed scheduler and the other schemes, $\Delta V$, is much larger for heterogeneous system ($\Delta V = 0.33$) compared to homogeneous system ($\Delta V = 0.19$), especially at higher $l_{pu}^{d1}$. This is because the impact of priority scheduling for different PU and ED classes in proposed scheduler becomes more prominent when the traffic classes are diverse.

### B. Impact of choice of ED threshold, $\gamma_1$

Fig. 6 shows the impact of selecting optimal threshold $\gamma_1$ for ED1 class on the system utility. We note that the performance of proposed scheduler is significantly improved ($\Delta V = 0.60$) compared to Fig. 5 ($\Delta V = 0.33$) where $\gamma_1$ was arbitrarily set to $b_1 + (4/a_1)$, especially at higher $l_{ed}^1$. This is because selecting optimal $\gamma_1$ avoids latency of ED2 to becomes very large when $l_{ed}^1$ becomes very large relative to $l_{ed}^2$.

### C. Impact of penalizing consecutive PU drops

So far, we have assumed there is no penalty for consecutive PU drops, i.e., $\gamma_1 = \gamma_2 = 1$. We now study the impact of penalizing consecutive PU drops ($\gamma_1 > 1$) on the performance of different schedulers. Fig. 7 shows the plot of system utility when the PU drop penalty is small and equal for PU1 and PU2 ($\gamma_1 = \gamma_2 = 1$). We observe that while the performance of proposed schedulers does not change appreciably, the performance of other schedulers drops down considerably and becomes close to 0 at $\lambda_{ed}^1 = 0.25$. This is because the proposed scheduler easily adapts to the change in PU penalty by altering the latency thresholds accordingly. But the other schedulers cannot do so and their performance suffer. This degradation in performance is more pronounced when the PU drop penalty for higher priority class PU1 is increased from $\gamma_1 = 1.2$ to 2 as illustrated in Fig. 8.
In this paper, we presented an online delay-efficient packet scheduler for uplink M2M traffic with heterogeneity in latency requirements both within and across sensors. The data from each sensor is classified as either PU or ED packets. The varying latency requirements of PU and ED class for different sensors is represented using different utility functions. We then proposed a novel proportionally-fair multi-class packet scheduler that primarily gives priority to ED data as long as the latency deadline is met for the PU data. However as the size of networks increases, PU packets increasingly fail to meet their deadlines. We remove the failed PU packets from the system to reduces congestion and thus improve overall system utility. We also introduce a novel penalty function as part of PU utility function so as to reduce burst PU failures. Using extensive simulations, we did a comparative analysis of the proposed scheme with various state-of-the-art scheduling policies such as FCFS, EDD, Preemptive priority etc. We note that as the uplink traffic increases, the proposed scheduler outperforms other schedulers. The performance gap further increases with increasing heterogeneity in latency requirements and increase in penalty for successive PU drops.

VII. CONCLUSIONS

In this paper, we presented an online delay-efficient packet scheduler for uplink M2M traffic with heterogeneity in latency requirements both within and across sensors. The data from each sensor is classified as either PU or ED packets. The varying latency requirements of PU and ED class for different sensors is represented using different utility functions. We then proposed a novel proportionally-fair multi-class packet scheduler that primarily gives priority to ED data as long as the latency deadline is met for the PU data. However as the size of networks increases, PU packets increasingly fail to meet their deadlines. We remove the failed PU packets from the system to reduces congestion and thus improve overall system utility. We also introduce a novel penalty function as part of PU utility function so as to reduce burst PU failures. Using extensive simulations, we did a comparative analysis of the proposed scheme with various state-of-the-art scheduling policies such as FCFS, EDD, Preemptive priority etc. We note that as the uplink traffic increases, the proposed scheduler outperforms other schedulers. The performance gap further increases with increasing heterogeneity in latency requirements and increase in penalty for successive PU drops.

REFERENCES

Algorithm 1 Proposed Online Packet Scheduler

Inputs
\( l_i, l_i \forall i = \{1, 2, \ldots, N\} \) are latency deadlines and thresholds for \( N \) PU classes.
\( \delta_i \forall i = \{1, 2, \ldots, N\} \) are the latency threshold for \( N \) ED classes.
\( T_{\text{sim}} \) is the duration of simulation and \( \Delta T \) is the time resolution at which the algorithm operates.

Local
\( T_c \) stores current system time.
\( A \) and \( B \) denotes the set of all PU and ED classes.
\( A^e \) and \( B^e \) denotes the set of all PU and ED classes that exceed their corresponding threshold \( l_i \) and \( \delta_i \) respectively.
\( \bar{B} \) denotes the set \( B - B^e \).
\( A^d \) denotes the set of all PU classes that fail to meet their deadlines \( l_i \).
\( A^h, A^e_h, B^h, \bar{B}^h \) denote the class with highest priority in sets \( A, A^e, B, \bar{B} \) respectively.
\( z \) denote class of current packet being served.

Initialization
Advance current time \( T_c \) to be the arrival time of next packet in system.
Set \( A^d = \emptyset \). Set \( A^h, B^h \) from given set of PU and ED classes.

Begin Algorithm:
1: while \( T_c \leq T_{\text{sim}} \) do
2:   Determine \( z \) and the set \( A^d \).
3:   if \( A^d \neq \emptyset \) then
4:     Remove the Head-of-Line (HoL) packet from all PU queues in set \( A^d \). go to 15.
5:   end if
6:   Determine the set \( A^e, B^e, \bar{B} \) and the classes \( A^e_h, B^h \).
7:   if \( A^e = \emptyset \) & \( z \notin \bar{A}^e \) \& \( (z \in \bar{A}^e & z \neq A^e_h) \) then
8:     Preempt the current packet with HoL of class \( A^e_h \).
9:   elseif \( A^e = \emptyset \) & \( B^e = \emptyset \) & \( z \neq B^h \) then
10:    Preempt the current packet with HoL packet of class \( B^h \).
11:   elseif \( A^e = \emptyset \) & \( B^e \neq \emptyset \) & \( z \neq \bar{B}^h \) then
12:      Preempt the current packet with HoL of class \( \bar{B}^h \).
13:   else
14:      \( \triangleright \) No preemption occurs; wait till current packet leaves server.
15:      Advance current time \( T_c \) to the service completion event.
16:     Recalculate the sets \( A^e, B^e, \bar{B} \) and the classes \( A^e_h, B^h \).
17:     if \( A^e = \emptyset \) & \( B^e = \emptyset \) then \( \triangleright \) None of the PU or ED class exceed latency threshold
18:      Serve the HoL packet from class \( B^h \).
19:     elseif \( A^e = \emptyset \) & \( B^e \neq \emptyset \) then \( \triangleright \) None of the PU class exceed latency threshold
20:      Serve the HoL packet from class \( B^h \).
21:     else \( \triangleright \) Some of the PU class exceed latency threshold
22:        Serve the HoL packet from class \( A^e_h \).
23:     end if
24:    go to 1.
25:   end if
26: end while
27: \( T_c = T_c + \Delta T \).