Computer Networks 75 (2014) 351–366

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Optimal energy efficient packet scheduling with arbitrary individual deadline guarantee

Feng Shan [a,*], Junzhou Luo [a], Xiaojun Shen [b]

[a] School of Computer Science and Engineering, Southeast University, Nanjing, Jiangsu 210096, China
[b] School of Computing and Engineering, University of Missouri-Kansas City, Kansas City, MO 64110, USA

A R T I C L E I N F O

Article history:
Received 30 November 2013
Received in revised form 18 May 2014
Accepted 22 October 2014
Available online 29 October 2014

Keywords:
Energy efficiency
Rate control
Wireless communication
Densest interval first
Deadline guarantee
Scheduling algorithm

A B S T R A C T

Given a rate adaptive wireless transmitter, a challenging problem is to design a rate control policy for it such that the energy consumption is minimized at transmitting a set of dynamically arrived pack2;ets with arbitrary individual deadlines. In a decade, researches have partially made progress on this topic. A latest work offers an optimal algorithm that allows packets to have arbitrary deadlines but requires them to follow the order they arrive. This paper first presents the Densest Interval First (DIF) policy which repeatedly locates the densest data interval and determines its transmission rate. This policy is proved to be optimal for the most general model that allows arbitrary arrival times as well as arbitrary deadlines. Then, this paper presents a simple EDF (earliest deadline first) algorithm to actually schedule the transmission time for each packet. It is proved that the EDF always guarantees every packet to complete transmission before its deadline with minimum energy consumption which is computed and required by DIF. Finally, this paper also proposes a novel online policy named Density Guided Cooling (DGC) policy which models Newton's Law of Cooling. Simulations show that online DGC policy constantly produces a rate scheduling that on average consumes energy within 110% of the minimum value obtained by the offline DIF.

1. Introduction

Most wireless networks, e.g., sensor networks, ad hoc networks and cell networks, rely upon limited energy supply such as batteries to support their operations. Therefore, how to efficiently use the limited energy is a crucial issue in all aspects of the network design and operations, which often determines the length of wireless devices' working period or the network lifetime. Tremendous research efforts have been made in designing energy efficient routing, energy efficient data gathering, etc.

Because very often packets from various real-time applications that have different arrival times and different delay constraints need be transmitted through a common channel, a challenging research task is to develop a transmission rate control policy and a scheduling algorithm such that a minimum energy is used in transmitting all arrived packets before their deadlines.

1.1. Related work

Prabhakar, Uysal-Biyikoglu, and El Gamal are among the first group of researchers who formulated the energy efficient packet transmission problem more than a decade ago ([1] and its extension [2]), which has drawn considerable interests from researchers in the field of wireless communications. In [1,2], they considered an offline case

* Corresponding author.
E-mail addresses: shanfeng@seu.edu.cn (F. Shan), jluo@seu.edu.cn (J. Luo), shenx@umkc.edu (X. Shen).

http://dx.doi.org/10.1016/j.comnet.2014.10.022
1389-1286/

where the arrival time and size of every packet are known prior to the scheduling and all packets have a common deadline. They presented an optimal scheduling algorithm that guarantees to deliver all packets before the deadline with minimum energy consumption. In their proof, several important optimality properties were introduced which are useful and inspiring for following researches. Uysal-Biyikoglu and El Gamal [3] also generalized the problem by taking multiple-access channels and channel fading into consideration. They proposed the *FlowRight* algorithm to find an optimal offline schedule for the generalized problem but still assumed that all packets have a common deadline. As pointed out by Chen et al. [4,5], the single deadline model does not explicitly consider individual packet delay performance.

In [6], Khojastepour and Sabharwal started to look at the energy efficient packet transmission problem where each packet could have its own deadline. They proposed the *water-filling* method to find an optimal rate control policy. However, this method is applicable only for the case where all packets have arrived and have been waiting in buffer before scheduling. Obviously, this is an easy case but a good initial work for dealing with individual packet deadlines.

The energy efficient packet transmission problem with individual packet deadlines has also been studied by other researchers. Chen et al. ([4] and journal version [5]) proposed an offline optimal scheduling algorithm that handles individual deadlines. However, a restriction was imposed that all packets must have equal delay constraints which means that the length of time interval from the arrival time to its deadline is the same for every packet. Later, they extended the algorithm to an online algorithm [7] and then to a fading channel ([8] and journal version [9]). Although the authors claimed the result can be extended to scenarios with unequal delay constraints, it seems not an easy job.

Zafer and Modiano ([10] and journal version [11]) presented an optimal algorithm that allows each packet to have an arbitrary size, an arbitrary arrival time and an arbitrary deadline. They claimed that earlier results on the energy efficient packet transmission problem can be recovered as special cases. They used cumulative curves to trace packet arrivals and packet departures. The key observation is that a feasible departure curve always lies between the arrival curve and minimum departure curve. Displayed in the cumulative data-time diagram, their idea is intuitive and easy to understand. Later, they extended their results by considering fading effects [12]. Their work has made a true progress on the energy efficient packet transmission problem. However, they still need to make an undesirable assumption that a packet arriving earlier carries an earlier deadline. In other words, the cumulative curves fail to handle the case where a packet arrived later may have a more urgent deadline.

In addition to above research results that are directly related to our paper, some important extended work is also observed. For example, recently, Yang and Ulukus [13] investigated the energy efficient packet transmission problem in an energy harvesting system where the energy used by the transmitter can get recharged, time by time, to support long life operation. However, it is still necessary to consider how to control the transmission rate to minimize energy consumption because the recharged energy is limited each time and the amount of data to be transmitted may be large. They assume that the time and amount of energy received from each harvesting are known in advance and the size and arrival time of each packet are also known in advance. There is no deadline considered, but they presented an optimal algorithm that guarantees to finish transmission of all data in a shortest time span. Later the result for a single channel was extended to the case of multi-access channels [14], to the case of broadcasting channels [15,16], and to the case with fading channels [17,18]. Interested readers may find more related work such as [19–21]. We omit details here.

Most papers we introduced above also provided online algorithms [1–3,7–9,11,18] as an extension of their offline algorithms. Basically, they follow more or less a similar approach, that is, based on current known information, use the offline algorithm to set transmission rates until a new packet arrives. When a new packet arrives, the online algorithm re-calculates the best rates using the offline algorithm.

1.2. Contributions

We can conclude from the above related works that an unsolved challenging open problem in the past 10 years is how to design an optimal energy efficient rate control policy for a single channel for transmitting a sequence of packets each of which has an arbitrary individual arrival time, arbitrary size and arbitrary individual deadline. The technique of cumulative curves seems not applicable to this more general model. We need a new method. Our contributions can be summarized as follows.

1. We have solved the above open problem. An optimal rate control policy named *Densest Interval First* (*DIF*) is presented, which is inspired by the YDS algorithm proposed by Yao et al. [22].
2. The DIF approach opens a new avenue to obtaining optimal results for other similar rate adaption problems in fading channels, energy harvesting systems, multi-channel systems, etc. in the future.
3. We prove that, once the transmission rate for each time interval is determined by DIF, the Earliest Deadline First (EDF) scheduling algorithm produces an actual schedule for each individual packet to complete its transmission before its deadline with the minimum energy allowed by DIF.
4. We also present an online policy called Density Guided Cooling (DGC) policy that models Newton's Law of Cooling. Simulations show that this policy constantly produces a rate scheduling that on average consumes energy within 110% of the minimum.

The YDS algorithm [22] is designed to solve task-scheduling problems for processors, in which tasks may have arbitrary arrival time and arbitrary deadline. The YDS algorithm inspired us to design the DIF policy, which solves the 10-year open question in wireless communication. Although the two methods share some similar idea, the optimality proofs are different. In this paper, we present a direct and independent optimality proof for the scenario of wireless date transmission. Moreover, after the transmission rate is determined by DIF, we discovered that applying EDF (earliest deadline first) rule for all packets can actually guarantee every deadline within the energy limit imposed by DIF.

The rest of the paper is organized as follows. In Section 2, the system model and the problem formulation are presented. In Section 3, we introduce the notion of data interval and the *DIF* policy. The correctness and optimality of *DIF* policy are also presented. Section 4 discusses the EDF scheduling and proves that EDF guarantees every packet to meet its deadline. Section 5 introduces the online DGC policy and presents simulation results. Section 6 concludes this paper.

2. System model and problem formulation

We consider a single point to point transmission channel over which the transmitter needs to send a set of n packets to the receiver. We first model the data set and its delay constraints.

2.1. Data set and delay constraints

We assume $P = \{P_1, P_2, \ldots, P_n\}$ is a set of n packets to be transmitted. Each packet P_i has an arbitrary arrival time $a_i \geqslant 0$, an arbitrary deadline $d_i\,(>a_i)$, and an arbitrary packet size $B_i > 0$, denoted by a triple $P_i = (B_i, a_i, d_i)$, $1 \leqslant i \leqslant n$. The time interval $[a_i, d_i)$ is called the *waiting interval* of P_i. Note that we will use a left-closed, right-open interval to represent a time interval throughout this paper. It is allowed that multiple packets arrive at the same time or multiple packets have a common deadline. A packet size is measured by a number of bytes that could be transmitted at any rate. The transmission of packet P_i, $1 \leqslant i \leqslant n$, can begin only after its arrival time a_i and must finish by its deadline d_i. This is called *causality constraints* [13]. A packet may be transmitted in several segments, but must finish before its deadline. For the offline problem, we assume all information for P_i, including B_i, a_i, d_i, $1 \leqslant i \leqslant n$, are known before we determine the transmission rate and scheduling. For convenience, we assume packet arrival times are sorted such that $0 \leqslant a_1 \leqslant a_2 \leqslant \cdots \leqslant a_n$. Before scheduling, we also sort their deadlines such that $d_{h_1} \leqslant d_{h_2} \leqslant \cdots \leqslant d_{h_n}$ where $h_1, h_2, \ldots, h_n$ is a permutation of $1, 2, \ldots, n$. Let $T = d_{h_n}$ denote the largest deadline in the sequence.

Given sequences $0 \leqslant a_1 \leqslant a_2 \leqslant \cdots \leqslant a_n$, $0 \leqslant d_{h_1} \leqslant d_{h_2} \leqslant \cdots \leqslant d_{h_n} = T$, and associated B_i, $1 \leqslant i \leqslant n$, the data set to be transmitted is completely defined. If a packet arrives at time t, then we say that *an arrival event* occurs at t and the t is called *an arrival* (*event*) *point*. Similarly, if a deadline occurs at time t, then we say that *a deadline event* occurs at t, and the t is called *a deadline* (*event*) *point*. Therefore, from $t = 0$ to T, there are $2n$ events in total. An arrival point may also be a deadline point. The number m of distinct event points may be less than $2n$, $m \leqslant 2n$, if multiple events occur at the same time. We assume that the m distinct event points e_k, $1 \leqslant k \leqslant m$, are sorted in the order they occur, $0 \leqslant e_1 < e_2 < \cdots < e_m$. Obviously, $e_1 = a_1$, $e_m = d_{h_n} = T$. The time interval between two adjacent event points is called an epoch, and epoch $E_k = [e_k, e_{k+1})$, $1 \leqslant k \leqslant m - 1$, is said to have rank k in the epoch sequence. Fig. 1 shows an example of a data set for $n = 4$ where $P_1 = (10\text{K}, 2, 6)$, $P_2 = (8\text{K}, 3, 12)$, $P_3 = (20\text{K}, 5, 9)$, $P_4 = (7\text{K}, 7, 11)$, $T = 12$, $m = 8$. The 7 epochs are $E_1 = [2, 3)$, $E_2 = [3, 5)$, $E_3 = [5, 6)$, $E_4 = [6, 7)$, $E_5 = [7, 9)$, $E_6 = [9, 11)$, $E_7 = [11, 12)$.

We use function ξ to map each arrival or deadline point to its rank in the event sequence. Thus, if $a_i = e_k$ then $\xi(a_i) = k$. Similarly, if $d_{h_i} = e_k$ then $\xi(d_{h_i}) = k$. Function ξ is easy to obtain and known before scheduling.

Obviously, this model for delay constraints is the most general model with no restrictions on the order of events and the size of packets. All previous packet models in literature can be recovered as special cases of this model.

2.2. The system model

Following the same model used by previous researches [1–5,7,8,10,11,13,14], we consider a single point to point transmission channel and make the same assumption that the transmitter can adaptively change its transmission rate r, which is related to transmission power p through a function $r = g(p)$. The function $g(p)$ is called the *power-rate* function and satisfies the convex property.

Definition 1 [13]. A *power-rate* function $g(p)$ is said to satisfy *the convex property* if it satisfies the following 3 conditions:

(i) $g(0) = 0$ and $g(p) \to \infty$ as $p \to \infty$;
(ii) $g(p)$ increases monotonically and strictly concave in p;
(iii) $g(p)$ is continuously differentiable.

The convex property is satisfied in many systems with realistic encoding/decoding schemes, such as the optimal random coding in single-user additive White Gaussian Noise (AWGN) channel, where $g(p) = \frac{1}{2}\log(1 + p/N)$, N is the thermal noise level and often assumed $N = 1$ [8,13].

Definition 2. The *packet transmission rate function* $r_i(t)$ of packet P_i,: $\mathbf{R}^+ \to \mathbf{R}^+$ is defined as the transmission rate for packet P_i at time t, $0 \leqslant t < T$, $1 \leqslant i \leqslant n$.

By causality constraints, the packet transmission rate function must satisfy the following equation for $1 \leqslant i \leqslant n$.

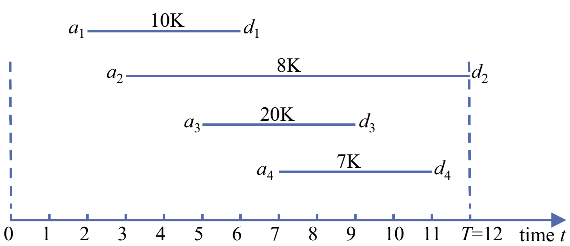


Fig. 1. An example of a data set with $n = 4$.

$$\int_0^T r_i(t)dt = \int_{a_i}^{d_i} r_i(t)dt = B_i \tag{1}$$

Definition 3. Given a set of n packet transmission rate functions, $r_i(t)$, $0 \leqslant t < T$, $1 \leqslant i \leqslant n$, the *overall rate function* $r(t)$ is defined as the sum of all packet transmission rate functions, that is $r(t) = \sum_{i=1}^{n} r_i(t)$, $0 \leqslant t < T$.

According to the transmission model, we have $r(t) = g(p(t))$, $0 \leqslant t < T$. Obviously, an overall rate function $r(t)$, $0 \leqslant t < T$, uniquely defines a transmission rate policy for the time interval $[0, T)$. Given an overall rate function $r(t)$, the corresponding total energy consumption can be calculated by the following integration [11].

$$E = \int_0^T g^{-1}(r(t))dt \tag{2}$$

2.3. Problem formulation

Given a data set as described above, we need to find an optimal feasible rate policy to satisfy the causality constraints. Let us define a feasible solution first.

Definition 4. Given a data set of n packets, $P_i = (B_i, a_i, d_i)$, $1 \leqslant i \leqslant n$, and a system model described above, a set S of n packet transmission rate functions, $S = \{r_i(t), 0 \leqslant t < T | 1 \leqslant i \leqslant n\}$ is called a feasible solution if Eq. (1) is satisfied for each $r_i(t)$, $1 \leqslant i \leqslant n$.

Definition 5. Given a data set of n packets, $P_i = (B_i, a_i, d_i)$, $1 \leqslant i \leqslant n$, and a system model described above, a feasible solution S is called optimal if its overall rate function $r(t)$, $0 \leqslant t < T$, minimizes the energy consumption defined by (2). Such an overall rate function $r(t)$, $0 \leqslant t < T$, is called an *optimal overall rate function*, denoted by $r^{opt}(t)$, $0 \leqslant t < T$.

Now, the problem we will study can be defined as follows.

Definition 6. The *offline energy efficient packet transmission problem* is to find an optimal feasible solution $S = \{r_i(t), 0 \leqslant t < T | 1 \leqslant i \leqslant n\}$ for a given data set of n packets, $P_i = (B_i, a_i, d_i)$, $1 \leqslant i \leqslant n$.

3. The Densest Interval First (DIF) policy

Before we introduce *Densest Interval First* (DIF) policy, we need to discuss some basic properties that an optimal transmission rate policy, $r^{opt}(t)$, $0 \leqslant t < T$, must have. Some of them have been known from previous research, some are new.

3.1. Basic properties of an optimal rate policy

It is easy to see [13] that, in any epoch $[e_k, e_{k+1})$ $1 \leqslant k \leqslant m-1$, only one transmission rate should be used because of the convexity of the *power-rate* function. If two rates $r_1 < r_2$ were used, we can always find a single rate r, $r_1 < r < r_2$, to transmit the same amount of data with less energy. This method is called *equalization*. Therefore, finding an optimal rate policy is to find a constant rate for each epoch such that the total energy used is minimized and all deadlines are guaranteed. The equalization is the most important notion for designing optimal rate control policy. The reader is suggested to refer [13] to get familiar with this notion.

Because of the causality constraints, from $t = 0$ to any time $t > 0$ the total amount of delivered data could not exceed the total amount of arrived data, and should not be less than the amount of data whose deadlines have expired. Thus, the following inequality must hold:

$$\forall t \in [0, T), \quad \sum_{d_i \leqslant t} B_i \leqslant \int_0^t r^{opt}(x)dx \leqslant \sum_{a_i < t} B_i \tag{3}$$

Now, the following two lemmas introduce some more properties that an optimal rate policy must have.

Lemma 1. *Any optimal overall rate function $r^{opt}(t)$, $0 \leqslant t < T$, increases rate only at an arrival point and decreases rate only at a deadline point.*

Proof. We have already explained that an optimal rate policy $r^{opt}(t)$, $0 \leqslant t < T$, can change rate only at event points. Now, for the sake of contradiction, we assume the optimal rate policy increases rate at an event point e_j, but no packet arrives at e_j. Obviously, $j > 1$, since $e_1 = a_1$ is an arrival point. Suppose the policy uses rate r_1 in the epoch $[e_{j-1}, e_j)$ and uses rate r_2 for the epoch $[e_j, e_{j+1})$, and $r_2 > r_1$. Then, we can equalize them to use a rate r, $r_1 < r < r_2$, to transmit the same amount of data in interval $[e_{j-1}, e_{j+1})$ with less energy. That is, some amount of data previously transmitted in epoch $[e_j, e_{j+1})$ is moved to be transmitted in the earlier epoch $[e_{j-1}, e_j)$. By doing so, no deadline will be missed, because we transmit more data in an earlier epoch; no data will be transmitted before its arrival either because they all arrived before or at e_{j-1}. This change does not cause any violation of causality constraint, but reduces the energy consumption. This contradicts the optimality of $r^{opt}(t)$.

Similarly, for the sake of contradiction, if the policy uses rate r_1 in epoch $[e_{i-1}, e_i)$ but decreases the rate to r_2 ($< r_1$) for epoch $[e_i, e_{i+1})$, while no deadline event is at e_i, then we can equalize them to use a rate r, $r_1 > r > r_2$, to transmit the same amount of data in interval $[e_{i-1}, e_{i+1})$ with less energy. That is, some amount of data previously transmitted in epoch $[e_{i-1}, e_i)$ is moved to be transmitted in later epoch $[e_i, e_{i+1})$. By doing so, no data will be transmitted before its arrival, no deadline will be missed either because they all have a deadline after or at e_{j+1}. Thus this change does not cause any violation of causality constraint, but reduces the energy consumption. This is also a contradiction. □

Here we would like to acknowledge that previous work [11] presented similar lemmas to Lemma 1 without proof for a previous packet model. Since it is not so obvious in our more general case, a formal proof here would be helpful.

Lemma 2. *Let $P_k = (B_k, a_k, d_k)$ be any packet transmitted according to an optimal rate policy $r^{opt}(t)$. Let H be the set of all epochs contained in time interval $[a_k, d_k)$, and $H' \subseteq H$ be the subset of H in which $r_k(t) \neq 0$. The following two statements are true:*

(1) *The overall rate $r^{opt}(t)$ used for any epoch of H' must be the same rate r.*
(2) *The overall rate $r^{opt}(t)$ used for any epoch of $H - H'$ must be higher or equal to the rate r.*

Proof. We prove (1) by contradiction. Suppose two different rates, $r_1 < r_2$, are used for epoch 1 and epoch 2, respectively, in set H'. Then, we use the method *equalization* to move certain amount of data of P_k that are transmitted in epoch 2 to epoch 1. By doing so, we equalize the rate in both epochs and reduce the total energy consumption, which contradicts to the optimality of rate $r^{opt}(t)$. Now, we prove (2). Suppose there is an epoch x in $H - H'$ for which the rate $r^{opt}(t) < r$, then we can remove certain amount of data of P_k that are transmitted in an epoch $y \in H'$ and transmit them in epoch x. By doing so, we reduce the rate for epoch y, increase the rate for epoch x, and reduce the total energy consumption, while keeping the deadlines guaranteed. This contradicts the optimality of rate $r^{opt}(t)$. Therefore, (2) must be true also. □

3.2. Data intervals and densest data interval

In this subsection, we introduce the key notions, namely the *data interval* and *densest data interval*. To better sense these notions, let us first briefly outline the *DIF* policy. It works in iterations. In each iteration, the *DIF* policy does three things:

(1) Identify a set S of unassigned packets and assign a set E of currently available epochs to them.
(2) Assign a single transmission rate r to every epoch in E which will be exclusively used for transmitting packets of set S.
(3) Mark all packets in S "*assigned*", mark all epochs in E "*unavailable*" to remaining unassigned packets.

Detailed discussions will be given in the next subsection. Now, we define data interval and densest (data) interval.

Definition 7. Given a data set $P_k = (B_k, a_k, d_k)$, $1 \leqslant k \leqslant n$, a *data interval* $I[i,j]$ is defined to be the time interval from the arrival time a_i to deadline d_{h_j}. That is, $I[i,j) = [a_i, d_{h_j})$, if $a_i < d_{h_j}$, $1 \leqslant i, j \leqslant n$, otherwise it is undefined.

Because multiple packets may share a common arrival point or a common deadline point, we may have redundant data intervals, $I[i,j) = I[u,v)$ while $i \neq u$ and/or $j \neq v$. For example, if $a_1 = 5$, $a_2 = 7$, $a_3 = 7$, $d_1 = 9$, $d_2 = 9$, $d_3 = 12$, then we have $d_{h_1} = 9, d_{h_2} = 9, d_{h_3} = 12$. The data intervals are:

$I[1,1) = [5,9)$, $I[1,2) = [5,9)$, $I[1,3) = [5,12)$,
$I[2,1) = [7,9)$, $I[2,2) = [7,9)$, $I[2,3) = [7,12)$,
$I[3,1) = [7,9)$, $I[3,2) = [7,9)$, $I[3,3) = [7,12)$.

The redundancy will not hurt our policy at all because once any of the redundant intervals has been chosen in an iteration, other redundant intervals will be updated to have empty packet sets. With empty packet set, a data interval becomes inactive. Details will be discussed in the next section.

Definition 8. For each data interval $I[i,j)$, we define four variable parameters as follows which may dynamically change during the rate determination process.

(1) Its *data set* $S[i,j]$ is the set of packets whose waiting intervals are contained inside $I[i,j)$ and have not been assigned epochs yet. Initially, $S[i,j] = \{P_k | [a_k, d_k) \subseteq I[i,j)\}$.
(2) Its *data load* $B[i,j]$ is the total amount of data contained in $S[i,j]$, that is $B[i,j] = \sum_{P_k \in S[i,j]} B_k$.
(3) Its *available time length* $L[i,j]$ is the total amount of time of all epochs in interval $I[i,j)$ that are currently available. $L[i,j] = (d_{h_j} - a_i)$ initially.
(4) Its *density* $D[i,j]$ is defined as $D[i,j] = \frac{B[i,j]}{L[i,j]}$ if $L[i,j] > 0$ and $D[i,j] = 0$ otherwise.

Note that we use left-closed, right-open interval notations only for time intervals, not for parameters. Let us look at an example. Given the data set shown in Fig. 1, the 4 initial parameters of each data interval are shown in Table 1.

Table 1
Parameters for data intervals of packet set of Fig. 1.

$S[i,j]$	$d_{h_1} = d_1$ 6	$d_{h_2} = d_3$ 9	$d_{h_3} = d_4$ 11	$d_{h_4} = d_2$ 12
$a_1 = 2$	$P_1(10k,2,6)$	$P_1(10k,2,6)$ $P_3(20k,5,9)$	$P_1(10k,2,6)$ $P_3(20k,5,9)$ $P_4(7k,7,11)$	$P_1(10k,2,6)$ $P_3(20k,5,9)$ $P_4(7k,7,11)$ $P_2(8k,3,12)$
$a_2 = 3$	$\varnothing$	$P_3(20k,5,9)$	$P_3(20k,5,9)$ $P_4(7k,7,11)$	$P_3(20k,5,9)$ $P_4(7k,7,11)$ $P_2(8k,3,12)$
$a_3 = 5$	$\varnothing$	$P_3(20k,5,9)$	$P_3(20k,5,9)$ $P_4(7k,7,11)$	$P_3(20k,5,9)$ $P_4(7k,7,11)$
$a_4 = 7$	Undefined	$\varnothing$	$P_4(7k,7,11)$	$P_4(7k,7,11)$
$B[i,j]$ $L[i,j]$ $D[i,j]$	$d_{h_1} = d_1$ 6	$d_{h_2} = d_3$ 9	$d_{h_3} = d_4$ 11	$d_{h_4} = d_2$ 12
$a_1 = 2$	$B = 10k$ $L = 6 - 2 = 4$ $D = 10/4 = 2.5$	$B = 30k$ $L = 9 - 2 = 7$ $D = 30/7 = 4.286$	$B = 37k$ $L = 11 - 2 = 9$ $D = 37/9 = 4.111$	$B = 45k$ $L = 12 - 2 = 10$ $D = 45/10 = 4.5$
$a_2 = 3$	$B = 0$ $L = 6 - 3 = 3$ $D = 0$	$B = 20k$ $L = 9 - 3 = 6$ $D = 20/6 = 3.333$	$B = 27k$ $L = 11 - 3 = 8$ $D = 27/8 = 3.375$	$B = 35k$ $L = 12 - 3 = 9$ $D = 35/9 = 3.889$
$a_3 = 5$	$B = 0$ $L = 6 - 5 = 1$ $D = 0$	$B = 20k$ $L = 9 - 5 = 4$ $D = 20/4 = 5$	$B = 27k$ $L = 11 - 5 = 6$ $D = 27/6 = 4.5$	$B = 27k$ $L = 12 - 5 = 7$ $D = 27/7 = 3.857$
$a_4 = 7$	Undefined	$B = 0$ $L = 9 - 7 = 2$ $D = 0$	$B = 7k$ $L = 11 - 7 = 4$ $D = 7/4 = 1.75$	$B = 7k$ $L = 12 - 7 = 5$ $D = 7/5 = 1.4$

Theorem 1 (*Basic Densest Interval Theorem*). *Given a data set $P = \{P_i | 1 \leqslant i \leqslant n\}$, where $P_i = (B_i, a_i, d_i)$, if the density $D[i,j]$ of data interval $I[i,j)$ is the largest among all data intervals, then $I[i,j)$ is called the densest interval, and the following statements are true.*

(1) *Any optimal rate policy must assign rate $r = D[i,j]$ to every epoch in time interval of $[a_i, d_{h_j})$.*
(2) *Any optimal solution must deliver exactly the packet set $S[i,j]$ during the time interval of $[a_i, d_{h_j})$.*

Proof. We prove (1) by contradiction. Let $r^{opt}(t)$ be the rate used by an optimal rate policy for the time interval $[a_i, d_{h_j})$. Suppose $r^{opt}(t) \neq D[i,j] = r$ in some epoch in $[a_i, d_{h_j})$. We claim that there must be at least one epoch $[e_k, e_{k+1}) \subseteq [a_i, d_{h_j})$ such that the $r^{opt}(t)$ in this epoch is r_{ek} and $r_{ek} > r$. This is because, otherwise we would have $r^{opt}(t) \leqslant r$ for entire interval $[a_i, d_{h_j})$ and $r^{opt}(t) < r$ for some epoch in $[a_i, d_{h_j})$, which implies that $\int_{a_i}^{d_{h_j}} r^{opt}(x)dx < r \times (d_{h_j} - a_i) = B[i,j]$ and some data would miss their deadlines. Let $r^{opt}(t) = r_{ek} > r$ in epoch $[e_k, e_{k+1}) \subseteq [a_i, d_{h_j})$. We extend $[e_k, e_{k+1})$ to a larger interval $[e_u, e_v)$. Let $[e_u, e_v)$ $(\supseteq [e_k, e_{k+1}))$ be the longest time interval in which every epoch has the rate $r^{opt}(t) \geqslant r$. Note that $[e_u, e_v)$ may not contain $[a_i, d_{h_j})$, or vice versa.

Obviously, $r^{opt}(t)$ increases rate at e_u and decreases rate at e_v, for otherwise we could extend to an even larger interval. By Lemma 1, e_u must be an arrival point and e_v must be a deadline point. So, $[e_u, e_v)$ is a data interval and its density is no *larger* than r because r is the densest one. We have $r^{opt}(t) \geqslant r$ for entire interval $[e_u, e_v)$ and $r^{opt}(t) > r$ for epoch $[e_k, e_{k+1}) \subseteq [e_u, e_v)$, thus $\int_{e_u}^{e_v} r^{opt}(x)dx > \int_{e_u}^{e_v} r dx \leqslant B[u, v] = \sum_{[a_k, d_k) \subseteq [e_u, e_v)} B_k$, the optimal policy must have transmitted more data than $B[u, v]$ in the time interval $[e_u, e_v)$. Obviously, $[e_u, e_v) \neq [0, T)$ for otherwise the optimal policy would transmit more data than the total load of all packets, which is impossible. Thus, the optimal policy must have transmitted a packet P_x that arrived before e_u or have a deadline larger than e_v in time interval $[e_u, e_v)$. This contradicts Lemma 2. Therefore, any optimal rate policy must use a single rate $r = D[i,j]$ for the time interval $[a_i, d_{h_j})$. Part (1) is proved.

Part (2) of the theorem directly follows from part (1) because rate $r = D[i,j]$ is just enough to finish all packets in $S[i,j]$. □

If $I[i,j) = [a_1, T)$, then our job is done. Otherwise, we need to continue to find an optimal rate policy for the remaining intervals $([0,T) - I[i,j)) = [0, a_i) \cup [d_{h_j}, T)$ for transmitting the remaining set of packets $P' = P - S[i,j]$. We can see that the same problem occurs if we treat the remaining set of packets just like a new set of packets. The only difference is that, this time, the epochs in $I[i,j)$ are not allowed to use, because it has been assigned to $S[i,j]$ already.

3.3. Densest Interval First (DIF) policy

As we pointed out, the *DIF* policy computes transmission rate in iterations. Specifically, in each iteration, it does the following:

(1) Re-compute the data set $S[i,j]$ for every interval $I[i,j)$, $1 \leqslant i, j \leqslant n$. The data set consists of all packets whose waiting intervals are inside $[a_i, d_{h_j})$ and currently remain *unassigned*.
(2) Re-compute the data load $B[i,j]$, the available time length $L[i,j]$, and the density $D[i,j]$ for every interval $I[i,j)$, $1 \leqslant i, j \leqslant n$.
(3) Find the densest interval $I[i,j)$, and assign the single rate $r = D[i,j]$ to every currently available epoch in $I[i,j)$ which will be exclusively used by $S[i,j]$.
(4) Mark all epochs in $I[i,j)$ to be *unavailable* to next iteration. Mark all packets in $S[i,j]$ *assigned*.

It is assumed that $r = 0$ in time interval $[0, a_1)$.

Definition 9. A data interval $I[i,j)$ is defined as *active* if it has a non-empty data set $S[i,j]$, otherwise, it is *inactive*.

During execution of *DIF* policy, packets always change from *unassigned* to *assigned*; epochs always change from *available* to *unavailable*; date intervals always change from *active* to *inactive*. The *DIF* algorithm stops when all data intervals become *inactive*.

We use $M[i,j] = 1$ and $M[i,j] = 0$ to denote data interval $I[i,j)$ being active and inactive, respectively. We use $R[i] = 0$, $1 \leqslant i \leqslant n$, to denote that packet P_i has been assigned and $R[i] = 1$ otherwise. We use $E[k] = 0$, $1 \leqslant k \leqslant m - 1$, to denote that kth epoch is unavailable if it has been assigned to a packet set and $E[k] = 1$ if it is still available to remaining packets. Both $R[i]$ and $E[k]$ need be updated after each iteration. Initially, $R[i] = 1$, $1 \leqslant i \leqslant n$, $E[k] = 1$, $1 \leqslant k \leqslant m - 1$. Major notations used in this paper are summarized in Table 2 for the reader's convenience.

The following is the pseudo code for updating $R[i]$, $1 \leqslant i \leqslant n$, and $E[k]$, $1 \leqslant k \leqslant m - 1$, if data interval $I[i,j)$ has been found to have the largest density in an iteration. Basically, we will mark those packets whose waiting intervals are inside $[a_i, d_{h_j})$ with "assigned" and those epochs inside $[a_i, d_{h_j})$ with "unavailable."

Availability-Update($E[\,]$, $R[\,]$, $I[i,j)$)

```
for k ← 1 to m − 1 do
    if [e_k, e_{k+1}) ⊆ [a_i, d_{h_j}) then
        E[k] ← 0 //It is possible that E[k] = 0 already
    endif
endfor
for i = 1 to n do
    if [a_i, d_i) ⊆ [a_i, d_{h_j}) then
        R[i] ← 0 //It is possible that R[i] = 0 already
    endif
endfor
```

Obviously, the time complexity of *Availability-Update* is $O(n)$. Let us continue the example of Fig. 1. From Table 1, we find the densest interval is $I[3,2) = [5,9)$ which has density 5. Therefore, we assign packet P_3 with transmission rate $r = 5$ and we also assign epochs $E_3 = [5,6)$ $E_4 = [6,7)$ and $E_5 = [7,9)$ to packet P_3. After the assignment, we set $R[3] = 0$ which means P_3 has been assigned with a rate and should be excluded from remaining packet set. We also set $E[3] = E[4] = E[5] = 0$, which means that epochs 3,

Table 2
Major notations and their explanations.

Notation	Explanation
P_k	The k-th packet, $P_k = (B_k, a_k, d_k)$, with size B_k, arrival time a_k, and deadline d_k
E_k	The k-th epoch, $E_k = [e_k, e_{k+1})$, the time interval between event points e_k and e_{k+1}
$R[k]$	=0, packet P_k is *assigned* with epochs, or =1, packet P_k is *unassigned* with epochs
$E[k]$	=0, epoch E_k is *unavailable*, or =1, epoch E_k is *available* to remaining packets
$I[i,j]$	$=[a_i, d_{h_j})$, the time interval from the arrival time a_i to the deadline d_{h_j}
$S[i,j]$	The set of *unassigned* packets whose waiting intervals are contained inside $I[i,j)$
$B[i,j]$	The total amount of data contained in $S[i,j]$
$L[i,j]$	The total amount of time of all epochs in interval $I[i,j)$ that are currently available
$D[i,j]$	$=B[i,j]/L[i,j]$, the data density of $I[i,j)$
$M[i,j]$	=0, data interval $I[i,j)$ is *inactive*, or =1, data interval $I[i,j)$ is *active*
$T[i,j]$	The set of *available* epochs in interval $I[i,j)$

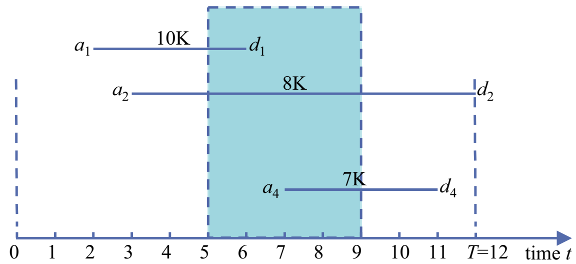


Fig. 2. An illustration of remaining packets and available epochs after applying the first iteration by *DIF-Policy* on the packet set of Fig. 1. Epochs in shaded area are not available for remaining packets.

4, and 5 are not available to remaining packets. Finally, we should set $M[3,2] = 0$, which means interval $I[3,2)$ becomes inactive. Once we have found an interval has empty packet set or undefined, we set this interval to be inactive. Fig. 2 shows the remaining unassigned packets and available epochs to these packets after the first iteration, where the time covered by shaded area is not allowed to use for the remaining packets. As shown in Fig. 2, there are 3 packets remaining. Since they cannot use the shaded area, P_1 must finish by time 5, although its deadline is 6; P_4 cannot start transmission until time 9, although it arrives at 7; P_2 can use time intervals $[3,5)$ and $[9,12)$ so that it may be divided into two segments to transmit accordingly.

As we can conclude from Fig. 2, after each iteration, we need to re-compute the parameters for each active interval because the set of remaining unassigned packets and the set of available epochs change after each iteration. For this updating, we follow the order of $I[i,1), I[i,2), \ldots, I[i,n)$, $i = 1, 2, \ldots, n$. This order allows us to apply an efficient greedy approach when we go from $I[i,j)$ to $I[i,j+1)$.

Because $I[i,j) = [a_i, d_{h_j})$ and $I[i,j+1) = [a_i, d_{h_{j+1}})$, we have $I[i,j) \subseteq I[i,j+1)$. Thus, we have the following two sequences of *containment* relations:

$$I[i,1) \subseteq I[i,2) \subseteq \cdots \subseteq I[i,j) \subseteq I[i,j+1) \subseteq \cdots \subseteq I[i,n) \tag{4}$$

and

$$S[i,1] \subseteq S[i,2] \subseteq \cdots \subseteq S[i,j] \subseteq S[i,j+1] \subseteq \cdots \subseteq S[i,n] \tag{5}$$

Moreover, we also have the following sequence of *less than or equal to* relations:

$$L[i,1] \leqslant L[i,2] \leqslant \cdots \leqslant L[i,j] \leqslant L[i,j+1] \leqslant \cdots \leqslant L[i,n] \tag{6}$$

Based on above observations, the parameters for all intervals can be computed efficiently by the following procedure *Density-Update*. This procedure will mark $M[i,j] = 1$ if interval $S[i,j]$ is not empty, and mark $M[i,j] = 0$ for other cases. However, parameters $S[i,j]$ and $B[i,j]$ will be computed for any interval $I[i,j)$ even if it is marked with $M[i,j] = 0$ because we need to extend them through entire sequence, from $S[i,1]$ to $S[i,n]$ and from $B[i,1]$ to $B[i,n]$. It is easy to see that undefined intervals always occur in the beginning segment of the sequence (4). We set $L[i,j] = 0$ and $D[i,j] = 0$ if $I[i,j)$ is undefined.

Density-Update($E[\,]$, $R[\,]$, n)

for $i \leftarrow 1$ **to** n **do** // initialization loop
  for $j \leftarrow 0$ **to** n **do** //j starts from 0 for looping purpose
    $M[i,j] \leftarrow 0$ // initially, every $I[i,j)$ is *inactive*
    $S[i,j] \leftarrow \varnothing$ //$\varnothing$ = empty
    $B[i,j] \leftarrow 0$
    $L[i,j] \leftarrow 0$
  endfor
endfor
for $i \leftarrow 1$ **to** n **do**
  $d_{h_0} \leftarrow a_i$ //Dummy d_{h_0} is for looping purpose
  for $j \leftarrow 1$ **to** n **do**
    if $R[h_j] = 1$ **and** $a_{h_j} \geqslant a_i$ **then**
      //P_{h_j} is unassigned and $[a_{h_j}, d_{h_j}) \subseteq I[i,j)$
      $S[i,j] \leftarrow S[i,j-1] \cup \{P_{h_j}\}$
      $B[i,j] \leftarrow B[i,j-1] + B_{h_j}$
    else
      $S[i,j] \leftarrow S[i,j-1]$
      $B[i,j] \leftarrow B[i,j-1]$
    endif
    $L[i,j] \leftarrow L[i,j-1]$ //Start adding new epochs
    for $k \leftarrow \max\{\xi(d_{h_{j-1}}), \xi(a_i)\}$ **to** $\xi(d_{h_j}) - 1$ **do**
      if $E[k] = 1$ **then** //Epoch k is available
        $L[i,j] \leftarrow L[i,j] + (e_{k+1} - e_k)$
      endif
    endfor
  if $S[i,j] \neq \varnothing$ **then**
    $M[i,j] \leftarrow 1$ // $I[i,j)$ is active
    $D[i,j] = B[i,j]/L[i,j]$
  endif
  endfor
endfor
End

Clearly, the procedure *Density-Update* handles the case of redundant intervals smoothly. Let us continue the example of Fig. 1. After applying the *Density-Update* to the remaining set of packets after the first iteration, the densities of all active intervals are shown in Table 3. *Density-Update* can also be used to compute parameters of all data intervals prior to the first iteration.

It is not difficult to see that the time complexity for *Density-Update* is $O(n^2)$. First the initialization takes $O(n^2)$ steps. Second, for each i in the loop at line 9, the index j runs from 1 to n. Moreover, for each index j, the **for** loop at line 20 only checks epochs between $d_{h_{j-1}}$ and d_{h_j}. Therefore, each epoch is checked at most once through entire loop for all values of j. Since there are $m-1<2n$ epochs, only $O(n)$ time for checking is needed for each index i. Therefore, the time complexity of *Density-Update* is $O(n^2)$.

Based on *Availability-Update* and *Density-Update*, the algorithm of *DIF* policy is presented in the following. We assume the input to the algorithm consists of:

(1) $P_i=(B_i,a_i,d_i),\ 1\leqslant i\leqslant n$.
(2) $0\leqslant a_1\leqslant a_2\leqslant\cdots\leqslant a_n,\ 0\leqslant d_{h_1}\leqslant d_{h_2}\leqslant\cdots\leqslant d_{h_n}=T$.
(3) $0\leqslant e_1\leqslant e_2\leqslant\cdots\leqslant e_m$.
(4) Function ξ that maps each arrival time a_i or deadline d_{h_j} to its epoch rank.

DIF-Policy$(P[\,],\ n,\ m,\ \xi)$

```
1  for k ← 1 to m − 1 do
2    E[k] ← 1      //Mark all epochs available
3  endfor
4  for k ← 1 to n do
5    R[k] ← 1      //Mark all packets unassigned
6  endfor
7  Density-Update(E[], R[], n)
8  find densest active interval I[i,j]
     // D[i,j] = max{D[u,v]|M[u,v] = 1}
9  while D[i,j] > 0 do
10    T[i,j] ← ∅      //set of epochs assigned to S[i,j]
11    for u ← ξ(a_i) to ξ(d_{h_j}) − 1
12      if E[u] = 1 then
13        T[i,j] ← T[i,j] ∪ {epoch E_u}
14      endif
15    endfor
16    assign rate r = D[i,j] to all epochs of T[i,j]
17    Availability-Update(E[], R[], I[i,j])
18    Density-Update(E[], R[], n)
19    find densest active interval I[i,j]
     // D[i,j] = max{D[u,v]|M[u,v] = 1}
20  endwhile
21  for k ← 1 to m − 1 do
22  if E[k] ← 1 then
       //All remaining unused available epochs
23    assign rate r = 0 to epoch E_k
24  endif
25  endfor
26  End
```

For the example of Fig. 1, we observe from Table 3 that the densest interval is $I[1,4]$ in the second iteration. Since $S[1,4]=\{P_1,P_4,P_2\}$, we assign epochs $E_1=[2,3)$, $E_2=[3,5)$, $E_6=[9,11)$, and $E_7=[11,12)$ to packet set $\{P_1,P_4,P_2\}$, using transmission rate $r=25/6=4.167$. Obviously, the *DIF-Policy* for this example finishes after two iterations.

Table 3
The updated densities of active intervals after the first iteration for the packet set of Fig. 1.

$S[i,j]$	$d_{h_1}=d_1$ 6	$d_{h_2}=d_3$ 9	$d_{h_3}=d_4$ 11	$d_{h_4}=d_2$ 12
$a_1=2$	$P_1(10k,2,6)$	$P_1(10k,2,6)$	$P_1(10k,2,6)$ $P_4(7k,7,11)$	$P_1(10k,2,6)$ $P_4(7k,7,11)$ $P_2(8k,3,12)$
$a_2=3$	$\varnothing$	$\varnothing$	$P_4(7k,7,11)$	$P_2(8k,3,12)$ $P_4(7k,7,11)$
$a_3=5$	$\varnothing$	$\varnothing$	$P_4(7k,7,11)$	$P_4(7k,7,11)$
$a_4=7$	Undefined	$\varnothing$	$P_4(7k,7,11)$	$P_4(7k,7,11)$
$B[i,j]$ $L[i,j]$ $D[i,j]$	$d_{h_1}=d_1$ 6	$d_{h_2}=d_3$ 9	$d_{h_3}=d_4$ 11	$d_{h_4}=d_2$ 12
$a_1=2$	$B=10k$ $L=1+2=3$ $D=10/3=3.333$	$B=10k$ $L=3$ $D=10/3=3.333$	$B=17k$ $L=3+2=5$ $D=17/5=3.4$	$B=25k$ $L=5+1=6$ $D=25/6=4.167$
$a_2=3$	$M=0$ inactive	$M=0$ inactive	$B=7\,k$ $L=2+2=4$ $D=7/4=1.75$	$B=15\,k$ $L=4+1=5$ $D=15/5=3$
$a_3=5$	$M=0$ inactive	$M=0$ inactive	$B=7\,k$ $L=2$ $D=7/2=3.5$	$B=7\,k$ $L=2+1=3$ $D=7/3=2.333$
$a_4=7$	$M=0$ inactive	$M=0$ inactive	$B=7\,k$ $L=2$ $D=7/2=3.5$	$B=7\,k$ $L=2+1=3$ $D=7/3=2.333$

Because there are n packets and at least one packet becomes assigned by each iteration of the **while** loop at line 9, the *DIF-Policy* needs at most n iterations to finish. Since procedures *Availability-Update* and *Density-Update* needs $O(n^2)$ steps, the *DIF-Policy* has time complexity of $O(n^3)$.

Before we prove its correctness, we make few observations.

Observation 1. The rate assigned to $T[i,j]$ in line 16 is no larger than the rate assigned in the previous iteration in the **while** loop, because in every loop, the densest interval is removed by *Availability-Update*.

Observation 2. *DIF* actually partitions the n packets into several groups. When the densest interval $I[i,j]$ is identified in each iteration, all currently *unassigned* packets contained in $S[i,j]$ are separated from other *unassigned* packets. The packets in $S[i,j]$ are to be transmitted with the rate $r=D[i,j]$ during *available* epochs in $I[i,j]$.

Observation 3. The packets in $S[i,j]$ are transmitted during *available* epochs in $T[i,j]$ with the rate $r=D[i,j]$, where r is the lowest rate among all epochs in the entire interval $I[i,j]$ according to Observation 1. In other words, for any $S[i,j]$, its $T[i,j]$ is composed of epochs with lowest rate among all epochs in $I[i,j]$.

Observation 4. If data interval $I[u,v) \subset I[i,j)$ is an subinterval of $I[i,j)$ when $I[i,j)$ is identified to be the densest interval, then its density is less than or equal to the density of $I[i,j)$, that is, $D[u,v] \leqslant D[i,j]$, for otherwise, $I[i,j)$ should not be the densest.

The correctness of *DIF* policy will be proved if we can show that in line 16 of **while** loop, any optimal overall rate for *available* epochs in $T[i,j]$ must equal to $r = D[i,j]$. This claim is stated in Theorem 2 which is a generalization of Theorem 1.

Theorem 2 (*Generalized Densest Interval Theorem*). *In any iteration of the while loop in algorithm DIF-Policy, let $I[i,j)$ be the data interval such that $D[i,j] = max\{D[u,v]|M[u,v] = 1\}$. The following are true:*

(1) *Any optimal rate policy must assign rate $r = D[i,j]$ to every available epoch in $T[i,j]$.*
(2) *Any optimal solution must deliver exactly the set of packets in $S[i,j]$ during epochs of $T[i,j]$.*

Proof. We prove this by induction on iterations. By Theorem 1, the claim of Theorem 2 is true for the first iteration, which serves as the induction basis. Suppose Theorem 2 is true for the first k iterations, $k \geqslant 1$, we prove that it is also true for the $(k+1)$st iteration.

Let $I[i,j)$ be the data interval such that $D[i,j] = max\{D[u,v]|M[u,v] = 1\}$ at the beginning of the $(k+1)$st iteration. If $D[i,j] = 0$ then all packets must have been assigned a transmission rate in previous iterations and assigning rate $r = 0$ to every unused epoch at line 23 is the only choice for any optimal policy. So, we assume $D[i,j] > 0$. Let $T[i,j]$ be the set of all available epochs in time interval $I[i,j) = [a_i, d_{h_j})$ at the beginning of the $(k+1)$st iteration.

Suppose $r^{opt}(t)$, $a_i \leqslant t < d_{h_j}$, is the rate used by an optimal policy. By induction, $r^{opt}(t)$ equals to the rate given by the *DIF-Policy* for those unavailable epochs at the beginning of $(k+1)$st iteration.

If $r^{opt}(t) \neq D[i,j] = r$ in some epoch of $T[i,j]$, then we claim that there must be an epoch $[e_k, e_{k+1}) \subseteq T[i,j]$ such that $r^{opt}(t) > r$. Otherwise, we would have $r^{opt}(t) \leqslant r$ for all epochs in $T[i,j]$ and $r^{opt}(t) < r$ in some epoch, which implies $\sum_{[e_k,e_{k+1})\in T[i,j]} \int_{e_k}^{e_{k+1}} r^{opt}(t)dt < r \times L[i,j] = B[i,j]$ and some data would miss their deadlines. Suppose $r^{opt}(t) = r_{ek} > r$ is used in epoch $[e_k, e_{k+1}) \subseteq T[i,j]$. We extend time interval $[e_k, e_{k+1})$ to a larger interval. Let $[e_u, e_v)$ be the longest data interval such that $[e_k, e_{k+1}) \subseteq [e_u, e_v)$ and $r^{opt}(t) \geqslant r$ in $[e_u, e_v)$. Note that interval $[e_u, e_v)$ can include both available and unavailable epochs. $r^{opt}(t)$ increases rate at e_u and decreases rate at e_v.

By Lemma 1, e_u must be an arrival point a_x and e_v must be a deadline point d_{h_y}, thus, $[e_u, e_v)$ is a data interval $I[x,y)$ and its density $D[x,y]$ is no larger than r at the beginning of $(k+1)$st iteration, because r is the densest density. Obviously $I[x,y)$ is active at the beginning of $(k+1)$st iteration because it contains $[e_k, e_{k+1})$. Moreover, if time interval $[e_u, e_v)$ contains an unavailable epoch marked by a previous iteration $k' \leqslant k$ for some densest interval $I[x', y')$, then the time interval $I[x', y')$ must be entirely contained inside $I[x,y)$, that is, $I[x', y') \subseteq I[x,y)$, because all marked epochs have a higher rate $r' \geqslant r$ by Observation 1.

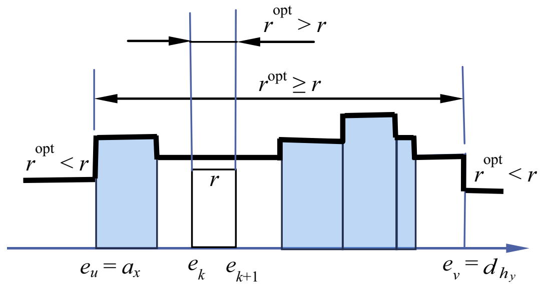


Fig. 3. An illustration of rate $r^{opt}(t)$ in interval $[e_u, e_v)$, where the shaded area represents epochs that have been assigned to packets in previous iterations and are unavailable to the $(k+1)$st iteration.

Fig. 3 shows the relation between intervals $[e_k, e_{k+1})$ and $[e_u, e_v)$ and the relation between $r^{opt}(t)$ and r in interval $[e_u, e_v)$.

We can divide all packets whose waiting intervals are inside $I[x,y)$ into two groups. The first group consists of those packets that have been assigned epochs and transmission rates by *DIF-Policy* before the $(k+1)$st iteration, and the second group consists of remaining unassigned packets at the beginning of $(k+1)$st iteration. Let $B^{(1)}$ and $B^{(2)}$ represent the data loads in these two groups, respectively. Then, at the beginning of $(k+1)$st iteration, the density of $I[x,y)$ is $D[x,y] = B^{(2)}/L[x,y]$, where $L[x,y]$ is the length of total available time in $T[x,y]$. Because $r^{opt}(t) \geqslant r$ in entire interval $[e_u, e_v)$ and $r^{opt}(t)$ is strictly larger than r in $[e_k, e_{k+1})$, the amount of data transmitted by the optimal rate in interval $[e_u, e_v)$ is:

$$B = B^{(1)} + \sum_{[e_k,e_{k+1})\in T[x,y]} \int_{e_k}^{e_{k+1}} r^{opt}(t)dt > B^{(1)} + r \times L[x,y].$$

Since the density $D[i,j] = r$ is the largest at the beginning of $(k+1)$st iteration, we have $r \times L[x,y] \geqslant D[x,y] \times L[x,y] = B^{(2)}$. Therefore, we have $B > B^{(1)} + B^{(2)}$, which implies that the optimal policy transmits some packet P_z whose arrival time a_z is smaller than a_x or whose deadline d_z is larger than d_{h_j}. Either case contradicts Lemma 2. Part (1) is proved.

Part (2) of the theorem directly follows from part (1) because rate $r = D[i,j]$ is just enough to finish all packets in $S[i,j]$. □

4. Optimal individual packet scheduling

Once the rate for the densest interval $I[i,j)$ is determined, we need to schedule the transmission time for each individual packet that belongs to the set $S[i,j]$. Inappropriate schedule may cause violation of causality constraint. Let us continue to look at the example of Fig. 1 from last section. After *DIF* policy has identified the intervals [2,5) and [9,12) that use rate 4.167 for transmitting P_1, P_2, and P_4, how should we schedule them? If we schedule them in the order of P_1, P_2, and P_4, one by one, continuously until finish, then some data in P_4 would miss the deadline. This problem becomes more complicated when we have many packets in $S[i,j]$. The *DIF-Policy* only determines the optimal transmission rate we should use for each epoch, we need to arrange actual transmission for individual packets.

Unless we can find a schedule that allows every individual packet to complete its transmission before its deadline using the rate determined by *DIF-policy*, we cannot claim that we have solved the optimization problem given in Definition 6. Fortunately, we have found that the well-known *EDF* (Earliest Deadline First) algorithm is an optimal algorithm for this job. We could apply the EDF algorithm for packet set $S[i,j]$ when *DIF* has identified the densest interval $I[i,j)$ in each iteration, but a nicer way is to apply *EDF* algorithm just once from $t = 0$ to $t = T$, continuously using the rate determined by *DIF-Policy* for each epoch. We present its pseudo code below.

EDF-Schedule(P, m)

```
build a min-heap H using deadlines as the keys.
   //Use arrival time to break a tie.
for i ← 1 to m − 1 do
  if e_i is an arrival point then
    insert those packets that arrive at e_i into H
  endif
  r ← r(E_i)      //the rate assigned to E_i by DIF-Policy
  s_t ← e_i       //Starting time for sending next packet
  while s_t < e_{i+1} and H ≠ ∅ do
    P_k ← packet at the root of H
     //Get the packet without extracting it from heap H
     f_t ← s_t + B_k/r      //expected finish time
     if f_t > e_{i+1} then f_t ← e_{i+1} endif
     transmit P_k at rate r in time interval [s_t,f_t)
     if r × (f_t − s_t) < B_k then
       B_k ← B_k − r × (f_t − s_t)   //remaining size of P_k
     else
       extract P_k from H
     endif
     s_t ← f_t   //Next starting time
   endwhile
 endfor
 End
```

Let us continue to discuss the example of Fig. 1. By the *DIF-Policy*, epochs have been given the following rates: $r[2,3) = r[3,5) = 4.167$, $r[5,6) = r[6,7) = r[7,9) = 5$, $r[9,11) = r[11,12) = 4.167$. The *EDF* takes the following steps:

1. At $t = 2$, P_1 has arrived, transmit P_1 in $[2,3)$ at rate 4.167, remaining size = 10 − 4.167 = 5.833.
2. At $t = 3$, insert P_2. P_1 is still at the root of H. $f_t = 3 + 5.833/4.167 = 3 + 1.4 = 4.4 < 5$. P_1 finishes at 4.4. Delete P_1.
3. At $t = 4.4$, P_2 starts transmission at rate 4.167. $f_t = 4.4 + 8/4.167 = 4.4 + 1.92 = 6.32$. Since $6.32 > 5$, transmit P_2 until $t = 5$. The remaining size = 8 − 4.167(5 − 4.4) = 8 − 2.5 = 5.5.
4. At $t = 5$, insert P_3. P_3 is at the root. $f_t = 5 + 20/5 = 9 > 6$, transmit P_3 at rate 5 to $t = 6$. The remaining size = 20 − 5 = 15.
5. At time 6, no insertion. Continue to transmit P_3 at rate 5. $f_t = 6 + 15/5 = 9 > 7$, transmit P_3 to $t = 7$. The remaining size = 15 − 5 = 10.
6. At $t = 7$, insert P_4, but P_3 is still at the root. Continue to transmit P_3 at rate 5. $f_t = 7 + 10/5 = 9$ = next event time. P_3 finishes at $t = 9$. Delete P_3.
7. At $t = 9$, no insertion. Because P_4 is at the root, transmit P_4 at rate 4.167. $f_t = 9 + 7/4.167 = 9 + 1.68 = 10.68 < 11$. P_4 finishes at $t = 10.68$. Delete P_4.
8. At $t = 10.68$, P_2 is at the root. $f_t = 10.68 + 5.5/4.167 = 10.68 + 1.32 = 12 > 11$, transmit P_2 to 11. The remaining size = 5.5 − 4.167(11 − 10.68) = 5.5 − 1.333 = 4.167.
9. At $t = 11$, no insertion. Continue to transmit P_2 to 12 at rate 4.167. Delete P_2.

In the rest of this section, we show the correctness of the *EDF-Schedule*.

Lemma 3. *For a DIF-Policy identified interval I[i,j), if the assigned epochs in T[i,j] are exclusively used only by packets of S[i,j] with rate D[i,j], then all packets in S[i,j] can finish transmission before or at their deadlines by the EDF-Schedule.*

Proof. Suppose for the sake of contradiction, packet $P_k = (B_k, a_k, d_k)$ is the first to miss its deadline d_k. According to *EDF-Schedule*, only those packets in $S[i,j]$ whose deadlines are d_k or earlier have been transmitted in $T[a_k, d_k]$, where $T[a_k, d_k] \subset T[i,j]$ is the set of available epochs in the time interval $[a_k, d_k)$. Moreover, the transmission in $T[a_k, d_k]$ must be continuous without stopping because there are still unfinished data in P_k at time d_k. Now, we extend this interval to $[t, d_k)$ by finding the earliest time $t \leqslant a_k$ such that in the available epochs in $[t, d_k)$, only packets of $S[i,j]$ with deadlines d_k or earlier have been transmitted and the transmission has been continuous. The time t must be an arrival event. (If t were a deadline event, we could find an even earlier t.) Let $t = a_u \leqslant a_k$. Further, we can see that in the time interval $[a_u,d_k)$, only packets arrived at a_u or later have been transmitted, for otherwise, we could further extend a_u to an even earlier time. Therefore, during the available epochs in $[a_u,d_k)$ the *EDF-Schedule* has continuously transmitted packets that arrived at a_u or later with deadlines d_k or earlier at rate $D[i,j]$, but still has missed the deadline of P_k. This implies that interval $[a_u, d_k)$ must have higher density than $D[i,j]$, contradicting Observation 4. □

Corollary. *For a DIF identified interval I[i,j), if the assigned epochs in T[i, j] are used exclusively by packets in S[i,j] only, then there are always sufficient data in S[i,j] ready to be transmitted at rate D[i,j].*

This observation is true because the available time in T[i,j] and rate D[i,j] assigned to S[i,j] are just enough to finish the data load B[i,j]. Since, by Lemma 3, no packet would miss its deadline, then the transmission in T[i,j] must be continuous without stopping, which implies there are always sufficient data in S[i,j] ready to be transmitted at rate D[i,j].

Theorem 3. *Given a data set of n packets, $P_i = (B_i, a_i, d_i)$, $1 \leqslant i \leqslant n$, the EDF-Schedule guarantees that all packets are transmitted before or at their deadlines with the transmission rate determined by the DIF policy.*

Proof. By Lemma 3, we only need to show that *EDF-Schedule* guarantees that for a *DIF* identified interval $I[i,j)$, the assigned epochs in $T[i,j]$ are used exclusively by packets of $S[i,j]$ only.

For the sake of contradiction, let epoch $E_h = [e_h, e_{h+1}) \subset I[u, v)$ be the first epoch that is assigned to $S[u, v]$ of interval $I[u, v)$ but is used by *EDF-Schedule* to transmit a packet $P_k = (B_k, a_k, d_k)$ that belongs to $S[i,j]$ of another interval $I[i,j)$. It is clear that $I[u, v)$ must have been identified by *DIF* at an earlier time than that of $I[i,j)$. Thus we have the following arguments.

First, we claim that $d_k \leqslant d_v$, for otherwise the *EDF-Schedule* would not transmit P_k in $I[u, v)$ because, by the Corollary, there are always sufficient data from $S[u, v]$ to be transmitted in $I[u, v)$, and data of $S[u, v]$ have earlier deadlines than d_k. It would be impossible to transmit P_k inside $I[u, v)$. Second, since P_k belongs to $S[i,j]$, we must have $a_k < a_u$, for otherwise, P_k would belong to $S[u, v]$. Third, we must have $d_k > a_u$, because P_k is transmitted in $E_h \subset I[u, v)$. Thus we have $[a_k, d_k) - I[u, v) = [a_k, a_u)$. Since in entire time interval of $I[u, v)$, no epoch is available for $S[i,j]$, including P_k, P_k must have missed its deadline already at time a_u. However, since no violation of the claim occurs before E_h, this contradicts Lemma 3. □

Theorem 3 shows that the *DIF-Policy* together with the *EDF* scheduling have optimally solved the energy efficient packet scheduling problem defined by Definition 6. Note that, the optimal rate for every epoch is unique as we have proved in Theorem 2. However, the schedule for transmitting individual packets produced by *EDF-Schedule* may not be the only way to implement individual packet transmission to meet the deadline requirement using the rete determined by the *DIF-Policy*. For example, if we have 3 packets, $P_1 = (2K, 0, 2)$, $P_2 = (3K, 0, 3)$, $P_3 = (1K, 2, 3)$. The optimal rate is 2K per unit time. The *EDF-schedule* will transmit P_1 from $t = 0$ to $t = 1$, then transmit P_2 from $t = 1$ to $t = 2.5$, and transmit P_3 from $t = 2.5$ to $t = 3$. Another feasible way is to transmit P_2 from $t = 0$ to $t = 1$ with 1K data remaining, then transmit P_1 from $t = 1$ to $t = 2$, transmit P_3 from $t = 2$ to $t = 2.5$, and finally, transmit remaining P_2 from $t = 2.5$ to $t = 3$. The *EDF-Schedule* is only one way to implement individual packet scheduling to satisfy the deadline requirement and the rate required by *DIF-Policy* to guarantee minimum energy consumption. Obviously, *EDF-Scheduling* is the most efficient and convenient way to implement individual packet schedule after the rate is determined.

5. Online policy and simulation results

In previous sections, we have obtained an offline optimal rate control policy as well as deadline guaranteed individual packet scheduling for the energy efficient packet transmission problem. Based on the offline policy, in this section we develop an online rate control policy and packet scheduling with no knowledge of any information of arriving packets, including arrival time, deadline, packet size, and distribution of inter-arrival time.

5.1. Previous online policies

An intuitive online policy is whenever a new packet arrives, apply the offline *DIF-policy* and *EDF-Schedule* to the new packet together with remaining unfinished backlog packets, and start to transmit according to the new schedule. This method turns out to be similar to the Backlog Adaptive (BA) policy proposed in [11] and Online Flush (OF) scheduler proposed in [5]. Since BA policy and OF scheduler use the same idea, we refer them as BA-OF policy.

Basically, the BA-OF policy maintains a transmission (backlog) queue to buffer all unfinished packets. They are ordered according to their deadlines. Based on the information of these backlog packets, the BA-OF policy applies the offline algorithm to compute the best rate. Whenever a new packet arrives, it is inserted into the backlog queue, and the transmission rate is re-calculated accordingly. Once the rate is computed, packets in the queue will be transmitted in order.

The way BA-OF policy calculates the rate is as follows. Suppose current time is t_0, and there are k packets, $P_i = (B_i, a_i, d_i)$, $i = 1, 2, \ldots, k$, buffered. Then, the BA-OF policy computes rate r_0 and index j according to the following formulas:

$$r_0 = \max_{0 \leqslant d_j \leqslant d_k} \frac{\sum_{i=1}^{j} B_i}{d_j - t_0} \qquad d_j = \arg\max_{0 \leqslant d_j \leqslant d_k} \frac{\sum_{i=1}^{j} B_i}{d_j - t_0} \tag{7}$$

Once the pair (r_0, d_j) is found, the transmission rate r_0 will be used until packet P_j is entirely delivered or a new packet arrives. Then, new transmission rate will be computed again according to (7). It was shown in [11] that if no more packets arrive, then this policy achieves the same optimal result as the offline policy can achieve.

The formula (7) actually finds the densest interval for all intervals starting from current time t_0. Moreover, this online policy uses EDF scheduling for the current densest interval. Therefore, the previous online policies match our offline optimal strategy very well.

5.2. Density Guided Cooling(GGC) policy

Although the BA-OF policy works well, it inclines to a more conservative side. If the next packet has a large size and an urgent deadline, it will be forced to use a much higher rate that costs a lot of energy, which could be avoided or reduced by better prediction and pre-planning. Fig. 4 illustrates two scenarios. In (a.1) and (a.2), two types of intersection of packets P_1 and P_2 are shown respectively. At $t = a_1$, there is one packet P_1 in the queue, so BA-OF decides to use low rates, as shown in epoch 1 of (b.1) and (b.2). At $t = a_2$, a large packet P_2 arrives. There are two cases, $d_2 < d_1$ and $d_2 \geqslant d_1$. In both cases, as shown by (b.1) and (b.2), in epochs 2 and 3, BA-OF would be forced to use higher rates to transmit P_2 and the remaining part of P_1. In (c.1) and (c.2), optimal rates are shown for transmitting these two packets. By comparing (c.1) to (b.1) and (c.2) to (b.2), we conclude that sometimes it is better to use higher

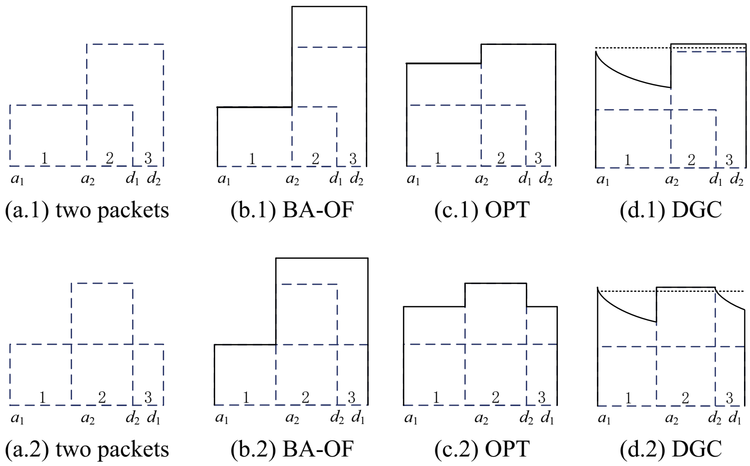


Fig. 4. A comparison among BA-OF policy, optimal policy OPT, and the DGC policy.

rate to transmit current packets in the queue if large incoming packets are anticipated. However, the difficulty is that we do not know when the next packet will come and how large it will be. It is crucially important to make good prediction and strategy that help to decide on how much more data should be sent if a higher rate is used.

Fig. 4(d.1) and (d.2) shows the transmission rates by our online policy which we explain as follows. We use the history average density as a dividing line to classify large packets and small ones. Specifically, we first apply Eq. (7) to calculate rate r_0 and ending time d_j for current remaining unfinished backlog packets. If r_0 is larger than history average density (rate), which means current packets are within a high density interval, then our policy adopts the same rate for the same epochs as BA-OF does. However, if r_0 is smaller than history average density, we anticipate large packets may come soon, thus we will send more data by following a well-designed rate function instead of a constant rate used by BA-OF. This rate function starts with the history average density which is a higher rate than r_0 and gradually reduces to a lower rate, just like the temperature drops of a hot object put in a low temperature environment. The formula of dropping temperature according to Newton's Law of Cooling is shown in Fig. 5, namely, $r = (a - b)e^{-\lambda(t-t_0)} + b$, where a is the temperature of hot object, b is the lower temperature of the environment and λ is the cooling factor. In our experiment, we redefine

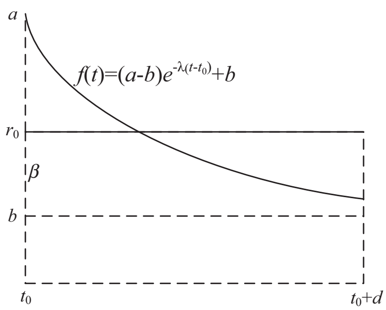


Fig. 5. The rate function follows an exponential decay, starts at a, goes down below r_0 but guarantees a minimum rate b.

these three parameters and use the same formula. Thus, we call our online policy the Density Guided Cooling (*DGC*) policy. Compared to BA-OF policy, the DGC policy is more adaptive to large or small incoming packets, and also guarantees all packets meet their deadlines. The details are presented below.

We discuss the definition of the three parameters, a, b and λ, one by one. First, let a be the history average density.

Definition 10 (*History Average Density*). Suppose $r(t)$ is the history transmission rate, and $t_p(>0)$ is the time (an arrival event point) previous average rate a_{pre} was computed. At time $t_0(>t_p)$, the new average a_{new} is computed.

$$a_{pre} = \frac{\sum_{e_k<t_p} r(e_k)(e_{k+1} - e_k)}{t_p} \quad a_{new} = \frac{\sum_{t_p\leqslant e_k<t_0} r(e_k)(e_{k+1} - e_k)}{t_p - t_0}$$

Set $\alpha = \frac{t_p}{t_0}$, the history average density a is computed as:

$$a = \alpha a_{pre} + (1 - \alpha)a_{new} \tag{8}$$

It is easy to see, Eq. (8) accurately computes the average density over a finite time interval $[0, t_0)$. Note, in real practical situation where t_0 can become too large or goes to infinite, we set α to be a constant instead. The similar idea is also used in computing TCP round trip delay time. The history average density is thus computed in both finite and infinite scenarios.

Since the history average density is the 'high temperature' a from which the object temperature starts to drop, we define the 'low environment temperature' b at which the object temperature eventually stops.

In case $r_0 < a$, we define the *minimum guaranteed rate b* to be the lower bound of the rate function when it drops. Note that rate b is always smaller than rate r_0. In order to determine rate b, we introduce the *invasion ratio* β $(0 < \beta < 1)$ which is an adjustable constant parameter defined by Eq. (9). We have tried different values of β and observed similar results. The simulation results reported in this paper are obtained by setting $\beta = 0.5$.

$$\beta = \frac{r_0 - b}{a - b}, \quad 0 \leqslant b < r_0 < a \tag{9}$$

According to Eq. (9), the *minimum guaranteed rate b* can be calculated by (10).

$$b = \begin{cases} \frac{r_0-\beta a}{1-\beta}, & \text{if } \beta a \leqslant r_0 < a \\ 0, & \text{if } r_0 < \beta a \end{cases} \tag{10}$$

Given the 'high temperature' a and the 'low environment temperature' b, we determine the 'cooling factor' λ, as follows.

$$\lambda = \lambda(d) = \frac{A}{d} \tag{11}$$

where A is a value that satisfies the following equation:

$$1 - e^{-A} = \beta A \tag{12}$$

The value A can be easily pre-computed given a fixed β. d is the length of the definition domain of current rate function that is larger than $d_j - t_0$. How to decide d will be discussed later in the next subsection.

Definition 11 (*Rate Prediction Function*). At time $t = t_0$, given the history average density a, the rate r_0 and its ending time d_j, the minimum guaranteed rate b, the cooling factor λ, the rate prediction function applied to period $[t_0, d_j)$ is defined as follows:

$$f(t) = \begin{cases} (a-b)e^{-\lambda(t-t_0)} + b & r_0 < a \\ r_0 & r_0 \geqslant a \end{cases} \tag{13}$$

This formula follows an exponential decay. The extensive use of exponential decay can be found in many nature sciences, e.g. fluid dynamics, radioactive and heat transfer.

Some rationales for using the proposed rate prediction function are: (1) In a long run, the densities for future packets are expected to be around the history average a statistically. Therefore, our prediction function starts with rate a. (2) Most packet arrivals follow Poisson processes, in which, the inter-arrival time follows an exponential distribution. (3) As time goes by, the remaining data load becomes less and less, with no need to transmit more data in advance. The prediction function needs to go below r_0 to save energy instead. (4) The prediction rate should always be larger than a minimum rate for a given period. This minimum rate is b which depending on the situation could be zero.

Our DGC online policy consists of 4 steps:

(1) Compute the history average density a by Eq. (8), utilize Eq. (7) to compute (r_0, d_j), and calculate the minimum guaranteed rate b by Eq. (10).
(2) Rate prediction function $f(t)$ is set as in Eq. (13), in which $\lambda = \lambda(d)$ is obtained by (11).
(3) Transmit packets at rate $f(t)$ from time t_0 until d_j.
If all packets are finished before d_j, pause until time d_j.
If a new packet arrives at any time before d_j, insert it into the queue in the order of their deadlines.
(4) Goto step (1).

Theorem 4. *The DGC online policy guarantees that all packets meet their own deadlines.*

Proof. The BA-OF policy by (7) guarantees that all packets meet their own delay constraints [5]. Thus, we only need to show, our DGC policy guarantees any packet be finished earlier than or equal to the time by BA-OF policy. This is obvious for $r_0 \geqslant a$, we prove this for the case of $r_0 < a$ as follows.

Since in DGC online policy, $\lambda = \frac{A}{d}$, thus $A = \lambda \times d$. According to (11) and (12), we have

$$1 - e^{-\lambda d} = \beta \lambda d$$

We further compute the total amount of data transmitted by the rate function of (13) in the definition domain of $[t_0, t_0 + d)$ as follows:

$$\begin{aligned} \int_{t_0}^{t_0+d} f(t)dt &= \int_{t_0}^{t_0+d} ((a-b)e^{-\lambda(t-t_0)} + b)dt \\ &= bd + (a-b)\left(\frac{1}{\lambda} - \frac{e^{-\lambda d}}{\lambda}\right) = bd + (a-b)d \\ &= bd + (r_0 - b)d = r_0 d \end{aligned}$$

This means that the total amount of data transmitted by the rate function of (13) in time interval $[t_0, t_0 + d)$ is exactly equal to the amount that would be transmitted by the constant rate r_0. Since $[t_0, d_j) \subseteq [t_0, t_0 + d)$ and the rate function of (13) is a decreasing function, at any point $t_x \in [t_0, d_j)$, we must have

$$\int_{t_0}^{t_x} f(t)dt > \int_{t_0}^{t_x} r_0 t dt$$

That is, the data transmitted during interval $[t_0, t_x)$ by DGC online policy exceeds that by BA-OF policy, thus, DGC policy guarantees any packet be finished earlier than that by BA-OF policy. □

5.3. Simulation results

In our simulations, we set the invasion ratio $\beta = 0.5$, as discussed before. We then have $\frac{r_0-b}{a-b} = 0.5$ and $A = 1.59$. According to (11), $\lambda = \frac{A}{d} = \frac{1.59}{d}$, so, to compute λ, we need to decide d first. Let $c = d_j - t_0$. We set d as follows.

$$d = \begin{cases} 2c, & \text{if } c > avg(delay) \\ 2avg(delay), & \text{otherwise} \end{cases} \tag{14}$$

where $avg(delay)$ is the average delay constraint of all arrived packets, which can be easily computed similar to (8). Note that d is re-calculated each time when a new rate prediction function is calculated, and it may change over time, because both c and $avg(delay)$ are also updated dynamically. In summary, the specific rate function used in the simulation is:

$$f(t) = \begin{cases} 2(a-r_0)e^{-\frac{1.59(t-t_0)}{d}} + 2r_0 - a & r_0 \leqslant a \\ r_0 & r_0 > a \end{cases}$$

From (14), we can see $d > c$, thus Theorem 4 holds: all packets can meet their deadlines.

In the simulation, we also carefully model packet related parameters, including packet arrival, packet size and packet delay constraint. (i) Following most previous works, we model packet arrival as a Poisson process. (ii) We assume packet size follows normal distribution $\mathcal{N}(s, 0.1s)$, where s is the average packet size. (iii) We imitate arbitrary packet delay constraint by modeling it to be a mixed random distribution of three distributions: uniform distribution $\mathcal{U}(0.1q, 1.9q)$, normal distribution $\mathcal{N}(q, 0.3q)$ and a modified exponential distribution $EXP(0.9q) + 0.1q$, where q is the average delay constraints. All packets are with delay constraints larger than $0.1q$. We make this assumption because packets must hold a minimum allowed period $[0, 0.1q)$ to transmit.

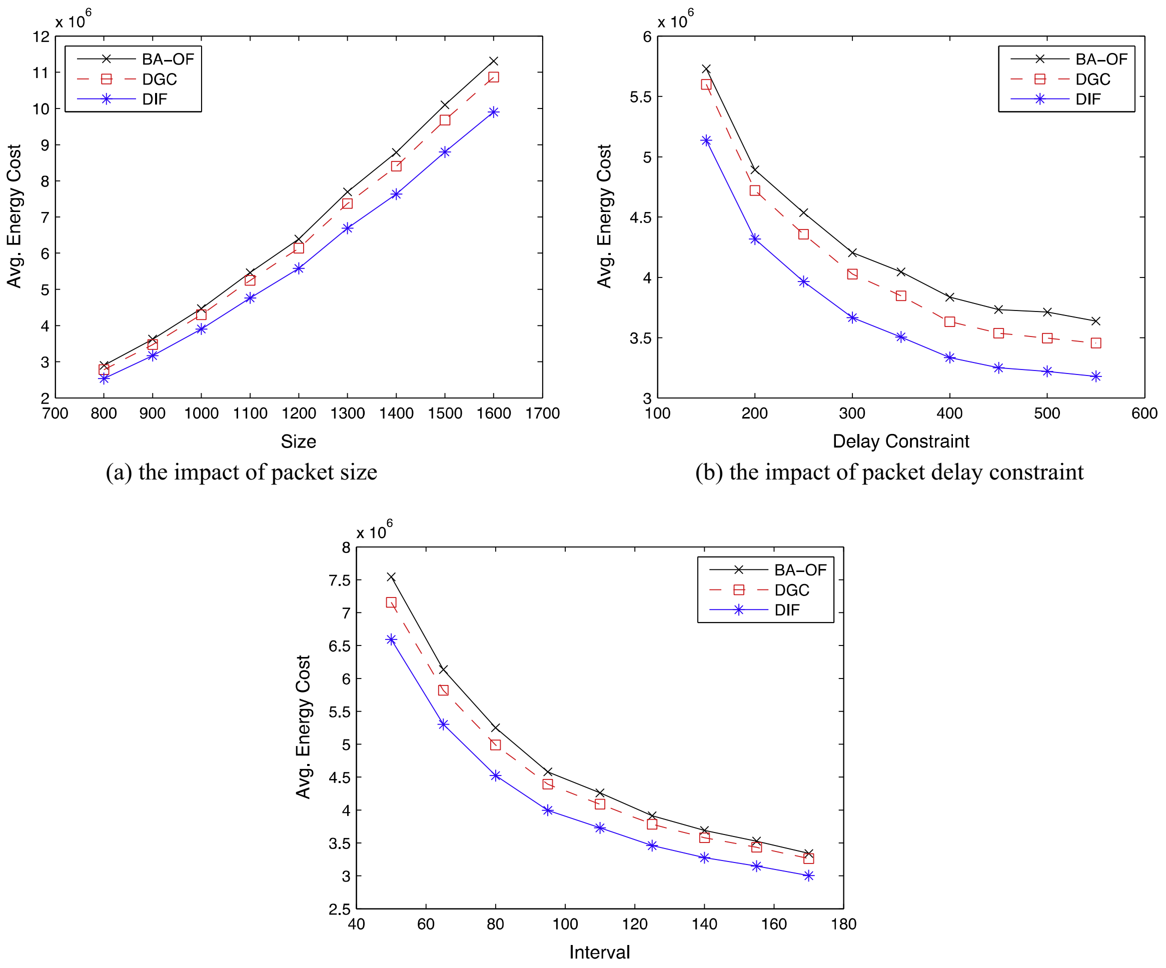
Each value shown in all the following figures and tables is the mean value of simulation results from 40 random instances, and in each instance, 300 packets are generated according to the above model.

We first investigate the performance of our DGC policy against BA-OF policy.

From Table 4, we can see the BA-OF policy, as well as our DGC policy, tends to produce near offline optimal (DIF) results when the ratio between average packet inter-arrival time and average packet delay constraint becomes bigger. This is because, when the ratio becomes

Table 4
Energy consumption under different ratio between average packet inter-arrival time and average packet delay constraint.

	0.2	0.4	0.6	0.8	1.0	1.2	1.4	1.6
DIF(10^6)	6.59	3.93	3.20	2.84	2.67	2.52	2.48	2.35
BA-OF(10^6)	7.58	4.51	3.59	3.12	2.90	2.70	2.63	2.48
DGC(10^6)	7.18	4.33	3.50	3.07	2.87	2.68	2.62	2.48
BA-OF/DIF (%)	115.0	114.8	112.0	110.0	108.5	107.2	106.3	105.7
DGC/DIF (%)	108.9	110.4	109.2	108.1	107.3	106.5	105.9	105.5



(a) the impact of packet size

(b) the impact of packet delay constraint

(c) the impacket of pacekt inter-arrival time

Fig. 6. The impact of different factors to the average energy consumptions of optimal DIF policy, BA-OF policy and our online policy DGC. The default setting is packet size 1000, average packet delay constraint 250, average packet inter arrival time 100.

bigger than 1, packet delay constraint is smaller than inter-arrival time, thus packets tend to have deadline before the next packet arrives, as a result, there is little or no need for scheduling and all algorithms obtain near optimal results. However, when the ratio is smaller, BA-OF policy failed to keep results close to the offline optimal. In these scenarios, we observe that our DGC policy outperforms BA-OF policy and output results are within constant ratio to the offline optimal.

In Fig. 6, the default setting is that the average packet size is 1000 unit, average packet delay constraint is 250 unit time and average packet inter arrival time is 100 unit. These three parameters are changed one at a time to study their impacts. We can see from Fig. 6(a)–(c) that the DGC policy constantly outperforms BA-OF policy under all settings. And in almost all cases, the DGC policy outputs results that are within 110% of the offline optimal. In Fig. 6(a), the average energy consumption of DGC rises as average packet size increase, but it is almost constantly around 109% of the offline optimal. In Fig. 6(b), the average energy consumption decreases when the packet delay constraint increase. This is because the longer the packet delay constraints, the less urgent these packets are, thus a lower rate can be used to consume less energy for transmission. In Fig. 6(c), the average energy consumption decreases when the inter-arrival time increases. This is because a large inter-arrival time means a lower arrival rate, thus fewer packets arrive in a unit time, therefore less energy is consumed. In both (b) and (c), the curve of DGC is almost parallel to the curve of offline optimal, which means its performance is stable.

We have done quite extensive simulations and obtained rich results beyond Fig. 6 can show. All results show the DGC policy is more adaptive to incoming packets with parameters dynamically changing and preforms better than previous BA-OF algorithm.

6. Conclusions

This paper has optimally solved the energy efficient packet transmission problem for transmitting a sequence of packets with arbitrary deadlines. A notion of *data interval* is introduced and a new technique called *densest interval first* (DIF) is proposed to capture the nature of this difficult problem. The EDF (Earliest Deadline first) scheduling is proven to be optimal and efficient to schedule each individual packet after the *DIF* policy has determined the transmission rate for each densest interval. Finally, this paper has proposed an online policy called DGC algorithm. Simulations show that by better prediction and pre-planning, DGC policy constantly produces a rate function that is within 110% of the optimal result. The combination of *DIF-Policy* and *EDF-Schedule* would provide a generic approach for other energy efficient research problems with different system models such as energy harvesting systems, multi-channel systems, and fading channels.

Acknowledgments

The authors would like to thank the anonymous referees and the editor for their constructive comments and valuable suggestions which have helped improve the quality and presentation of the paper. This work is supported by National High-tech R&D Program under Grant No. 2013AA013503, National Natural Science Foundation of China under Grants No. 61370208, No. 61300024, No. 61320106007 and No. 61402104, China Specialized Research Fund for the Doctoral Program of Higher Education under Grants No. 20110092130002, Jiangsu Provincial Natural Science Foundation of China under Grants No. BK20130634 and No. BK20140648, Jiangsu Provincial Key Laboratory of Network and Information Security under Grant No. BM2003201, and Key Laboratory of Computer Network and Information Integration of Ministry of Education of China under Grants No. 93K-9.

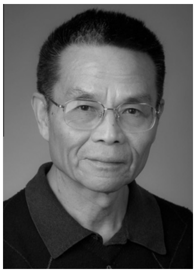
References

[1] B. Prabhakar, E. Uysal Biyikoglu, A. El Gamal, Energy-efficient transmission over a wireless link via lazy packet scheduling, in: Proceedings IEEE INFOCOM, vol. 1, April 2001, pp. 386–394.
[2] E. Uysal-Biyikoglu, B. Prabhakar, A. El Gamal, Energy-efficient packet transmission over a wireless link, IEEE Trans. Netw. 10 (4) (2002) 478–499.
[3] E. Uysal-Biyikoglu, A. El Gamal, On adaptive transmission for energy efficiency in wireless data networks, IEEE Trans. Inf. Theory 50 (12) (2004) 3081–3094.
[4] W. Chen, U. Mitra, Energy efficient scheduling with individual packet delay constraints, in: Proceedings of IEEE INFOCOM, April 2006.
[5] W. Chen, M.J. Neely, U. Mitra, Energy-efficient transmissions with individual packet delay constraints, IEEE Trans. Inf. Theory 54 (5) (2008) 2090–2109.
[6] M.A. Khojastepour, A. Sabharwal, Delay-constrained scheduling: power efficiency, filter design, and bounds, in: Proceedings of IEEE INFOCOM, March 2004, pp. 1938–1949.
[7] W. Chen, M.J. Neely, U. Mitra, Energy efficient scheduling with individual packet delay constraints: offline and online results, in: Proceedings of IEEE INFOCOM, May 2007, pp. 1136–1144.
[8] W. Chen, U. Mitra, M. Neely, Energy-efficient scheduling with individual delay constraints over a fading channel, in: Proceedings of WiOpt, April 2007.
[9] W. Chen, U. Mitra, M. Neely, Energy-efficient scheduling with individual packet delay constraints over a fading channel, Wireless Netw. 15 (5) (2009) 601–618.
[10] M. Zafer, E. Modiano, A calculus approach to minimum energy transmission policies with quality of service guarantees, in: Proceedings of IEEE INFOCOM, March 2005, vol. 1, pp. 548–559.
[11] M. Zafer, E. Modiano, a calculus approach to energy-efficient data transmission with quality-of-service constraints, IEEE Trans. Netw. 17 (3) (2009) 898–911.
[12] M. Zafer, E. Modiano, Optimal rate control for delay-constrained data transmission over a wireless channel, IEEE Trans. Inf. Theory 54 (9) (2008) 4020–4039.
[13] J. Yang, S. Ulukus, Optimal packet scheduling in an energy harvesting communication system, IEEE Trans. Comm. 60 (1) (2012) 220–230.
[14] J. Yang, S. Ulukus, Optimal packet scheduling in a multiple access channel with rechargeable nodes, in: Proceedings of IEEE ICC, June 2011.
[15] J. Yang, O. Ozel, S. Ulukus, Optimal packet scheduling in a broadcast channel with an energy harvesting transmitter, in: Proceedings of IEEE ICC, June 2011.
[16] M. Akif Antepli, E. Uysal-Biyikoglu, H. Erkal, Optimal packet scheduling on an energy harvesting broadcast link, IEEE J. Sel. Area Comm. 29 (8) (2011) 1721–1731.
[17] O. Ozel, K. Tutuncuogl, J. Yang, S. Ulukus, A. Yener, Resource management for fading wireless channels with energy harvesting nodes, in: Proceedings of IEEE INFOCOM, April 2011, pp. 456–460.
[18] O. Ozel, K. Tutuncuogl, J. Yang, S. Ulukus, A. Yener, Transmission with energy harvesting nodes in fading wireless channels: optimal policies, IEEE J. Sel. Areas Comm. 29 (8) (2011) 1732–1743.
[19] P. Nuggehalli, V. Srinivasan, R.R. Rao, Delay constrained energy efficient transmission strategies for wireless devices, in: Proceedings of IEEE INFOCOM, vol. 3, June 2002, pp. 1765–1772.

[20] A. Fu, E. Modiano, J. Tsitsiklis, Optimal energy allocation for delay-constrained data transmission over a time-varying channel, in: Proceedings of IEEE INFOCOM, vol. 2, April 2003, pp. 1095–1105.
[21] Michael J. Neely, Energy optimal control for time-varying wireless networks, IEEE Trans. Inf. Theory 52 (7) (2006) 2915–2934.
[22] F. Yao, A. Demers, S. Shenker, A scheduling model for reduced CPU energy, in: Proceedings of 36th Annual Symposium on Foundations of Computer Science, 1995, pp. 374–382.

Feng Shan received his B.S. degree in Computer Science from Hohai University, Nanjing, China, in 2008. He is currently pursuing the Ph.D. degree in computer science and engineering at Southeast University, Nanjing, China. He joined the School of Computing and Engineering, University of Missouri-Kansas City, Kansas City, MO, United States, from 2010 to 2012 as a visiting scholar. His research interests are in the areas of energy harvesting, algorithm design, and wireless multi-hop networks.

Junzhou Luo received the BS degree in applied mathematics and the M.S. and Ph.D. degrees in computer network all from Southeast University, China, in 1982, 1992, and 2000, respectively. He is a Full Professor in the School of Computer Science and Engineering, Southeast University, Nanjing, China. His research interests are next generation network architecture, network security, cloud computing, and wireless LAN. He is a member of the IEEE Computer Society and co-chair of IEEE SMC Technical Committee on Computer Supported Cooperative Work in Design, and he is a member of the ACM and chair of ACM Nanjing Chapter.



Xiaojun Shen (SM'02) received the B.S. degree in numerical analysis from Tsinghua University, Beijing, China, in 1968, the M.S. degree in computer science from the Nanjing University of Science and Technology, China, in 1982, and the Ph.D. degree in computer science from the University of Illinois at Urbana–Champaign in 1989. He is a professor in the School of Computing and Engineering, University of Missouri-Kansas City. His current research focuses on fundamental scheduling problems in wired and wireless computer networks.