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

Article info
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

Abstract
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 packets 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.

© 2014 Elsevier B.V. All rights reserved.

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
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 charged 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
they occur, 0 ≤ e₁ < e₂ < · · · < eₘ. Obviously, e₁ = a₁, eₘ = dₘ = T. The time interval between two adjacent event points is called an epoch, and epoch Eₖ = [eₖ, eₖ₊₁), 1 ≤ k < 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 P₁ = (10K, 2, 6), P₂ = (8K, 3, 12), P₃ = (20K, 5, 9), P₄ = (7K, 7, 11), T = 12, m = 8. The 7 epochs are E₁ = [2, 3), E₂ = [3, 5), E₃ = [5, 6), E₄ = [6, 7), E₅ = [7, 9), E₆ = [9, 11), E₇ = [11, 12).

We use function ζ to map each arrival or deadline point to its rank in the event sequence. Thus, if aₖ = eₖ, then ζ(aₖ) = k. Similarly, if dₘ = eₖ then ζ(dₘ) = k. Function ζ 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) → ∞ as p → ∞;
(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) = 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(t) of packet Pᵢ: R⁺ → R⁺ is defined as the transmission rate for packet Pᵢ at time t, 0 ≤ t < T, 1 ≤ i ≤ n.

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

![Fig. 1. An example of a data set with n = 4.](image-url)
\[
\int_0^\tau r_i(t) dt = \int_{n_i}^{d_i} r_i(t) dt = B_i
\]

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

According to the transmission model, we have \( r(t) = g(p(t)), 0 \leq t < T \). Obviously, an overall rate function \( r(t), 0 \leq t < T \), 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 \([11]\).

\[
E = \int_0^\tau g^{-1}(r(t)) dt
\]

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

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

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 \( S = \{r_i(t), 0 \leq t < T\} \) for a given data set of \( n \) packets, \( P_i = (B_i, a_i, d_i), 1 \leq i \leq n \).

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, \( r_{opt}(t), 0 \leq t < T \), 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 \([e_k, e_{k+1})\), \( 1 \leq k \leq m - 1 \), 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:

\[
\forall t \in [0,T), \sum_{a_i \leq t} B_i \leq \int_0^t r_{opt}(x) dx \leq \sum_{a_i \leq t} B_i
\]

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

Lemma 1. Any optimal overall rate function \( r_{opt}(t), 0 \leq t < T \), 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 \( r_{opt}(t), 0 \leq t < T \), 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 \( e_1 \) 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 \([e_{j-1}, e_{j+1}]\) 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 \([e_{j-1}, e_j]\) but decreases the rate to \( r_2 \) before the deadline at \( e_j \), while no deadline event is at \( e_j \), 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 \([e_{j-1}, e_j]\) is moved to be transmitted in later epoch \([e_j, e_{j+1}]\). 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. \( \square \)

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 \( P_h = (B_h, a_h, d_h) \) 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 \([a_h, d_h]\), and \( H' \subseteq H \) be the subset of \( H \) in which \( r_h(t) \neq 0 \). 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 \( P_0 \) 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 \( P_0 \) 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 \( P_k = (B_k, a_k, d_k) \), \( 1 \leq k \leq n \), a data interval \( I(i, j) \) is defined to be the time interval from the arrival time \( a_i \) to deadline \( d_j \). That is, \( I(i, j) = [a_i, d_j] \), if \( a_i < d_j \), and \( i, j \leq n \), 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 \( a_1 = 5, a_2 = 7, a_3 = 9, d_2 = 9, d_3 = 12 \), then we have \( d_{h_1} = 9, d_{h_2} = 9, d_{h_3} = 12 \). The data intervals are:

\[
\begin{align*}
I(1, 1) &= [5, 9], I(1, 2) = [5, 9], I(1, 3) = [5, 12], \\
I(2, 1) &= [7, 9], I(2, 2) = [7, 9], I(2, 3) = [7, 12], \\
I(3, 1) &= [7, 9], I(3, 2) = [7, 9], I(3, 3) = [7, 12].
\end{align*}
\]

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, \( S[i, j] = \{P_k | (d_k - a_k) \subseteq I(i, j) \} \).
2. Its *data load* \( B[i, j] \) is the total amount of data contained in \( S[i, j] \), that is \( B[i, j] = \sum_{P_k \in S[i, j]} P_k \).
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. \( L[i, j] = (d_j - a_i) \) 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>
<thead>
<tr>
<th>Table 1</th>
<th>Parameters for data intervals of packet set of Fig. 1.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S[i, j] )</td>
<td>( d_0 = d_1 )</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>( a_1 = 2 )</td>
<td>( P_0(10k, 2, 6) )</td>
</tr>
<tr>
<td>( a_2 = 3 )</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>( a_3 = 5 )</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>( a_4 = 7 )</td>
<td>Unassigned</td>
</tr>
<tr>
<td>( B[i, j] )</td>
<td>( d_0 = d_1 )</td>
</tr>
<tr>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>( a_1 = 2 )</td>
<td>( B = 10k )</td>
</tr>
<tr>
<td>( L = 6 - 2 = 4 )</td>
<td>( L = 9 - 2 = 7 )</td>
</tr>
<tr>
<td>( D = 10 )</td>
<td>( D = 30 )</td>
</tr>
<tr>
<td>( a_2 = 3 )</td>
<td>( B = 0 )</td>
</tr>
<tr>
<td>( L = 6 - 3 = 3 )</td>
<td>( L = 9 - 3 = 6 )</td>
</tr>
<tr>
<td>( D = 0 )</td>
<td>( D = 20 )</td>
</tr>
<tr>
<td>( a_3 = 5 )</td>
<td>( B = 0 )</td>
</tr>
<tr>
<td>( L = 6 - 5 = 1 )</td>
<td>( L = 9 - 5 = 4 )</td>
</tr>
<tr>
<td>( D = 0 )</td>
<td>( D = 20 )</td>
</tr>
<tr>
<td>( a_4 = 7 )</td>
<td>Unassigned</td>
</tr>
<tr>
<td>( L = 9 - 7 = 2 )</td>
<td>( L = 11 - 7 = 4 )</td>
</tr>
<tr>
<td>( D = 0 )</td>
<td>( D = 7 )</td>
</tr>
<tr>
<td>1.75</td>
<td>2</td>
</tr>
</tbody>
</table>
Theorem 1 (Basic Densest Interval Theorem). Given a data set \( P = \{P[i] \mid 1 \leq i \leq n\} \), 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 \([a_i, d_i]\).
2. Any optimal solution must deliver exactly the packet set \( S[i,j] \) during the time interval \([a_i, d_i]\).

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

Obviously, \( r^{opt}(t) \) increases at \( e_n \) and decreases rate at \( e_n \), for otherwise we could extend to an even larger interval. By Lemma 1, \( e_n \) must be an arrival point and \( e_n \) must be a deadline point. So, \([e_n, e_n]\) is a data interval and its density is no larger than \( r \) because the rate is the densest one. We have \( r^{opt}(t) \geq r \) for entire interval \([e_n, e_n]\) and \( r^{opt}(t) > r \) for epoch \([e_k, e_{k+1}] \subseteq [a_i, d_i]\), thus \( \int_{e_k}^{e_{k+1}} r^{opt}(t) dt > \int_{e_k}^{e_{k+1}} r dt \leq B[i,j] \). Obviously, \([e_n, e_n] \neq [0, T] \) 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 \( P_o \) that arrived before \( e_n \) or have a deadline larger than \( e_n \) in time interval \([e_n, e_n]\). This contradicts Lemma 2. Therefore, any optimal rate policy must use a single rate \( r = D[i,j] \) for the time interval \([a_i, d_i]\).

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 \( I[i,j] = [a_i, T] \), then our job is done. Otherwise, we need to continue to find an optimal rate policy for the remaining intervals \((0, T) - [I[i,j]] = [0, a_i] \cup [d_i, T] \) 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 \( I[i,j] \), \( 1 \leq i, j \leq n \). The data set consists of all packets whose waiting intervals are inside \([a_i, d_i]\) and currently remain unassigned.
2. Re-compute the data load \( B[i,j] \), the available time length \( I[i,j] \), and the density \( D[i,j] \) for every interval \( I[i,j] \), \( 1 \leq i, j \leq n \).
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 assigned 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 \( R[i] = 0 \), \( 1 \leq i \leq n \), to denote that packet \( P[i] \) has been assigned and \( R[i] = 1 \) otherwise. We use \( E[k] = 0 \), \( 1 \leq k \leq m - 1 \), to denote that \( k \)th 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, \( R[i] = 1 \), \( 1 \leq i \leq n \), \( E[k] = 1 \), \( 1 \leq k \leq m - 1 \). Major notations used in this paper are summarized in Table 2 for the reader’s convenience.

The following is the pseudo code for updating \( R[i] \), \( 1 \leq i \leq n \), and \( E[k] \), \( 1 \leq k \leq m - 1 \), 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 \([a_i, d_i]\) with “assigned” and those epochs inside \([a_i, d_i]\) with “unavailable.”

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

1. for \( k \leftarrow 1 \) to \( m - 1 \) do
2. if \( e_k, e_{k+1} \subseteq [a_i, d_i] \) then
3. \( E[k] \leftarrow 0 \) //It is possible that \( E[k] = 0 \) already
4. endif
5. endfor
6. for \( i \leftarrow 1 \) to \( n \) do
7. if \( [a_i, d_i] \subseteq [a_i, d_i] \) then
8. \( R[i] \leftarrow 0 \) //It is possible that \( R[i] = 0 \) already
9. endif
10. 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 \( P_1 \) with transmission rate \( r = 5 \) and we also assign epochs \( E_3 = [5,6] \) and \( E_4 = [6,7] \) to packet \( P_2 \). 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,
Major notations and their explanations.

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

![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.](image)

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 $I[i,j] = [a_i, d_j]$ and $I[i,j+1] = [a_i, d_{j+1}]$, we have $I[i,j] \subset I[i,j+1]$. Thus, we have the following two sequences of containment relations:

$$(4) \quad I[i, 1] \subset I[i, 2] \subset \cdots \subset I[i,j] \subset I[i,j+1] \subset \cdots \subset I[i,n]$$

and

$$(5) \quad S[i, 1] \subset S[i, 2] \subset \cdots \subset S[i,j] \subset S[i,j+1] \subset \cdots \subset S[i,n]$$

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

$$(6) \quad L[i, 1] \leq L[i, 2] \leq \cdots \leq L[i,j] \leq L[i,j+1] \leq \cdots \leq L[i,n]$$

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($E[i]$, $R[i]$, $n$)**

1. for $i = 1$ to $n$ do // initialization loop
2. for $j = 0$ to $n$ do // $j$ starts from 0 for looping purpose
3. $M[i,j] = 0$ // initially, every $I[i,j]$ is inactive
4. $S[i,j] = \emptyset$ // $\emptyset$ = empty
5. $B[i,j] = 0$
6. $L[i,j] = 0$
7. endfor
8. endfor
9. for $i = 1$ to $n$ do
10. $d_{h_0} = a_i$ // Dummy $d_{h_0}$ is for looping purpose
11. for $j = 1$ to $n$ do
12. if $R[h_j] = 1$ and $a_{h_j} \geq a_i$ then // $P_{h_j}$ is unassigned and $(a_{h_j}, d_{h_j}) \subseteq I[i,j]
13. $S[i,j] = S[i,j-1] \cup (P_{h_j})$
15. else
16. $S[i,j] = S[i,j-1]$
18. endif
19. $L[i,j] = [L[i,j-1] \cup [a_i, d_j])$ //Start adding new epochs
20. for $k = \max(\{\zeta(d_{h_{k-1}}), \zeta(a_{\bar{i}})\})$ to $\zeta(d_{h_0})$ - 1 do
21. if $E[k] = 1$ then //Epoch $k$ is available
22. $L[i,j] = L[i,j] + (e_{k+1} - e_k)$
23. endif
24. endfor
25. if $S[i,j] \neq \emptyset$ then
26. $M[i,j] = 1$ // $I[i,j]$ is active
27. $D[i,j] = B[i,j]/L[i,j]$
28. endif
29. endfor
30. endfor
31. End
Table 3
The updated densities of active intervals after the first iteration for the packet set of Fig. 1.

<table>
<thead>
<tr>
<th>n</th>
<th>d_h_1</th>
<th>d_h_2</th>
<th>d_h_3</th>
<th>d_h_4</th>
<th>d_h_5</th>
<th>d_h_6</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>a_1 = 2 P(10k,2,6)</td>
<td>9</td>
<td>11</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>a_2 = 2 P(10k,2,6)</td>
<td>P(10k,2,6)</td>
<td>P(10k,2,6)</td>
<td>P(10k,2,6)</td>
<td>P(10k,2,6)</td>
<td>P(10k,2,6)</td>
</tr>
<tr>
<td>11</td>
<td>a_3 = 2 P(10k,2,6)</td>
<td>P(10k,2,6)</td>
<td>P(10k,2,6)</td>
<td>P(10k,2,6)</td>
<td>P(10k,2,6)</td>
<td>P(10k,2,6)</td>
</tr>
<tr>
<td>12</td>
<td>a_4 = 2 P(10k,2,6)</td>
<td>P(10k,2,6)</td>
<td>P(10k,2,6)</td>
<td>P(10k,2,6)</td>
<td>P(10k,2,6)</td>
<td>P(10k,2,6)</td>
</tr>
</tbody>
</table>

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_1 and d_h_6. Therefore, each epoch is checked at most once through entire loop for all values of j. Since there are m = 1 < n 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 \( \leq i \leq n \).
(2) 0 \( \leq a_1 \leq a_2 \leq \cdots \leq a_m \), 0 \( \leq d_h_1 \leq d_h_2 \leq \cdots \leq d_h_6 = T \).
(3) 0 \( \leq e_i \leq e_2 \leq \cdots \leq e_m \).
(4) Function \( \xi \) that maps each arrival time \( a_i \) or deadline \( d_h_6 \) to its epoch rank.

\[
\text{DIF-Policy}(P, n, m, \xi)
\]

1. for \( k \leftarrow 1 \) to \( m - 1 \) do
2. \( E[k] \leftarrow 1 \) //Mark all epochs available
3. endfor
4. for \( k \leftarrow 1 \) to \( n \) do
5. \( R[k] \leftarrow 1 \) //Mark all packets unassigned
6. endfor
7. Density-Update\((E[,], R[,], n)\)
8. find densest active interval \( \text{I}[i,j] \)
9. \[ D[i,j] = \max\{D[u,v] || M[u,v] = 1\} \]
10. while \( D[i,j] > 0 \) do
11. \( T[i,j] \leftarrow 1 \) //Set of epochs assigned to \( S[i,j] \)
12. for \( u \leftarrow \xi(a_i) \) to \( \xi(d_h_6) \) \( - 1 \)
13. if \( E[u] = 1 \) then
14. \( T[i,j] \leftarrow T[i,j] \cup \{\text{epoch } E_u\} \)
15. endif
16. endfor
17. assign rate \( r = D[i,j] \) to all epochs of \( T[i,j] \)
18. Availability-Update\((E[,], R[,], I[i,j])\)
19. Density-Update\((E[,], R[,], n)\)
20. find densest active interval \( I[i,j] \)
21. \[ D[i,j] = \max\{D[u,v] || M[u,v] = 1\} \]
22. endwhile
23. for \( k \leftarrow 1 \) to \( m - 1 \) do
24. if \( E[k] = 1 \) then
25. assign rate \( r = 0 \) to epoch \( E_k \)
26. endif
27. endfor
28. 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 \( E_1 = \{2,3\}, E_2 = \{3,5\}, E_3 = \{5,7\}, E_4 = \{6,8\} \), and \( E_5 = \{9,11\} \). thus, 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 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 a 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, $D[u,v] \leq D[i,j]$, 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 $D[i,j] = \max\{D[u,v] | I[u,v] = I[i,j]\}$. The following are true:

1. Any optimal rate policy must assign rate $r = D[i,j]$ to every available epoch in $T[i,j]$.

**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, $k \geq 1$, we prove that it is also true for the $(k + 1)$st iteration.

Let $I[i,j]$ be the data interval such that $D[i,j] = \max\{D[u,v] | I[u,v] = I[i,j]\}$ 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 $I[i,j] = [a_0, b_k]$ at the beginning of the $(k + 1)$st iteration.

Suppose $r^{opt}(t) = a_i \leq t < b_k$, 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 the $(k + 1)$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 $r^{opt}(t) \leq r$ for all epochs in $T[i,j]$ and $r^{opt}(t) \leq r$ in some epoch which implies $\sum_{t\in [e_k, e_{k+1}]} r^{opt}(t) dt < r \times I[i,j] = B[i,j]$ 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 $[e_k, e_{\infty}]$ be the longest data interval such that $[e_k, e_{\infty}] \subseteq [e_k, e_{e}]$ and $r^{opt}(t) \geq r$ in $[e_k, e_{\infty}]$. Note that interval $[e_k, e_{\infty}]$ can include both available and unavailable epochs. $r^{opt}(t)$ increases rate at $e_k$ and decreases rate at $e_{\infty}$.

By Lemma 1, $e_k$ must be an arrival point $a_0$ and $e_{\infty}$ must be a deadline point $b_k$, thus, $[e_k, e_{\infty}]$ is a data interval $I[x,y]$ and its density $D[x,y]$ is no larger than $r$ at the beginning of the $(k + 1)$st iteration, because $r$ is the densest density. Obviously $I[x,y]$ is active at the beginning of the $(k + 1)$st iteration because it contains $[e_k, e_{\infty}]$. Moreover, if time interval $[e_k, e_{\infty}]$ contains an unavailable epoch marked by a previous iteration $k' < k$ 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 $r' > r$ by Observation 1.

Fig. 3 shows the relation between intervals $[e_u, e_{n+1})$ 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 the $(k + 1)$st iteration. Let $B^{(1)}$ and $B^{(2)}$ represent the data loads in these two groups, respectively. Then, at the beginning of the $(k + 1)$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 $r^{opt}(t) \geq r$ in entire interval $[e_u, e_{v}]$ and $r^{opt}(t)$ is strictly larger than $r$ in $[e_u, e_{v+1}]$, the amount of data transmitted by the optimal rate in interval $[e_u, e_{v}]$ is:

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

Since the density $D[i,j] = r$ is the largest at the beginning of the $(k + 1)$st iteration, we have $r \times L[x,y] \geq D[x,y] \times L[x,y] = B^{(2)}$. Therefore, we have $B > B^{(1)} + B^{(2)}$, which implies that the optimal policy transmits some packet $P_2$ whose arrival time $a_0$ is smaller than $a_1$ or whose deadline $d_2$ is larger than $d_1$. 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 $P_3$, how should we schedule them? If we schedule them in the order of $P_1$, $P_2$, and $P_3$, one by one, continuously until finish, then some data in $P_3$ 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$(P, m)$

1. build a min-heap $H$ using deadlines as the keys.
   //Use arrival time to break a tie.
2. for $i = 1$ to $m - 1$ do
3.  if $e_i$ is an arrival point then
4.    insert those packets that arrive at $e_i$ into $H$
5.  endif
6.  $r ← r(E_i)$ //the rate assigned to $E_i$ by DIF-Policy
7.  $s_t ← e_t$ //Starting time for sending next packet
8.  while $s_t < e_{t+1}$ and $H ≠ ∅$ do
9.    $P_k ←$ packet at the root of $H$
10.  //Get the packet without extracting it from heap $H$
11.  $f_t ← s_t + B_k/r$ //expected finish time
12.  if $f_t > e_{t+1}$ then $f_t ← e_{t+1}$ endif
13.  transmit $P_k$ at rate $r$ in time interval $[s_t, f_t]$
14.  if $r × (f_t - s_t) = B_k$ then
15.    $B_k ← B_k - r × (f_t - s_t)$ //remaining size of $P_k$
16.  else
17.    extract $P_k$ from $H$
18.  endif
19.  $s_{t+1} ← f_t$ //Next starting time
20. endwhile
21. 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 $P_3$ is at the root, transmit $P_3$ at rate 4.167. $f_t = 9 + 7/4.167 = 9 + 1.68 = 10.68 < 11$. $P_3$ finishes at $t=10.68$. Delete $P_3$.
8. At $t=10.68$, $P_2$ is at the root. $f_t = 10.68 + 5/4.167 = 10.68 + 1.32 = 12 > 11$, transmit $P_2$ to 11. The remaining size = 5.5 – 4.167(11 – 10.68) = 5.5 – 1.333 = 4.167.

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 $P_k = (A, a_k, d_k)$ 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 $T[a_k, d_k] ⊂ T[i, j]$ 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 $t < a_k$ 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 $t = a_k < a_k$. Further, we can see that in the time interval $[a_k, d_k]$, only packets arrived at $a_k$ or later have been transmitted, for otherwise, we could further extend $a_k$ to an even earlier time. Therefore, during the available epochs in $[a_k, d_k]$ the EDF-Schedule has continuously transmitted packets that arrived at $a_k$ 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 $[a_k, d_k]$ 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, \( P_i = (B_i, a_i, d_i) \), \( 1 \leq i \leq n \), 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 \( E_k = [e_k, e_{k+1}) \subset I[u,v] \) 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 \( d_k \leq d_u \), 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 \( a_u < a_k \), for otherwise \( P_k \) would belong to \( S[u,v] \). Third, we must have \( d_k > a_u \) because \( P_k \) is transmitted in \( E_k \subset I[u,v] \). Thus we have \( \{a_u, d_k\} \cap I[u,v] = \{a_u, a_u\} \). 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 \( E_k \), 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 rate 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 \( P_3 \) 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:

\[
r_0 = \max_{0 < d_i < B_j} \left( \frac{\sum_{i=1}^{j} B_i}{d_j - t_0} \right) \quad d_j = \arg \max_{0 < d_i < B_j} \left( \frac{\sum_{i=1}^{j} B_i}{d_j - t_0} \right)
\]

(7)

Once the pair \((r_0, d_j)\) is found, the transmission rate \( r_0 \) will be used until packet \( P_i \) 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 decision 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 \( d_2 \geq d_1 \). 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
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 $d_1$ 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, within a high density interval, then our policy adopts the 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 much more data should be sent if a higher rate is used.

We discuss the definition of the three parameters, $a$, $b$ and $\lambda$, 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.

$$a_{pre} = \frac{\sum_{t_0 \leq t \leq t_p} r(e_k) (e_{k+1} - e_k)}{t_p}$$

$$a_{new} = \frac{\sum_{t_0 \leq t \leq t_p} r(e_k) (e_{k+1} - e_k)}{t_p - t_0}$$

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

$$a = \lambda a_{pre} + (1 - \lambda) a_{new}$$

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 $\lambda$ 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.

**Fig. 5.** The rate function follows an exponential decay, starts at $a$, goes down below $r_0$ but guarantees a minimum rate $b$.
According to Eq. (9), the minimum guaranteed rate $b$ can be calculated by (10).

$$b = \begin{cases} \frac{t_k - t_0}{\beta b} & \text{if } |b| \leq r_0 < a \\ 0 & \text{if } r_0 < |b| \end{cases} \quad (10)$$

Given the ‘high temperature’ $a$ and the ‘low environment temperature’ $b$, we determine the ‘cooling factor’ $\lambda$, as follows.

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

where $A$ is a value that satisfies the following equation:

$$1 - e^{-\lambda} = \beta A \quad (12)$$

The value $A$ can be easily pre-computed given a fixed $\beta$. $d$ 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 $\lambda$, the rate prediction function applied to period $[t_0, d_j]$ is defined as follows:

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

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 $\hat{\lambda} (d)$ 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.

**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 $r_0 \geq a$, we prove this for the case of $r_0 < a$ as follows.

Since in DGC online policy, $\lambda = \frac{\pi}{2}$, thus $A = \lambda \times d$. According to (11) and (12), we have

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

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:

$$\int_{t_0}^{t_0 + d} f(t) dt = \int_{t_0}^{t_0 + d} ((a - b)e^{-\lambda(t - t_0)} + b)dt$$

$$= bd + (a - b)\left(1 - e^{-\lambda d}\right) = bd + (a - b)d$$

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

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

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 $\frac{r_0 - d_j}{\beta b} = 0.5$ and $A = 1.59$. According to (11), $\lambda \approx \frac{\pi}{2} = 1.59$, so, to compute $\lambda$, we need to decide $d$ first. Let $c = d_j - t_0$. We set $d$ as follows.

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

where $\text{avg}(\text{delay})$ 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 $\text{avg}(\text{delay})$ are also updated dynamically. In summary, the specific rate function used in the simulation is:

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

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 $U(0.1q, 1.9q)$, normal distribution $\mathcal{N}(q, 0.3q)$ and a modified exponential distribution $\text{EXP}(0.9q) + 0.1q$, 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>
<thead>
<tr>
<th>$\frac{\text{avg. packet inter-arrival time}}{\text{avg. packet delay constraint}}$</th>
<th>0.2</th>
<th>0.4</th>
<th>0.6</th>
<th>0.8</th>
<th>1.0</th>
<th>1.2</th>
<th>1.4</th>
<th>1.6</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIF($10^6$)</td>
<td>6.59</td>
<td>3.93</td>
<td>3.20</td>
<td>2.84</td>
<td>2.67</td>
<td>2.52</td>
<td>2.48</td>
<td>2.35</td>
</tr>
<tr>
<td>BA-OF($10^6$)</td>
<td>7.58</td>
<td>4.51</td>
<td>3.59</td>
<td>3.12</td>
<td>2.90</td>
<td>2.70</td>
<td>2.63</td>
<td>2.48</td>
</tr>
<tr>
<td>DGC($10^6$)</td>
<td>7.18</td>
<td>4.33</td>
<td>3.50</td>
<td>3.07</td>
<td>2.87</td>
<td>2.68</td>
<td>2.62</td>
<td>2.48</td>
</tr>
<tr>
<td>BA-OF/DIF (%)</td>
<td>115.0</td>
<td>114.8</td>
<td>112.0</td>
<td>110.0</td>
<td>108.5</td>
<td>107.2</td>
<td>106.3</td>
<td>105.7</td>
</tr>
<tr>
<td>DGC/DIF (%)</td>
<td>108.9</td>
<td>110.4</td>
<td>109.2</td>
<td>108.1</td>
<td>107.3</td>
<td>106.5</td>
<td>105.9</td>
<td>105.5</td>
</tr>
</tbody>
</table>

Table 4

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

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


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.