



# Delay Fairness Using the Burst Assembly for Service Differentiation

Viet Minh Nhat Vo , Van Hoa Le  and Manh Thanh Le

**Using various offset times to separate differential services is the most common form of service differentiation in optical burst switching networks. In this approach, a larger offset time is given to a higher priority burst, but it causes this burst to have a longer delay. One solution to this problem is to adjust the burst assembly time so that the buffering delay of the higher priority burst is always shorter than that of the lower priority burst. However, this adjustment causes another problem, called delay unfairness, for bursts with differential priorities that share the same path to their destination. This article proposes a new solution for delay fairness using the burst assembly.**

**Keywords:** Burst assembly, Delay fairness, Delay reduction, Ingress OBS node, Service differentiation.

## I. Introduction

Optical burst switching (OBS) is considered to be the best solution for optical switching because it exploits the high capacity of fibers and relaxes the complexity of electronic processes [1]. In an OBS network, the control channels have to go through O/E/O conversions at its intermediate nodes, while data loads are completely optically switched at burst granularity in the data channels; therefore, data transparency and statistical multiplexing can be achieved simultaneously. The basic principle of communication in OBS networks is that the burst control packet (BCP) is separate from its data load. Specifically, the data arriving at an ingress node (for example, IP packets and asynchronous transfer mode cells) that have the same destination (for example, the same egress node) are aggregated into larger carriers, called “bursts.” When a burst is completed, its BCP is sent ahead of an offset time. This is so that the BCP has enough time to configure the switches so that when its burst arrives, the switches will be ready. At its egress node, the burst is demultiplexed to recover the original data.

The end-to-end delay of data through an OBS network is mainly caused by four components: (1) the assembly delay, (2) the offset time, (3) the switching delay at core nodes, and (4) the propagation delay inside the OBS network [2]. The last two delays usually depend on the selected path and its available bandwidth, so they cannot be reduced with an implemented protocol. Only the first two components, the assembly delay and offset time, which are combined into a common component, the buffering delay, can be reduced. This article focuses on reducing the buffering delay.

Some burst assembly models have been proposed to reduce the buffering delay [3]–[5], in which the main idea is to send the BCP early, before the completion of its

---

Manuscript received Aug. 10, 2017; accepted Apr. 9, 2018.

Viet Minh Nhat Vo (corresponding author, vvmnhat@hueuni.edu.vn) and Van Hoa Le (levanhhoa@hueuni.edu.vn) are with the Hue University, Vietnam.

Manh Thanh Le (lmthanh@hueuni.edu.vn) is with the Department of Information Technology, University of Sciences, Hue University, Vietnam.

This is an Open Access article distributed under the term of Korea Open Government License (KOGIL) Type 4: Source Indication + Commercial Use Prohibition + Change Prohibition (<http://www.kogil.or.kr/info/licenseTypeEn.do>).

burst. In order to support service differentiation, the authors in [5] combined the model of sending the BCP early with service differentiation based on offset time. One problem of offset time-based service differentiation is that a higher priority assigned to the burst leads to a longer buffering delay, because a larger offset time is assigned to the higher priority burst. In fact, the priority of a burst depends on the data being carried within it; for example, bursts carrying video or VoIP traffic will have high priority while bursts with best-effort traffic are given low priority [6]. Thus, the long delay of a higher priority burst may violate the upper delay bound of the data. To resolve this problem, the authors in [5] proposed a solution to adjust the burst assembly time so that the buffering delay of the high priority burst is always shorter than that of the low priority burst. However, this adjustment causes another problem known as delay unfairness for bursts with differential priorities that share the same path to their destination. This article proposes a new solution for delay unfairness using the burst assembly.

The remainder of this paper is organized as follows: Section II briefly presents the previous proposals for burst assembly-based for delay reduction in which the model for service differentiation is considered. Based on the delay unfairness of the previous models, a burst assembly model for delay fairness is proposed in Section III, which includes a definition of delay fairness, measure for delay fairness, 2-phase burst assembly model, and the burst assembly algorithm for delay fairness. The simulation results are compared and analyzed in Section IV, and the conclusion is given in Section V.

## II. Related Work

To reduce the buffering delay, the authors in [3]–[5] proposed a burst assembly model in which the BCP is sent early, before the completion of its burst. As shown in Fig. 1b, this model reduces the period of offset time in comparison with that of the basic burst assembly model (Fig. 1(a)).

However, because the burst length information must be carried in the BCP, the authors in [3]–[5] have proposed various approaches for estimating the length of the burst that will be completed (called the estimated length). Specifically, the authors in [3] determine the estimated length based on the rate of data arriving during the estimation time ( $T_e$ ), which is  $T_a - T_o$ . The authors in [4] calculate this length based on the difference between the arrival data rate of the current assembly with that of the previous assembly while the authors in [5] use an auto adaptive regression linear filter and the arrival data rate

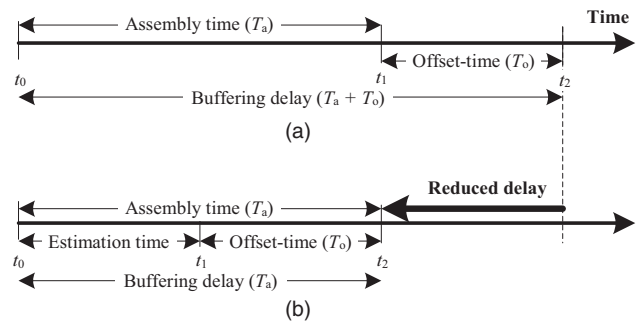


Fig. 1. Comparison of the buffering delays of the (a) basic burst assembly and (b) delay-reduction burst assembly models.

during  $T_e$ . In summary, the estimation time plays an important role in which a longer estimation time leads to a more accurate estimated length.

In the abovementioned models of burst assembly, only the one in [5] is for service differentiation. Specifically, the authors in [5] set various offset times for differential priority bursts (Fig. 2) and adjust the assembly times so that the higher priority given to a burst leads to a shorter buffering delay. This adjustment was interpreted as a solution to delay unfairness. However, there are some drawbacks to this approach: (1) higher priority bursts have shorter estimation times (as shown in Fig. 2), leading to a lower the estimation accuracy; (2) no solution was proposed to determine (or adjust) the assembly times for delay fairness among the various levels of priority of bursts; and (3) no measure was suggested to evaluate the level of delay fairness achieved. Our following proposals include a definition of delay fairness, measure for delay fairness, 2-phase burst assembly model, and the burst assembly algorithm for delay fairness.

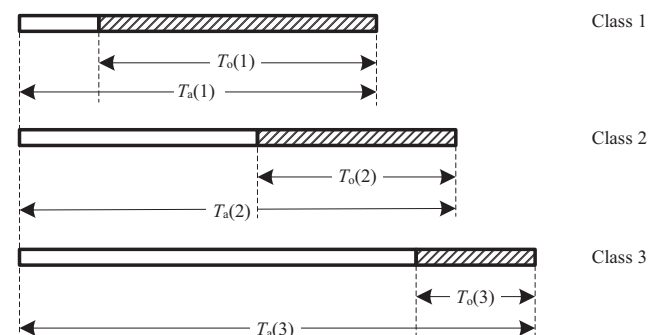


Fig. 2. Example of three assembly times (thresholds) and three offset times for three classes of priority, in which the highest priority burst (class 1) has the least assembly time and largest offset time, while the lowest priority burst (class 3) has the largest assembly time and least offset time.

### III. Our Solution For Delay Fairness Using The Burst Assembly

#### 1. Definition of Delay Fairness

There are several interpretations of the concept of fairness, but their main idea is satisfaction for individuals, such as the allocation of network resources to various applications so that fairness is achieved when these resources are allocated fairly [7]. In OBS networks, fairness has been studied in the form of rate fairness [8] and distance fairness [9]. Specifically, rate fairness refers to fair bandwidth allocation for connections and is based on the ratio of the allocated bandwidth to the available bandwidth of outgoing links; distance fairness is the issue of fairness between long-path bursts and short-path bursts, where the long-path burst has a higher blocking probability because of its longer path. The concept of delay fairness proposed in [5] is interpreted as giving higher priority bursts shorter buffering delays. However, this interpretation does not yet express the essence of satisfaction for individuals in the concept of fairness. Therefore, we propose a definition of delay fairness for differential priority bursts that share the same path to their destination in an OBS network as follows.

Delay fairness is the satisfaction of the delay for differential priority bursts so that the ratio of the average end-to-end delay to its upper delay bound for each priority is approximate. In addition, to meet the requirements of delay-based priority differentiation and upper end-to-end delay bounds in OBS networks, two constraints for the differential priority bursts are added:

- Higher priority bursts have shorter end-to-end delays;
- The end-to-end delay of each burst is not longer than its upper delay bound (for example, the round-trip time (RTT) of the packets included in the burst).

#### 2. Measure For Delay Fairness

As mentioned in Section I, the end-to-end delay of a burst through an OBS network depends primarily on the buffering delay at ingress nodes. Therefore, the fairness of buffering delay for differential priority bursts is the main cause of the fairness of their end-to-end delay. In this article, we focus on optimizing the delay fairness using the burst assembly.

Given  $D(i)$ , the average delay of the data that waits in queue  $i$  before being aggregated in a burst, and  $T_a(i)$ , the assembly time of the queue  $i$ ,  $x_i = D(i)/T_a(i)$  reflects the rate of data delay in queue  $i$ . We propose the delay fairness index (DFI) to measure the delay fairness for

differential priority bursts, which is based on Jain's formula [10], as follows.

$$DFI = \frac{\left(\sum_{i=1}^n \sigma_i x_i\right)^2}{n \sum_{i=1}^n (\sigma_i x_i)^2}, \tag{1}$$

where  $n$  is the number of priority classes.

The fairness increases when  $DFI$  approaches 1 and reaches its maximum if  $DFI = 1$ . This is equivalent to

$$\sigma_1 x_1 \approx \sigma_2 x_2 \approx \dots \approx \sigma_n x_n, \tag{2}$$

where  $\sigma_i$  is the weight of  $x_i$ ,  $0 < \sigma_i < 1$  and  $\sum_{i=1}^n \sigma_i = 1$ . In this article, these weights are assumed to be equal, so  $\sigma_i$  is excluded from (1).

Note that  $D(i)$  is a variable component that depends on the rate of data arriving at queue  $i$ . If we want to push  $DFI$  to 1, we must adjust  $T_a(i)$  so that all  $x_i$  are close together. As in the example shown in Fig. 3, with the distribution of each  $x_i$  in the space of  $(D, T_a)$ , the adjustment of  $DFI$  to 1 is equivalent to the movement of all  $x_i$  to their center of mass. Specifically, each  $x_i$  is moved as follows:

- Determine the center of mass of all  $x_i$ :  $\bar{x} = (\sum_{i=1}^n x_i)/n$
- Move to the center of mass:  $x_i = x_i + \eta \cdot (\bar{x} - x_i)$ , where  $\eta$  is the movement step,  $0 \leq \eta \leq 1$ . We chose  $\eta = 0.1$  for our simulation implementation (Section IV).

#### 3. Two-Phase Burst Assembly Model

Our burst assembly model is also based on the idea of sending the control packet early, as in [3]–[5], but our improvements come from the combination of two phases of burst assembly: Phase 1 is estimation time-based assembly and Phase 2 is estimated length-based assembly. Specifically, the model of the 2-phase burst assembly is as follows:

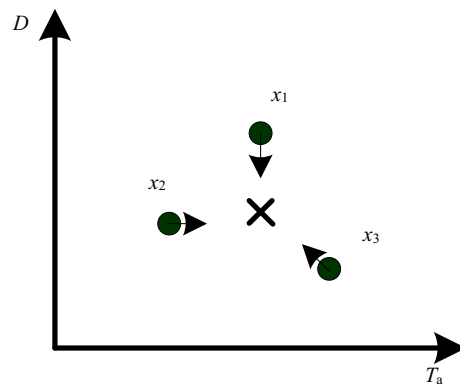


Fig. 3. Example of three priority bursts and the distribution of their  $x_i$  in the space of  $(D, T_a)$ .

• **Phase 1:** As the first packet arrives at the empty queue  $i$  (Line 3 of the burst assembly for delay fairness (BADF) algorithm in Section III.4), its timer is triggered. The control packet is sent only when the timer reaches the estimation time of  $T_e(i) = T_a(i) - T_o(i)$  (Line 9). The estimated length of burst  $L_e(i)$  is then calculated using the time-window-based exponentially weighted moving average (TW-EWMA) algorithm [11] (Lines 11–13) as follows.

$$L_e(i) = T_a(i) \times ((1 - \alpha(i)) \times \lambda_{avg}(i) + \alpha(i) \times \lambda_{cur}(i)), \tag{3}$$

where  $\alpha(i)$  is a weight factor,  $\lambda_{avg}(i)$  is the average rate of previously arriving packets, and  $\lambda_{cur}(i)$  is the average rate of currently arriving packets. The latter is calculated using  $\lambda_{cur}(i) = L(i)/T_e(i)$ , where  $L(i)$  is the number of packets arriving during the estimation time. However, we flexibly adjust the value of  $\alpha(i)$  depending on the increase/decrease in incoming traffic using the formula of  $\alpha(i) = \lambda_{cur}(i) \times (\lambda_{avg}(i) + \lambda_{cur}(i))$  (Line 14), instead of keeping it fixed, as in the original TW-EWMA algorithm.

• **Phase 2:** The algorithm of burst assembly is still continued, but now based on the estimated length  $L_e(i)$  or the assembly time  $T_a(i)$ . A burst is only completed if one of these thresholds ( $L_e(i)$  or  $T_a(i)$ ) is reached (Line 16).

The estimated length-based approach of Phase 2 will help minimize the estimation error (the difference between the estimated and completed lengths); specifically, it is zero when all arriving packets have the same size or estimated length  $L_e(i)$  is a common multiple of all arriving packet sizes. In the case when the estimated length is reached and the arriving packets have various sizes, the condition for completing a burst is  $L_e(i) - \max_p(i) \leq |b(i)| < L_e(i)$ , where  $|b(i)|$  is the length of the completed burst  $b$  and  $\max_p(i)$  is the possible maximum size of the packets arriving in queue  $i$ . This approach will cause a bit of waste in terms of bandwidth (when the bandwidth reserved based on an estimated length that is larger than the real length of the completed burst), but it ensures that no excess packet is moved to the next burst and, therefore, there is no additional delay. In the case when the assembly time of  $T_a(i)$  is reached first, the estimation error should increase because the length of the completed burst is shorter than the estimated one; but no delay is added to the burst.

Another action in Phase 2 to increase the fairness delay is to adjust  $T_a(i)$  so that all  $x_i$  are close together (see Section III.2). This is done by first determining the center of all  $x_i$  (Line 22) and then modifying  $T_a(i)$  for the next assembly based on the movement of  $x_i$  (Line 23).

#### 4. Burst Assembly Algorithm For Delay Fairness

Based on the abovementioned analysis, our algorithm for BADF is described as follows:

---

##### BADF Algorithm

**Input:** the packets arriving at queue  $i$ .

**Output:**  $D_{avg}(i)$  // the average delay of the packets in queue  $i$

**Begin**

```

1:  $\alpha(i) = 1; R_E(i) = 0; b(i) = \emptyset;$  // initiate the parameters
2:  $D_{avg}(i) = T_a(i)/2;$ 
3: while a packet  $p$  arrives do
4:   if  $b(i) = \emptyset$  then // the packet  $p$  arrives at empty queue  $i$ .
5:      $t(i) = s_p;$  // turn on timer  $t(i)$  using the arrival time of  $p$ .
6:      $T_e(i) = T_a(i) - T_o(i);$  // reset the parameters
7:      $t_1(i) = T_e(i) + s_p;$ 
8:   end if
9:    $b(i) = b(i) + \{p\};$  // enter  $p$  into queue  $i$ 
10:  if  $t(i) \geq t_1(i)$  then // Phase 1: send BCP
11:     $L(i) = |b(i)|;$  // measure the current length
12:     $\lambda_{cur}(i) = L(i)/T_e(i);$ 
13:     $\lambda_{avg}(i) = (1 - \alpha(i)) \times \lambda_{avg}(i) + \alpha(i) \times \lambda_{cur}(i);$ 
14:     $L_e(i) = L(i) + T_o(i) \times \lambda_{avg}(i);$  // estimate the completed length
15:     $\alpha(i) = \lambda_{cur}(i) / (\lambda_{cur}(i) + \lambda_{avg}(i));$  // update the parameters
16:     $t_1(i) = \infty;$ 
17:  end if
18:  if  $(|b(i)| \in [L_e(i) - \max_p(i), L_e(i)])$  or  $(t(i) \geq T_a(i))$  then
    // Phase 2: send burst
19:     $L(i) = |b(i)|;$  // measure the completed length
20:     $b(i) = \emptyset;$  // send the completed burst
21:     $x_i = D(i)/T_a(i);$  //  $D(i)$  is the measured delay of
    // the current burst assembly
22:     $\bar{x} = \frac{\sum_{j=1}^n x_j}{n};$  // determine the center.
    // adjust  $T_a(i)$  for the next assembly
23:     $T_a(i) = \begin{cases} T_a(i) & \text{if } D(i)/(x_i + \eta \cdot (\bar{x} - x_i)) \leq T_o(i) \\ D(i)/(x_i + \eta \cdot (\bar{x} - x_i)) & \text{if } T_o(i) < D(i)/(x_i + \eta \cdot (\bar{x} - x_i)) < RTT(i) \\ RTT(i) & \text{otherwise} \end{cases}$ 
    // calculate the estimation error:
24:     $R_E(i) = (1 - \alpha(i)) \times R_E(i) + \alpha(i) \times |L(i) - L_e(i)|/L(i);$ 
25:     $D_{avg}(i) = (D_{avg}(i) + D(i))/2;$  // adjust the average delay
26:  end if
27: end while
End

```

---

The complexity of BADF for one cycle of burst assembly is  $O(\log N(i))$ , where  $N(i)$  is the number of packets arriving in one cycle of burst assembly (which is also the average completed length  $L(i)$  in Phase 2).

### IV. Simulation Results and Analyses

The BADF algorithm and Sui’s algorithm [5] were implemented in NS2, with the support of package obs0.9a, on a PC with a 2.4 GHz Intel Core 2 CPU and 2G RAM.

The given arriving packets belong to three priority classes ( $n = 3$ ); thus, there are three queues used for burst assembly at the ingress nodes. The arrival process of the packets at the queues is assumed to be Poisson and their sizes are uniformly distributed in the interval [500, 1,000]. Assuming that the upper delay bounds (in ms) of the arriving packets (for example, their RTT) are distributed in the interval [0.4, 1.0], the packets with upper delay bounds in the interval [0.4, 0.6) are allocated to queue 1 (high priority), those with upper delay bounds in the interval [0.6, 0.8) are allocated to queue 2 (medium priority), and the rest, with upper delay bounds in the interval [0.8, 1.0], are allocated to queue 3 (low priority). The offset times of 0.3, 0.2 and 0.1 ms are also assigned to queues 1, 2 and 3, respectively.

We consider three periods of simulation (in s) with various arriving loads: equal loads of 0.2 (Erlang) for three priority classes in the period of [0.1, 0.3]; loads of 0.3, 0.2, and 0.1 arriving at queues 1, 2, and 3, respectively, in the period of [0.4, 0.6]; and loads of 0.1, 0.2 and 0.3 arriving at queues 1, 2, and 3, respectively, in the period of [0.7, 0.9].

Our simulation goals are to

- Compare the DFI of BADF and Sui’s algorithm;
- Analyze the effect of delay fairness on the assembly time  $T_a(i)$  and buffering delay;
- Compare the estimation errors of BADF and Sui’s algorithm.

#### 1. Comparison of DFI for BADF and Sui’s Algorithm

As shown in Fig. 4, the DFI of BADF is close to 1, which is better than that of Sui’s algorithm. This means that the delay fairness of BADF approaches the optimum.

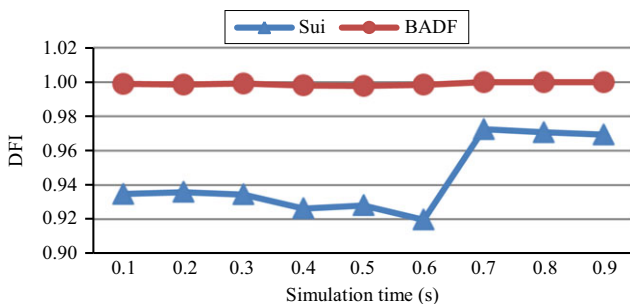


Fig. 4. Comparison of DFI for BADF and Sui’s algorithm.

To clarify which factor affects the delay fairness, we consider the rate of data delay in each queue ( $x_i = D(i)/T_a(i)$ ) of both BADF and Sui’s algorithm. As shown in Fig. 5, the three classes of Sui’s algorithm are far from each other in all periods of simulation (dashed rectangles), but they are very close in BADF (solid rectangles). This reflects the fact that the  $x_i$  of all classes are pushed closely toward their center of mass in BADF.

#### 2. Analysis of the Influence of DFI on Assembly Time $T_a(i)$ and Buffering Delay

As shown in Fig. 6, assembly time  $T_a(i)$  decreases when the rate of arriving packets increases, as with class 1 in the period of [0.4, 0.6] and with class 3 in the period of [0.7, 0.9]. This is understandable because when the number of arriving packets increases, the length threshold is usually reached first, so the value of  $T_a(i)$  is decreased to the real assembly time and, as a result, the gap between  $T_a(i)$  and the average delay of packets in their queue is narrowed. When arriving packets decrease, as in the period of [0.4, 0.6] for class 3 and in the period of [0.7, 0.9] for class 1,

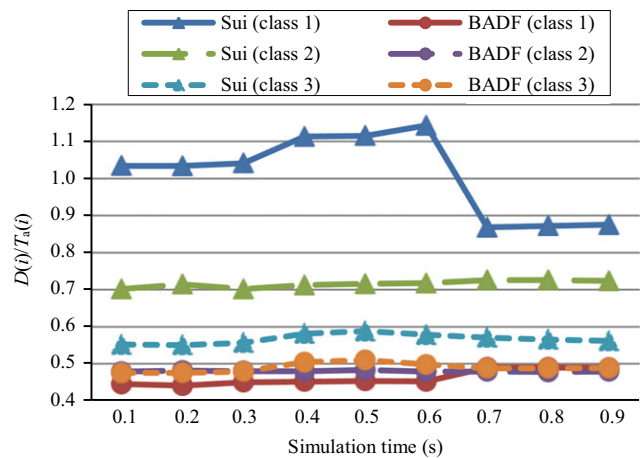


Fig. 5. Comparison of  $x_i = D(i)/T_a(i)$  among three classes for BADF and Sui’s algorithm.

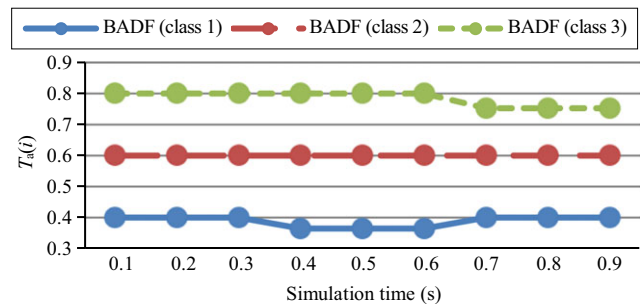


Fig. 6. Comparison of  $T_a(i)$  for three classes in BADF.

their  $T_a(i)$  are likely to increase, but because of the limit of their upper bounds (0.4 for class 1 and 0.8 for class 3), no change occurs, as shown in Fig. 6.

Optimizing the DFI also has a significant impact on the buffering delay of the priority classes. Specifically, the average buffering delay of all three classes of BADF is lower than that of Sui's algorithm (Fig. 7). This is more evident when considering each individual class, as shown in Fig. 8, in which the buffering delay of each class of BADF is always lower than that of Sui's algorithm.

However, one thing to note is that there is a significant estimation error (see Section IV.3) in Sui's algorithm when the number of arriving packets increases, and thus the completed length is longer than the estimated one. Hence the excess packets in the current assembly must be transferred to the next assembly. As a result, the excess packets are subjected to an additional delay that is equal to  $T_a(i)$ . Thus, the real average buffering delay of Sui's algorithm is much larger, as shown in Fig. 9. Figure 10 also shows the real average buffering delay of

the three classes of Sui's algorithm in comparison with BADF when considering the additional delay.

### 3. Comparison of the Estimation Error of the Burst Assembly Models

The estimation error is defined as the difference between the completed length and the estimated one. In this article, we define the average estimation error rate ( $R_E$ ) of  $M$  successive burst assemblies as follows:

$$R_E = \frac{\sum_{i=1}^M (|L(i) - L_e(i)|/L(i))}{M}, \quad (4)$$

where  $L(i)$  and  $L_e(i)$  are the completed and estimated lengths, respectively, in the  $i$ th burst assembly.

Figure 11 compares the estimation errors of the BADF and Sui's algorithms, in which the estimation error of the BADF algorithm is much smaller than that of Sui's algorithm. This is achieved because of our proposed 2-

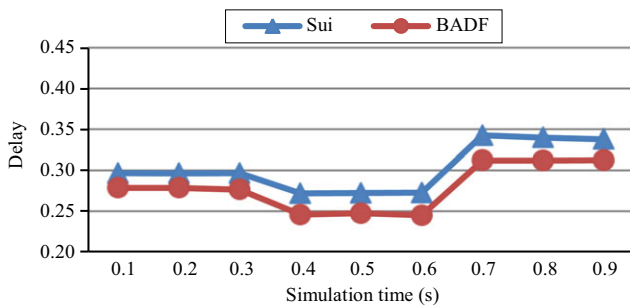


Fig. 7. Comparison of the average buffering delay of BADF and Sui's algorithm without considering the additional delay.

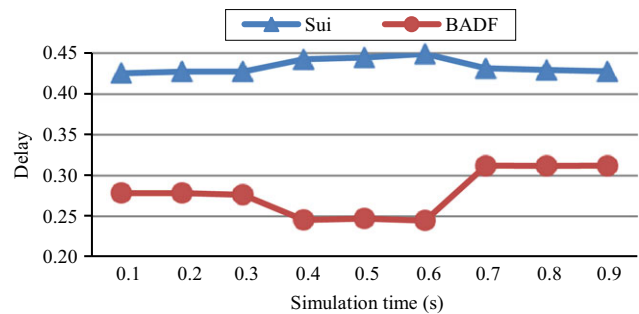


Fig. 9. Comparison of the average buffering delay of BADF and Sui's algorithm considering the additional delay.

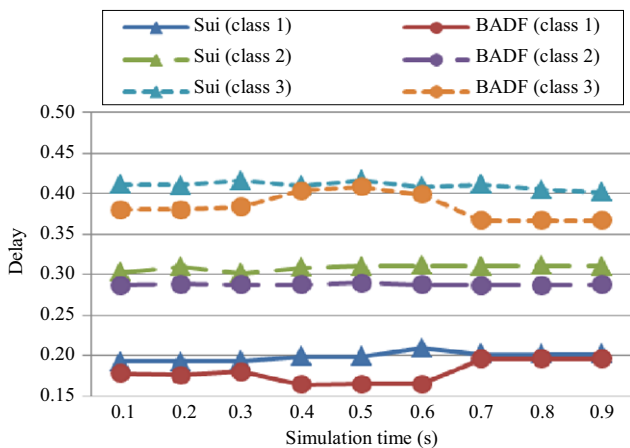


Fig. 8. Comparison of the buffering delay of three classes of BADF and Sui's algorithm without considering the additional delay.

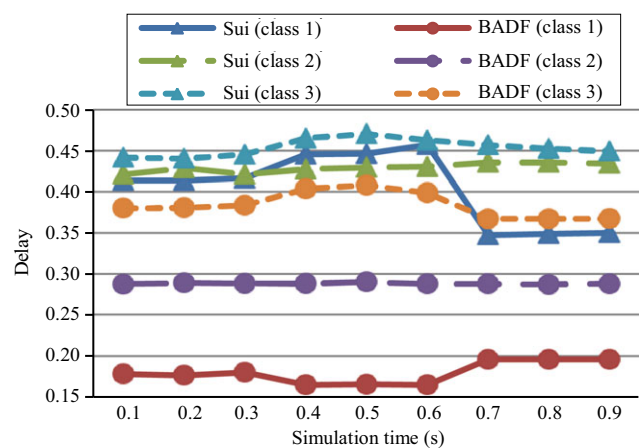


Fig. 10. Comparison of the average buffering delay of three classes of BADF and Sui's algorithm considering the additional delay.

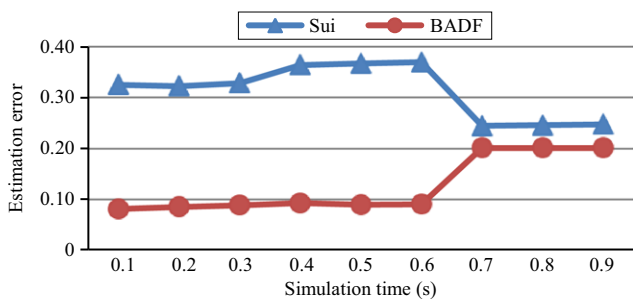


Fig. 11. Comparison of estimation error of BADF and Sui's algorithm.

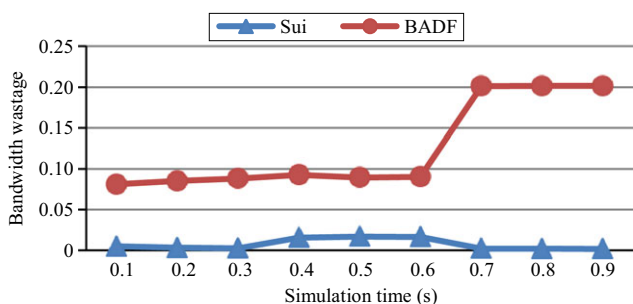


Fig. 12. Comparison of bandwidth wastage of BADF and Sui's algorithm.

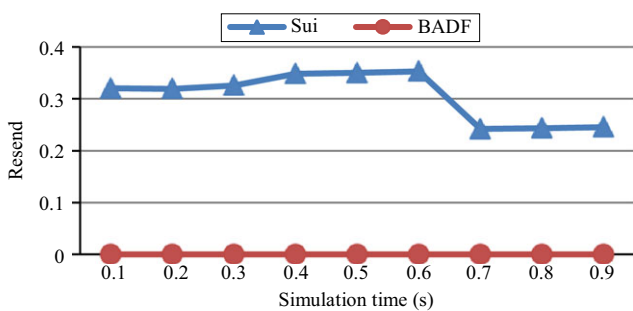


Fig. 13. Comparison of the excess packets of BADF and Sui's algorithm.

stage assembly algorithm in which a burst is formed when the estimated length or time threshold is reached. If the estimated length is reached first, no estimation error occurs; but if the time threshold is reached first, there is an estimation error because the real number of arriving packets is less than the estimated one. This estimation error will waste bandwidth, as shown in Fig. 12, but it does not generate any excess packets (Fig. 13); thus, no additional delay is created.

A comparison of the estimation error for each class in the BADF algorithm and Sui's algorithm is also shown in Fig. 14. With the BADF algorithm, the estimation error of class 1 is increased in the period of [0.7, 0.9] because the reduction of the class 1 packet rate causes the time threshold

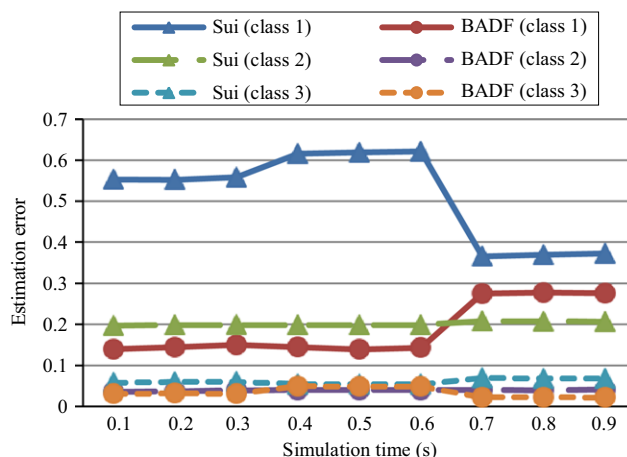


Fig. 14. Comparison of the estimation error for each class of BADF and Sui's algorithm.

to always be reached first. In Sui's algorithm, because class 1's time threshold is fixed, when the rate of the class 1 packets increases (in the period of [0.4, 0.6]) or decreases (in the period of [0.7, 0.9]), there is always a certain increase or decrease, respectively, in estimation errors.

### V. Conclusion

The burst assembly has a significant impact not only on the end-to-end delay but also on the delay fairness for the priority bursts through OBS networks that support service differentiation. In this paper, we proposed a burst assembly model for delay fairness, named BADF, that includes a definition of delay fairness, measure for delay fairness, 2-phase burst assembly model, and burst assembly algorithm for delay fairness. The simulation results show that our BADF algorithm achieves near optimal delay fairness, reduced buffering delay, minimal estimation error, and zero excess packets in comparison with previously proposed models.

### References

- [1] Y. Chen, C. Gico, and X. Yu, "Optical Burst Switching: A New Area in Optical Networking Research," *IEEE Netw.*, vol. 18, no. 3, May 2004, pp. 16–23.
- [2] G. Hu and M. Kohn, "Evaluation of Packet Delay in OBS Edge Nodes," *Int. Conf. Transpar. Opt. Netw.*, Nottingham, UK, June 18–22, 2006, pp. 66–69.
- [3] T. Hashiguchi, X. Wang, H. Morikawa, and T. Aoyama, "Burst Assembly Mechanism with Delay Reduction for OBS Networks," *Conf. Opt. Internet/ Australian Conf. Opt. Fiver Technol.*, Melbourne, Australia, July 13–16, 2003, pp. 664–666.

- [4] T. Mikoshi and T. Takenaka, "Improvement of Burst Transmission Delay Using Offset Time for Burst Assembly in Optical Burst Switching," *Asia-Pacific Symp. Inform. Telecommun. Technol.*, Bandos Island, Maldives, Apr. 22–24, 2008, pp. 13–18.
- [5] Z. Sui, Q. Zeng, and S. Xiao, "Adaptive Assembly on Delay Fairness in Optical Burst Switched Networks," *J. Opt. Commun.*, vol. 27, no. 1, 2006, pp. 35–38.
- [6] S. Askar, G. Zervas, and D. Hunter, "Service Differentiation for Video Applications Over OBS Networks," *Eur. Conf. Netw. Opt. Commun.*, Newcastle-Upon-Tyne, UK, July 20–22, 2011, pp. 200–203.
- [7] R. Denda, A. Banchs, and W. Effelsberg, "The Fairness Challenge in Computer Networks," *First COST 263 Int. Work. Quality Future Internet Services*, Berlin, Germany, Sept. 25–26, 2000, pp. 208–220.
- [8] T. Orawiwattanakul, Y. Ji, Y. Zhang, and J. Li, "Fair Bandwidth Allocation in Optical Burst Switching Networks," *J. Lightw. Technol.*, vol. 27, no. 16, Aug. 2009, pp. 3370–3380.
- [9] C.-F. Hsu and L.-C. Yang, "On the Fairness Improvement of Channel Scheduling in Optical Burst-Switched Networks," *Photonic Netw. Commun.*, vol. 15, no. 1, Feb. 2008, pp. 51–66.
- [10] R. Jain, D.-M. Chiu, and W.R. Hawe, "A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer System," *DEC Technical Report TR301*, vol. cs.NI/9809, no. DEC-TR-301, 1984, pp. 1–38.
- [11] K. Salad and F. Haidari, "On the Performance of a Simple Packet Rate Estimator," *IEEE/ACS Int. Conf. Comput. Syst. Applicat.*, Doha, Qatar, Mar. 31–Apr. 4, 2008, pp. 392–395.



**Viet Minh Nhat Vo** received his MS degree in communication engineering from the Franco Institute for Computer Science (IFI), Vietnam, in 2000 and his PhD in cognitive informatics from the University of Quebec in Montreal, Canada, in 2007. Since 2008, he has been with Hue University, Vietnam, where he is now an associate professor. His research interests include optical packet/burst-based switching, quality of service, optimization, and traffic engineering.



**Van Hoa Le** received his MS degree in computer science from Hue University, Vietnam, in 2013. Since 2009, he has been with Hue University, Vietnam, where he is now a lecturer. His research interests are in the fields of all-optical networks with emphasis on packet/burst-based switching, burst assembly, fairness, and quality of service.



**Manh Thanh Le** received his PhD in Mathematics and Physics from Otvos Lorand University (ELTE), Hungary, in 1994. Since 1995, he has been with the Department of Information Technology, University of Sciences, Hue University, Vietnam, where he is now an associate professor. His main research interests include data mining, decision support systems, image retrieval, logic programming, deductive databases, knowledge-based systems, and computer networks.