Improved adaptive-threshold burst assembly in optical burst switching networks

Jiuru Yang (杨九如)[1,2], Gang Wang (王 钢)[1], and Shilou Jia (贾世楼)[1]

[1]*Communication Research Center, Harbin Institute of Technology, Harbin 150001*
[2]*Institute of Electronic Engineering, Heilongjiang University, Harbin 150080*

Received January 4, 2007

An improved adaptive-threshold burst assembly algorithm is proposed to alleviate the limitation of conventional assembly schemes on data loss and delay. The algorithm will adjust the values of assembly factors according to variant traffic regions. And the simulation results show that, by using the adaptive-factor adaptive assembly scheme, the performance of networks is extensively enhanced in terms of burst loss probability and average queuing delay.

OCIS codes: 060.4250, 060.4510.

As fine granularity and the non-buffer nature, optical burst switching (OBS) has been regarded as one of the main supporting technologies for next-generation optical internet, which combines the best of optical circuit switching and optical packet switching[1]. In OBS networks, contention resolution is one of the hot topics and much work has been devoted to them, such as channel scheduling, deflection routing, burst segment/drop etc.[2]. But most of these methods require the support of either tunable wavelength converters (TWCs) or optical buffer devices (e.g. fiber delay lines (FDLs)). It is no doubt that the addition of TWCs and FDLs must bring more complex and larger switch fabric at each core node. Also, as far as the current immature technologies on optical buffers and converters, in the near future, it will be unpractical to introduce the abundant expensive TWCs and FDLs into transparent optical networks. A feasible and reasonable way has been proposed for the current optical networks to simplify the complexity of core nodes as much as possible and let much work has been done at edge nodes electronically[3]. Thus, adopting burst assembly implemented at edge nodes to avoid burst contention has attracted much attention recently, because of the processing flexibility and abundant cheap electronic random-access memory (RAM)[4−8]. In this letter, we propose a novel assembly scheme called adaptive-factor adaptive-threshold-assembly (AFATA) to alleviate the weakness of conventional adaptive threshold assembly in terms of burst loss probability and average queuing delay.

Basically threshold-based burst assembly schemes can be categorized into two types: 1) fixed threshold assembly (FTA)[4] and 2) adaptive threshold assembly (ATA)[5,6], whose performances are highly determined by the design parameters, such as burst size and assembly period etc.[7]. In the FTA scheme, the burst threshold is a constant in all traffic states, e.g., fixed-assembly-period (FAP) and fixed-assembly-size (FAS). Taking the dynamic traffic into account, it is obvious that the FTA scheme is too rigid to improve transmission efficiency, though it is easy to be implemented with low complexity. Alternatively, in order to enhance the bandwidth usage on the premise that low loss and delay are guaranteed, adaptive-assembly-size (AAS) algorithm[5] is proposed, which adjusts the values of threshold according to the dynamic traffic. For the AAS algorithm, the authors design a control-window on burst size (BS) with upper bound $Q_{\rm high}$ and lower bound $Q_{\rm low}$ (see Fig. 1). Specially, under a certain traffic condition, when the designed assembly threshold is $Q_{\rm low}$, a low burst loss rate (e.g. 10^{-6}) should be guaranteed. In addition, because a data burst variation is assumed not over 2% in Ref. [5], the value of $(Q_{\rm high} - Q_{\rm low})$ is equal to $Q_{\rm low} \times 2\%$. Then, within the predetermined assembly period T, if $Q_{\rm low} < {\rm BS} < Q_{\rm high}$, the size-window will be unchanged. Once ${\rm BS} > Q_{\rm high}$, the window will increase by one step, that is, $Q_{\rm high} = Q_{\rm high} + \Delta{\rm size}$ and $Q_{\rm low} = Q_{\rm low} + \Delta{\rm size}$, where Δsize denotes the change of burst size per step. Similarly, when ${\rm BS} < Q_{\rm low}$, $Q_{\rm high} = Q_{\rm high} - \Delta{\rm size}$ and $Q_{\rm low} = Q_{\rm low} - \Delta{\rm size}$. But the value of $Q_{\rm high}$ should be less than the maximum ${\rm BS}_{\rm max}$ and $Q_{\rm low}$ is larger than the minimum ${\rm BS}_{\rm min}$. Whereas, considering the bursty of traffic, for example, the traffic load is suddenly changed from light (heavy) load to heavy (light) load, this step-by-step assembly scheme is clearly not enough flexible. Comparatively, based on the time-sensitive nature of TCP flow, Cao *et al.* proposed a more flexible and proportional-based assembly scheme called adaptive-

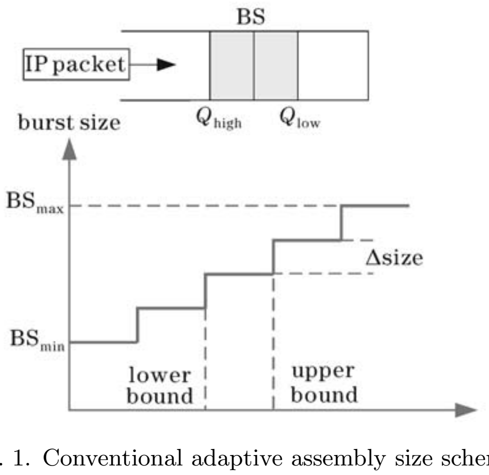

Fig. 1. Conventional adaptive assembly size scheme.

assembly-period (AAP)[6]. Briefly, the AAP algorithm can be depicted as

$$\mathrm{AP}_{\mathrm{qsd}} = \alpha \times \frac{\mathrm{AvgBL}_{\mathrm{qsd}}}{\mathrm{Bandwidth} \times \mathrm{Channel}}, \quad (1)$$

where $\mathrm{AP}_{\mathrm{qsd}}$ and $\mathrm{AvgBL}_{\mathrm{qsd}}$ are the assembly period and average burst length of queue q_{sd} respectively, α is the constant called assembly factor.

However, since the controlling-object is TCP flow, it is certain that the performance on edge delay and bandwidth usage, which is directly related to the assembly period, is dramatically improved by using AAP. But the price is that continuous blocking and huge variation in burst size are caused easily[5]. Then, to overcome the weakness in AAP, we propose a proportional-based adaptive-assembly-size (PAAS) algorithm that adjusts the threshold of burst size proportionally in accordance with the change of IP traffic. And this algorithm can be depicted as

$$\Delta T_{\mathrm{p}} = T_{\mathrm{p}} - \mathrm{Avg}[T_{\mathrm{b}}], \quad (2)$$

$$[\mathrm{Th}i]_{\mathrm{b},i} = \mathrm{Avg}[L_{\mathrm{b}}] - F\frac{\Delta T_{\mathrm{p}}}{\mathrm{Avg}[T_{\mathrm{b}}]} \cdot \mathrm{Avg}[L_{\mathrm{b}}], \quad (3)$$

where $\mathrm{Avg}[T_{\mathrm{b}}]$ is the average assembly time and $\mathrm{Avg}[L_{\mathrm{b}}]$ is the burst length when assembly time is $\mathrm{Avg}[T_{\mathrm{b}}]$ at traffic load $\rho = 0.4$ (traffic load is defined as $\rho = \sum \lambda / m\mu$, where m is the number of data channels); $T_{\mathrm{p}} = \sum_{i=1}^{\mathrm{Avg}[L_{\mathrm{b}}]} (\frac{1}{\lambda_i} + \frac{1}{\mu})$ is the sum of arrival time of multiple packets sampled, λ_i is packet arrival rate with mean λ, μ is the service rate, ΔT_{p} is the change of T_{p}, $[\mathrm{Th}i]_{\mathrm{b},i}$ is the length-threshold of Burst i, and F is called assembly factor. From Eq. (2), it is apparent that T_{p} characterizes the real-time traffic state of IP flow. So, according to ΔT_{p}, the assembler will adjust the value of $[\mathrm{Th}i]_{\mathrm{b},i}$ at edge nodes. And once the queue length of a burst, $L_{\mathrm{b},i}$, reaches or exceeds $[\mathrm{Th}i]_{\mathrm{b},i}$, this burst will be sent immediately. Our work is then concentrated on the quantification of the PAAS algorithm on burst loss probability (BLP) and average queuing delay (AQD). Besides, to alleviate the synchronous effect and overlarge edge delay, we set the lower and upper limits of burst length denoted by MBL and MAL respectively for all bursts. That means, if $L_{\mathrm{b},i} > \mathrm{MAL}$, we set $L_{\mathrm{b},i} = \mathrm{MAL}$, and when $L_{\mathrm{b},i} < \mathrm{MBL}$, then $L_{\mathrm{b},i} = \mathrm{MBL}$. Especially, from Ref. [7], when BS = 10 kB, the loss rate reaches 10^{-2} even at $\rho = 0.3$, and if BS > 200 kB, the block performance is its asymptotic value in all traffic states. So, in this letter, assuming each IP packet is 500 bytes, we then set MBL = 30 and MAL = 250. Furthermore, to examine the capability of assembly algorithms, we assume that the offset time for all bursts is the same and the original burst length L_0 is equal to 100 at $\rho = 0.4$.

Then the simulation results of the effect of assembly factor F on data loss and queuing delay are shown in Fig. 2 from a typical simulation scenario used in Ref. [8]. In Fig. 2(a), comparing the numerical results when F is equal to 0.5 and 5 respectively, a lower BLP no higher than 10^{-5} is obtained when $F = 0.5$ in light load. Conversely, under the condition of $\rho = 1.0$, the BLP when

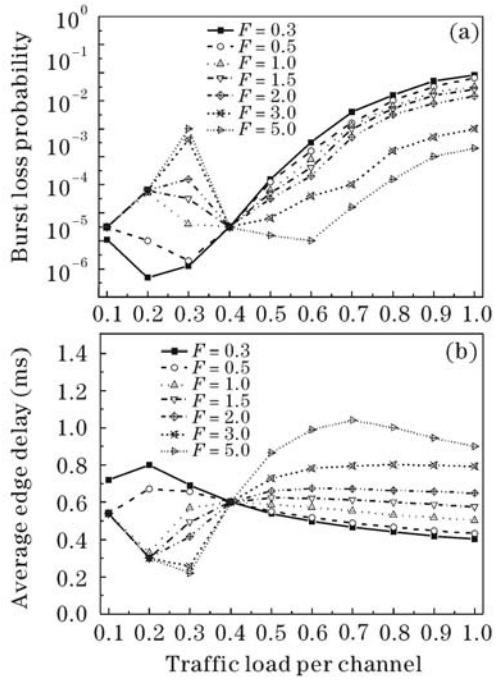

Fig. 2. Performance evaluation on the PAAS algorithm.

$F = 5$ is just about 1/100 that when $F = 0.5$. And similar results are also gained in Fig. 2(b) on AQD. Such an interesting result on the selection of F is mainly caused by the negative exponential distribution of assembly period when we assume the arrival of IP packets is a Poisson process. For instance, in light load ($\rho < 0.4$), due to $T_{\mathrm{p}} > \mathrm{Avg}[T_{\mathrm{b}}]$, from Eq. (1), a positive and large value of ΔT_{p} is gotten. Substituting this large value into Eq. (3), if F is small (large), $[\mathrm{Th}i]_{\mathrm{b},i}$ is large (small), we then get a low (high) loss but large (small) queuing delay. On the other hand, in heavy load ($\rho > 0.4$), ΔT_{p} is a negative and small value. Thereby, if F is small (large), $[\mathrm{Th}i]_{\mathrm{b},i}$ is small (large), and a high (low) loss but small (large) delay is obtained. Consequently, for the PAAS algorithm, the performance of networks is absolutely limited by assembly factor F, and there may be an inevitable contradiction that always exists in getting low BLP and AQD in all traffic states.

Fortunately, from Fig. 2, in a certain traffic region, the PAAS algorithm with variant values of F presents a better performance on both BLP and AQD. As a result, in order to improve the weakness of PAAS with fixed-factor and provide high quality of service (QoS), a novel adaptive-assembly size algorithm called adaptive-factor adaptive-assembly-size (AFAAS) is proposed, which adjusts the values of F in variant traffic states accordingly. First, the input traffic load at each edge node is divided into three variant traffic regions — light load (ρ : 0.1 − 0.3), normal load (ρ : 0.3 − 0.7), and heavy load (ρ : 0.7 − 1.0). Then the AFAAS algorithm is applied by selecting variant values of F in variant traffic regions: in light load, a small value of F denoted by F_1 is enough to get a low BLP with an assured low AQD; similarly, to guarantee the lower BLP, two larger assembly factors should be selected in normal load and heavy load, denoted by F_2 and F_3, respectively. So Eq. (2) is substituted by

$$[\mathrm{Th}i]_{\mathrm{b},i} = \begin{cases} \mathrm{Avg}[L_\mathrm{b}](1 - F_1 \cdot \eta_{T_\mathrm{p}}) & \rho \in 0.1 - 0.3 \\ \mathrm{Avg}[L_\mathrm{b}](1 - F_2 \cdot \eta_{T_\mathrm{p}}) & \rho \in 0.3 - 0.7 \\ \mathrm{Avg}[L_\mathrm{b}](1 - F_3 \cdot \eta_{T_\mathrm{p}}) & \rho \in 0.7 - 1.0 \end{cases}, \quad (4)$$

where $\eta_{T_\mathrm{p}} = \Delta T_\mathrm{p}/\mathrm{Avg}[T_\mathrm{b}]$ is the change ratio of ΔT_p.

Here we evaluate the performance of the AFAAS algorithm in terms of BLP and AQD. Based on the analysis above and Fig. 2, the assembly values are chosen as $F_1 = 0.5$, $F_2 = 3$, and $F_3 = 5$. Furthermore, aiming at comparing with other assembly schemes with different granularities, for the FTA schemes, we select middle granularity (FAS: $L_\mathrm{b} = 100$) and large granularity (FAP: $T_\mathrm{b} = 2$ ms). And for the assembly factor F in PAAS, a middle value ($F = 2$) and a large value ($F = 5$) are chosen respectively. Then, the numerical results by simulation are shown in Fig. 3. From Fig. 3, whether the algorithm FAS or FAP is adopted, a large and intolerable AQD exists though a very low BLP ($< 10^{-6}$) is gotten in light load. Instead, comparing with PAAS, we get a low BLP ($< 10^{-5}$) but acceptable AQD in light load by using AFAAS. Again, when $0.4 < \rho < 0.7$ and $0.7 < \rho < 1$, BLP $< 10^{-4}$ and AQD < 0.8 ms, BLP $< 10^{-3}$ and AQD < 1 ms, respectively. That indicates that, with an appropriate decision on assembly factors, the AFAAS algorithm multi-assembly-factor based has better performance than other assembly algorithms in that data loss and queuing delay. Note that, for the AFAAS algorithm, a better numerical result may be obtained if we choose various values of assembly factor in each traffic state. However, considering the computational complexity of assembly algorithm, it is not necessary for the overmuch traffic partition.

Further, we set a target requirement on BLP and AQD

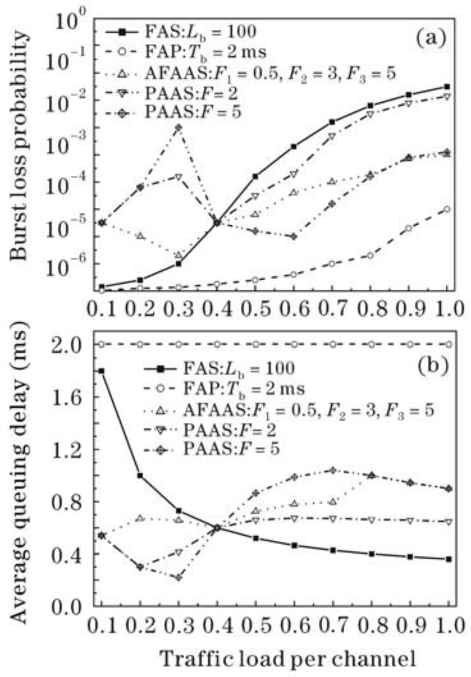

Fig. 3. Performance comparison of various burst assembly.

(Table 1) and the evaluating results for various assembly schemes are shown as Table 2. It is apparent that, from Table 2, AFAAS is the optimal scheme to satisfy the given target requirement except for AQD in light load. But, from Fig. 3, the value of AQD in AFAAS, not over 0.7 ms, is very near to the target even in light load.

Table 1. Target Requirement

Traffic	Light	Normal	Heavy
Loss Rate	$< 10^{-5}$	$10^{-5}- < 10^{-4}$	$< 10^{-3}$
Queuing Delay	< 0.6 ms	$0.6 - 0.8$ ms	< 1 ms

Table 2. Performance Evaluation for Target Requirement

		FAS[a]	FAP[b]	AFAAS	PAAS	
				(0.5,3,5)	$F = 2$	$F = 5$
Light	BLP	√	√	√		
	AQD				√	√
Normal	BLP		√	√		
	AQD	√		√	√	
Heavy	BLP		√	√		√
	AQD	√		√	√	√

a: $L_\mathrm{b} = 100$; b: $T_\mathrm{b} = 2$ ms.

In conclusion, in order to overcome the limitation of assembly parameters and enhance the flexibility of assembly algorithm, based on the variant traffic states, an improved algorithm AFAAS with adaptive-factor is proposed. And the numerical results by simulation indicate that a good trade-off between low burst loss and queuing delay is obtained within all traffic states by using our algorithm and it is also the unique assembly scheme to satisfy the given target requirement.

This work was partially supported by the Space Flight Supporting Technology Foundation of China under Grant No. 03-1.4. J. Yang's e-mail address is totti9555@163.com.

References

1. M. Yoo, C. Qiao, and S. Dixit, IEEE. J. Sel. Area in Commun. **18,** 2062 (2000).
2. Y. Chen, C. Qiao, and X. Yu, IEEE Network Magazine **18,** 16 (2004).
3. W. A. Vanderbauwhede and D. A. Harle, J. Lightwave Technol. **23,** 2215 (2005).
4. A. Ge, F. Callegati, and L. S. Tamil, IEEE Commun. Lett. **4,** 98 (2000).
5. S. Oh, H. Hong, and M. Kang, ETRJ Journal **24,** 311 (2002).
6. X. Cao, J. Li, Y. Chen, and C. Qiao, in *Proceedings of IEEE GLOBECOM '2002* **21,** 2815 (2002).
7. J. Y. Choi, J. S. Choi, and M. Kang, IEICE Trans. Commun. **E88-B,** 3855 (2005).
8. J. Yang, G. Wang, and S. Jia, in *Proceedings of IEEE ICCT '2006* **1,** 428 (2006).