A New Method for Control-plane Congestion Control in Chip-scale Optical Networks

Shi Xu[1,a], Renfa Li[1], Quanyou Feng[2, b] , Wenhua Dou[1]

[1]Department of Information Science and Engineering, Hunan University, Changsha, 410073, China

[2] School of Computer Science, National University of Defense Technology, Changsha, 410073, China

[a]email:xushi9018@aliyun.com, [b]email: fengquanyou@nudt.edu.cn

Keywords: Many-core, Optical On-chip Network, Control Congestion

Abstract. Recently, various optical networks have been proposed for chip-scale communication on large-scale many-core chips, including optical burst-switching (OBS). The overall throughput of OBS network is dually constrained by congestion and loss in both the data-plane (i.e., burst contention) and the control-plane (i.e., header packet losses). While previous studies mainly focus on maximizing data-plane throughput, little work has been done on the control-plane congestion problem. In chip-scale OBS, massive fine-grained data bursts, stringent transmission delay budget and moderate network operation frequency have actually constrained the electronic header processing capability, leading to serious network congestion. This paper proposes a new approach based on ingress regulation to overcome the control-plane congestion problem. In our scheme, before being injected, every concurrent packet flow is globally regulated and coordinated so that the aggregated flows do not exceed the header processing capacity of routing nodes, leading to the alleviation of control-plane congestion. We use real application traces to evaluate our method. Simulation results validate that the proposed approach can effectively minimize the control-plane congestion and improve system performance in terms of transmission delay and burst loss rate.

1. Introduction

In the coming Exascale computing era [1], limited power budget and enormous memory bandwidth demand are two important challenges for high performance processors design. Base on the rapid development of silicon photonic devices [2], many optical network technologies have been used to provide energy-efficient and high throughput chip-scale interconnection network for manycore system, for example wavelength routing [3], photonic circuit-switching [4], and chip-scale optical burst switching (OBS) [5]. Among these paradigms, OBS technology combines the flexibility and maturity of electronic processing with the huge bandwidth potential and scalability of all-optical switching [6]; it is theoretically a very plausible candidate for the design of low-power and high-bandwidth manycore interconnection network.

However, in general OBS network, due to the random nature of burst arrivals at core nodes, control-plane congestion occurs when the short-term arrival rate of headers at a core node exceeds the maximum rate at which they can be processed. Finite processing capacity of the electronic header processors thus significantly affect the maximum throughput that can be supported [7]. This problem gets even worse in chip-scale OBS. Unlike macro OBS networks (e.g., metropolitan OBS network and the Internet backbone OBS network), 1) chip-scale OBS network is characterized by massive short bursts (fine-grained control messages, like memory read/write requests) that have stringent requirements on communication delay; 2) the operating frequency of chip-scale OBS network is constrained by thermal constraint and limited power budget, and therefore cannot be very high. All these features definitely intensify the burden and complexity of control-plane operations, resulting in frequent network congestion and finally limited system throughput.

Although there have been many studies that aim to maximize throughput by resolving contention and avoiding burst collisions in the OBS data-plane[9], there have been very few studies that consider the loading and congestion of the control planes. That is because control-plane congestion

in general macro OBS network happens infrequently. To date, the pioneer work on this problem is done by N. Barakat and T. E. Darcie [7] [8]. Using queuing theory, they studied the control-plane stability constraint in a single core node and proposed some useful guidelines.

In this paper, we focus on chip-scale OBS network and propose a new approach to address the control-plane congestion problem by flow regulation [13]. Using network calculus[10], our work first study the worst-case end-to-end communication delay and minimum loss rate for each OBS header packets flow. And then, we develop a non-linear optimization model to select the optimal regulator parameters. Simulation results with real application traces show that our approach can effectively resolve the control-plane congestion and achieve considerable performance improvements in terms of network delay and burst losses rate.

The paper's outline is as follows. In Section 2, we introduce the basics of network calculus briefly. We present the worst-case delay bound model and minimum burst loss rate analysis of OBS traffic flows in Section 3. Based on these models, we formulate our new approach as an optimization problem and present our optimization method in Section 4. In Section 5, we present simulation results and discussions. The paper is concluded in Section 6.

2. Background -- Network Calculus

Cruz [11] and Chang [12] have pioneered the study of network calculus [10], which is a mathematical framework to derive worst case bounds on maximum latency, backlog, and minimum throughput. Le Boudec and Thiran [10] summarized the results of network calculus and their applications in Internet and ATM. The authors in [13] proposed a network calculus-based flow regulation and defined a regulation spectrum as a design instrument for SoC architects to control QoS.

In network calculus, a flow f is generally an infinite stream of unicast traffic sent from a source node and flow j is denoted as f_j. $f_j(t)$ represents the accumulated number of bits transferred in the time interval $[0,t]$. To obtain the average and peak characteristics of a flow, traffic specification[13] is used. With traffic specification, f_j is characterized by an arrival curve $\alpha_j(t) = \min(M_j + p_j t, \sigma_j + \rho_j t)$ in which M_j is the maximum transfer size, p_j the peak rate ($p_j \geq \rho_j$), σ_j the burstiness ($\sigma_j \geq M_j$), and ρ_j the average (sustainable) rate. We denote it as $f_j \propto (M_j, p_j, \sigma_j, \rho_j)$.

Network calculus uses the abstraction of service curve to model a network element (node) which processes traffic flows. A well-formulated service model is the latency-rate function $\beta_{R,T} = R(t-T)^+$, where R is the minimum service rate and T is the maximum processing latency of the node [10]. Notation $x^+ = x$ if $x > 0$; $x^+ = 0$ otherwise.

3. Network calculus model for OBS control-plane network

3.1 An OBS core node

Figure 1 illustrates a general architecture of the core node in chip-scale OBS mesh network, in which the JET [6] signaling protocol is implemented. Data channels are connected to the optical cross connect and control channels are terminated at the header packet router after O/E conversion. According JET, header packets are issued out an offset time earlier before their data burst payload. The header packet router computes the routing Table based on pre-defined protocols (such as, dimension order routing in optical meshes) and forwards the control packet to the input queue of corresponding burst scheduler. There is a scheduler for each output link, which runs the LAUC [6] scheduling algorithm. Before the arrival of data burst, the switch controller configures the optical cross connect according to the commands issued by the schedulers.

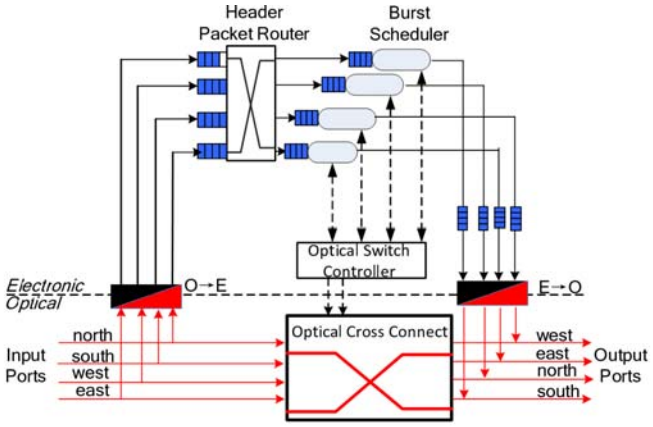

Fig.1. A general architecture of OBS router

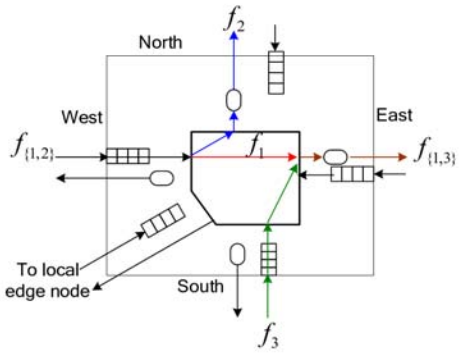

Fig.2. Resource sharing due to flow aggregation

Since transmission of headers and payloads are physically separate, bursts can get lost because of congestion in the outgoing data-plane wavelength channels, excessive delay in the header packet routers or failure operation of the LAUC scheduler due to intra-channel burst overlapping [14]. Since our work focuses on control-plane issues, we assume, throughout the paper, there are enough wavelengths and wavelength convertors in chip-scale OBS so that data-plane congestion due to channel shortage will not happen. So, our work focuses on studying the impact of excessive delay in the header packet routers which are caused by finite electronic processing capacity. In this section, based on network calculus, we derive worst-case delay bounds and minimum burst loss rate model for each header traffic flow. In the following section, we elaborate our flow regulation method that addresses the control-plane congestion issues encountered by massive concurrent OBS flows.

3.2 Worst-Case Delay Bounds

When we apply network calculus to compute the worst-case end-to-end delay bound for an OBS flow, the basic idea is to transform the original network into a feed-forward network, in which a tandem of network elements (i.e., buffers and schedulers) provide exclusive services to a single flow without loops. Then, we can apply the tandem theorem[10] to compute equivalent service curve of the route path. As can be observed in Figure 1 and Figure 2, header packets are delivered via shared wavelength channels and buffers; different flows may interfere with each other along the overlapped route. These aggregated scheduling and interference must be resolved first in order to derive end-to-end delay bound.

At each router, we identify two types of resource sharing, namely, *channel sharing* and *buffer sharing*. *Channel sharing* means that multiple flows from different buffers share the same outport and thus the output link bandwidth. As shown in Figure 2, flows, f_1 and f_3, which are from the West, and South inport, respectively, share the East output link. The switch arbitrates and sends the two flows to the East outport. *Buffer sharing* means that an aggregate flow, which are to be split, share a buffer. As also illustrated in Figure 2, flow $f_{\{1,2\}}$ shares the buffer at the West inport, which is an aggregate of two flows, f_1, and f_2 , and to be split by the switch to the East, and North outport, respectively.

Analysis of Link Sharing

Without loss of generality, we consider two flows f_1 and f_3 share one output link as shown in Figure 2. We assume that the router performs the weighted round-robin scheduling; the flows are served according to their configured weight, ϕ_i for flow f_i. In each round, for a non-empty buffer encountered, the router will try to serve up to ϕ_i bursts before moving to the next buffer. Assume the service rate of the router is C bursts/cycle per link, the maximum length of a round is consequently equal to $\frac{\sum_i \phi_i}{C}$ cycles and the time for flits of flow f_i to be forwarded within a round is bounded by $\frac{\phi_i}{C}$ cycles. The service offered to one flow completely depends on the weight of the flow. In weighted round-robin scheduling, the worst-case appears when a flow just misses its slot in

the current round. Consequently it will have to wait for its slot at the next round. In the worst case, flits will have to wait up to $\frac{\sum_{j\neq i}\phi_j}{C}$ cycles to be served. Hence the equivalent service curve for f_1 can be derived as

$$\overline{\beta}_1^R = \frac{\phi_1}{\phi_1+\phi_3}\beta^R \otimes \delta_{\frac{\phi_3}{C}} \tag{1}$$

Analogously, the equivalent service curve for f_3, equals to

$$\overline{\beta}_3^R = \frac{\phi_3}{\phi_1+\phi_3}\beta^R \otimes \delta_{\frac{\phi_1}{C}} \tag{2}$$

Analysis of Buffer Sharing

As drawn in Figure 2, an aggregate flow $f_{\{1,2\}}$ sharing the same input buffer is to be split to different outports. Flits of $f_{\{1,2\}}$ are stored in the input buffer and served by the switch of the router in the FIFO order. The equivalent service curve for an individual flow f_i depends also on the arrival curve of its contention flows at the ingress of the buffer. For f_1, the equivalent service curve can be derived as $\overline{\beta}_1^R = \varepsilon(\beta^R, \alpha_2)$, where $\varepsilon(..,..)$ is a function to compute the equivalent service curve. Thus, according to [10], we obtain the output arrival curve of f_1 as $\alpha_1^* = \alpha_1 \otimes \varepsilon(\beta^R, \alpha_2)$. In the same way, we can obtain f_2 's equivalent service curve $\overline{\beta}_2^R = \varepsilon(\beta^R, \alpha_1)$ and output arrival curve as $\alpha_2^* = \alpha_2 \otimes \varepsilon(\beta^R, \alpha_1)$.

We give an example to compute $\varepsilon(..,..)$. If the router provides a latency-rate service curve [10], i.e., $\beta^R(t) = \gamma_{0,C} \otimes \delta_T(t) = C(t-T)^+$, and f_2 is constrained by an affine arrival curve, i.e., $\alpha_2(t) = \gamma_{b_2,r_2}(t) = r_2 t + b_2$, then applying Corollary 4.5 in [10], the equivalent service curve for f_1 can be computed as:

$$\overline{\beta}_1^R = \varepsilon(\beta^R, \alpha_2) = \gamma_{C.s,C-r2} \otimes \delta_{T+\frac{b_2}{C}+s,} \quad s \geq 0 \tag{3}$$

where s is an intermediate argument for computing the least upper delay bound [10].

Besides these two sharing model, the burst scheduler also provides services to each flow. We assume that a scheduler is only responsible for the configuration of a single wavelength channel, i.e., it works on aggregated outgoing flows. So we may view it as a constant delay element β^γ for each flow.

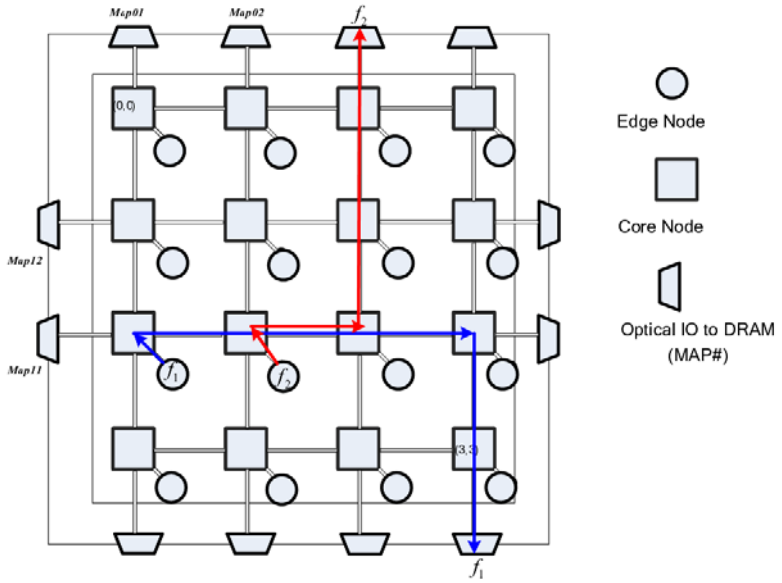

Fig.3. A 4x4 OBS mesh network

Now, based on the service curve for input buffer sharing β^α, outing channel scheduling β^φ

and burst scheduler β^{γ}, we can compute the equivalent service curve β^{R} of a control-plane router for OBS flow f as

$$\beta^{R} = \beta^{\alpha} \otimes \beta^{\varphi} \otimes \beta^{\gamma} \tag{4}$$

End-to-end delay bound analysis

To compute worst-case delay bound for a flow of OBS header packets, we first analyze the service curve of individual router $\beta^{Ri}(i=1...n)$ along the route and then, by applying the tandem theorem again, we get the equivalent service curve β.

$$\beta = \beta^{R1} \otimes \beta^{R2} \otimes \mathrm{L} \otimes \beta^{Rn} \tag{5}$$

For example, Figure 3 shows a 4x4 OBS mesh network, in which dimensional order routing is used to transfer data between on-chip core clusters and off-chip memory. There are 2 flows, f_1, and f_2. Flow f_1 is injected from ingress of edge node $(2,0)$ and leaves at Map03;it traverses a tandem of routers $\{R_{(2,0)}, R_{(2,1)}, R_{(2,2)}, R_{(2,3)}, R_{(3,3)}\}$, loosely denoted as $\{R_{(2,0)} \to R_{(3,3)}\}$; f_2 is injected from ingress of edge node $(2,1)$ and leaves at Map07;it traverses $\{R_{(2,1)}, R_{(2,2)}, R_{(1,2)}, R_{(0,2)}\}$ (denoted as $\{R_{(2,1)} \to R_{(0,2)}\}$); At router $R_{(2,1)}$, f_1 and f_2 share the same output link. At router $R_{(2,2)}$, f_1 and f_2 share the same input buffer. We transform the original network into a fee-forward model as shown in Figure 4.

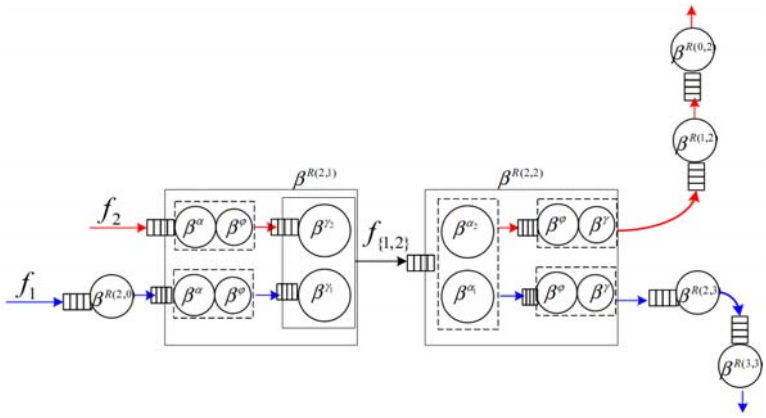

Fig.4. feed-forward network

Base on the feed-forward network, we get the service curve for f_1 as

$$\beta_1^{R_{\{(2,0)\to(3,3)\}}} = \beta^{R(2,0)} \otimes \beta_{f_1}^{R(2,1)} \otimes \beta_{f_1}^{R(2,2)} \otimes \beta^{R(2,3)} \otimes \beta^{R(3,3)} \tag{6}$$

where $\beta_{f_1}^{R(2,1)} = \beta^{\alpha} \otimes \beta^{\varphi} \otimes \frac{\phi_1}{\phi_1+\phi_2}\beta^{\gamma} \otimes \delta_{\frac{\phi_2}{C}}$ and $\beta_{f_1}^{R(2,2)} = \varepsilon(\beta^{\alpha}, \alpha_2) \otimes \beta^{\varphi} \otimes \beta^{\gamma}$.The service curve for f_2 is

$$\beta_2^{R_{\{(2,1)\to(0,2)\}}} = \beta_{f_2}^{R(2,1)} \otimes \beta_{f_2}^{R(2,2)} \otimes \beta^{R(1,2)} \otimes \beta^{R(0,2)} \tag{7}$$

where $\beta_{f_2}^{R(2,1)} = \beta^{\alpha} \otimes \beta^{\varphi} \otimes \frac{\phi_2}{\phi_1+\phi_2}\beta^{\gamma} \otimes \delta_{\frac{\phi_1}{C}}$ and $\beta_{f_2}^{R(2,2)} = \varepsilon(\beta^{\alpha}, \alpha_1) \otimes \beta^{\varphi} \otimes \beta^{\gamma}$, thus the delay bound for f_1 and f_2 can be derived as $D_1 = H(\alpha_1, \beta_1^{R_{\{(2,0)\to(3,3)\}}})$ and $D_2 = H(\alpha_2, \beta_2^{R_{\{(2,1)\to(0,2)\}}})$, in which $H(..,..)$ is the function to compute the maximum horizontal distance between the arrival curve and the service curve.

3.3 Minimum burst losses rate

Due to the random nature of burst arrivals at core nodes, control-plane congestion can occur in an OBS network when the short-term arrival rate of headers at a core-node exceeds the maximum rate at which they can be processed. To minimize data loss during these periods, arriving headers are generally placed in a header queue until they can be processed. The overall delay experienced by an arriving header at an OBS core node is therefore equal to the sum of its own header-processing duration and the duration spent waiting in the processing queue. If the cumulated

delay along its path exceeds the header's delay budget as defined by its offset time, the header will be dropped because of insufficient time for burst scheduling before its burst's arrival, and the optical burst will get lost. This control-plane loss is different from the commonly studied data-plane loss that occurs when all of the channels on outgoing links are full. In order to minimize overall throughput, a practical objective in any OBS network design should be to properly provision the network such that the number of headers that expire before service is negligible. In this section, we apply network calculus on the analysis of control-plane losses and derive the minimum burst loss rate bound for each OBS flow.

Recall that in our assumptions, with enough wavelengths and convertors, the LAUC scheduler can always find an eligible wavelength channel. The only scenario where a data burst gets lost happens when the cumulated delay of its header packet violates the JET offset time budget. So the burst loss rate here is the same as the header packet loss rate. Furthermore, we also assume that the delay bound in a single flow is the same among packets, because they all take the same route path and the offset-time is also the same. So, all the network nodes encountered by a header flow can be deemed to be a delay bounded network element [10] equivalently.

Losses of a delay bounded element in network calculus, can be modeled using the "clipper" model, i.e., a lossless system proceeded by a "clipper", as shown in Figure 5. The clipper drops some data when a delay constraint d would otherwise be violated. We consider the tandem of routers in section 3.2 as an equivalent network element offering a service curve β and having a delay constraint d. We denote by a the incoming traffic. We derive a representation formula for a scenario when data are discarded because of a delay constraint: any entering data must have exited the system after at most d unit of time, otherwise it is discarded.

Let $x(t)$ be the part of $a(t)$ that does actually enter the system, and let $y(t)$ be its output. The literature [10] has proven that, the amount of lost data $L(t)$ in the interval [0,t] is given by

$$L(t) \le \sup_{n \in N} \{ \sup_{0 \le s_{2n} \le \ldots \le s_2 \le s_1 \le t} \{ \sum_{i=1}^{n} [a(s_{2i-1}) - a(s_{2i}) - \beta(s_{2i-1} + d - s_{2i})] \} \} \tag{8}$$

Theorem 1(minimum loss rate): Consider a system with delay constraint d, offering a service curve β to a flow constrained by an arrival α. Then the loss rate $l(t) = \frac{L(t)}{a(t)}$ is bounded above by

$$\hat{l}(t) = \left[1 - \inf_{0<s<t} \frac{\beta(s+d)}{a(s)} \right]^{+} \tag{9}$$

Proof:

With $\hat{l}(t)$ defined by (9), we have that for any $0 \le \mu < \nu \le t$, $1 - \hat{l}(t) = \inf_{0<s\le t} \frac{\beta(s+d)}{a(s)} \le \frac{\beta(\nu - \mu + d)}{a(\nu - \mu)} \le \frac{\beta(\nu - \mu + d)}{a(\nu) - a(\mu)}$. Due to definition of the arrival curve, we know $a(\nu) - a(\mu) \le a(\nu - \mu)$. Therefore, for any $0 \le \mu < \nu \le t$, $a(\nu) - a(\mu) - \beta(\nu - \mu + d) \le \hat{l}(t) \cdot [a(\nu) - a(\mu)]$. For any $n \in ¥_0 = \{1, 2, 3 \ldots\}$ and any sequence $\{s_k\}_{1 \le k \le 2n}$ with $0 \le s_{2n} \le \mathrm{L} \le s_1 \le t$, setting $\nu = s_{2i-1}$, $\mu = s_{2i}$, and summing over i, we obtain $\sum_{i=1}^{n} [a(s_{2i-1}) - a(s_{2i}) - \beta(s_{2i-1} + d - s_{2i})] \le \hat{l}(t) \cdot \sum_{i=1}^{n} [a(s_{2i-1}) - a(s_{2i})]$. Because s_k are increasing with k, the right hand side of this inequality is always less than, or equal to $\hat{l}(t) \cdot a(t)$. Therefore, we have $L(t) \le \sup_{n \in N} \{ \sup_{0 \le s_{2n} \le \ldots \le s_2 \le s_1 \le t} \{ \sum_{i=1}^{n} [a(s_{2i-1}) - a(s_{2i}) - \beta(s_{2i-1} + d - s_{2i})] \} \} \le \hat{l}(t) \cdot a(t)$, which shows that $\hat{l}(t) \ge l(t) = L(t) / a(t)$. □

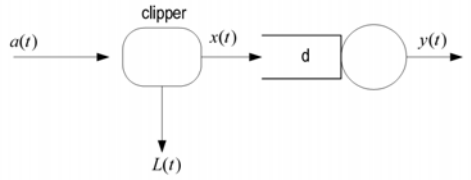

Fig.5. system losses in a bounded delay element

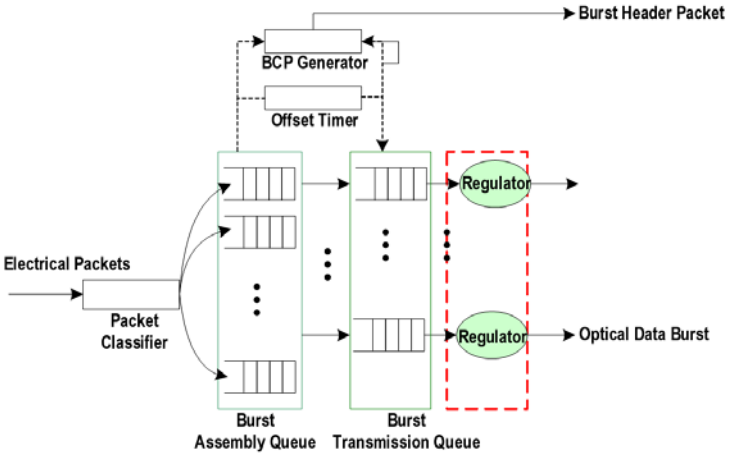

Fig.6. Schematic of the regulation method

4. The Flow regulation Method

As discussed in section 3.1, bursts in OBS network can get lost due to the congestion in data-plane and control-plane. The latter could be avoided via regulating the packets arrival rate to the electronic router, as advocated by X.Yin et al [14]. For any node in an OBS network, there generally exist an infinite number of feasible combinations of burst length and offset size that satisfy the control-plane blocking constraint [8]. Specifically, there is a trade-off between the required average burst duration and offset size in each node. A reasonable design objective is therefore to select from this feasible set the operating point that corresponds to the average minimum end-to-end control latency in the network.

4.1 Flow regulator

Our regulation method is shown in Figure 6. At the ingress node, traffic flows are first classified by the packet classifier according to the destination, and then sent into different queues to be aggregated into bursts. For simplicity in hardware implementation, bursts to different destination nodes are served by different burst assembly queues (BAQs). Bursts destined to a particular egress node, are sent to the same burst transmission queues (BTQs). Before a flow is scheduled onto the outgoing wavelength channel, a regulator is used to implement the regulation policy so that along its traversal, the flow suffers the minimum loss rate while it is ensured against violating an acceptable delay bound. The regulator uses the burst transmission queue to store incoming traffic, so it can also reduce the backpressure of burst source nodes.

The regulator is implemented using the token-bucket mechanism [15] as shown in Figure 7. The token queue has a size of σ. Initially the token queue is full. The 1-packet/token server admits one header packet by de-asserting the "stall" signal as long as the token queue is not empty. The token queue is realized by a saturating credit counter that increments at rate ρ, and saturates when it reaches a count of σ. A packet can be transmitted if and only if the credit counter is positive (at least one token available). Each time a header packet is sent, the counter is decremented by 1.

Figure 8 shows that an input flow f_j reshaped by a regulation component $R_j(p_{Rj},\sigma_{Rj})$ results in an output flow f_{Rj}. We assume the regulator has the same input and output data unit and the same input and output capacity C packets/cycle. We also assume that f_j's average bandwidth requirement must be preserved. The output flow f_{Rj} is characterized by the four parameters $(M_j, p_{Rj}, \sigma_{Rj}, \rho_j)$, where $p_{Rj} \in [\rho_j, p_j], \sigma_{Rj} \in [M_j, \sigma_j]$. f_{Rj} can be reshaped without loss by the regulator, meaning that f_{Rj} has the same M and average rate ρ as f_j. The two intervals $p_{Rj} \in [\rho_j, p_j]$ and $\sigma_{Rj} \in [M_j, \sigma_j]$ are called the regulation spectrum[13], where the former is for the regulation of peak rate and the latter for the regulation of traffic burstiness. The regulation

spectrum defines the upper and lower limits of regulation. Selecting appropriate p_{Rj} and σ_{Rj} is very effective in performance and cost of communications. These parameters will be calculated by the optimization problem in Section 4.2.

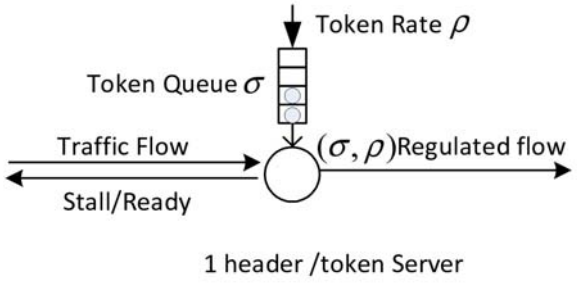

Fig.7. (ρ,σ)-based regulation mechanism

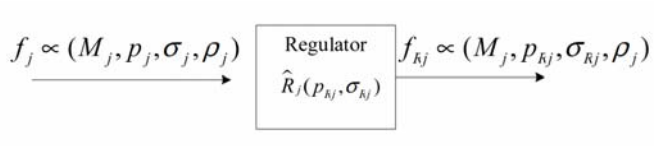

Fig.8. network calculus model for flow regulator

Note that a regulator also introduces a time delay D_{reg} for each header packet. Its bound can also be calculated by network calculus, which is $D_{reg} = H(f_j, \hat{R}_j(p_{R_j}, \sigma_{R_j}))$, where $H(..,..)$ is the function to compute the maximum horizontal distance between the arrival curve and the service curve.

4.2 The Optimization Problem

Using ingress regulation, our objective is to avoid potential congestion in the control-plane network. That is, to minimize the end-to-end control latency of each flow under the constraint of minimum burst loss rate by selecting the optimal regulator parameters (i.e., peak rate and traffic burstiness). Thus, the minimization problem $f(p_{Rj}, \sigma_{Rj})$ can be formulated as follows.

Let D_{reg_j} denote the regulator delay and D_j denotes the end-to-end delay bound. Given a set of flows $F = \{f_j \propto (M_j, p_j, \sigma_j, \rho_j)\}$, routing matrix *R*, the minimum loss rate that each flow can suffer in the network $l = \{\bar{L}_j\}$ for $\forall f_j \in F$, find the regulator parameters, peak rate p_{Rj} and traffic burstiness σ_{Rj} for $\forall f_j \in F$, such that

$$\min_{p_{Rj}, \sigma_{Rj}} \sum_{\forall f_j \in F} (D_{reg\,j} + D_j) \tag{10}$$

subject to

$$l_j \le \bar{L}_j \quad \forall f_j \in F \tag{11}$$

$$\rho_j \le p_{Rj} \le p_j \quad \forall f_j \in F \tag{12}$$

$$M_j \le \sigma_{Rj} \le \sigma_j \quad \forall f_j \in F \tag{13}$$

where p_{Rj} and σ_{Rj} for $\forall f_j \in F$ are optimization variables.

Equation (10) is the objective function of this optimization problem which minimizes end-to-end delay requirements. Constraint (11) says that the burst loss rate of each flow *j* cannot exceed the maximum budget that it can suffer in the network $\bar{L}_j$. Since we measured the flow performance in terms of its losses, we can consider $\bar{L}_j$ as a criterion of minimum guaranteed performance for flow *j*. Constraints (12) and (13) are related to two intervals $p_{Rj} \in [\rho_j, p_j]$ and $\sigma_{Rj} \in [M_j, \sigma_j]$. We see clearly that by following the above mentioned equations, we can understand the effect of optimization variables on the objective function and all constraints of the defined problem.

In the literature, (10) is called a nonconvex NLP problem [16]. There are different methods for solving this kind of optimization problems. In particular, we will solve the optimization problem (10) using interior point method for constrained NLP problems [16].

4.3 Optimization method

The interior-point approach to constrained minimization is to solve a sequence of approximate minimization functions, namely the barrier functions. Based on (10), For each $\mu > 0$, the approximate problem is

$$\min_{p_{Rj},\sigma_{Rj}} \sum_{\forall f_j \in F} (D_{reg\,j} + D_j) - \mu \sum_{i=1}^{5|F|} \ln(s_i) \tag{14}$$

subject to

$$l_j - \overline{L}_j + s_i = 0 \ \forall f_j \in F \ i = 1,\mathrm{L}\ ,|F| \tag{15}$$

$$\rho_j - p_{Rj} + s_i = 0 \ \forall f_j \in F \ i = 1,\mathrm{L}\ ,2|F| \tag{16}$$

$$p_{Rj} - p_j + s_i = 0 \ \ \forall f_j \in F \ i = 1,\mathrm{L}\ ,3|F| \tag{17}$$

$$M_j - \sigma_{Rj} + s_i = 0 \ \ \forall f_j \in F \ i = 1,\mathrm{L}\ ,4|F| \tag{18}$$

$$\sigma_{Rj} - \sigma_j \ + s_i = 0 \ \ \forall f_j \in F \ i = 1,\mathrm{L}\ ,5|F| \tag{19}$$

The approximate problem is a sequence of equality constrained problems. These are easier to solve than the original inequality-constrained problem. There are as many slack variables s_i as there are inequality constraints g. The s_i are restricted to be positive to keep $ln(si)$ bounded. As μ decreases to zero, the minimum of $f(\mu)$ should approach the minimum of f. The added logarithmic term is called a barrier function.

To simplify discussions, we transform it into vector form. Define $p_R = [p_{R1},\mathrm{L}\ ,p_{R|F|}]^T$, $\sigma_R = [\sigma_{R1},\mathrm{L}\ ,\sigma_{R|F|}]^T$, $s = [s_1,\mathrm{L}\ ,s_{5|F|}]^T$ and assume $g(p_R,\sigma_R) = [g_1(p_R,\sigma_R),\mathrm{L}\ ,g_{5|F|}(p_R,\sigma_R)]^T$ so that $g(p_R,\sigma_R)+s$ is a vector that its element is the constraints(15)-(19). The problem (14) can be rewritten as

$$\min_{p_R,\sigma_R,s} f_\mu(p_R,\sigma_R) = \min_{p_R,\sigma_R,s} f(p_R,\sigma_R) - \mu \sum_{i=1}^{5|F|} \ln(s_i) \tag{20}$$

subject to

$$g(p_R,\sigma_R) + s = 0 \tag{21}$$

To solve the problem (20), we first consider the first-order optimality condition, namely the Karush-Kuhn-Tucker (KKT) conditions. The KKT conditions are analogous to the condition that the gradient must be zero at a minimum, modified to take constraints into account. The KKT conditions use the auxiliary Lagrangian function:

$$L(p_R,\sigma_R,s,\lambda) = f(p_R,\sigma_R) - \mu \sum_{i=1}^{5|F|} \ln(s_i) + \lambda^T (g(p_R,\sigma_R) + s) \tag{22}$$

where $\lambda = (\lambda_1,\mathrm{L}\ \lambda_{5|F|})^T$ is the vector of Lagrange multipliers. At an optimal solution (p_R,σ_R,s) of the barrier problem, we have

$$\nabla_{p_R} L(p_R,\sigma_R,s,\lambda) = \nabla_{p_R} f(p_R,\sigma_R) + \Phi(p_R,\sigma_R)\lambda = 0 \tag{23}$$

$$\nabla_{\sigma_R} L(p_R,\sigma_R,s,\lambda) = \nabla_{\sigma_R} f(p_R,\sigma_R) + \overline{\Phi}(p_R,\sigma_R)\lambda = 0 \tag{24}$$

$$\nabla_s L(p_R,\sigma_R,s,\lambda) = -\mu S^{-1} e + \lambda = 0 \tag{25}$$

where $\Phi(p_R,\sigma_R) = (\nabla_{p_R} g_1(p_R,\sigma_R),\mathrm{L}\ ,\nabla_{p_R} g_{5|F|}(p_R,\sigma_R)$ and $\overline{\Phi}(p_R,\sigma_R) = (\nabla_{\sigma_R} g_1(p_R,\sigma_R),\mathrm{L}\ ,\nabla_{\sigma_R} g_{5|F|}(p_R,\sigma_R)$ are the matrixes of constraint gradients with respect to p_R and σ_R, respectively, and where $e = (1,1,1,1)^T$ and S is the diagonal matrix with its non-zero element $s_i, i = 1,\mathrm{L}\ ,5|F|$.

To solve the approximate problem, we generate a step d for displacement at an iterate k, where $d = (d_{p_R}, d_{\sigma_R}, d_s)^T$. The algorithm uses one of two main types of steps at each iteration:

- A Newton step in (p_R,σ_R,s). This step attempts to solve the KKT equations, for the approximate problem via a linear approximation.
- A CG (conjugate gradient) step, using a trust region.

By default, the algorithm first attempts to take a newton step. If it cannot, it attempts a CG step. One case where it does not take a direct step is when the approximate problem is not locally convex near the current iteration.

At each iteration, the algorithm decreases a merit function, that is

$$\psi = \sum_{\forall f_j \in F} (D_{reg\,j} + D_j) - \mu \sum_{i=1}^{5|F|} \ln(s_i) + \nu \left\| g(p_R, \sigma_R) + s \right\| \tag{26}$$

The parameter ν may increase with iteration number in order to force the solution towards feasibility. If an attempted step does not decrease the merit function, the algorithm rejects the attempted step, and attempts a new step. The objective and constraints must yield proper (double) values at the initial point.

In this paper, the proposed optimization model is realized in Matlab. Regulator parameters computed by this model are used to implement our regulation method in simulations. Further details about the optimization method can be found in [16].

5. Simulation results

5.1 Simulation Environment

We used PhoenixSim[17], which is a simulator for modeling and analyzing the performance of manycore systems that use photonic networks. The OBS network model for manycore-to-DRAM communication and regulation methods were implemented and integrated into PhoenixSim. Throughout the experiment, we considered an OBS network with 1GHZ header processing frequency. We also used a hybrid timer/size based burst assembly algorithm, which aims to introduce negligible assembly delay and guarantee that all data bursts are of the same size, 128 bytes. A header packet has a fixed size of 8 bytes.

Table 1important parameters about the cores and memory subsystem

Feature		Value
Cores		Out of order execution, 16 node per cluster
Memory hierarchy	L1 cache	Private, 32 KB (per tile), 64 byte line size, 8-way associativity, LRU replacement
	L2 cache	Private, 3 MB (per tile), 64 bytes line size, 24-way associativity, LRU replacement
	Cache coherence	Full-map directory based
DRAM	Base DRAM Frequency (MHz)	1333
	Total Memory Per MAP(GB)	2
	Bandwidth per DIMM (Gb/s)	256

Table 2 manual mapping of real benchmark races

App name	Ingress Cluster	MAP no	Traffic characteristic $(L_j, p_j, \sigma_j, \rho_j)$
cholesky	(3,3)	*Map01*	(1,1,3, 0.3)
fft	(1,2)	*Map03*	(0.9,1,2, 0.4)
fmm	(1,1)	*Map04*	(0.8,1,2, 0.4)
Lu_non_con	(0,3)	*Map06*	(1,0.9,4, 0.5)
Ocean_con	(1,3)	*Map07*	(0.9,0.7,3, 0.3)
radix	(2,3)	*Map09*	(0.7,0.9,2, 0.6)
Water_spatial	(3,2)	*Map11*	(1,0.8,2, 0.4)

We used real data traces of the SPLASH2 [18] benchmarks and mapped them onto a 4x4 OBS mesh network (see Figure 3). Each edge node is connected to an 8- core cluster. Memory requests and responses are relayed by the OBS network using XY routing between core clusters and off-chip DRAMs.

5.2 Splash2 traffic patterns

Chip-scale OBS network is actually devoted to moving data between the last layer cache(L2, or L3 cache) and off-chip DRAM banks. The impact of memory hierarchies must be considered when we generate real application traces. Therefore, we used the Graphite simulator[19], which provides accurate modeling for the memory subsystems (including cache hierarchies with full cache coherence) and cores. Table 1 lists some important parameters of our simulation setup, which are selected to mimic real execution environment.

We first run parallel applications of SPLASH2 on Graphite and collected their traces, including instructions and events from the core, network, and memory subsystem. We then manually mapped these traces onto the core cluster (see Table 2) and rerun these traces in the Phoenixsim simulator, in which the control-plane congestion phenomenon and our regulation method are modeled. Furthermore, our simulation focused on the "Cluster-to-Dram" flows, since they are more prone to control-plane congestion than "Dram-to-Cluster" flows due to their smaller transfer granularity (e.g., read /write request, bytes to be written to Dram). Table 2 also lists the traffic characteristics for "Cluster-to-Dram" flows used in our simulation.

As we mentioned before, a regulator limits the flow injection process with two parameters (peak rate and burstiness). Since there are 7 flows in the example, 14 parameters have to be assigned to regulators. To show that how these parameters heavily affect the loss rate and communication delay, we consider two different regulator sets. 1) Optimized regulators, which are optimized based on the proposed optimization problem (10). 2) Random regulation, which are not optimized. Obviously, there is a number of unoptimized configurations. Therefore we choose randomly the optimization parameters in the regulation spectrum. Then, the total maximum delay and average loss rate are calculated and depicted in Tables 3, along with values for a system without regulators.

From Table 3, we see that the optimized regulation scheme leads to larger reduction in average loss rate and smaller total maximum delay when compared with the no-regulation scheme. Also the Table shows that random regulation method somewhat decrease the loss rate and delay because of reducing the contention for shared resources, although their regulator parameters are not configured appropriately. That is because these suboptimal parameters are in the same regulation spectrum as the optimized ones. However, when we compare these suboptimal results with the ones in the optimized regulation, we see the optimal regulation scheme shows a large improvement. As a result, we can minimize total network delay and improve communications performance by assigning the peak and burstiness parameters of regulators in a wiser manner.

Table 3 Comparison for all flows in terms of delay and network loss rate

	Regulator delay	Total traversal delay	Average loss rate
No regulation	0	4583	17%
Random regulation	42	4376	16%
Optimized regulation	25	890	7%

To go into more detail, we also depict the peak rate and traffic burstiness of each flow for our regulation schemes. Figure 9 and Figure 10 show that regulators dramatically reduce peak rate and traffic burstiness of flows, respectively.

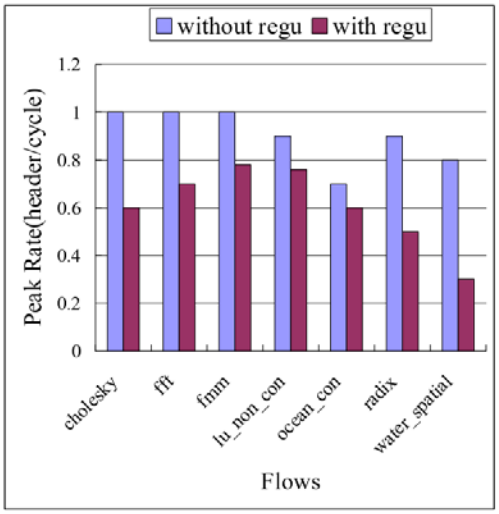

Fig.9. peak rate of flows

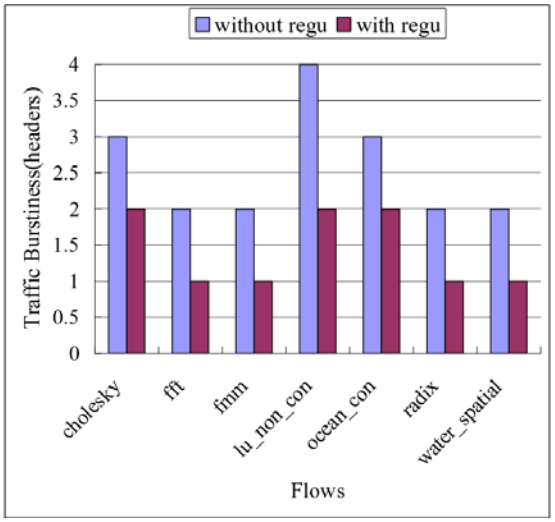

Fig.10. traffic burstiness of flows

6. Conclusions and Future Works

In chip-scale optical burst-switched networks, massive fine-grained data bursts, stringent delay requirement and constrained network operation frequency result in finite electronic header

processing capacity, which leads to serious control-plane congestion. Long transmission delay and large burst loss rate caused by the congestion may ultimately limit the maximum system throughput. So, in this paper, we propose a new approach using flow regulation to address the problem. In our approach, concurrent traffic flows are globally regulated and coordinated before being injected so that we can minimize the end-to-end delay of each control packets flow without violation of the constraint of minimum burst loss rate. Based on network calculus, we build an optimization model to select the optimal regulator parameters (i.e., flow rate and traffic burstiness). Simulation results with benchmark traces show that our approach can effectively minimize the control-plane congestion and improve system performance. Our future work will focus on dynamic regulation which can configure regulator parameters on-the-fly.

***Acknowledgments**

This research is partially supported by the grants from National Program on Key Basic Research Project of China (Grant No. 2012CB933504), National Natural Science Foundation of China (Grant No. 61402502).

References

[1] T. Agerwala. Exascale computing: The challenges and opportunities in the next decade[C], IEEE HPCA16, 2010.

[2] M. J. R. Heck, et al. Hybrid Silicon Photonics for Optical Interconnects[J], IEEE Journal of Selected Topics in Quantum Electronics, 2011.

[3] D. Vantrease, et al. Corona: System Implications of Emerging Nanophotonic Technology[C], IEEE 35th International Symposium on Computer Architecture, 2008.

[4] G. Hendry, et al. Circuit-Switched Memory Access in Photonic Interconnection Networks for High-Performance Embedded Computing[C], International Conference for High Performance Computing, Networking, Storage and Analysis (SC), 2010.

[5] F.Quanyou, et al. A Hybrid Photonic Burst-Switched Interconnection Network for Large-Scale Manycore System[J], IEICE Transactions on Information and System, vol. E95-D, pp. 2908-2918, 2012.

[6] C. Qiao and M. Yoo. Optical burst switching (OBS) - a new paradigm for an optical Internet[J], Journal of High Speed Networks, 1999.

[7] N. Barakat and T. E. Darcie. The Control-Plane Stability Constraint in Optical Burst Switching Networks[J], IEEE Communications Letters, vol.11, 2007.

[8] N. Barakat and T. E. Darcie, Control-Plane Congestion in Optical-Burst-Switched Networks[J], IEEE/OSA Journal of Optical Communications and Networking, vol.1, 2009.

[9] T. Venkatesh and C. S. R. Murthy, An Analytical Approach to Optical Burst Switched Networks, Springer Publishing Company, 2009.

[10] J.-Y. Le Boudec and P. Thiran, Network Calculus: A Theory of Deterministic Queuing Systems for the Internet. New York: Springer-Verlag, 2001.

[11] R. L, Cruz. A calculus for network delay, part I: Network elements in isolation[J], IEEE Transaction on Information Theory, vol.37, pp.114-C131, 1991.

[12] C. Chang. Performance Guarantees in Communication Networks, Springer-Verlag, 2000.

[13] Z. Lu, M. Millberg, A. Jantsch, A. Bruce, P. van der Wolf, and T. Henriksson. Flow regulation for on-chip communication[C], Proceedings of DATE, 2009.

[14] X. Yijun, et al. Control architecture in optical burst-switched WDM networks[J], IEEE Journal on Selected Areas in Communications, vol.18, pp.1838-1851, 2000.

[15] P. P. Tang and T. Y. C. Tai. Network traffic characterization using token bucket model[C], Proceedings of IEEE INFOCOM, 1999.

[16] D. P. Bertsekas, Nonlinear Programming. Belmont, MA: Athena Scientific, 1999.

[17] J. Chan, et al. PhoenixSim: A simulator for physical-layer analysis of chip-scale photonic interconnection networks[C], Proceedings of Design, Automation and Test, 2010.

[18] S. C. Woo, M. Ohara, et. al. The SPLASH-2 Programs: Characterization and Methodological Considerations[C], Proceedings of ISCA, 1995.

[19] J. E. Miller, et al. Graphite: A distributed parallel simulator for multicores[C], Proceedings of International Symposium on High Performance Computer Architecture , 2010.