Work-Conserving Fair-Aggregation Across Multiple Core Networks

Jorge A. Cobb Zhe Xu
Department of Computer Science
The University of Texas at Dallas
Richardson, TX 75083-0688
Email: {cobb,xuzhe}@utdallas.edu

***Abstract*—In the effort of reducing or eliminating per-flow state at routers, hence making QoS schedulers scalable in the core Internet, Flow Aggregation outperforms Dynamic Packet State by providing better end-to-end delay guarantee. Work-Conserving Flow Aggregation (WCFA) has the advantage of gaining work-conserving property at the cost of an extra per-hop delay. Most research on work-conserving flow aggregation so far has focused on a single level of flow aggregation within a single aggregation domain. In this paper, we investigate the per-hop behavior of flow aggregation over multiple aggregation domains. Moreover, in each aggregation domain, multiple aggregation levels are considered. We show that the aggregating cost happens only once in each aggregation domain over all aggregation levels, and higher level of flow aggregation provides lower per-hop delay guarantee.**

I. Introduction

Traditionally, the Internet only provides best effort service to all applications. Although this service works well for elastic applications such as file transfer, email, web browsing, etc., Quality of Service (QoS) guarantees are required for emerging applications such as real-time audio and video conferencing, Video on Demand (VoD), IP Telephony, etc.. In order to support new inelastic applications, the network must reserve resources for an individual flow so that QoS is guaranteed at each hop. The IETF has defined two service disciplines, namely, Integrated Services (IntServ) [3] and Differentiated Services (DiffServ) [12], [13].

In the IntServ approach, QoS is provided by real-time scheduling algorithms such as Virtual Clock (VC) [10], [25] and Weighted Fair Queuing (WFQ) [17]. Zhang provided an excellent review of such schedulers [24] in 1995. Schedulers such as VC and WFQ reserve bandwidth along the path of each flow, and provide similar end-to-end delay guarantees to each packet of a flow. Since the delay guarantee is related to the reserved rate, they belong to the same scheduler family called Guaranteed Rate Schedulers (GRS).

GRS schedulers need to maintain per-flow state at each hop along the path of a flow in order to schedule packets going out of the same link of a router, in which a scheduler is implemented for each outgoing link. While the number of flows is manageable in an access network, routers in the core network do not have necessary resources to keep track of each individual flow. In other words, GRS schedulers do not scale well. This leads to the design of DiffServ scheduling algorithms. Dynamic Packet State (DPS) [22], [14] and Flow Aggregation [4] are among approaches to make GRS schedulers more scalable.

In DPS, scheduling information is carried in the packet header. Each router along the path of the flow extracts out the scheduling information and updates the flow state in the packet header. By storing and updating per-flow state in the packet header, DPS eliminates the burden of maintaining per-flow state in the routers, hence makes the scheduling algorithms scalable. However, this limited amount of state provides only a coarse allocation of resources, and falls short of the QoS level available in IntServ [22].

Fair aggregators provide better performance guarantees to individual flows [4]. In flow aggregation, all flows that are going into and coming out of the core network through the same ingress and egress routers are aggregated together. Per-flow state is only maintained in routers of access networks. In contrast, routers in the core network do not know or simply choose to ignore the existence of individual flows. In other words, Core routers only need to maintain states for aggregate flows. Hence, the number of flows that a core router needs to manage is drastically reduced, making the scheduling algorithms scalable. With flow aggregation, scheduling and signaling is also simplified [9]. Moreover, fair aggregators also provide a delay bound that is inversely proportional to the reserved rate of the aggregate flow while it is inversely proportional to the reserved rate of the individual flow in the case of no flow aggregation. Since we expect the reserved rate of the aggregate flow to be much larger than that of each individual flow, flow aggregation has the advantage of providing a much lower delay guarantee across the core network. However, fair aggregators are non-work-conserving, which means a scheduler might be idle although packets are queued waiting for service.

Work-Conserving Flow Aggregation (WCFA) has been proposed recently [7]. WCFA follows the same network model as fair aggregators do. In WCFA, each packet is tagged at the aggregator with its Virtual Finishing Time calculated according to Virtual Clock. Then at each hop along the path through the core network, packets within the aggregate flow are sorted by their tags. This packet reordering eliminates the effect of burstiness of other individual flows in the same aggregate flow, hence provides flow isolation for each constituent flow of the

aggregate. WCFA requires scheduling algorithms to be fair in the core network [19], [2]. Although WCFA gains the work-conserving property at the cost of larger per-hop delay, the end-to-end delay guarantee is still inversely proportional to the reserved rate of the aggregate flow.

Although flow aggregation over multiple domains has been studied in [5], most research so far has concentrated on a single level of flow aggregation over a single aggregation domain. This is especially true for work-conserving flow aggregation. In this paper, we investigate work-conserving flow aggregation over multiple domains. In each domain, we consider the case of multiple levels of flow aggregation. We show that the per-hop delay increase is inversely proportional to the reserved rate of the highest level aggregation flow, which implies a lower end-to-end delay guarantee compared to one single level flow aggregation. Moreover, the aggregation cost is only paid once in each aggregation domain, no matter how many levels of aggregation there are in the domain.

The rest of this paper is organized as follows. In Section II, we introduce the QoS model on which our work is based on. Section III introduces WCFA. Multi-Level Work-Conserving Flow Aggregation is shown in Section IV. Then in Section V, we investigate WCFA over multiple domains. We also briefly review other approaches to scalable scheduling algorithms and compare them with our model in Section VI. Finally, Section VII concludes the paper.

II. Quality of Service Model

In this section, we define the QoS model that the network will provide to each real-time flow. We base our service model on the models of [11], [20].

A. Virtual Finishing Times and Guaranteed-Rate Schedulers

A *flow* is a sequence of packets generated by an application. Each output channel of a computer is equipped with a scheduler, whose function is to schedule packets in an order which guarantees QoS to each input flow. We say a packet exits/arrives from/to a scheduler when the last bit of the packet is transmitted/received by the scheduler. For simplicity, we assume the propagation delay between schedulers is zero.

Each flow is characterized by its reserved packet rate and its maximum packet size. We adopt the following notation for each flow f and each scheduler s along the path of f.

C^s	output channel bit rate of s
R_f	bit rate reserved for flow f
$f.i$	i^{th} packet of flow f
$A^s_{f.i}$	arrival time of $f.i$ at s
$E^s_{f.i}$	exit time of $f.i$ from s
$L_{f.i}$	length of packet $f.i$
$L^{max}_{f.i}$	maximum of $L_{f.j}$, where $1 \leq j \leq i$
L^s_{max}	maximum packet size at s

Consider a scheduler s and a flow f. We define the *virtual finish time* [1] $F^s_{f.i}$ of packet $f.i$ at scheduler s as follows. Assume s were to forward the packets of f at exactly R_f bits/sec.. Then, $F^s_{f.i}$ is the time at which the last bit of $f.i$ is forwarded by s. More formally, let f be an input flow of scheduler s. Then,

$$\begin{aligned} F^s_{f.1} &= A^s_{f.1} + L_{f.1}/R_f \qquad (1)\\ F^s_{f.i} &= max(A^s_{f.i}, F^s_{f.(i-1)}) + L^s_{f.i}/R_f, \text{ for every } i,\ i > 1 \end{aligned}$$

Because scheduler s will forward the packets of f at a rate at least R_f, each packet $f.i$ exits from s close to $F^s_{f.i}$. Schedulers with this property are known as *guaranteed-rate schedulers* [11]. More formally, a scheduler s is a guaranteed-rate (GR) scheduler if and only if, for every input flow f of s and every i, $i \geq 1$,

$$E^s_{f.i} \leq F^s_{f.i} + \beta^s_f \qquad (2)$$

for some constant β^s_f. We refer to β^s_f as the scheduling constant of f at s.

Since the virtual finishing time of a packet determines its exit time from a scheduler, then a bounded end-to-end delay requires a bounded per-hop increase in the virtual finishing time. This bound is well known (it was shown in [11] and also follows from the results in [6], [20]) and is as follows. Let $t^1, t^2, \ldots, t^k$ be a sequence of k GR schedulers traversed by flow f. For all i,

$$F^{t^k}_{f.i} \leq F^{t^1}_{f.i} + \sum_{x=1}^{k-1} \left(\frac{L^{max}_{f.i}}{R_f} + \beta^{t^x}_f \right) \qquad (3)$$

B. Flow Aggregation

To reduce the amount of state managed by each router, multiple flows can be combined together to form a single aggregate flow [4], [5], [9], [18].

An *aggregate* flow g is obtained by merging, at a single point in the network, the packets of multiple flows $f^1, f^2, \ldots, f^n$. In this case, $f^1, f^2, \ldots, f^n$ are said to be the *constituents* of g. A flow f is *simple* if it is not an aggregate, i.e., if f is not the constituent of any other flow.

The reserved rate, R_g, of aggregate flow g is at least the sum of the reserved rates of the immediate constituent flows of g. Schedulers after the aggregation point are not aware of the constituents of an aggregate flow. At a later point in the network, the aggregate flow is separated again into its constituent flows.

A scheduler that receives as inputs a set of flows $f^1, f^2, \ldots, f^n$, and produces as output a single aggregate flow g, by merging the packets of the input flows, is called an *aggregator*. Thus, ingress routers contain $N-1$ aggregators, one for each egress router. A scheduler whose set of input flows is the same as its set of output flows is called a *non-aggregating scheduler*, or simply *scheduler* for terseness. Thus, core routers contain schedulers but no aggregators.

We assume all schedulers, aggregating or not, are GR schedulers. Thus, for any scheduler s and any input flow h

[1] The virtual finishing time is also known as the guaranteed rate clock value in [11], and it is also equal to the time stamp assigned by a virtual clock scheduler [25].

of s (regardless of whether h is a simple or aggregate flow), every packet $p_{h.i}$ exits s no later than time $F^s_{h.i} + \beta^s_h$.

A *separator* is a process that receives as input an aggregate flow, and produces as output the set of *constituents* of the input flow. We assume a separator causes no packet delay by simply examining the packet's header.

Consider as an example a computer with four input/output channels as depicted in Figure 1. Here, flows c and e are the constituents of d, and they are separated from d through a separator. Input flows f and h are aggregated together to form flow g. Flows e and g are forward to the output channel by a non-aggregating scheduler, thus they remain separate in the output channel.

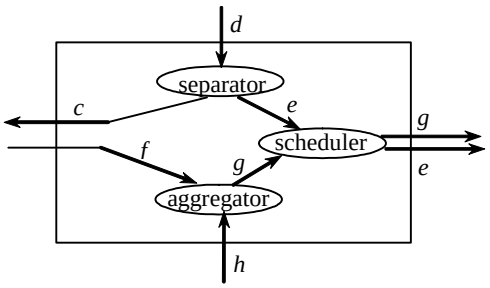


Fig. 1. Flow Aggregation Example

Even if an aggregator is a GR scheduler, it is not sufficient to guarantee a bounded end-to-end delay to its input flows. E.g., consider again Figure 1. Assume h generates packets at a rate greater than R_g, i.e., greater than $R_f + R_h$. On the other hand, f is generating few packets, if any, and the aggregator does not delay packets. Since the scheduler forwards the packets of g at a rate of $R_f + R_h$, the queue of g may grow arbitrarily large. Thus, the next packet of f arriving at the scheduler encounters a large queue of g (consisting of packets from h), causing an excessive delay for f.

To prevent the above, in addition to being a GR scheduler, aggregators must restrict their output rate, and thus be non-work-conserving [4]. In this case, the per-hop delay of a flow f as it traverses a scheduler t is

$$\frac{L^{max}_{(t,f)}}{R_{(t,f)}} + \beta^t \tag{4}$$

where (t, f) is the "root" flow of f at scheduler t, i.e., the highest level aggregate flow containing f. An additional delay of $\frac{L^{max}_{(s,f)}}{R_{(s,f)}}$ occurs for each aggregator s along the path of f.

Notice that the per-hop delay above is similar to the per-hop delay in (3). However, in general, $R_{(t,f)} \gg R_f$ and $L^{max}_{(t,f)} \approx L^{max}_f$, and hence, aggregation provides a much smaller per-hop delay.

III. Work-Conserving Flow Aggregation

As mentioned above, flow aggregation requires aggregators to be non-work-conserving [4].

In [7] we presented a work-conserving aggregation method whose per-hop delay is similar to Relation (4), and is thus independent of the leaky-bucket parameters of other flows. We overview this method in this section.

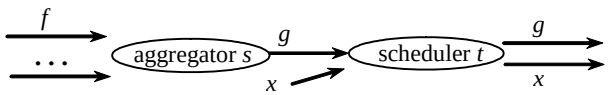


Fig. 2. Aggregator and scheduler.

Our original results were limited to a single level of aggregation. In Section IV we enhance the method to allow a multi-level aggregation, and thus reducing even further the per-hop delay and the number of flows visible to each router. In Section V we apply multi-level aggregation across a multi-core stateless network.

A. Tagging-aggregators and Non-FIFO Schedulers

We first describe how flows are aggregated together, and then discuss the behavior necessary from the schedulers after the point of aggregation. Note that, aggregators are internal, and thus their output channel capacity is, in principle, unbounded. Hence, we assume $C^s = \infty$ for any aggregator s.

Consider the general case of an aggregator s whose input flows include f, its aggregate output is g. Packet $f.i$ is one packet in g, and g is an input to scheduler t, as shown in Figure 2. If there is a large queue at g when packet $f.i$ arrives, the delay of $f.i$ could be kept small if the queue of g were not served in FIFO order. That is, if $f.i$ could be served first before other packets of other flows in the queue of g.

To implement the above, aggregator s assigns a tag $T_{f.i}$ to each input packet $f.i$. We choose a tag equal to the virtual finishing time of the packet at s, i.e., $T_{f.i} = F^s_{f.i}$. Scheduler t then sorts each of its input queues by tag value.

Note however that although t is aware of flow g, it is not aware of g's constituent flow f.

Also note that because t sorts each input flow by tag value, then the exit time of a packet $g.j$ of flow g depends not only on packets of g arriving before $g.j$, but also on packets of g arriving *after* $g.j$ whose tag is at most that of $g.j$.

B. Coordinated Virtual-Finishing-Time

To capture the above behavior, we define the *Coordinated Virtual-Finishing-Time,* Φ. Intuitively, $\Phi^t_{g.j}$ is the time at which $g.j$ would exit t if t served the packets of g at exactly the rate R_g, and, *furthermore,* t serves every packet of g whose tag is at most $T_{g.j}$ *before* it serves $g.j$.

We next provide a more detailed definition of Φ. We begin with some auxiliary definitions.

- Let $filter(g, t, \tau)$ return a flow that differs from g only by removing those packets of g whose tag is greater than τ.
- Let $advance(g, t, f.i)$ return a flow that differs from g with only the following difference. The only difference is that all packets in g that arrive after $f.i$, where $f.i$ is a packet of g, are moved immediately *ahead* of $f.i$ if their tag is at most that of $f.i$.

Definition 1: Let s be an aggregator with an input flow f and with output flow g. Let $f.i = g.j$, and let t be the next scheduler after s. Then,

$$\Phi^t_{g.j} = F^t_{g''.|g''|}$$

where $g' = filter(g, t, T_{f.i})$ and $g'' = advance(g', t, f.i)$. ■

C. Fair Schedulers

We argued in [7] that schedulers must be fair in order to support work-conserving flow aggregation. Consider again Figure 2. If scheduler t stops providing service to flow g, then even if the arriving packet of f is moved to the head of the queue of g, the packet will suffer excessive delay. Note that if the scheduler is a GR scheduler, but is unfair, such as the Virtual Clock protocol [25], [10], then it may stop providing service to g for an arbitrary length of time.

The amount of time that may elapse without a scheduler serving a flow can be formalized by the Worst-Case Fair Index (WFI), as defined in [2].

Definition 2: A scheduler t provides to an input flow g a Worst-Case Fair Index (WFI) of W^t_g if for any time τ, the delay of a packet arriving at τ is bounded above by

$$\frac{Q^t_g(\tau)}{R_g} + W^t_g$$

where $Q^t_g(\tau)$ is the queue of flow g at scheduler t at time τ. ■

In this manner, regardless of how many packets from g have been forwarded by t, i.e., even if g has exceeded its packet rate, at all times t will serve g at a rate at least R_g, except for an additional delay of at most W^t_g. This ensures an exit bound on all packets of g that is related to their coordinated virtual-finishing time.

D. End-to-End Delay

In [7], we considered a network with only a single level of aggregation. That is, each flow f would be aggregated once with other flows to become flow g, and g would not be aggregated any further. In Section IV, we will examine how to provide multi-level flow aggregation while maintaining a work-conserving system.

Under a single aggregation level, we have the following end-to-end delay.

Lemma 1: Let f be an input flow of an internal aggregator s, g be the output of s, and let g traverse schedulers $t^1, t^2, \ldots t^k$, and $f.i = g.j$. Then, the end-to-end delay of any packet $f.i$ of flow f is as follows.

$$\Phi^{t^k}_{g.j} \leq \Phi^{t^{k-1}}_{g.j} + W^{t^{k-1}}_g + \frac{L^{max}_g}{R_g} \tag{5}$$

$$E^{t^k}_{f.i} \leq F^s_{f.i} + \sum_{x=1}^{k} W^{t^x}_g + (k-1)\frac{L^{max}_g}{R_g} \tag{6}$$

■

We thus have that the end-to-end delay has a per-hop increase proportional to $\frac{L}{R_g}$, as in the case of regular flow aggregation (see (4)). However, we have the additional WFI term W^t_g. This term should be as small as possible to ensure a low end-to-end delay.

Virtual Clock has an unbounded WFI. On the other hand, Weighted Fair Queuing (WFQ) [17] has a WFI, which although bounded, equal to $\frac{L}{R_{min}}$, where R_{min} is the minimum rate among the flows at the scheduler [1]. If R_{min} is allowed to be very small, this will cause a significant end-to-end delay.

Although end-to-end delay is bounded with WFQ, we desire a tighter bound in proportion to $\frac{L}{R_g}$. In [1], WF^{2}Q is proposed as an alternative to WFQ, and it is shown that the WFI of a WF^{2}Q scheduler t is bounded as follows.

$$W^t_g \leq \frac{L^{max}_g}{R_g} + \frac{L^t_{max}}{C^t}$$

WF^{2}Q is part of a whole family of schedulers, called Shaped Rate Proportional (SRP) schedulers [21], [19], whose WFI is as above. Any work-conserving member of this family could be used as a scheduler in work-conserving flow aggregation.

We thus have that the end-to-end delay has a per-hop increase proportional to $\frac{L}{R_g}$, as in the case of regular flow aggregation (see Relation (4)) plus the small per-hop term $\frac{L^t_{max}}{C^t}$. However, most GR scheduling protocols have $\beta^t_g = \frac{L^t_{max}}{C^t}$. Thus, work-conservation increases the per-hop delay by $\frac{L}{R_g}$. However, this is a relative small increase that is outweighed by the advantages of a work-conserving system.

IV. Multi-level Work-Conserving Aggregation

We next consider multi-level flow aggregation. That is, an aggregate flow can have constituent flows which are themselves also aggregate flows. Multi-level aggregation has the advantage that the rate of the resulting aggregate flow increases with each aggregation level, and the per-hop delay is inversely proportional to the rate of the flow. However, because aggregation is work-conserving, aggregation and separation of flows has different consequences than the original results on non-work-conserving aggregation [4].

We address aggregation in two steps. We first consider aggregators and schedulers. We then consider the effects of separating flows.

A. Multi-level Aggregation

Because we address work-conserving flow aggregation, similar assumptions to those of Section III are made. That is, each packet of an aggregate flow g, regardless of the aggregation level, contains a tag $T_{g.j}$, and schedulers sort the queue of each flow by tag value. The coordinated virtual finishing time $\Phi_{g.j}$ is defined as before.

In Section III, we defined an aggregator that receives simple (i.e., non-aggregated) flows as input and produces a single aggregate flow as output. Below, we consider an aggregator whose input is a set of aggregate flows and whose output is a single (higher layer) aggregate flow.

Recall that an aggregator assigns tags to packets and then merges the packets over its output channel. When the inputs are aggregate flows, packets are already tagged. We have chosen to preserve this tag in the output flow. That is, if an aggregate flow g is aggregated with other flows to become flow h, and if $g.j = h.k$, then $T_{h.k} = T_{g.j}$. With respect to merging of the flows, this is done as before, i.e., in a FCFS manner.

We refer to a higher-layer aggregator as a "non-tagging" aggregator. It is basically a FCFS multiplexer, with the only difference that it modifies the packet header to reflect that the packet belongs to the new higher level flow h.

The exit time of a packet is tightly related to its Φ value at a scheduler. Thus, we must evaluate the effect of aggregation on Φ. This is given below.

Theorem 1: Let s be a non-tagging aggregator, and let t be a scheduler after s. For each aggregate input flow g of s, and for each packet $g.j$ of g, let $\Phi^s_{g.j} \leq T^s_{g.j}$. Then,

$$\Phi^t_{h.k} \leq T^s_{g.j} = T^t_{h.k}$$

where h is the output flow of s and $g.j = h.k$.

■

Due to the limited space, we will present the proof for theorems of this paper in [23].

Theorem 1 shows that if Φ is bounded by the tag of the packet then the same thing holds for the outgoing aggregated flow. This implies there is no additional penalty for aggregation, and flows can be aggregated without incurring any additional delay. This is contrary to [4], where each aggregation point introduce additional end-to-end delay.

Theorem 1 requires that Φ be bounded by the tag of the packet. However, from (5) in Lemma 1, Φ may increase at each hop. We therefore require the tag of each packet to be increased by $W^t_g + \frac{L^{max}_g}{R_g}$ at every scheduler t. The scheduler is aware of these values and therefore they need not be part of the packet header.

Under these conditions, we have the following bound on the end-to-end increase of Φ and on the end-to-end delay as a flow traverses multiple aggregators and schedulers.

Theorem 2: Consider a flow g that traverses multiple aggregators and schedulers along its path. Each scheduler is a fair scheduler that increases the packet tag as described above, and each aggregator merges the packets of its input flows but does not modify the tag. Let $t^1, t^2, \ldots, t^k$ be the sequence of schedulers traversed by f. Finally, let $\Phi^{t^1}_{g.j} \leq T^{t^1}_{g.j}$. Then,

$$\begin{aligned} E^{t^k}_{g.j} &\leq T^{t^1}_{g.j} + \sum_{x=1}^{k} W^{t^x}_{(t^x,g)} + \sum_{x=1}^{k-1} \frac{L^{max}_{(t^x,g)}}{R_{(t^x,g)}} \\ \Phi^{t_k}_{(t^k,g).j} &\leq T^{t^1}_{g.j} + \sum_{x=1}^{k-1} W^{t^x}_{(t^x,g)} + \sum_{x=1}^{k-1} \frac{L^{max}_{(t^x,g)}}{R_{(t^x,g)}} \end{aligned}$$

where (t, g) is the "root" aggregate flow of g at t, i.e., the highest level aggregate flow containing g.

■

The per-hop delay in Theorem 2 is inversely proportional to the rate of the root flow of g. In consequence, aggregation has the benefit of reducing per-hop delay. In [4], where non-work-conserving aggregation is used, the per-hop delay is also inversely proportional to the the rate of the root flow of g. However, there are significant differences.

In [4], a penalty of

$$\frac{L^{max}_{(s,g)}}{R_{(s,g)}}$$

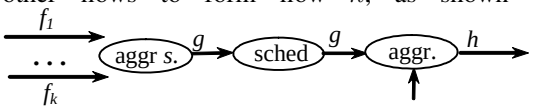
is incurred whenever g is the input to an aggregator s. No such penalty occurs in Theorem 2 above, regardless of the number of aggregation levels. On the other hand, the per-hop delay in [4] is simply

$$\frac{L^{max}_{(t,g)}}{R_{(t,g)}},$$

while the per-hop delay in Theorem 2 has the additional term $W^t_{(t,g)}$. However, since $W^t_{(t,g)}$ can be made very close to $L^{max}_{(t,g)}/R_{(t,g)}$, this increase is not of great significance, and it is outweighed by the benefits of work-conservation.

B. Flow Separation and End-to-End Delay

We next consider the effects on Φ of separating an aggregate flow into its constituent flows. Consider the following example. Assume k simple flows $f_1, f_2, \ldots f_k$ are aggregated to form flow g, and g is aggregated with other flows to form flow h, as shown below.



Flow h will then traverse several schedulers. Since each f_i flow is simple, the first aggregator s assigns tag $T^s_{f_i.j} = F^s_{f_i.j}$ to each packet $f_i.j$. From Theorem 2, packets of each flow f_i have a bounded exit time of $T^s_{f_i.j} + \Delta$ for some bound Δ which depends on the number of hops.

Assume that the first packets of each of flows arrive at the same time τ at aggregator s. For simplicity, assume these flows have the same rate R_f and the same packet length L. Thus, each receives a tag value T equal to $\tau + L/R_f$. By the definition of Δ, each of these packets will exit the last scheduler around time $T + \Delta$.[2]

Assume h is then separated into its constituents, and thus g is recovered. Consider the value of Φ_g for the packets mentioned above. Since they all arrive at time $T + \Delta$, a constant rate server of rate $R_g = k \cdot R_f$ would require $(k \cdot L)/(k \cdot R_f) = L/R_f$ seconds to forward all of these packets, and hence, for one of these packets,

$$\Phi_g = T + \Delta + L/R_f$$

Notice, however, that if h were separated into its simple constituents (and hence each of $f_1, f_2, \ldots f_k$ is recovered individually) then we would also have

$$\Phi_f = T + \Delta + L/R_f$$

[2]Bound Δ is reachable, this is not shown due to lack of space.

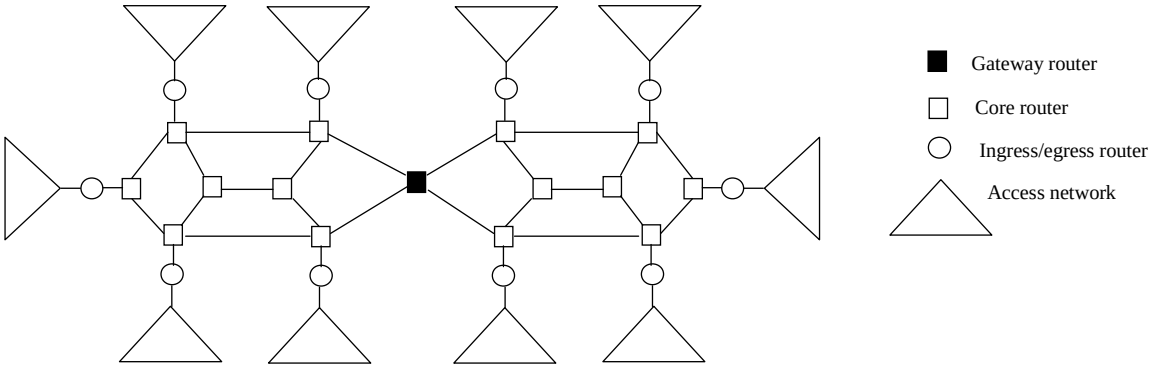


Fig. 3. Multi-Core Network

This is because the virtual time F of a simple flow (and hence its Φ also) increases by ε if it traverses a system where each packet is guaranteed to exit the system by its virtual time at the entrance of the system plus ε [11][6].

We conclude that there is no advantage on end-to-end delay bound in separating flow h into its aggregate constituents, such as g, over separating h into its simple flows such as $f_1, f_2, \ldots f_k$. This is because aggregation introduces no penalty, and any subset of the simple flows can be aggregated together again if desired. We thus assume that each separator always separate a flow into the simple flows that comprise it.

The cause of the additional L/R_f increase in Φ is due to the overlapping of tag values of the packets in g. That is, two packets $g.i$ and $g.j$ have overlapping tags if the two ranges $[T_{g.i} - \frac{L_{g.i}}{R_{g.i}}, T_{g.i}]$ and $[T_{g.j} - \frac{L_{g.j}}{R_{g.j}}, T_{g.j}]$ intersect. Note that in our example tags are overlapping since each packet in $f_1, f_2, \ldots f_k$ receives the same tag value.

We have argued that Φ increases by L^{max}/R_f when it is separated into its components. We thus assume that separators increase the tag of each packet of f by this amount. If separators are implemented without per-flow state, we assume that the value of L^{max}/R_f can be included in the header of the packet. This is the same assumption is made by other work in state-free networks [22], [14].

We thus have the following end-to-end behavior when flows are separated.

Theorem 3: Consider a flow f that traverses a sequence of aggregators and schedulers, and the schedulers along this path are $t^1, t^2, \ldots, t^k$, terminating at a separator u. Assume also that for each packet $f.i$, $\Phi_{f.i}^{t_1} \leq T_{f.i}^{t_1}$, and that flow f has non-overlapping tags. Then,

$$\Phi_{f.i}^{u} \leq T_{f.i}^{u} \quad (7)$$

$$T_{f.i}^{u} = T_{f.i}^{t^1} + \sum_{x=1}^{k} W_{(t^x,f)}^{t^x} + \sum_{x=1}^{k} \frac{L_{(t^x,g)}^{max}}{R_{(t^x,f)}} + \frac{L_f^{max}}{R_f} \quad (8)$$

■

V. Multi-domain Work-Conserving Aggregation

We discussed multiple level work-conserving flow aggregation in the previous section and claim that the per-hop delay guarantee is inversely proportional to the reserved rate of the highest level aggregation. Moreover, the price to pay for flow aggregation is L/R_f, where f is a simple flow. More importantly, we pay this aggregation price only once at the separator. In this section, we investigate work-conserving flow aggregation over multiple domains.

Fig. 3 shows the network architecture for multiple domain flow aggregation. Only routers in access networks keep per-flow information. At the ingress router, each packet is tagged with its virtual finishing time. Along with this tag, the packet header also carries the reserved rate of the simple flow to which the packet belong.

When a packet enters the core network, its flow f is aggregated with other flows traveling the same path, forming the aggregate flow g. The packets within the aggregate are sorted according to their tag values at each hop. Each core router updates the tag of a packet as follows.

$$T_{g.j}^{t+1} = T_{g.j}^{t} + W_g^t + \frac{L_g^{max}}{R_g}$$

Then, this first layer aggregate flow may be aggregated again with other simple or aggregate flows forming a higher layer aggregate flow h. The tag of a packet is updated with all parameters of its highest aggregation, namely, h. This procedure is repeated for more levels of aggregation.

As described in the previous section, we can not gain any advantage by separating multiple level flow aggregation one layer at a time. Hence, the aggregate flow is separated all the way down to simple flows, no matter how many levels it has. This is represented as the gateway in Fig. 3. It is worthing noting that the gateway does not need to maintain any per-flow information, and only a small amount of information is kept in the packet header. The gateway only needs to update the tag of a packet according to (8).

When a packet goes into the next aggregation domain, its flow can be aggregated freely with other simple or aggregate flows, and packets are sorted using the tags their headers carry. And the tag of a packet should be updated accordingly. If the flow goes through more aggregation domains in the core network, it will be separated into the original simple flow again at each gateway. By updating the tag of a packet

at the gateway and all routers on the path, a simple flow traverses the whole core network through multiple aggregation domains in multiple level aggregations in each domain. Again, all routers in the core network, including gateways between aggregation domains, do not need to maintain any per-flow state information for simple flows.

In the next section, we discuss other approaches of reducing number of flows maintained in the core network and compare major results to our aggregation model.

VI. Related Work

In Dynamic Packet State (DPS) [22], the core routers are totally relieved from the burden of maintaining per-flow state. Instead, a time stamp is carried in the header of a packet along with other necessary information for scheduling such as reserved rate of a flow, packet length, etc.. A router in the core network will update the time stamp with the information that how much time in advance the packet exited the previous router than its deadline. However, in order to calculate the time stamp at each hop, all routers in the core network need to be synchronized, or constant delay links are required. In the meanwhile, the delay guarantee is much larger than the one flow aggregation provides. Multiple level flow aggregation results in an even lower delay, since a higher level aggregation has a much larger reserved rate.

Core-Stateless Guaranteed Rate (CSGR) networks [15], [16] follow a similar idea as that in DPS. In CSGR, the ingress router will calculate a time stamp of a packet for each router in the core network on the path of a flow. Then, the packet header carries the whole vector of time stamps into the core network. Each core router on the path extracts the time stamp from the corresponding element in the vector, and sort all packets according to the time stamp. In this manner, all packets are guaranteed to exit the core network by their deadlines. The delay guarantee provided by CSGR is the same as that of DPS, hence is also larger than that of flow aggregation. Plus, the packet header needs to reserve more space for scheduling information. However, the deadline reuse technique [15], with corrected reuse conditions [8], provides a means for throughput guarantee.

Multi-level work-conserving flow aggregation over multiple domains not only provides better delay guarantee, but offers flexibility for aggregation as well. In each aggregation domain, flows are freely aggregated together, regardless of their aggregation level. And the aggregation cost is only paid once at the separator. Moreover, higer levels of aggregation also means fewer number of flows to maintain state for in the core network. These properties, along with work-conservation, make it an attractive scalable scheduling algorithm for the core network.

VII. Concluding remarks and Future Work

As discussed in [7], the fairness among the simple flows is limited to the aggregate rate. Similar to [8], we speculate that the "time stamp reuse" technique [15] could also be incorporated into multi-level flow aggregation. In the future work, we will study how to provide throughput guarantee, hence fairness guarantee, to simple flows in multi-level flow aggregation over multiple domains.

References

[1] J. C. Bennett and H. Zhang, "Hierarchical packet fair queueing algorithms," *IEEE/ACM Transactions on Networking*, vol. 5, no. 5, pp. 675–689, Oct. 1997.

[2] ——, "WF2Q: worst-case fair weighted fair queueing," in *IEEE INFOCOM Conference*, 1996.

[3] R. Braden, D. Clark, and S. Shenker, "Integrated services in the internet architecture," Internet RFC 1633.

[4] J. Cobb, "Preserving quality of service guarantees in-spite of flow aggregation," *IEEE/ACM Transactions on Networking*, vol. 10, no. 1, pp. 43–53, Feb. 2002.

[5] ——, "Scalable quality of service across multiple domains," *Computer Communications*, vol. 28, no. 18, pp. 1997–2008, Nov. 2005, Elsevier.

[6] J. Cobb and M. Gouda, "Flow theory," *IEEE/ACM Transactions on Networking*, vol. 5, no. 5, pp. 661–674, Oct. 1997.

[7] J. Cobb and Z. Xu, "Maintaining flow isolation in work-conserving flow aggregation," in *Proc. IEEE GLOBECOM Conference*, 2005.

[8] ——, "Guaranteed throughput in work-conserving flow aggregation through deadline reuse," in *Proceedings of the ICCCN Conference*, Oct. 2006.

[9] H. Fu and E. W. Knightly, "A simple model of real-time flow aggregation," *IEEE/ACM Transactions on Networking*, vol. 11, no. 3, June 2003.

[10] X. G. and L. S., "Delay guarantee of the virtual clock server," *IEEE/ACM Transactions on Networking*, pp. 683–689, Dec. 1995.

[11] P. Goyal, S. Lam, and H. Vin, "Determining end-to-end delay bounds in heterogeneous networks," in *Proc.of the NOSSDAV Workshop*, 1995.

[12] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "Assured forwarding phb group," Internet RFC 2597.

[13] V. Jacobson, K. Nichols, and K. Poduri, "An expedited forwarding phb," Internet RFC 2598.

[14] J. Kaur and H. M. Vin, "Core-stateless guaranteed rate scheduling algorithms," in *Proc. of the IEEE INFOCOM Conf.*, 2001.

[15] ——, "Core stateless guaranteed throughput networks," in *Proc. of the IEEE INFOCOM Conf.*, 2003.

[16] ——, "Providing deterministic end-to-end fairness guarantees in corestateless networks," in *Proc. IEEE International Workshop on Quality of Service (IWQoS)*, 2003.

[17] A. K. J. Parekh and R. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: The single node case," *IEEE/ACM Transactions on Networking*, vol. 1, no. 3, June 1993.

[18] J. Qiu and E. W. Knightly, "Measurement-based admission control with aggregate traffic envelopes," *IEEE/ACM Transactions on Networking*, vol. 9, no. 2, Apr. 2001.

[19] D. Stidialias and A. Varma, "Rate proportional servers: A design methodology for fair queuing algorithms," *IEEE/ACM Transactions on Networking*, Apr. 1998.

[20] D. Stiliadis and A. Varma, "Latency rate servers: a general model for analysis of traffic scheduling algorithms," *IEEE/ACM Transactions on Networking*, vol. 6, no. 5, pp. 611–624, 1998.

[21] ——, "A general methodology for designing efficient traffic scheduling and shaping algorithms," in *IEEE INFOCOM Conference*, 1997.

[22] I. Stoica and H. Zhang, "Providing guaranteed services without per-flow management," in *Proc. of the ACM SIGCOMM Conference*, 1999.

[23] Z. Xu, "Scalable scheduling for quality of service guarantees," Ph.D. dissertation, The University of Texas at Dallas, Aug. 2007.

[24] H. Zhang, "Service disciplines for guaranteed performance service in packet-switching networks," *Proceedings of the IEEE*, vol. 93, no. 10, Oct. 1995.

[25] L. Zhang, "Virtual clock: A new traffic control algorithm for packet-switched networks," *ACM Transactions on Computer Systems*, vol. 9, no. 2, pp. 101–124, May 1991.