GD-Aggregate: A WAN Virtual Topology Building Tool for Hard Real-Time and Embedded Applications

Qixin Wang*, Xue Liu†, Jennifer Hou*, and Lui Sha*
* Department of Computer Science, University of Illinois at Urbana-Champaign
Email: {qwang4, jhou, lrs}@uiuc.edu
† School of Computer Science, McGill University
Email: xueliu@cs.mcgill.ca

Abstract

The convergence of computer and physical world calls for next generation Wide Area Network *(WAN) infrastructures for hard real-time and embedded applications. Such networks need virtual topologies to achieve scalability, configurability, and flexibility. Virtual topologies are made of virtual links, for which, the state-of-the-art building tool is* Guaranteed Rate server based aggregates *(GR-aggregates). However, common-practice weight assignment scheme couples GR-aggregate* End-to-End *(E2E) delay bound with aggregate's data throughput inverse proportionally. This is undesirable for many hard real-time embedded sensing/actuating applications, whose traffic has small data throughput but requires short E2E delay. We propose* Guaranteed Delay server based aggregates *(GD-aggregates), which allow assigning weights according to* priorities *instead of data throughput. This decouples E2E delay guarantee from data throughput, hence meets the needs of hard real-time embedded applications. In addition, GD-aggregates can be analyzed with simple closed form formulae, and can be easily planned with optimization tools.*

1 Introduction

The trend of next generation networks is to converge computers and the physical world. This demands next generation *Wide Area Network* (WAN) for hard real-time and embedded applications (simplified as *Real-Time and Embedded WAN*, or RTE-WAN in the following). Efforts like the Real-Time and Embedded GENI [15] and the Cyber-Physical Systems [9] are calling for design proposals of such RTE-WANs, which must provide scalability, configurability, flexibility, and hard real-time.

To achieve scalability, configurability, and flexibility, the RTE-WAN needs virtual topologies: reconfigurable *virtual links* shall overlay on top of physical links, and hide physical links from higher level networking activities, such as routing, QoS provisioning, and applications; a virtual link may span one or several sequentially connected physical links, occupy part or all of the physical links' bandwidth, and may be reconfigured.

In addition to scalability, configurability, and flexibility, an RTE-WAN virtual link must also guarantee hard real-time E2E delay. Considering all these goals, Section 4 discusses different technological alternatives on how to build RTE-WAN virtual links, and shows *Guaranteed Rate server based aggregates* (GR-aggregates) proposed by Sun and Shin [19][18] is the best technology to start with.

Specifically, a virtual link is embodied by the GR-aggregates passing through it. A GR-aggregate (see Section 2.1) is basically an aggregation of flows that start from the virtual link's sender-end node, traverse the virtual link's intermediate nodes, and arrive at the virtual link's receiver-end node. A GR-aggregate aggregates flows of similar characteristics, and provides hard real-time E2E delay guarantee. Fig. 1 illustrates the relationship between GR-aggregates and their corresponding virtual link, where a virtual link over two physical links (AC and CB) consists of three GR-aggregates: $F1$, $F2$, and $F3$.

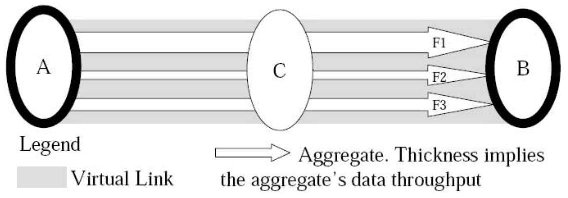

A
C
B
F1
F2
F3
Legend
Virtual Link
Aggregate. Thickness implies the aggregate's data throughput

Figure 1. A virtual link is embodied by the aggregates passing through it

Originally, the GR-aggregate design is for Internet traffic [19][18]. Though this design fits the Internet context well, when applied to RTE-WAN, GR-aggregates face a new challenge: using common-practice configuration method, a GR-aggregate's E2E delay bound is inverse-proportionally coupled with the aggregate's data throughput. This makes aggregates with small data throughput to have large E2E delay bounds, and vice versa. Such tight coupling is undesirable to many RTE-WAN traffics, especially the sensing/actuating traffic, which usually has small data throughput, but demands short E2E delay guarantee (see Section 2.2).

To solve this problem, we propose *Guaranteed Delay server based aggregates* (GD-aggregates). GD-aggregate design decouples (or partially decouples) E2E delay guarantee from data throughput. It also supports priorities, which facilitates real-time system design. GD-aggregates can be analyzed with simple closed form formulae, and can be easily planned with optimization tools.

We evaluate the performance of GD-aggregates in the context of underground mining, a representative RTE-WAN application with typical RTE-WAN traffic. The results show that GD-aggregates support RTE-WAN traffic better than GR-aggregates. However, due to page limits, the performance evaluation is moved to Appendix J of [20].

The rest of the paper is organized as follows: Section 2 describes the state-of-the-art GR-aggregate design and its coupling problem when using common-practice configuration methods; Section 3 presents our GD-aggregate solution; Section 4 discusses related work; Section 5 concludes the paper.

2 GR-aggregate and the Coupling Problem

In the following, Section 2.1 describes the GR-aggregate design [19][18] and its E2E delay bounds; and Section 2.2 explains GR-aggregate's coupling problem between E2E delay bound and data throughput.

Unless explicitly noted, we assume output queueing [14], and pick symbols consistent with [19][18] whenever possible.

2.1 GR-aggregate

The GR-aggregate is based on the concept of *Guaranteed Rate* (GR) server proposed by Goyal et al. [7]:

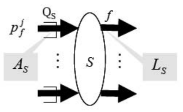


Figure 2. A Queueing Server

Definition 1 (Guaranteed Rate Server) *As shown in Fig. 2, suppose the jth ($j = 1, 2, \ldots$) packet p_f^j of flow f arrives at queue Q_S at time $A_S(p_f^j)$, and leaves server S at time $L_S(p_f^j)$. Suppose the length of p_f^j is ℓ_f^j. We define* Guaranteed Rate Clock *(*GRC*) of packet p_f^j at server S as*

$$\mathrm{GRC}(p_f^j) \overset{def}{=} \max\{A_S(p_f^j), \mathrm{GRC}(p_f^{j-1})\} + \frac{\ell_f^j}{r_f}, \quad \forall j = 1, 2, \ldots,$$

where constant r_f is called the Guaranteed Rate*, and* $\mathrm{GRC}(p_f^0) \overset{def}{=} 0$*. If server S can provide a guaranteed rate r_f for flow f, such that for each packet p_f^j ($j = 1, 2, \ldots$) of f, there is*

$$L_S(p_f^j) \le \mathrm{GRC}(p_f^j) + \alpha, \tag{1}$$

where α is a constant independent of p_f^j, then S is a Guaranteed Rate *(GR) server. In addition, we call α the* scheduling constant *of GR server S for f.* ■

Many existing scheduling servers, such as the *Weighted Fair Queueing* (WFQ) server [13] and the *Worst-Case Fair Weighted Fair Queueing* (WF^{2}Q) server [2], are GR servers [7]. *Without loss of generality, we assume all scheduling servers are WFQ servers.* A WFQ server S has a scheduling constant of $\alpha = \ell^{max}/C$, where ℓ^{max} is the maximum size of packets entering S, and C is the output capacity of S. If a flow f is assigned a WFQ scheduling weight of ϕ_f, its guaranteed rate $r_f = \phi_f C$.

Based on the above notions, the GR-aggregate design runs as follows [19][18]: A GR-aggregate starts by creating an aggregate of flows at a GR server (called "*low-end server*"), and letting the aggregate traverse several GR servers (called "*high-end servers*") as shown in Fig. 3. In the figure, flow f joins other flows at low-end server $S_L^{(1)}$ at Node 1. The output of $S_L^{(1)}$ is an aggregate, denoted as F. High-end server $S_H^{(1)}$ forwards F to Node 2, and $S_H^{(2)}$ forwards F to Node 3, so on and so forth, until F reaches Node K. As the receiver-end of aggregate F, the routing circuit of Node K forwards individual packets of aggregate F according to their original flow headers. Therefore packets of flow f are forwarded to their corresponding output port, where it may join another set of flows at low-end server $S_L^{(K)}$ to create another GR-aggregate F'. We denote $A_{SL}^{(i)}(p)$ and $L_{SL}^{(i)}(p)$ as the time when packet p reaches $Q_{SL}^{(i)}$ and leaves server $S_L^{(i)}$ at Node i respectively; and denote $A_{SH}^{(i)}(p)$ and $L_{SH}^{(i)}(p)$ as the time packet p reaches $Q_{SH}^{(i)}$ and leaves server $S_H^{(i)}$ at Node i respectively. The *End-to-End* (E2E) delay guarantee of packets traveling through GR-aggregates is restated in Theorem 1 (originally from [19][18]).

Theorem 1 (GR-aggregate E2E Delay) *Suppose a flow f joins an GR-aggregate F at $S_L^{(1)}$ as shown in Fig. 3, then the E2E delay d_f^j for any packet p_f^j ($j = 1, 2, \ldots$) of f satisfies*

$$\begin{aligned} d_f^j &\overset{def}{=} A_{SL}^{(K)}(p_f^j) - A_{SL}^{(1)}(p_f^j) \\ &= L_{SH}^{(K-1)}(p_f^j) - A_{SL}^{(1)}(p_f^j) \qquad (2) \\ &\le [\mathrm{GRC}_{SL}^{(1)}(p_f^j) - A_{SL}^{(1)}(p_f^j)] + (K-2)\frac{\ell_F^{max}}{R_F} \\ &\quad + \alpha_L^{(1)} + \sum_{i=1}^{K-1} \alpha_H^{(i)}, \qquad (3) \end{aligned}$$

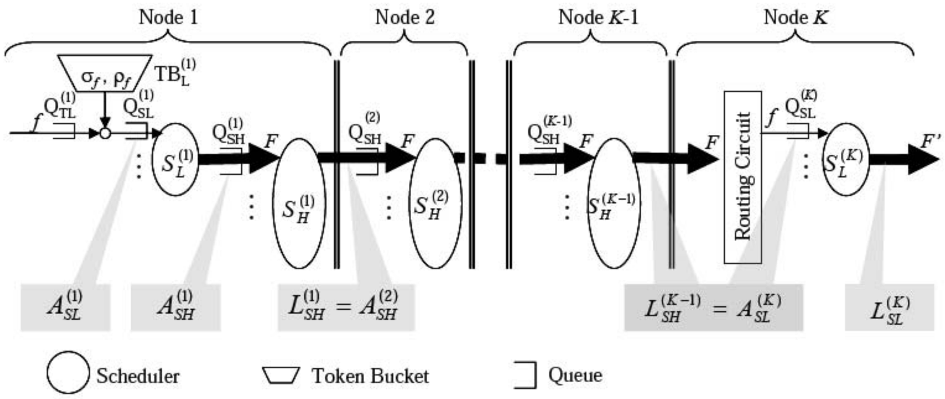


Figure 3. A GR server based aggregate. Labels in the figure explain the symbols denoting arrival and departure (leaving) time of packets. Routing circuits in Node $1 \sim (K-1)$ are omitted in the figure.

where ℓ_F^{max} and ℓ_f^{max} are the maximum packet length for aggregate F and flow f respectively; R_F is the guaranteed rate for aggregate F at $S_H^{(1)} \sim S_H^{(K-1)}$; r_f is the guaranteed rate for flow f at $S_L^{(1)}$ and $S_L^{(K)}$; the output capacity of $S_L^{(1)}$ is R_F; the output capacity of $S_H^{(i)}$ ($i = 1, 2, \ldots, (K-1)$) is $C^{(i)}$; $\alpha_L^{(1)}$ is the scheduling constant of GR server $S_L^{(1)}$; and $\alpha_H^{(i)}$ is the scheduling constant of GR server $S_H^{(i)}$. Particularly, if f is constrained by a token bucket (σ_f, ρ_f) before arriving at $Q_{SL}^{(1)}$ at Node 1, where $\rho_f \leq r_f$, then

$$d_f^j \leq \frac{\sigma_f}{r_f} + (K-2)\frac{\ell_F^{max}}{R_F} + \alpha_L^{(1)} + \sum_{i=1}^{K-1} \alpha_H^{(i)}. \quad (4)$$

Proof: See [19][18], particularly Appendix III of [18]. ■

With minor modification to the proof of the above theorem, we can improve the delay bound if packets entering $S_L^{(1)}$ are *Conflict-Free*, which is defined in the following:

Theorem 2 (E2E Delay for Conflict-Free Packet Arrival) *In Fig. 3, if packets entering $S_L^{(1)}$ follow* Conflict-Free *pattern, i.e., for any two packets p_1 and p_2 arriving at $S_L^{(1)}$ consecutively, the arrival time $A_{SL}^{(1)}(p_1)$ and $A_{SL}^{(1)}(p_2)$ satisfy $A_{SL}^{(1)}(p_2) \geq A_{SL}^{(1)}(p_1) + \frac{\ell_1}{C_L^{(1)}}$, where ℓ_1 is the length of p_1, $C_L^{(1)}$ is the output capacity of $S_L^{(1)}$, then*

$$d_f^j \leq (K-1)\frac{\ell_F^{max}}{R_F} + \sum_{i=1}^{K-1} \alpha_H^{(i)}. \quad (5)$$

Proof: See Appendix A of [20]. ■

Conflict-Free packet arrival pattern is common to RTE-WAN sensing/actuating end nodes. For example, if an end node polls N local sensors in round robin, then the sensor reading packets can arrive at this node one after another without temporal overlap. Such packet arrival pattern is Conflict-Free.

2.2 The Coupling Problem under Data Throughput Proportional Weight Assignment (DTPWA)

With the above E2E delay bounds, GR-aggregates effectively meet the needs of Internet, the GR-aggregates' original application context [19][18]. However, when applying GR-aggregates to RTE-WAN, a new challenge emerges:

The GR-aggregate E2E delay bound is inverse-proportionally coupled with the aggregate F and flow f's guaranteed rates R_F and r_f (see Inequality (3) $\sim$ (5)). Though the original GR-aggregate design [19][18] does not specify how to set R_F and r_f, as a common-practice, people assign WFQ scheduling weights ϕ_F and ϕ_f proportional to data throughput to avoid queue overflow. For simplicity, we refer to such common-practice as *Data Throughput Proportional Weight Assignment* (DTPWA).

Since guaranteed rate $R_F = \phi_F C$ and $r_f = \phi_f C$, where C is the server output capacity, DTPWA makes guaranteed rate proportional to aggregate/flow data throughput. Therefore, a GR-aggregate's E2E delay bound becomes inverse-proportionally coupled with the aggregate/flow data throughput. As a consequence, if aggregate F and flow f are of small data throughput, then guaranteed rates R_F and r_f are small, and E2E delay bound d_f^j becomes large; if F and f are of large data throughput, then R_F and r_f are large, and d_f^j becomes small.

This is undesirable for RTE-WAN traffic, which typically includes

1) *hard real-time sensing/actuating traffic, which usually has small data throughput but demands short E2E delay bounds*;

2) hard real-time video streams, which have large data throughput and demand short E2E delay bounds;

3) soft real-time traffic, such as FTP, which may have large data throughput, but only demands bounded E2E delays (does not have to be short).

Because of the inverse-proportional coupling of E2E delay bound and data throughput under DTPWA, hard real-time sensing/actuating traffic gets very large E2E delay bounds. This problem motivates us to decouple E2E delay bounds from data throughput.

3 GD-aggregate

We propose *Guaranteed Delay server based aggregates* (GD-aggregates), to allow non-DTPWA weight assignments, particularly *priority* based weight assignment, so as to decouple E2E delay bounds from data throughput.

In the following, Section 3.1 introduces basic building components for GD-aggregates; Section 3.2 describes the GD-aggregate design and gives its E2E delay bounds; Section 3.3 elaborates on how to assign weights according GD-aggregates' priorities, so as to decouple E2E delay bounds from data throughput, and shows how to analyze GD-aggregates with simple closed form formulae, and to plan GD-aggregates with optimization tools.

3.1 Guaranteed Delay Server

To guarantee a bounded E2E delay, it is enough to guarantee a packet transmission time bound at each intermediate node. If this time bound is decoupled from data throughput, the E2E delay is decoupled from data throughput. This observation motivates our proposal of *Guaranteed Delay* (GD) server as a basic building component:

Definition 2 (Guaranteed Delay Server) *Same as shown in Fig. 2, suppose the jth ($j = 1, 2, \ldots$) packet p_f^j of flow f arrives at queue Q_S at $A_S(p_f^j)$, and leaves server S at $L_S(p_f^j)$. Suppose the length of p_f^j is ℓ_f^j. We define* Guaranteed Delay Clock *(GDC) of packet p_f^j at server S as*

$$\mathrm{GDC}(p_f^j) \overset{def}{=} \max\{A_S(p_f^j), \mathrm{GDC}(p_f^{j-1})\} + \Delta(\ell_f^j),$$
$$\forall j = 1, 2, \ldots,$$

where the nonnegative monotonically nondecreasing *function $\Delta(\ell)$ is called the* Guaranteed Delay Function*, and $\mathrm{GDC}(\ell_f^0) \overset{def}{=} 0$. If server S can provide a nonnegative monotonically nondecreasing guaranteed delay function $\Delta(\ell)$ for flow f, such that for each packet p_f^j ($j = 1, 2, \ldots$) of f, there is*

$$L_S(p_f^j) \leq \mathrm{GDC}(p_f^j) + \alpha, \tag{6}$$

where α is a constant independent of p_f^j, then S is a Guaranteed Delay *(GD) server. In addition, we call α the* scheduling constant *of GD server S for f.* ■

We prove that any *Token Bucket Constrained WFQ* (TBC-WFQ) server is a GD server, and give its guaranteed delay function. The analysis uses the concept of *greedy starting*, and the liquid flow model based *Generalized Processor Sharing* (GPS) server [12].

A server S is called *Token Bucket Constrained* (TBC) if each of its input flows is constrained by a token bucket, as shown in Fig. 4. A flow f is called *greedy starting* from time τ, if $\forall t \geq \tau$, the number of bits pass through f's token bucket TB_f during $[\tau, t]$ equals $\sigma(\tau) + (t - \tau)\rho$, where $\sigma(\tau)$ is the number of tokens in TB_f at τ, and ρ is TB_f's token filling rate, which is also f's data throughput.

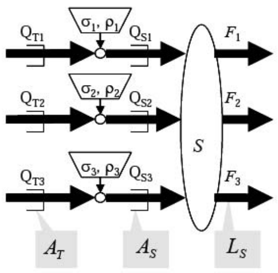


Figure 4. A Token-Bucket-Constrained GPS (TBC-GPS) or Token-Bucket-Constrained WFQ (TBC-WFQ) Server (depends on whether S is GPS or WFQ)

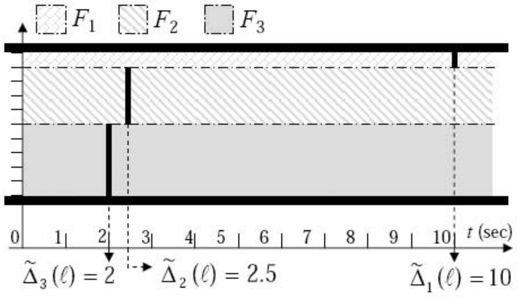
Theorem 3 (Critical Instance for Transmission Time) *Suppose under TBC-GPS a chunk p of ℓ continuous bits of flow f starts transmission (i.e. p reaches the head of the queue) at τ, and the transmission completes at $\tau + \tilde{\Delta}$, Then $\tilde{\Delta} \leq \hat{\Delta}(\ell)$, where $\hat{\Delta}(\ell)$ is the transmission time cost if p is the first ℓ bits of flow f to send at time 0, with all flows of the system greedy starting from time 0. Note, without loss of generality, this paper always assume the whole system is initiated at time 0, and all token buckets are full when initiated.*

Proof: The proof is based on Lemma 10 of [12]. See Appendix B of [20] for details. ■

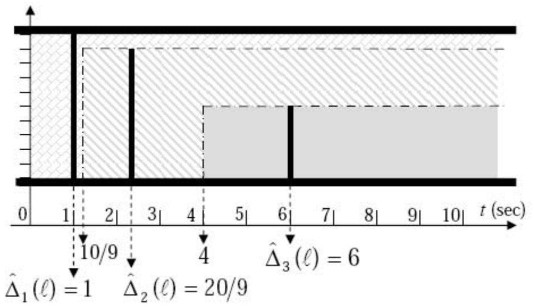
According to the above theorem, function $\hat{\Delta}(\ell)$ gives the transmission time bound for a packet of f with length ℓ. Hence, we call $\hat{\Delta}(\ell)$ the *Packet Transmission Time Bound Function*. $\hat{\Delta}(\ell)$ has the following property:

Property 1 *$\hat{\Delta}(\ell)$ is nonnegative monotonically nondecreasing. Particularly, $\forall 0 \leq \ell_1 \leq \ell_2$, $0 \leq \hat{\Delta}(\ell_1) \leq \hat{\Delta}(\ell_2)$, and $\forall 0 \leq \ell \leq \ell^{max}$, $\hat{\Delta}(\ell) \leq \hat{\Delta}(\ell^{max})$, where ℓ^{max} is the maximum packet size possible.* ■

Theorem 3 not only specifies how to calculate the packet transmission time bound function $\hat{\Delta}(\ell)$, but also implies



(a) Using common-practice *Data Throughput Proportional Weight Assignment* (DTPWA), packet transmission time bound is coupled (inverse proportional) with data throughput ρ.



(b) Using unconventional weight assignment (in fact, PAWA described in Section 3.3), packet transmission time bound still exists due to Theorem 3, and is decoupled from data throughput ρ.

Figure 5. Theorem 3's implications on decoupling packet transmission time bound from flow data throughput. Thick vertical solid line segments indicate the time when the first packets are transmitted.

this packet transmission time bound can be decoupled from flow's data throughput. The key to the decoupling is to assign proper scheduling weights. This is shown in the following example:

Example 1 *Consider a TBC-GPS server S with three input flows $F_1 \sim F_3$ as shown in Fig. 4. Suppose S's output capacity is 1, and each flow's data throughput (equivalent to the token bucket's token filling rate) is $\rho_1 = 0.1$, $\rho_2 = 0.4$, and $\rho_3 = 0.5$ respectively. For simplicity, suppose all flows' packet sizes are $\ell = 1$, and all flows' token bucket capacity equals ℓ.*

According to DTPWA, $F_1 \sim F_3$ are assigned weight $\phi_1 = 0.1$, $\phi_2 = 0.4$, and $\phi_3 = 0.5$ respectively, resulting in guaranteed rates $R_1 = \rho_1 = 0.1$, $R_2 = \rho_2 = 0.4$, and $R_3 = \rho_3 = 0.5$. With such guaranteed rates, a packet of $F_1 \sim F_3$ has a transmission time cost of $\tilde{\Delta}_1(\ell) = 10$, $\tilde{\Delta}_2(\ell) = 2.5$, and $\tilde{\Delta}_3(\ell) = 2$ respectively, as shown in Fig. 5(a). The per packet transmission time is inverse-proportional to flow's data throughput.

In contrast, suppose we assign weight $\phi_1 = 0.999$, $\phi_2 = 0.000999$, and $\phi_3 = 0.000001$. Then the greedy starting scenario is shown in Fig. 5(b), with $\hat{\Delta}_1(\ell) = 1$, $\hat{\Delta}_2(\ell) = 20/9$, and $\hat{\Delta}_3(\ell) = 6$, which are decoupled from data throughput. According to Theorem 3, $\hat{\Delta}_1(\ell) \sim \hat{\Delta}_3(\ell)$ bounds the packet transmission time for flow $F_1 \sim F_3$ respectively. Therefore, the packet transmission time is decoupled from data throughput. ■

Based on Theorem 3, we prove TBC-WFQ servers are GD servers:

Theorem 4 (TBC-WFQ Servers Are GD Servers) *If S is a TBC-GPS or TBC-WFQ server, define*

$$\mathrm{GDC}(p_f^j) \overset{def}{=} \max\{A_S(p_f^j), \mathrm{GDC}(p_f^{j-1})\} + \hat{\Delta}(\ell_f^j), \quad (7)$$

where $\mathrm{GDC}(p_f^0) = 0$ and $\hat{\Delta}(\ell)$ is the packet transmission time bound function derived from Theorem 3. Then

$$\begin{aligned} L_S^{GPS}(p_f^j) &\leq \mathrm{GDC}(p_f^j), & (8)\\ \textit{and} \quad L_S^{WFQ}(p_f^j) &\leq L_S^{GPS}(p_f^j) + \frac{\ell^{max}}{C} \\ &\leq \mathrm{GDC}(p_f^j) + \frac{\ell^{max}}{C}, & (9)\end{aligned}$$

where $L_S^{GPS}(p)$ and $L_S^{WFQ}(p)$ are the time when packet p leaves S when S is a GPS and WFQ respectively. That is, a TBC-WFQ server S is a GD server; its guaranteed delay function is its packet transmission time bound function $\hat{\Delta}(\ell)$; and its scheduling constant $\alpha = \frac{\ell^{max}}{C}$, where ℓ^{max} is the maximum size of packets entering S, and C is S's output capacity.

Proof: See Appendix C of [20]. ■

3.2 The GD-aggregate Design and E2E Delay Bound

Using TBC-WFQ servers, we build *GD server based aggregates* (denoted as *GD-aggregates*) as shown in Fig. 6. In Fig. 6, a GD-aggregate is created by aggregating several flows with *one GR* (not necessarily GD) server at the sender-end node, forwarded by *several GD* servers along intermediate nodes, and de-aggregated at receiver-end node by forwarding each packet according to its original flow header. We call the GR server that creates a GD-aggregate the "*low-end server*"; while the GD servers that forward the GD-aggregate the "*high-end servers*".

Specifically, a flow f joins other flows at *GR* server $S_L^{(1)}$ at Node 1 to output *GD-aggregate F*. $S_L^{(1)}$ guarantees f a rate of r_f, which is no less than f's data throughput. Particularly, if f complies with token bucket (σ_f, ρ_f) (denoted

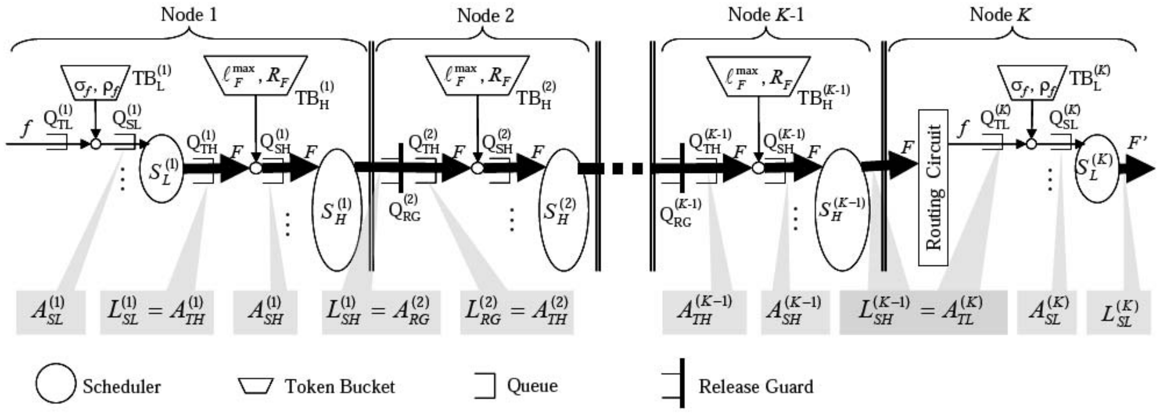


Figure 6. A Release-Guarded Token-Bucket-Constrained WFQ (TBC-WFQ) based aggregate. Labels in the figure explain the symbols denoting arrival and departure (leaving) time of packets. Routing circuits in Node $1 \sim (K-1)$ are omitted in the figure.

as $f \sim (\sigma_f, \rho_f)$) before arriving at $Q_{SL}^{(1)}$, then $r_f \geq \rho_f$. *We call $R_F \overset{def}{=} \sum_{f \in F} r_f$ and $\rho_F \overset{def}{=} \sum_{f \in F} \rho_f$ the capacity and the data throughput of GD-aggregate F respectively* (naturally $R_F \geq \rho_F$ since $r_f \geq \rho_f$). Note we use R_F as GD-aggregate capacity to maintain symbolic consistancy with [19][18]. We set the output capacity $C_L^{(1)}$ of $S_L^{(1)}$ to R_F. We also exploit the fact that $S_L^{(1)}$ is implemented in software to play a trick called *Time-Of-Scheduling-Equals-Time-Of-Leaving* (TOSETOL) [18]: once $S_L^{(1)}$ schedules a packet p, p is immediately output to $Q_{TH}^{(1)}$; meanwhile, $S_L^{(1)}$ waits another $\ell / C_L^{(1)} = \ell / R_F$ seconds before scheduling next packet, where ℓ is the length of p.

Once a packet p leaves $S_L^{(1)}$, it is regarded as a packet of GD-aggregate F, and is queued at $Q_{TH}^{(1)}$ to enter TBC-WFQ server $S_H^{(1)}$. The corresponding token bucket $TB_H^{(1)}$ has a bucket capacity of ℓ_F^{max} and a token filling rate of R_F, where $\ell_F^{max} \overset{def}{=} \max_{f \in F}\{\ell_f^{max}\}$, and ℓ_f^{max} is the maximum packet size of flow f. For simplicity, we denote $TB_H^{(1)} = (\ell_F^{max}, R_F)$.

Later, TBC-WFQ server $S_H^{(1)}$ will forward p via the physical output link to Node 2. At Node 2, p is first queued at *release guard* $Q_{RG}^{(2)}$ [11][17]. A release guard is a special kind of token bucket that *only* allows backlogged packets consuming tokens when the bucket is full, one packet at a time. For $Q_{RG}^{(2)}$, its bucket capacity and token filling rate are also ℓ_F^{max} and R_F respectively. For simplicity, we denote $Q_{RG}^{(2)} = (\ell_F^{max}, R_F)_{RG}$. Once p leaves release guard $Q_{RG}^{(2)}$, it is queued at token bucket $TB_H^{(2)} = (\ell_F^{max}, R_F)$ to enter TBC-WFQ server $S_H^{(2)}$. Later, $S_H^{(2)}$ will forward p to Node 3, which also has $Q_{RG}^{(3)} = (\ell_F^{max}, R_F)_{RG}$, $TB_H^{(3)} = (\ell_F^{max}, R_F)$, and so on and so forth, until p reaches Node K. At Node K, GD-aggregate F is de-aggregated, i.e., p is routed according to its original flow header to its corresponding output interface, where it may again join another set of flows at GR server $S_L^{(K)}$ to create another aggregate F'.

The above GD-aggregate architecture in Fig. 6 can be further simplified due to the following lemma:

Lemma 1 (All TB_Hs Can Be Ignored) *The queueing delay at $Q_{TH}^{(i)}$ ($i = 1, 2, \ldots, (K-1)$) are all* 0.

Proof: See proof of Lemma 3 in Appendix D of [20]. ■

This means all $Q_{TH}^{(i)}$ and $TB_H^{(i)}$ $(i = 1, 2, \ldots, (K-1))$ in Fig. 6 can be removed.

Based on Lemma 1, we derive the following theorem for GD-aggregate E2E delay bound:

Theorem 5 (E2E Delay$'$ with Prerequisites) *Suppose, as shown in Fig. 6, flow f joins GD-aggregate F from GR server $S_L^{(1)}$ at Node 1, traverses release guarded GD server $S_H^{(1)} \sim S_H^{(K-1)}$, and finally reaches Node K to be de-aggregated. Suppose the guaranteed delay function for aggregate F at $S_H^{(i)}$ is $\Delta_F^{(i)}(\ell)$ ($i = 1, 2, \ldots, (K-1)$); and for any valid packet length $\ell \in [\ell_F^{min}, \ell_F^{max}]$, $\Delta_F^{(i)}(\ell) \leq \frac{\ell}{R_F}$, where R_F is F's capacity. Then for jth ($j = 1, 2, \ldots$) packet of f, p_f^j, the E2E delay ${d_f^j}'$ (we use ${d_f^j}'$ instead of d_f^j to make* *symbols consistent with [19][18]) satisfies:*

$$\begin{aligned} d_f^{j\,\prime} &\overset{def}{=} L_{SH}^{(K-1)}(p_f^j) - A_{SL}^{(1)}(p_f^j) & (10)\\ &\leq [\mathrm{GRC}_{SL}^{(1)}(p_f^j) - A_{SL}^{(1)}(p_f^j)] + \sum_{i=1}^{K-1} \Delta_F^{(i)}(\ell_F^{max}) \\ &\quad + \alpha_L^{(1)} + \sum_{i=1}^{K-1} \alpha_H^{(i)}, & (11) \end{aligned}$$

where $A_{SL}^{(1)}(p)$ is when packet p arrives at $Q_{SL}^{(1)}$, $L_{SH}^{(K-1)}(p)$ is when packet p leaves $S_H^{(K-1)}$, $\alpha_L^{(1)}$ is the scheduling constant of GR server $S_L^{(1)}$ for flow f, and $\alpha_H^{(i)}$ is the scheduling constant of GD server $S_H^{(i)}$ for GD-aggregate F ($i = 1, 2, \ldots, (K-1)$). In addition, if packets arrive at $S_L^{(1)}$ in Conflict-Free pattern (defined in Theorem 2), then

$$d_f^{j\,\prime} \leq \sum_{i=1}^{K-1} \Delta_F^{(i)}(\ell_F^{max}) + \sum_{i=1}^{K-1} \alpha_H^{(i)}. \quad (12)$$

Proof: See Appendix D in [20]. ■

Corollary 1 *If flow f conforms to token bucket $TB_L^{(1)} = (\sigma_f, \rho_f)$ as shown in Fig. 6, and $S_L^{(1)}$ guarantees rate $r_f \geq \rho_f$, then Inequality (11) becomes:*

$$d_f^{j\,\prime} \leq \frac{\sigma_f}{r_f} + \sum_{i=1}^{K-1} \Delta_F^{(i)}(\ell_F^{max}) + \alpha_L^{(1)} + \sum_{i=1}^{K-1} \alpha_H^{(i)}. \quad (13)$$

Proof: Similar to the derivation of Inequality (43) of [7], we have $\mathrm{GRC}_{SL}^{(1)}(p_f^j) \leq \frac{\sigma_f}{r_f} + A_{SL}^{(1)}(p_f^j)$. ■

Corollary 2 *If, as shown in Fig. 6, flow f joins another GD-aggregate F' at Node K at GR server $S_L^{(K)}$, token bucket $TB_L^{(1)} = TB_L^{(K)} = (\sigma_f, \rho_f)$, and all other conditions are the same as those of Corollary 1, then E2E delay*

$$\begin{aligned} d_f^j &\overset{def}{=} A_{SL}^{(K)}(p_f^j) - A_{SL}^{(1)}(p_f^j) & (14)\\ &\leq \frac{\sigma_f}{r_f} + \sum_{i=1}^{K-1} \Delta_F^{(i)}(\ell_F^{max}) + \alpha_L^{(1)} + \sum_{i=1}^{K-1} \alpha_H^{(i)}. & (15) \end{aligned}$$

Proof: See Appendix E of [20]. ■

We have two observations on the E2E delay bounds in the above theorems and corollaries:

1) GD-aggregate E2E delay bound is decoupled (or partially decoupled) from data throughput: Due to Theorem 3, and as shown in Example 1, by assigning proper scheduling weight, $\Delta_F^{(i)}(\ell_F^{max})$ in Inequality (11) $\sim$ (15) can be decoupled from data throughput ρ_F. Therefore GD-aggregate E2E delay bound is decoupled from data throughput. Note if the GD-aggregate transports RTE-WAN hard real-time sensing/actuating traffic, packets can easily arrive in Conflict-Free pattern, therefore E2E delay bound is calculated with Inequality (12), which is completely decoupled from data throughput. If the GD-aggregate transports RTE-WAN hard real-time video traffic or soft real-time traffic, the E2E delay bound is partially decoupled from data throughput. But the improvement is still significant, with all $(K-2)\frac{\ell_F^{max}}{R_F}$ terms removed. What is more, the coupling problem is not a prominent defect for RTE-WAN hard real-time video traffic or soft real-time traffic anyway.

2) GD-aggregate is a generalization of GR-aggregate (with a bounded error): In Fig. 6, if we stick to DT-PWA, that is, assigning WFQ weight proportional to input flow/aggregate's data throughput (ρ_f and ρ_F), then $\Delta_F^{(i)}(\ell_F^{max}) = \hat{\Delta}_F^{(i)}(\ell_F^{max}) \leq \ell_F^{max}/R_F$, and Inequality (11) implies Inequality (3), with only a maximal possible error of ℓ_F^{max}/R_F.

However, there are still three unsettled problems in order to use Inequality (11) $\sim$ (15):

1) How to assign proper scheduling weight so that the guaranteed delay function $\Delta_F^{(i)}(\ell)$ is decoupled from data throughput?

2) Given the proper scheduling weight, how to calculate the guaranteed delay function $\Delta_F^{(i)}(\ell)$?

3) A GD-aggregate must satisfy precondition $\forall \ell \in [\ell_F^{min}, \ell_F^{max}]$, $\Delta_F^{(i)}(\ell) \leq \ell/R_F$ ($i = 1 \sim (K-1)$) to use Inequality (11) $\sim$ (15). How to remove this precondition?

The next sub-section addresses these problems.

3.3 Priority Approximating Weight Assignment (PAWA) Scheme

To address the three problems proposed in the end of last sub-section, we propose the *Priority Approximating Weight Assignment* (PAWA) scheme, with following features:

1) Introduces priorities into GD-aggregates. A GD-aggregate F's priority decides F's scheduling weight, and hence decides the guaranteed delay function $\Delta_F^{(i)}(\ell)$. Particularly, $\Delta_F^{(i)}(\ell)$ is decoupled from data throughput, and a higher priority corresponds to shorter $\Delta_F^{(i)}(\ell_F^{max})$.

2) Guaranteed delay function $\Delta_F^{(i)}(\ell)$ can be calculated with a closed-form linear formula.

3) The precondition of Inequality (11) $\sim$ (15) can be removed (at the cost of a larger E2E delay bound).

In addition, PAWA scheme allows planning with classic optimization tools, and enables simple admission tests.

The details are as follows:

The Scheme

Under PAWA scheme, a high-end[1] TBC-WFQ server

[1]As mentioned in Section 3.2, we call the GR server that creates a GD-aggregate the "*low-end server*"; while the GD servers that forward the GD-aggregate the "*high-end servers*".

S supports a set of priorities: $1, 2, \ldots, \Pi$, where smaller number means higher priority. Each priority π ($\pi < \Pi$) corresponds to three parameters: packet transmission time bound $\Delta^\star_\pi$, total aggregates' capacity $R^\star_\pi$, and total maximum packet size $\ell^\star_\pi$. During configuration time, system administrator can set these three parameters to any real numbers as long as they satisfy the following constraints:

$$\Delta^\star_0 \overset{def}{=} 0, \quad 0 < \Delta^\star_1 < \Delta^\star_2 < \ldots < \Delta^\star_{\Pi-1}; \tag{16}$$
$$R^\star_\pi \geq R^{\star min}_\pi > 0, \qquad \pi = 1 \sim (\Pi - 1); \tag{17}$$
$$\ell^\star_\pi \geq \ell^{\star min}_\pi > 0, \qquad \pi = 1 \sim (\Pi - 1); \tag{18}$$
$$\textstyle\sum_{\pi=1}^{\Pi-1} R^\star_\pi < C, R^\star_\Pi \overset{def}{=} C - \sum_{\pi=1}^{\Pi-1} R^\star_\pi; \tag{19}$$
$$C^\star_0 \overset{def}{=} 0, \quad C^\star_1 \overset{def}{=} C; \tag{20}$$
$$C^\star_\pi \overset{def}{=} C^\star_{\pi-1} - R^\star_{\pi-1} = C - \textstyle\sum_{i=1}^{\pi-1} R^\star_i, \quad \pi = 2 \sim \Pi; \tag{21}$$
$$\ell^\star_1 = \Delta^\star_1 C^\star_1; \tag{22}$$
$$\ell^\star_\pi = \Delta^\star_\pi C^\star_\pi - \Delta^\star_{\pi-1} C^\star_{\pi-1}, \pi = 2 \sim (\Pi - 1); \tag{23}$$

where C is the output capacity of S; $R^{\star min}_\pi$ and $\ell^{\star min}_\pi$ are minimum limits for $R^\star_\pi$ and $\ell^\star_\pi$ set by administrator.

Each GD-aggregate entering S must pick one priority. Denote $\mathcal{F}_\pi$ as the set of GD-aggregates entering S with priority π. To simplify our analysis, the system will add a dummy GD-aggregate $\bar{F}_\pi$ to each $\mathcal{F}_\pi$, where $\bar{F}_\pi$ is constrained by a token bucket of $(\ell^{max}_{\bar{F}_\pi}, R_{\bar{F}_\pi})$. For any $\pi < \Pi$, $\bar{F}_\pi$'s token bucket capacity $\ell^{max}_{\bar{F}_\pi} = \ell^\star_\pi - \sum_{F \in \mathcal{F}_\pi, F \neq \bar{F}_\pi} \ell^{max}_F$; for $\pi = \Pi$, $\ell^{max}_{\bar{F}_\pi}$ is set to an arbitrary constant that can be used as packet length. The token filling rate $R_{\bar{F}_\pi} = R^\star_\pi - \sum_{F \in \mathcal{F}_\pi, F \neq \bar{F}_\pi} R_F$. To insure the feasibility of setting $\ell^{max}_{\bar{F}_\pi}$ and $R_{\bar{F}_\pi}$, we require

$$\textstyle\sum_{F \in \mathcal{F}_\pi, F \neq \bar{F}_\pi} \ell^{max}_F \leq \ell^\star_\pi \quad (\forall \pi < \Pi); \tag{24}$$
$$\text{and } \textstyle\sum_{F \in \mathcal{F}_\pi, F \neq \bar{F}_\pi} R_F \leq R^\star_\pi \quad (\forall \pi = 1 \sim \Pi). \tag{25}$$

A new aggregate F is not admitted to $\mathcal{F}_\pi$ if its admission will violate Formulae (24) or (25).

The weight assignment rules run as follows:

Each $\mathcal{F}_\pi$ is assigned a total weight of ψ_π, such that

$$\psi_\pi / \psi_{\pi+1} = \Psi >> 1, \qquad \pi = 1 \sim \Pi - 1 \quad \text{and} \quad \textstyle\sum_{\pi=1}^{\Pi} \psi_\pi = 1, \tag{26}$$

where Ψ is a sufficiently large constant. For each GD-aggregate $F \in \mathcal{F}_\pi$ (including $\bar{F}_\pi$), its weight ϕ_F is

$$\phi_F = \begin{cases} \psi_\pi \ell^{max}_F / \ell^\star_\pi, & \text{when } \pi < \Pi; \\ \psi_\pi R_F / R^\star_\pi, & \text{when } \pi = \Pi. \end{cases} \tag{27}$$

Based on above rules, PAWA provides many desirable properties. First, it results in closed-form linear guaranteed delay functions:

Theorem 6 (PAWA Guaranteed Delay Function) *If TBC-WFQ server S complies with PAWA scheme, then $\forall F \in \mathcal{F}_\pi$ and $\forall \ell^{min}_F \leq \ell \leq \ell^{max}_F$, the PAWA GD server S provides F a guaranteed delay function*

$$\Delta^{(S)}_F(\ell) = \hat{\Delta}^{(S)}_F(\ell)$$
$$= \begin{cases} \Delta^\star_{\pi-1} \frac{C^\star_{\pi-1}}{C^\star_\pi} + \frac{\ell}{\ell^{max}_F} (\Delta^\star_\pi - \Delta^\star_{\pi-1} \frac{C^\star_{\pi-1}}{C^\star_\pi}), \\ \qquad \textit{when } \pi < \Pi; \\ \Delta^\star_{\Pi-1} \frac{C^\star_{\Pi-1}}{C^\star_\Pi} + \frac{\ell}{R_F}, \quad \textit{when } \pi = \Pi; \end{cases} \tag{28}$$

where $\hat{\Delta}^{(S)}_F(\ell)$ is the packet transmission time bound function mentioned in Theorem 3. Particularly, Equation (28) implies when $\pi < \Pi$, $\forall F \in \mathcal{F}_\pi$,

$$\Delta^{(S)}_F(\ell^{max}_F) = \hat{\Delta}^{(S)}_F(\ell^{max}_F) = \Delta^\star_\pi, \tag{29}$$

which is why we call $\Delta^\star_\pi$ the "packet transmission time bound" parameter[2].

Proof: See Appendix F of [20]. ■

Second, PAWA guarantees E2E delay without the $\Delta^{(i)}_F(\ell) \leq \frac{\ell}{R_F}$ prerequisite in Theorem 5. This is described in the following by Theorem 7 and 8:

Theorem 7 (PAWA TBC-WFQ Server is also GR) *Without loss of generality, suppose in Node i (e.g. $i = 1$) of Fig. 6, $Q^{(i)}_{TH}$, $TB^{(i)}_H$, $Q^{(i)}_{SH}$, and $S^{(i)}_H$ make up a PAWA TBC-WFQ server. Then for each $F \in \mathcal{F}_\pi$ ($\pi = 1 \sim \Pi$), $S^{(i)}_H$ is also a GR server with guaranteed rate R_F and scheduling constant*

$$\alpha'^{(i)}_H = \begin{cases} \Delta^\star_\pi C^\star_\pi / C^\star_{\pi+1} + \frac{\ell^{(i)max}_{SH}}{C}, & \textit{if } \pi < \Pi, \\ \Delta^\star_{\Pi-1} C^\star_{\Pi-1} / C^\star_\Pi + \frac{\ell^{(i)max}_{SH}}{C}, & \textit{if } \pi = \Pi, \end{cases} \tag{30}$$

where $\ell^{(i)max}_{SH}$ is the maximum packet length of all aggregates/flows entering $S^{(i)}_H$. That is, if define $\mathrm{GRC}^{(i)}_{SH}(p^j_F) \overset{def}{=} \max\{A^{(i)}_{SH}(p^j_F), \mathrm{GRC}^{(i)}_{SH}(p^{j-1}_F)\} + \ell^j_F / R_F$, then $L^{(i)}_{SH}(p^j_F) \leq \mathrm{GRC}^{(i)}_{SH}(p^j_F) + \alpha'^{(i)}_H$, where $L^{(i)}_{SH}(p)$ is the time when p leaves WFQ server $S^{(i)}_H$. Note $\mathrm{GRC}^{(i)}_{SH}(p^0_F) \overset{def}{=} 0$.

Proof: See Appendix G of [20]. ■

Theorem 8 (E2E Delay′ without Prerequisites) *Suppose flow f joins GD-aggregate F at $S^{(1)}_L$ and traverses $S^{(1)}_H$, $S^{(2)}_H$, ..., $S^{(K-1)}_H$, and $S^{(K)}_L$ as shown in Fig. 6. Suppose each TBC-WFQ server $S^{(i)}_H$ ($i = 1, 2, \ldots, K-1$) enforces PAWA scheme. According to Theorem 7, $S^{(i)}_H$ is also a*

[2]According to Theorem 3, every packet's transmission time under GPS is no more than $\hat{\Delta}^{(S)}_F(\ell^{max}_F)$.

GR server for F with a GR scheduling constant $\alpha_H^{'(i)}$ (see Formula (30)). Then for packet p_f^j, the E2E delay $d_f^{j\,'}$ satisfies:

$$\begin{aligned} d_f^{j\,'} &\overset{def}{=} L_{SH}^{(K-1)}(p_f^j) - A_{SL}^{(1)}(p_f^j) \\ &\leq [\mathrm{GRC}_{SL}^{(1)}(p_f^j) - A_{SL}^{(1)}(p_f^j)] + (K-1)\frac{\ell_F^{max}}{R_F} \\ &\quad + \alpha_L^{(1)} + \sum_{i=1}^{K-1} \alpha_H^{'(i)}, \end{aligned} \quad (31)$$

where $A_{SL}^{(1)}(p)$ is when packet p arrives at $Q_{SL}^{(1)}$; $L_{SH}^{(K-1)}(p)$ is when packet p leaves $S_H^{(K-1)}$; $\alpha_L^{(1)}$ is the GR scheduling constant at server $S_L^{(1)}$, and $\alpha_H^{'(i)}$ is the GR scheduling constant at server $S_H^{(i)}$. In addition, if packets arrive at $S_L^{(1)}$ in Conflict-Free pattern (defined in Theorem 2), then

$$d_f^{j\,'} \leq (K-1)\frac{\ell_F^{max}}{R_F} + \sum_{i=1}^{K-1} \alpha_H^{'(i)}. \quad (32)$$

Proof: See Appendix H of [20]. ∎

Corollary 3 *If flow f conforms to token bucket $TB_L^{(1)} = (\sigma_f, \rho_f)$ as shown in Fig. 6, and $r_f \geq \rho_f$, then Inequality (31) becomes:*

$$d_f^{j\,'} \leq \frac{\sigma_f}{r_f} + (K-1)\frac{\ell_F^{max}}{R_F} + \alpha_L^{(1)} + \sum_{i=1}^{K-1} \alpha_H^{'(i)}. \quad (33)$$

Proof: Similar to the derivation of Inequality (43) in Goyal et al. [7], we have $\mathrm{GRC}_{SL}^{(1)}(p_f^j) \leq \frac{\sigma_f}{r_f} + A_{SL}^{(1)}(p_f^j)$. ∎

Corollary 4 *If, as shown in Fig. 6, flow f joins another GD-aggregate F' at Node K at GR server $S_L^{(K)}$, token bucket $TB_L^{(1)} = TB_L^{(K)} = (\sigma_f, \rho_f)$, $r_f \geq \rho_f$, and all other conditions are the same as those of Corollary 3, then E2E delay*

$$\begin{aligned} d_f^j &\overset{def}{=} A_{SL}^{(K)}(p_f^j) - A_{SL}^{(1)}(p_f^j) \\ &\leq \frac{\sigma_f}{r_f} + (K-1)\frac{\ell_F^{max}}{R_F} + \alpha_L^{(1)} + \sum_{i=1}^{K-1} \alpha_H^{'(i)}. \end{aligned} \quad (34)$$

Proof: See Appendix I of [20]. ∎

PAWA Parameter Planning

During configuration time, a PAWA TBC-WFQ server administrator can plan the $\{\Delta_\pi^\star\}$, $\{R_\pi^\star\}$, and $\{\ell_\pi^\star\}$ parameters with classic optimization tools. Just to give an example:

Given C, $\{\Delta_\pi^\star\}$, desired total aggregates' capacity $\{\tilde{R}_\pi^\star\}$, desired total max packet size $\{\tilde{\ell}_\pi^\star\}$, weight (importance) w_π of getting $R_\pi^\star$ close to $\tilde{R}_\pi^\star$, and weight ϖ_π of getting $\ell_\pi^\star$ close to $\tilde{\ell}_\pi^\star$, derive optimal settings of $\{R_\pi^\star\}$ and $\{\ell_\pi^\star\}$.

The problem corresponds to the following convex optimization problem:

$$\min \textstyle\sum_{\pi=1}^{\Pi-1} \left(w_\pi (R_\pi^\star - \tilde{R}_\pi^\star)^2 + \varpi_\pi (\ell_\pi^\star - \tilde{\ell}_\pi^\star)^2 \right),$$

with convex linear constraint set (16) $\sim$ (23).

GD-aggregate Admission Test

To add a GD-aggregate F with priority π at PAWA TBC-WFQ server S, F only needs to pass the following three tests:

Test 1:

$$\ell_F^{max} + \sum_{f \in \mathcal{F}_\pi, f \neq \bar{F}_\pi} \ell_f^{max} \leq \ell_\pi^\star, \qquad \text{if } \pi < \Pi; \quad (35)$$

Test 2:

$$R_F + \sum_{f \in \mathcal{F}_\pi, f \neq \bar{F}_\pi} R_f \leq R_\pi^\star; \quad (36)$$

Test 3 (Theorem 5 Prerequisite): If $\forall \ell \in [\ell_F^{min}, \ell_F^{max}]$, $\Delta_F^{(S)}(\ell) \leq \ell / R_F$ ($\Delta_F^{(S)}(\ell)$ is derived from Equation (28)), then use Theorem 5, Corollary 1, or Corollary 2 to calculate E2E delay. Otherwise, use Theorem 8, Corollary 3, or Corollary 4 to calculate E2E delay.

Usually, we should assign RTE-WAN hard real-time sensing/actuating GD-aggregates with the highest priority. Because such GD-aggregates' maximum packet lengthes are small, such priority assignment will empirically always satisfy the Theorem 5 prerequisite. RTE-WAN hard real-time video GD-aggregates shall take lower priorities, which may or may not satisfy the Theorem 5 prerequisite. But the E2E delay bounds will still be satisfactory, because RTE-WAN hard real-time video traffic has large data throughput. RTE-WAN soft real-time GD-aggregates shall take lowest priorities, and will still get bounded E2E delay. All of these are illustrated by the underground mining case study described in Appendix J of [20].

4 Related Work

There are other candidate technologies for WAN virtual topologies (virtual links): Overlay network [3] also discusses virtual links. However, they are not hard real-time virtual links. DiffServ [1][21] is similar to aggregates: flows with similar QoS requirements are transmitted as one group. However, DiffServ uses FIFO scheduling, which is hard to guarantee hard real-time E2E delay when traffic is bursty. As pointed out by Wang et al. [21], even when token bucket ratio $\frac{\sigma}{\rho}$ is as low as 1.28, the maximal schedulable link utilization drops below 5%. Real-time virtual machines [5][10][16][4] can be a good candidate to support hard real-time virtual links. However, to our best knowledge, mutual exclusion is still an open problem: efficient system architecture and simple closed-form schedulability formulae are yet to be developed, especially for hierarchies with more than two levels.

In comparison, the GR-aggregate [19][18] scheme guarantees hard real-time E2E delay, assumes packetized (mutually exclusive) traffic model, supports hierarchical aggregation of arbitrary number of levels, provides closed-form analytical formulae, and can easily achieve 100% link utilization. Therefore, it is good to start RTE-WAN virtual link design on top of GR-aggregates.

There are other efforts on decoupling E2E delay bound from data throughput. Geogiadis et al. [6] also discover that the combination of per node traffic shapers (token buckets) and fair queueing weights can decouple E2E delay bound from flow data throughput. But Geogiadis et al. assume fluid model, and do not talk about aggregation. Goyal et al. [8] generalize the GR server notion to cases where guaranteed rates may differ between packets of the same flow. However, they do not talk about aggregation, and they assume the per packet guaranteed rates are either given a priori, or referring to the smallest instantaneous rates during the packets' transmission.

5 Conclusion

The convergence of computer and physical world calls for next generation WAN infrastructures for hard real-time and embedded applications. Such networks need virtual topologies to achieve scalability, configurability, and flexibility. Virtual topologies are made of virtual links, for which, the state-of-the-art building tool is *Guaranteed Rate server based aggregates* (GR-aggregates) [19][18]. However, common-practice weight assignment scheme couples GR-aggregate *End-to-End* (E2E) delay bound with aggregate's data throughput inverse proportionally. This is undesirable for many hard real-time embedded sensing/actuating applications, whose traffic has small data throughput but requires short E2E delay. We propose *Guaranteed Delay server based aggregates* (GD-aggregates) design, which allows assigning weight according to *priority* instead of data throughput. This decouples E2E delay guarantee from data throughput, hence meets the needs of hard real-time embedded applications. In addition, GD-aggregates can be analyzed with simple closed form formulae, and can be easily planned with optimization tools.

Acknowledgement

This work is supported in part by NSF CCR 03-25716, NSF CNS 06-49885 SGER, by ONR N00014-05-0739, and by a grant from Lockheed Martin and a grant from Rockwell Collins. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of sponsors. The authors thank anonymous reviewers for their advice on improving this paper.

References

[1] *An Architecture for Differentiated Services*. RFC 2475, 1998.

[2] J. C. R. Bennett et al. WF^{2}Q: Worst-case fair weighted fair queueing. *Proc. of INFOCOM'96*, pages 120–128, 1996.

[3] Y. Chu et al. A case for end system multicast. *Proc. of ACM SIGMETRICS*, 2000.

[4] R. Davis et al. Resource sharing in hierarchical fixed priority pre-emptive systems. *Proc. of IEEE RTSS'06*, 2006.

[5] Z. Deng and J. W.-S. Liu. Scheduling real-time applications in an open environment. *Proc. of IEEE RTSS'97*, 1997.

[6] L. Georgiadis et al. Efficient network qos provisioning based on per node traffic shaping. *IEEE/ACM Trans. on Networking*, 4(4), August 1996.

[7] P. Goyal et al. Determining end-to-end delay bounds in heterogeneous networks. *Multimedia Systems*, (5):157–163, 1997.

[8] P. Goyal et al. Generalized guaranteed rate scheduling algorithms: A framework. *IEEE/ACM Trans. on Networking*, 5(4):561–571, August 1997.

[9] R. Gupta and K. G. Shin. Working group summary: Infrastructure and building blocks. *NSF Cyber-Physical Systems Workshop*, October 2006.

[10] T. Kuo and C. Li. A fixed-priority-driven open environment for real-time applications. *Proc. of IEEE RTSS'99*, 1999.

[11] J. W. Liu. *Real-Time Systems*. Prentice-Hall, Inc., 2000.

[12] A. Parekh et al. A generalized processor sharing approach to flow control in integrated services networks: The single-node case. *IEEE/ACM Trans. on Networking*, 1(3):344–357, June 1993.

[13] A. K. Parekh. A generalized processor-sharing approach to flow control in integrated servcies networks. *PhD Thesis, EECS Dept., MIT*, 1992.

[14] L. L. Peterson et al. *Computer Networks: A Systems Approach (2nd Ed.)*. Morgan Kaufmann, 2000.

[15] L. Sha, A. Agrawala (Eds), T. Abdelzaher, C. D. Gill, R. Rajkumar, and J. A. Stankovic (Authors). Report of NSF workshop on distributed real-time and embedded systems research in the context of GENI. *GENI Design Document 06-32 (GDD-06-32)*, September 2006.

[16] I. Shin and I. Lee. Periodic resource model for compositional real-time guarantees. *Proc. of IEEE RTSS'03*, 2003.

[17] J. Sun. *Fixed Priority Scheduling of End-to-End Periodic Tasks*. Ph.D. Thesis, CS Dept., UIUC, 1997.

[18] W. Sun and K. G. Shin. End-to-end delay bounds for traffic aggregates under guaranteed-rate scheduling algorithms. *EECS Dept., Univ. of Michigan, Ann Arbor, Tech. Rep.*, (CSE-RT-484-03), 2003.

[19] W. Sun and K. G. Shin. End-to-end delay bounds for traffic aggregates under guaranteed-rate scheduling algorithms. *IEEE/ACM Trans. on Networking*, 13(5):1188–1201, October 2005.

[20] Q. Wang et al. Gd-aggregate: A WAN virtual topology building tool for hard real-time and embedded applications (appendices). *[Online] available at* `https://agora.cs.uiuc.edu/display/realTimeSystems/Recent+Publications`.

[21] S. Wang, D. Xuan, R. Bettati, and W. Zhao. Providing absolute differentiated services for real-tiem applications in static-priority scheduling networks. *IEEE/ACM Trans. on Networking*, 12(2):326–339, April 2004.