Scalable IP-VPN Flow Control Mechanism Supporting Arbitrary Fairness Criteria — Part 2: Simulation and Implementation —

Osamu Honda, Hiroyuki Ohsaki, Makoto Imase
Graduate School of Information Science and Technology
Osaka University, Osaka, Japan
E-mail: o-honda@ist.osaka-u.ac.jp
oosaki@ist.osaka-u.ac.jp
imase@ist.osaka-u.ac.jp

Junichi Murayama, Kazuhiro Matsuda
NTT Information Sharing Platform Laboratories
NTT Corporation, Tokyo, Japan
Email: murayama.junichi@lab.ntt.co.jp
matsuda.kazuhiro@lab.ntt.co.jp

***Abstract*— In recent years, IP-based virtual private networks (IP-VPNs), which provide a virtual privately owned network over an IP network, have attracted attention. With existing IP-VPNs, however, there is a serious problem that fairness among IP-VPN customers is not satisfied. In our previous work, we have proposed an IP-VPN fairness control mechanism called I2VFC (Inter- and Intra-VPN Fairness Control) that realizes fairness among IP-VPN customers. In this paper, we quantitatively show effectiveness of our I2VFC using simulation experiments and prototype system experiments. Focusing on inter-VPN fairness, intra-VPN fairness, and scalability, we extensively analyze the performance of I2VFC. Consequently, we show that I2VFC can realize both inter- and intra-VPN fairness under diverse control parameter configurations, indicating robustness and parameter insensitivity of our I2VFC. We also show that I2VFC has a practically sufficient scalability in terms of the transfer speed and the number of VPNs accommodated. For instance, measurement results using our prototype system show that with a modern desktop computer, I2VFC can support approximately 1.6 [Gbit/s] bandwidth and 1,300 numbers of VPNs.**

I. Introduction

In recent years, IP-based virtual private networks (IP-VPNs) [1-3], which provide a virtual privately owned network over an IP network, have attracted attention. A virtual private network can be constructed on an IP network at a lower cost than with conventional dedicated lines.

However, there is a serious problem that existing IP-VPNs cannot guarantee fairness among IP-VPN customers. This is because the IP network is a best-effort network, so the IP-VPN constructed on it is also a best-effort network. In our previous work [4], we have proposed I2VFC (Inter- and Intra-VPN Fairness Control) to achieve fair IP-VPN services within a layer 3 provider-provisioned VPN (L3-PPVPN) framework [5].

In the literature, there have been several approaches for achieving fairness among IP-VPN flows [6-9]. However, these approaches require a specific queue management mechanism be implemented at all core routers in the network. On the contrary, our I2VFC does not require such a specific queue management mechanism; I2VFC simply requires modification to Provider Edge (PE) routers.

In this paper, we quantitatively show effectiveness of our I2VFC using simulation experiments and prototype systems experiments. Focusing on inter- and intra-VPN fairness, we analyze the performance of I2VFC with simulation experiments. We show that I2VFC can realize both inter- and intra-VPN fairness under diverse control parameter configurations. We also show that I2VFC can realize arbitrary fairness including Max-Min fairness [10] in a network with multiple bottleneck links. We also show that although I2VFC does not perform active control for realizing intra-VPN fairness, the congestion point of the entire network is dispersed, so that the fairness of TCP flows operating between end hosts (i.e., intra-VPN fairness) is improved. In prototype system experiments, we show the validity of simulation experiments, and measure the CPU time and the amount of memories required for performing I2VFC control. Consequently, we show that our I2VFC has a practically sufficient scalability in terms of the transfer speed and the number of VPNs accommodated.

The structure of this paper is as follows. First, Section II explains overview of I2VFC with its key ideas. Section III quantitatively evaluates how inter- and intra-VPN fairness are achieved with the proposed I2VFC through simulation experiments. Section IV describes experimental results using an I2VFC prototype system for examining scalability in terms of the transfer speed and the number of VPNs. Finally, Section V concludes the paper and discusses future works.

II. I2VFC (Inter- and Intra-VPN Fairness Control)

This section explains an overview of our proposed inter- and intra-VPN fairness control (I2VFC), which achieves fair IP-VPN services within a layer 3 provider-provisioned VPN (L3-PPVPN) framework [5, 2]. Refer to [4] for details of I2VFC.

We show overview of I2VFC in Fig. 1. The core of I2VFC is an AIMD window flow control that operates among IP-VPN service provider's edge (PE) routers [2]. Specifically, multiple flows accommodated in the same VPN are aggregated into a single VPN flow, and stored in a logical queue for each VPN at ingress PE routers. The round-trip time and the packet loss rate

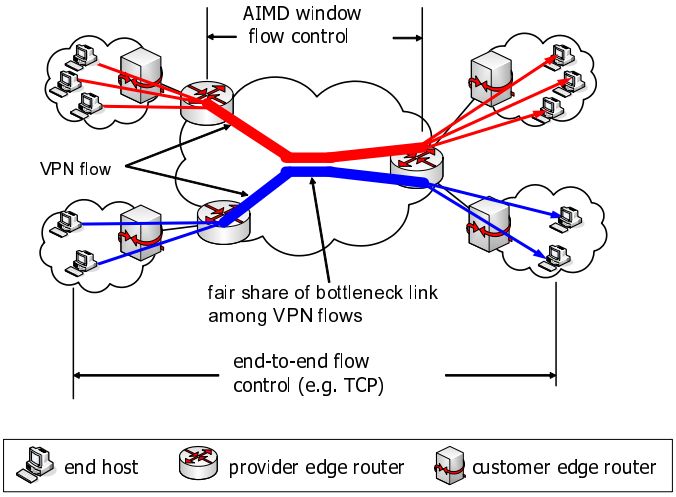

Fig. 1: Overview of I2VFC (Inter- and Intra-VPN Fairness Control)

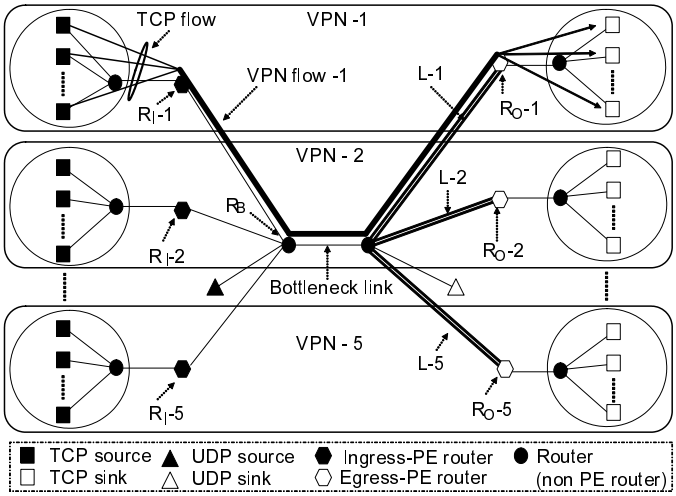

Fig. 2: Network topology with a single bottleneck link

of each VPN flow are periodically measured by exchanging management packets between ingress and egress PE routers.

Based on this information, the ingress PE router performs the AIMD window flow control [11] for adjusting the number of packets injected into the core network. Only the window flow control is performed between PE routers; i.e., neither retransmission control nor error recovery is performed. Note that VPN traffic is transferred bi-directionally, so window flow control is performed for VPN flows both upwards and downwards.

AIMD window flow control for each VPN is performed at PE routers, so inter-VPN fairness is achieved. Namely, parameters of AIMD window flow control (additive increase factor a and multiplicative decrease factor b [11]) are set appropriately based on the measured round-trip time and packet loss rate, and fairness criteria specified by the IP-VPN service provider. Thus, I2VFC can achieve arbitrary fairness criteria among VPN customers (inter-VPN fairness); i.e., the ratio of VPN flow throughputs can be arbitrary controlled by the service provider.

Intra-VPN fairness is achieved by simply relying on TCP's congestion control mechanism operating between end hosts. Namely, I2VFC itself does not perform any control for achieving intra-VPN fairness. The congestion control mechanism of TCP achieves sufficient intra-VPN fairness because flows accommodated in the same VPN have the same round-trip time and the same packet loss rate [4].

In I2VFC, AIMD window flow control operates between PE routers, and TCP window flow control operates between end hosts. That is, two independent feedback-based controls operate simultaneously between PE routers and between end hosts. As have been pointed out in TCP over ABR studies [12], there might be a risk of performance degradation due to mutual interference of different feedback-based controls. I2VFC's window flow control avoids such interference of feedback controls by operating at much larger timescale than TCP's one.

III. Simulation

In this section, we evaluate the effectiveness of I2VFC through simulation experiments. Evaluation is conducted by focusing, in particular, on inter- and intra-VPN fairness.

A weighted fairness index F defined by the following equation [8, 13] is used as a performance metric for inter-VPN fairness and intra-VPN fairness.

$$F = \frac{(\sum_i^N \frac{x_i}{r_i})^2}{N \sum_i^N (\frac{x_i}{r_i})^2} \quad (1)$$

x_i is the throughput of the i-th flow, r_i is the weight of the i-th flow (taking 1.0 when assessing intra-VPN fairness), and N is the number of flows in the network. The weighted fairness index F takes a value between 0 and 1, with $F = 1$ when fairness is completely satisfied and with F close to 0 when fairness is not satisfied.

I2VFC is implemented in OPNET simulator version 9.1A [14]. The simulation time is 50 seconds. For a given parameter set, simulation is repeated 10 times, and the average of the weighted fairness index F is calculated. In all simulations, the 95% confidence interval of the weighted fairness index F is within 2% of its average, so the confidence interval is not shown in the following results.

A. Case of a Single Bottleneck Link

First, we investigate inter- and intra-VPN fairness in a network with a single bottleneck link (Fig. 2). Data transfer is performed continuously using multiple TCP flows from a sending host to its receiving host starting at $t = 0$ [s]. The weight of each VPN and the propagation delay of each link used in simulations are shown in Tabs. I and II, respectively.

UDP traffic is generated on the bottleneck link as background traffic. The average arrival rate of background traffic

TABLE I: Weight of each VPN flow (case of a single bottleneck link)

VPN flow	weight (r_i)
VPN 1	1.0
VPN 2	2.0
VPN 3	2.0
VPN 4	3.0
VPN 5	4.0

TABLE II: Propagation delay of each link (case of a single bottleneck link)

link	propagation delay [s]
L-1	0.050
L-2	0.025
L-3	0.075
L-4	0.050
L-5	0.025

is 30% of the bottleneck link bandwidth and the packet length is fixed at 1,500 [Byte]. The inter-packet time is exponentially distributed. Unless otherwise noted, the following parameters are used in simulations; the number of VPN flows is 5, the bottleneck link bandwidth is 50 [Mbit/s], the router buffer size is 50 [packet], the number of TCP flows in each VPN flow is 30, the management packet interval (i.e., the number of packets sent between two consecutive management packets) is $\Delta = 4$, and the propagation delay of links except $linkL-1 \sim L-5$ is very small (i.e., 5.06×10^{-6} [s]).

First, the effect of the additive increase factor a and the multiplicative decrease factor b on inter-VPN fairness is investigated. TCP's congestion avoidance phase operating on end hosts corresponds to AIMD window flow control with $a = 1.0$ and $b = 0.5$. Hence, it is expected that I2VFC's window flow control operates satisfactorily with parameter settings of $a < 1.0$ and $b < 0.5$ [4].

Simulation results are shown for the additive increase factor $a = 0.01, 0.1$, and 1 and for the multiplicative decrease factor b being set as in Tab. I so that fairness can be realized. Figures 3 and 4 show evolutions of the fairness index for inter-VPN fairness when the multiplicative decrease factor for VPN flow 1 is respectively set to $b = 0.1$ and $b = 0.25$. Values of the multiplicative decrease factor b for other VPN flows are chosen based on Eq. (1) according to the measured packet loss rate and round-trip time [4]. In these figures, throughput of the VPN flow (VPN throughput) every 1 [s] and the weighted fairness index F are plotted. For comparison purpose, simulation results without I2VFC are also plotted. We note that in all simulations, utilization of the bottleneck link is almost 100 [%].

These figures show that inter-VPN fairness can be achieved with an extremely high accuracy ($F > 0.9$) regardless of settings of additive increase factor a and multiplicative decrease factor b. In addition, focusing on variations of inter-VPN fairness index, one can find that the transient performance is not good when the additive increase factor a is 1.0. One can also find that transient performance is not affected when

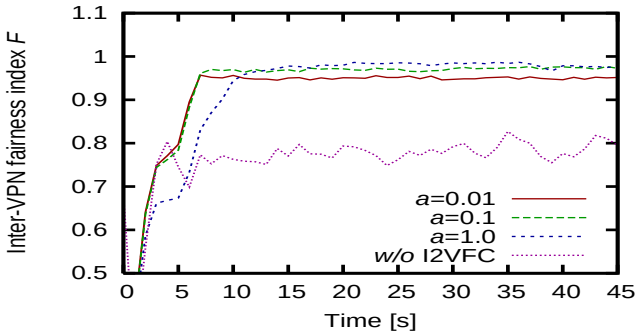

Fig. 3: Evolution of inter-VPN fairness index (multiplicative decrease factor $b = 0.1$ for VPN flow 1)

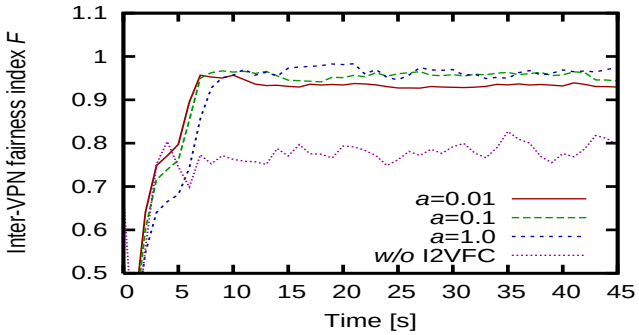

Fig. 4: Evolution of inter-VPN fairness index (multiplicative decrease factor $b = 0.25$ for VPN flow 1)

$a \leq 0.5$. This is probably because I2VFC's window flow control with $a = 1.0$ interferes with TCP's window flow control operating on end hosts. In addition, comparison of Figs. 3 and 4 shows that settings of the multiplicative decrease factor b have no substantial effect on inter-VPN fairness.

Based on these observations, we conclude that I2VFC's window flow control achieves satisfactory inter-VPN fairness with parameter settings of $a < 1.0$ and $b < 0.5$.

Next, the effect of the additive increase factor a and the multiplicative decrease factor b on intra-VPN fairness is investigated. There exist 4 VPN flows, and the additive increase factor is changed to $a = 0.1, 0.5, 1$, and 5. The multiplicative decrease factor is changed to $b = 0.1, 0.5$, and 0.75. The fairness index F for intra-VPN fairness is shown in Figs. 5 and 6. Figure 5 is the result with two TCP flows in each VPN flow. On the contrary, Fig. 6 is the result with 10 TCP flows in each VPN flow. For comparison purposes, simulation results without I2VFC are also included in all figures. The results show that the fairness index is 0.748 for two TCP flows in each VPN flow, and 0.604 for 10 TCP flows in each VPN flow.

Figures 5 and 6 also show that intra-VPN fairness is substantially affected by values of the additive increase factor a and multiplicative decrease factor b. One can find that intra-VPN fairness improves in particular when values of a and b are

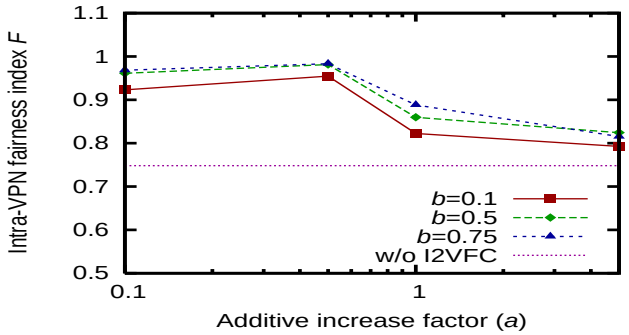

Fig. 5: Weighted fairness index for intra-VPN fairness (two TCP connections in each VPN flow)

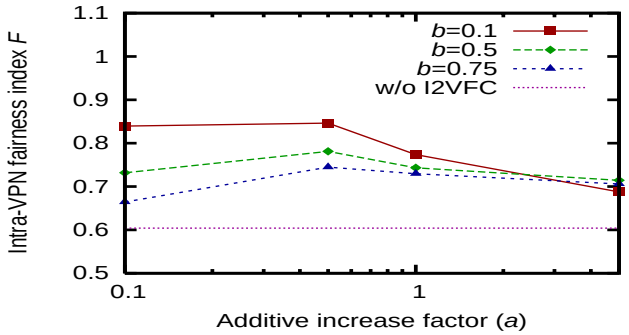

Fig. 6: Weighted fairness index for intra-VPN fairness (10 TCP connections in each VPN flow)

small. Such phenomenon can be explained by the interference between I2VFC's window flow control and TCP's window flow control; i.e., when values of a and b are large (e.g., $a \geq 1$ and $b \geq 0.5$), I2VFC's window flow control interferes with TCP's window congestion control. On the contrary, when values of a and b are small, two congestion controls works almost independently, leading fair bandwidth allocation to TCP flows in each VPN flow.

Note that in Figs. 5 and 6, intra-VPN fairness is improved by introducing I2VFC's window flow control regardless of settings of the additive increase factor a and multiplicative decrease factor b. For example, intra-VPN fairness is improved from 0.748 to 0.815 for $a = 5.0$ and $b = 0.75$ in Fig. 5 even when frequency of I2VFC's window flow control is comparable to that of TCP's window flow control. Such fairness improvement can be explained by dispersion of congestion at the bottleneck link; i.e., by introducing I2VFC's window flow control between ingress and egress routers, the bottleneck link is less congested than the case without the I2VFC's control. Therefore, packets from TCP connections in each VPN flow are less likely to be dropped, leading more stable behavior (e.g., less timeouts) of TCP connections.

Based on these observations, we conclude that introducing I2VFC's window flow control improves intra-VPN fairness regardless of the settings of the additive increase factor a and multiplicative decrease factor b. This is a surprising result; even though I2VFC does not perform any active control to improve intra-VPN fairness, it disperses the congestion and consequently results in much better fairness among TCP connections than the case without I2VFC's *inter-VPN fairness* control.

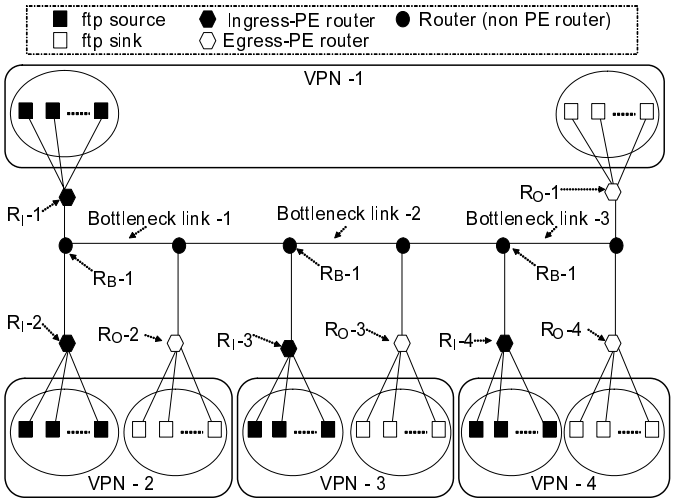

Fig. 7: Network topology with multiple bottleneck links

TABLE III: Weight of each VPN flow (case of multiple bottleneck links)

	B_3 [Mbit/s]		
VPN flow	10	20	30
VPN 1	1.0	1.0	1.0
VPN 2	3.0	3.0	3.0
VPN 3	1.0	1.0	1.0
VPN 4	1.0	3.0	5.0

B. Case of Multiple Bottleneck Links

Next, inter-VPN fairness in a generic network with multiple bottleneck links (Fig. 7) is investigated. We show that Max-Min fairness can be achieved with our I2VFC in a network with multiple bottleneck links.

In the following simulations, we set the bandwidth of link 1 to 20 [Mbit/s] and that of link 2 to 10 [Mbit/s] while changing the bandwidth of link 3 (B_3) to 10, 20, and 30 [Mbit/s]. The weight of each VPN is calculated to satisfy the Max-Min fairness as shown in Tab. III. The router's buffer size is 200 [packet], there exist 30 TCP flows in each VPN flow, and propagation delays of all links are very small (i.e., 5.06×10^{-6} [s]).

The additive increase factor for all VPN flows is equally set to $a = 0.5$. The multiplicative decrease factor for VPN flow 1 is set to $b = 0.01$. Values of the multiplicative decrease factor b for other VPN flows are configured according to the measured packet loss rate and round-trip time [4]. Evolution of the fairness index for inter-VPN fairness is plotted in Fig. 8.

Figure 8 shows that in all cases inter-VPN fairness can be achieved with an extremely high accuracy (i.e., $F >$

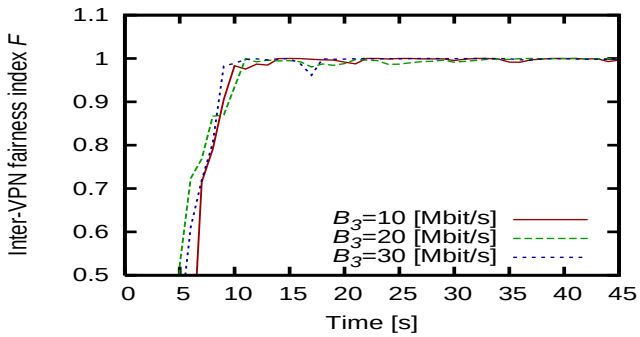

Fig. 8: Evolution of weighted fairness index for inter-VPN fairness (case of multiple bottleneck links)

0.95) regardless of the bandwidth of the link 3. Namely, the proposed I2VFC achieves arbitrary fairness by setting the additive increase and multiplicative decrease factor, a and b, based on the packet loss rate and round-trip time, even when each VPN flow traverses different bottleneck links.

We then focus on how quickly inter-VPN fairness converges. Figure 8 shows that the fairness index for inter-VPN fairness converges after 10 [s] since all VPN flows start their data transmissions regardless of the bandwidth of the link 3. Since the round-trip time of longest VPN flow (VPN flow 1) is 0.312 [s] when the bandwidth of the link 3 is 10 [Mbit/s], one can find that good transient performance (i.e., convergence in approximately 16 times of the round-trip time) is realized.

Based on these observations, we conclude that using I2VFC realizes arbitrary fairness including the Max-Min fairness even in a network with multiple bottleneck links. Also, we conclude that inter-VPN fairness has good transient performance.

IV. Experiments with Prototype System

A. Prototype System Overview

Using C programming language, an I2VFC prototype system has been implemented as an application running in the user space (Fig. 9). An ingress PE router consists of three processes to respectively process packet sending, packet receiving, and window flow control. An egress PE router consists of two types of processes to respectively process packet sending and packet receiving. The libpcap version 0.6.2 is used for packet receiving, and packet sending is implemented using raw sockets. Inter-process communication is implemented using shared memory.

The topology of the network used in measurements with the prototype system is shown in Fig. 10. The following equipments/softwares are used:

- Sending host and receiving host
  A computer running a Linux operating system is used, and multiple TCP flows are generated by a TCP benchmark software [15]. Every VPN is identified by the destination port number.
- Ingress PE router and egress PE router
  A computer running a Linux operating system is used.

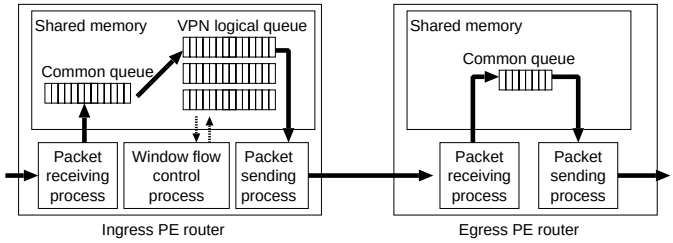

Fig. 9: I2VFC prototype system overview

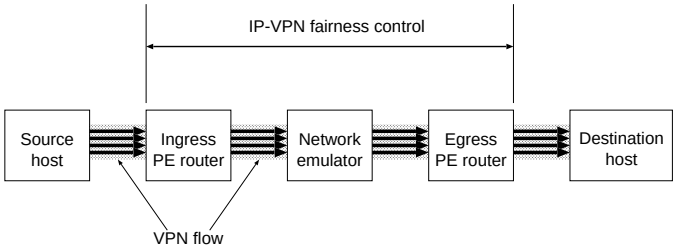

Fig. 10: Network topology used in experiments with prototype system

- Network emulator
  A network emulator is used to simulate various network environments; the bandwidth and delay for the bottleneck link are changed. A computer running a FreeBSD operating system is used, and dummynet [16] is used as a network emulator. With dummynet, the bandwidth, delay, and buffer size can be configured.

Specifications for respective equipments/softwares (e.g., CPU, memory, type of OS) are shown in Tab. IV. Unless otherwise noted, parameter settings shown in Tab. V are used in experiments.

B. Evaluation of Inter- and Intra-VPN Fairness

In experiments using the prototype system, there are 4 VPN flows; there is one TCP flow in VPN flow 1 and VPN flow 3, and two TCP flows in VPN flow 2 and VPN flow 4. Data transfer is initiated from VPN flow 1 to VPN flow 4 in order every 5 [s], and the VPN throughput and TCP throughput are measured.

TABLE V: Parameter configuration in experiments with prototype system

Number of VPN flows	4	
Number of TCP flows comprising VPN flows	1, 2	
Weight of VPN flows	1.0	
Buffer size for each VPN flow	128	[packet]
Additive increase factor a	0.1	
Multiplicative decrease factor b	0.1	
Interval of management packets Δ	4	
Bandwidth of the network emulator	50	[Mbit/s]
Delay of the network emulator	2	[ms]
Buffer size of the network emulator	50	[packet]

TABLE IV: Equipment/software specifications used in experiments with prototype system

	CPU	memory	OS	NIC driver
source host	Celeron 1.06 GHz	384 MByte	Linux 2.4.22	e100-2.3.18
destination host	Celeron 1.06 GHz	384 MByte	Linux 2.4.22	e100-2.3.18
ingress PE router	Pentium4 1.70 GHz	256 MByte	Linux 2.4.20	e1000-4.4.19
egress PE router	Celeron 2.00 GHz	512 MByte	Linux 2.4.20	epic100-1.11, 8139to-0.9.24
network emulator	Pentium4 2.26 GHz	256 MByte	FreeBSD 5.2.1	fxp, tx

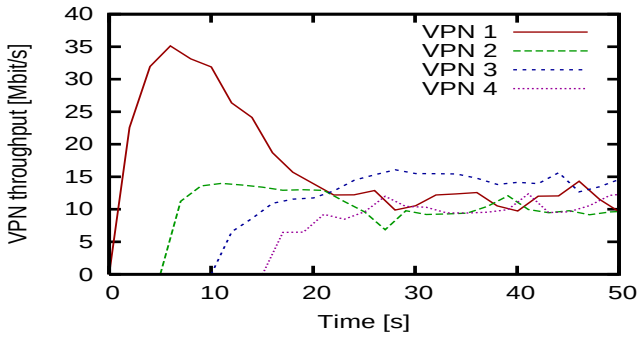

Fig. 11: Evolution of instantaneous VPN throughput

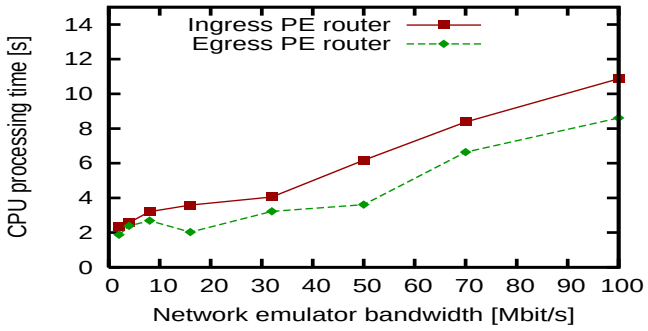

Fig. 12: Relation between network emulator bandwidth and total CPU processing time consumed by ingress/egress PE router

To examine in detail the behavior of each VPN flow, evolution of the VPN throughput is measured instead of the weighted fairness index F. Evolution of the VPN throughput is shown in Fig. 11. Variation of the instantaneous VPN throughput for every 2 [s] is plotted in this figure. TCP throughput in each VPN flow is also shown in Fig. 11. The figure shows that inter-VPN fairness is achieved; i.e., VPN throughput is almost equal at approximately $t = 25$ [s] regardless of the number of TCP flows comprising each VPN flow. Comparison with simulation results (see Fig. 3) shows good agreement in terms of the convergence time of VPN throughput.

Based on results in prototype system experiments (Fig. 11), the weighted fairness index F for intra-VPN fairness is calculated. The result is that the weighted fairness index for VPN flow 2 is $F = 0.99$; the weighted fairness index for VPN flow 4 is $F = 0.99$. From these results, we find that intra-VPN fairness is also achieved. These values roughly agree with the simulation results ($F = 0.92$ in Fig. 5).

These observations confirm that inter- and intra-VPN fairness are achieved in the prototype system. Moreover, we also confirm that simulation results and measurement results with the prototype system mostly agree. Thus, the validity of simulation experiments and measurement experiments using the prototype system is confirmed.

C. Evaluation of Scalability

To evaluate I2VFC scalability, the CPU time and the amount of memories used by the ingress and egress PE routers are measured using the prototype system. For the CPU time, the running time of each module is measured using a C compiler profiler. Specifically, data transfer is lasted for 180 [s] after all VPN flows started their data transmission, and the total amount of CPU time is measured. I2VFC does not perform dynamic memory allocation; i.e., it performs only static memory allocation. Thus, memory required by I2VFC is calculated as the total memory statically allocated by the I2VFC prototype.

To evaluate scalability regarding bandwidth and the number of VPNs, experiments are repeated by changing the network emulator's bandwidth as 10–100 [Mbit/s] and changing the number of VPN flows as 2–100. Either one or two TCP flows comprise each VPN flow. Regarding other parameters, values in Tab. V are used.

First, the CPU time used by the ingress and egress PE routers, when the number of VPN flows is fixed at 100 and the network emulator's bandwidth is changed, is shown in Fig. 12. This figure shows that the total CPU time consumed by the ingress and egress PE routers during 180 [s]. This figure shows that the CPU time used by the ingress and egress PE routers increase almost linearly to the bandwidth. For instance, the CPU time used by the ingress PE router is 3.12 [s] when the network emulator's bandwidth is 100 [Mbit/s], although this is approximately 1.73% when converted to CPU utilization. Thus, bandwidth up to approximately 5,700 [Mbit/s] can be supported using the devices used in experiments when there are 100 VPN flows.

Next, the CPU time used by the ingress and egress PE routers, when the network emulator's bandwidth is fixed at 100 [Mbit/s] and the number of VPN flows is changed, is shown in Fig. 13. The figure shows that the CPU time

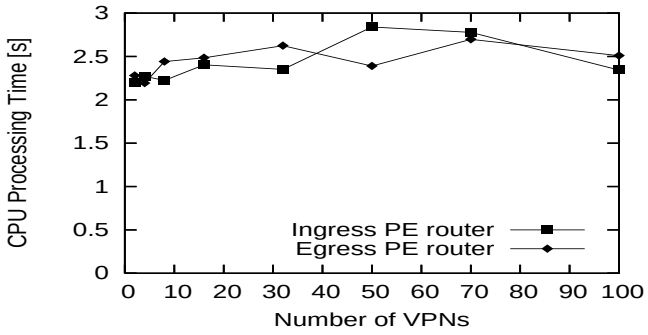

Fig. 13: Relation between the number of VPN flows and total CPU processing time consumed by ingress/egress PE router

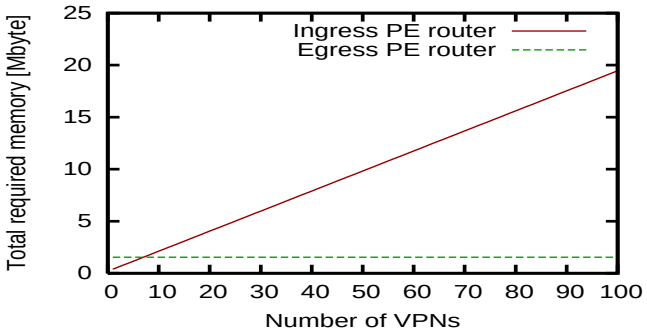

Fig. 14: Relation between the number of VPN flows and memory usage in ingress/egress PE router

used by the ingress PE routers slightly increase according to the number of VPN flows, and that the CPU time used by the egress routers are almost constant regardless of the number of VPN flows. For instance, the CPU time used by the ingress router when there are 100 VPN flows is 3.12 [s] although this is approximately 1.73% when converted to CPU utilization. Thus, up to approximately 16,000 VPN flows can be supported when the bandwidth is 100 [Mbit/s] using equipments/softwares used in our experiments.

Finally, the memory used by the ingress and egress PE routers, when the network emulator's bandwidth and the number of VPN flows are changed, is shown in Fig. 14. One can find that the memory used by the ingress and egress PE routers is constant regardless of the bandwidth. This figure shows that the memory used by the ingress PE router is almost proportional to the number of VPN flows. In contrast, the memory used by the egress PE router is almost constant regardless of the number of VPN flows. This is because most of the memory used is allocated as a buffer needed for window flow control by the ingress PE router. For example, the memory used by the ingress PE router is 19.5 [MByte] and that used by the egress PE router is 1.55 [MByte] when there are 100 VPN flows. Thus, up to approximately 1,300 VPN flows can be supported using the devices used in our experiments.

V. Conclusion

This paper quantitatively evaluates the effectiveness of I2VFC proposed in our previous work [4] through simulation experiments and prototype system experiments. The results demonstrated that (1) inter-VPN fairness can be achieved with an extremely high accuracy regardless of settings of I2VFC control parameters, (2) arbitrary fairness including the Max-Min fairness can be achieved even in a network with multiple bottleneck links, (3) congestion of the bottleneck link is dispersed, and consequently fairness among TCP connections (intra-VPN fairness) is improved, and (4) our I2VFC has a practically sufficient scalability in terms of the transfer rate and the number of VPNs accommodated.

Our future work includes tuning of other control parameter; i.e., the management packet interval Δ for optimizing the performance of I2VFC. Our future work also includes investigation on effectiveness of I2VFC for multimedia traffic such as VoIP and audio/video streaming.

References

[1] B. Gleeson *et al.*, "A framework for IP based virtual private networks," *Request for Comments (RFC) 2764*, Feb. 2000.

[2] M. Carugi and D. McDysan, "Service requirements for layer 3 provider-provisioned virtual private networks (PPVPNs)," *Request for Comments (RFC) 4031*, Apr. 2005.

[3] A. Nagarajan, "Generic requirement for provider-provisioned virtual private networks (PPVPN)," *Request for Comments (RFC) 3809*, June 2004.

[4] O. Honda, H. Ohsaki, M. Imase, J. Murayama, and K. Matsuda, "Scalable IP-VPN flow control mechanism supporting arbitrary fairness criteria — part 1: Architecture design —," in *Proceedings of Fourteenth International Conference on Computer Communications and Networks (ICCCN 2005)*, Oct. 2005, pp. 173–178.

[5] R. Callon and M. Suzuki, "A framework for layer 3 provider-provisioned virtual private networks PPVPNs," *Request for Comments (RFC) 4110*, July 2005.

[6] I. Khalil and T. Braun, "Edge provisioning and fairness in VPN-DiffServ networks," *JNSM*, vol. 10, no. 1, pp. 11–38, Mar. 2002.

[7] A. Sang, H. Zhu, and S. qi Li, "Weighted fairness guarantee for scalable diffserv assured forwarding," *Computer Communications Journal*, vol. 8, pp. 2365–2369, Mar. 2001.

[8] R. Pletka, A. Kind, M. Waldvogel, and S. Mannal, "Closed-loop congestion control for mixed responsive and non-responsive traffic," in *Proceedings of IEEE GLOBECOM 2003*, Dec. 2003, pp. 4180–4186.

[9] H. T. Kung and S. Y. Wang, "TCP trunking: Design, implementation, and performance," in *Proceedings of IEEE International Conference on Network Protocols '99*, Oct. 1999.

[10] D. Bertsekas and R. Gallager, *Data Networks*. Englewood Cliffs, New Jersey: Prentice-Hall, 1987.

[11] D.-M. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *Computer Networks and ISDN Systems*, vol. 17, no. 1, pp. 1–14, June 1989.

[12] S. Kalyanaraman, R. Jain, S. Fahmy, R. Goyal, J. Jiang, and S.-C. Kim, "Performance of TCP over ABR on ATM backbone and with various VBR traffic patterns," in *Proceedings of IEEE ICC '97*, June 1997.

[13] R. Jain, *The Art of Computer Systems Performance Analysis*. New York: Wiley-Interscience, Apr. 1991.

[14] Opnet Technologies, Inc., "OPNET," http://www.opnet.com/.

[15] "The TCP/UDP bandwidth measurement tool," http://dast.nlanr.net/Projects/Iperf/.

[16] L. Rizzo, "Dummynet: a simple approach to the evaluation of network protocols," *ACM Computer Communication Review*, vol. 27, no. 1, pp. 31–41, Jan. 1997.