# Defending Against High-Bandwidth Traffic Aggregates

**Takanori Komatsu[†] and Akira Namatame[††],**

Dept. of Computer Science - National Defense Academy of Japan

**Summary**

Network flows should adjust their sending rates to avoid a congestion collapse. Congestion collapses can be mitigated using improved packet scheduling based on a crowd control or an active queue management. However, the problem is associated with dynamic conditions such as underlying network topology, network load, and the reactions of transport protocols to congestion. Therefore, we have to evaluate what type of control mechanisms can solve this problem most effectively.

The research aim of this paper is to evaluate the effectiveness of the congestion control schemes. Adaptive flows adjust the rate, while unresponsive flows do not respond to congestion and keep sending packets. Unresponsive flows waste resources by taking their share of the upstream links of a domain and dropping packets later when the downstream links are congested. For instance, random early detection (RED) exemplifies this class of algorithms. A router only maintains a simple FIFO queue for all traffic flow and drops the arriving packet randomly during congestion. The probability to drop a packet increases with the queue length. By keep the output queue size small, RED can reduce the delay time for most of the traffic flow. However, RED cannot penalize the misbehaving traffic flows.

We evaluate, the congestion control schemes such ach Drop Tail, RED, CHOKe, and ACC with push back using unresponsive flows and in presence of short and long-lived background traffic. We use several network topologies to identify unresponsive flows that cause packet drops in other flows. We also simulate how various queuing algorithms implemented in a network router perform during an attack, and whether legitimate users can obtain desired service. The simulations show CHOKe and ACC with push back are successful in providing bandwidth requested by the legitimate user during DDoS attack.

***Key words:***
*DDoS attack, complex network, bandwidth control.*

## 1. Introduction

There exist various Internet-related security risks. One such risk is the DDoS attack, which causes the network to become congested and causes servers to be down by sending huge packets.

In general, DDoS attacks have the following two properties.

(1) They force the target computer(s) to reset or consume its resources such that it can no longer provide its intended service (e.g., SYN Flood).

(2) They obstruct the communication media between other users and the target computer such that adequate communication is no longer possible (e.g., UDP flood).

In the present paper, we use simulation to investigate bandwidth controls to mitigate the UDP flood problems. A UDP flood attack is a denial-of-service (DoS) attack using User Datagram Protocol (UDP). An attack on a specific host causes extreme network congestion in addition to decreased performance of the host. As a result, normal flows are restricted by DoS attack flows.

UDP does not have a packet control mechanism (a source will reduce its sending-rate when it infers through packet loss when the network is unable to sustain the current sending-rate, as in TCP). To make matters worse, the inherent design of the current Internet imposes no penalty or charge on unresponsive traffic. In turn, this may cause more attackers to implement unresponsive transmission policies for efficient attack.

There have been several approaches proposed for handling DDoS attacks. Congestion-based approach [1], anomaly-based approach [2] and source-based approach [3] are main approach in terms of attack detection. In addition, several bandwidth control methods have been proposed to mitigate the damage of congestion. However, they are basically simulated only in a simple network (e.g., lattice model, few nodes network).

The purpose of the present paper investigates the effectiveness of bandwidth controls, and we compare bandwidth control methods. We simulate DDoS attacks in several complex networks that represent the properties of internet topology, including the tiers model [4], the transit-Stub model [5][6], and the Barabasi-Albert model [7]. These networks are, respectively, characterized as hierarchy, domain architectures, and scale-free networks.

We analyze three queue methods and one pushback method [8] in two types DDoS attack simulations.

In order to improve the quality of the fairness simulation (consider eight flows of the same rate), we first use the droptail queue method. In this case, some flows barely reach the destination node, as a result of a DDoS Attack. Second, we use the Random Early Drop (RED) [9] queue method. The RED queue method improves the

Manuscript received February 5, 2007
Manuscript revised February 25, 2007

fairness between flows. Finally, we use CHOKe, which is a stateless active queue management method [10]. CHOKe accomplishes the best performance among the three queue methods in all networks. Pushback with the aggregate congestion control method performs as well as CHOKe in all network topologies. As a result, we show that CHOKe (simple queue method) has the same ability as pushback with ACC (complex bandwidth control) for maintaining the fairness between flows.

The remainder of the paper is organized as follows. Section 2 describes related literature. In Section 3, the congestion control methods are presented, and in Section 4, we discuss the results of two types of simulation. Finally, a discussion and conclusions are presented in Sections 5 and 6, respectively.

## 2. Related literature

### 2.1 CITRA

The CITRA (Cooperative Intrusion Traceback and Response Architecture) architecture[12] was designed to mitigate the effects of DoS attacks by using a rate-limiting mechanism, which is quite similar to the aggregate congestion control with a pushback system presented in the next chapter.

The latest published version of CITRA has a two-level organization. At the highest level, administrative domains controlled by a component called the Discovery Coordinator (DC) are called CITRA communities. A DC is a device with human oversight that controls and monitors activity throughout a community. One community is then divided into CITRA neighborhoods. A neighborhood is a set of CITRA-enabled devices that are directly adjacent, i.e., that are not separated by any CITRA-enabled boundary controller, such as routers or firewalls. Every CITRA-enabled device collects network audit data. If one device detects an attack, it sends the attack identification data to its neighbors and requests that they check whether they are also on the attack path. Neighbors compare the attack pattern with their own audited data and determine whether they are on the attack path. If they are on the attack path, they repeat the request to their own neighbors. Thus, the attack is gradually traced back to its source or to the boundary of the CITRA system. In addition to tracing the attack, each CITRA-enabled device also performs an automated response defined according to a certain policy. Possible actions include blocking the traffic and limiting its authorized bandwidth.

The CITRA architecture is implemented and tested [12]. The tests deal only with well-identified traffic aggregates. Only the attack traffic suffered rate-limiting, while the legitimate traffic passed through the system without penalties.

However, perfect traffic aggregate identification is not currently possible. The performances of IDSs suffer from false positives. However, if perfect attack detection were possible, why would rate-limiting be used when blocking would be more effective? Rate-limiting is used in order to avoid the effect of by blocking on the legitimate traffic that is mis-detected as belonging to an attack. This aspect has not been taken into account in the tests of the study [12].

### 2.2 Monitoring the Macroscopic Effect of DDoS Flooding Attacks

This method, which is presented in [13], is efficient for early attack detection. Using only a few observation points, this method can monitor the macroscopic effect of DDoS flooding attacks. The previous research reported that such macroscopic-level monitoring might be used to capture shifts in spatiotemporal traffic patterns caused by various DDoS attacks, and could then describe where and when a DDoS attack may occur in transit or source networks [13]. This monitoring method enables DDoS attack detection without any traffic observation in the victim network.

This method consists of five steps. In the first step, observation points are deployed in the network. Consider the case in which there are N subnets in test-network, where L ($L < N$) subnets routers are deployed as observation points to log outbound traffic. In the second step, the flow vector is calculated. Each element of this flow vector is itself a vector defining the number of packets flowing into the corresponding subnet from each of the observation subnets during a given time interval. Each element of the vector is normalized by its mean and standard deviation. In the third step, cross correlation between flow vectors is measured in order to obtain a cross correlation matrix. In the fourth step, the eigenvalue and vector of the cross correlation matrix are calculated. The eigenvector $w$ corresponding to the largest eigenvalue $\lambda$ often has special significance in many applications. In the final step, the weight vector, which represents a summary of the network-wide traffic load, is defined using the eigenvector $w$.

In addition, this method is tested under several DDoS attack types with many background flows. In all DDoS attack types, this method roughly counteracts the effects of background traffic (legitimate traffic) by cross correlation.

This result is helpful in simulating DDoS attacks. In general, computer simulation requires complex situation to simulate real problems. This simulation usually requires a great deal of time and needlessly complicates the problem. However, essentially, we need not take the background

traffic into consideration when using the cross correlation method. Therefore, we herein use the main flow to simulate a DDoS Attack.

## 3. Congestion Control Methods

Several rate-limiting congestion control methods have been proposed to mitigate internet traffic. In the present study, we used methods of the following forms.

### 3.1 Droptail

Droptail has a finite queue and implements FIFO scheduling. This is typical of most present-day Internet routers. Droptail is a rather simple discipline that does not rely on estimating traffic properties. If the queue is full, no incoming packets can enter the queue until the buffer space becomes available. Thus, sometimes the queue is filled by only one flow. Droptail does not have a congestion avoidance mechanism. Traffic bursts are common in packet networks, and, hence, an almost full droptail queue may cause multiple packet drops.

### 3.2 Random Early Discard: RED

RED[9] is an advanced queue method. RED drops packets from the queue with a certain probability, which increases with the average queue length. Thus, the queue is not filled by only one flow (which will happen in Droptail). RED does not classify traffic. Efficient packet dropping requires several configuration parameters: buffer capacity, lower threshold $\min_{th}$, upper threshold $\max_{th}$, and weight coefficient wq. RED continuously estimates the average queue length (*avg*) and instantaneous queue length (*q*):

$$avg_i = (1 - w_q)avg_{i-1} + + w_q q \qquad (1)$$

Threshold parameters $\min_{th}$ and $\max_{th}$ divide the buffer into three areas, as shown in Fig.4. The value of *avg* controls the behavior of the RED management. No packets are discarded if *avg* is smaller than the $\min_{th}$ threshold. RED acts if *avg* is between the lower $\min_{th}$ and upper $\max_{th}$ thresholds by dropping packets with a drop probability that is linearly proportional to the average queue size. These probabilistic drops are called early drops. They serve as an indication of an imminent congestion. An optimal operation of the RED mechanism should maintain the queue length average within the ($\min_{th}$, $\max_{th}$) area. RED functions as a droptail when the average queue length increases beyond $\max_{th}$.

### 3.3 CHOKe

CHOKe differentially penalizes unresponsive and unfriendly flows [10].

The CHOKe mechanism in the present simulation (modified CHOKe) is slightly different from the originally proposed CHOKe method [10]. Modified CHOKe does not use the previous state completely, and instead uses the current state. The instantaneous queue size is used to calculate the congestion level in modified CHOKe, instead of the average queue size, which is used in original CHOKe. Moreover, when the queue size exceeds the queue capacity, the incoming packet is used only for preferential drop and cannot enter the queue.

Therefore, the difference between modified CHOKe and droptail is only the use of a preferential packet drop mechanism when the queue size exceeds $\min_{th}$ and a packet drop mechanism when the queue size exceeds $\max_{th}$.

The queue of CHOKe is divided into three regions by two threshold values ($\min_{th}$, $\max_{th}$). If the queue size is less than $\min_{th}$, each arriving packet is queued into the FIFO buffer.

If the queue size is larger than $\min_{th}$, each arriving packet is compared with a randomly selected packet, called the drop candidate packet, from the FIFO buffer. If these packets have the same flow ID, they are both dropped (referred to herein as the preferential drop mechanism). Otherwise, the randomly chosen packet is kept in the buffer (in the same position as before), and the arriving packet is queued. In addition, the queued packet is dropped with the same probability as that of RED. If the average queue size is greater than $\max_{th}$, each arriving packet is compared with a randomly selected packet, called the drop candidate packet, from the FIFO buffer. If these packets have the same flow ID, they are both dropped. Otherwise, the randomly chosen packet is kept in the buffer (in the same position as before) and the arriving packet is dropped. This returns the queue occupancy to below $\max_{th}$.

We use the instantaneous queue length to detect congestion, instead of the average queue size, as in RED. Because the essence of CHOKe is a preferential drop mechanism and we want to validate the complete stateless queue method by not using the previous average queue size. In addition, incoming packets can be queued after sever check when the queue size exceeds $\max_{th}$. If the queue size becomes full, preferential drop is performed and all incoming packets are dropped.

Consider two type flows (large and small) that enter the same router. If the aggregated incoming rate is smaller than the output link capacity, the queue size does not increase to $\max_{th}$. If the aggregated incoming rate is grater than the output link capacity, the queue size increases. In addition, the size of each packet depends on each flow rate. In fact, in the queue, the number of packets belonging to a large flow is larger than the number of packets belonging to a small flow. Therefore, more packets of a large flow are dropped by the process of packet comparison. This mechanism is very simple, but must be realized using a preferential drop mechanism.

### 3.4 Pushback with ACC

The pushback method can cut off large flows early, which cause congestions when they are close to the source node[8]. The router that implements a pushback mechanism sends messages to neighbor routers when the router detects sever congestion and cannot be controlled locally by the rate-limit (bandwidth control) method. The messages include information about large flows that cause sever congestion. Neighbor routers try to rate-limit large flows until the router no longer receives rate-limit messages. When neighbor routers decide that those routers cannot solve congestion also, the routers send pushback messages to upstream routers again. As a result, the pushback method can rate-limit large flows near their source nodes.

For example, as shown in Fig.1, if router R0 detects congestion and maintains the situation for a constant time, router R0 asks neighbor routers to start rate-limiting flows that occupy a large amount of link capacity. The same is true for routers R2 and R3.

Aggregate congestion control is a method of assigning the same label to several flows that have the same properties for congestion control. For example, ACC considers flows that have the same destination address as one flow. If each flow size is small, normal pushback cannot regard the flow as a large flow. However, considering such flows as a single flow allows pushback to regard these flows as malicious flows. In this case, which label should we use for flow classification? This is a difficult problem. In the DDoS attack problem, we cannot use the source address as a label because attack flows usually have a spoofed source address. However, in the present simulation, we use source address ad flow ID expediently, because we do not use spoofing packets. The efficient flow ID is a matter for future research.

## 4. Network topologies used for simulations

The real Internet is considered to consist of several topologies, depending on the point of view. We thus take into account all properties needed to simulate DDoS Attacks. In this section, we discuss network topologies used to simulate DDoS Attacks.

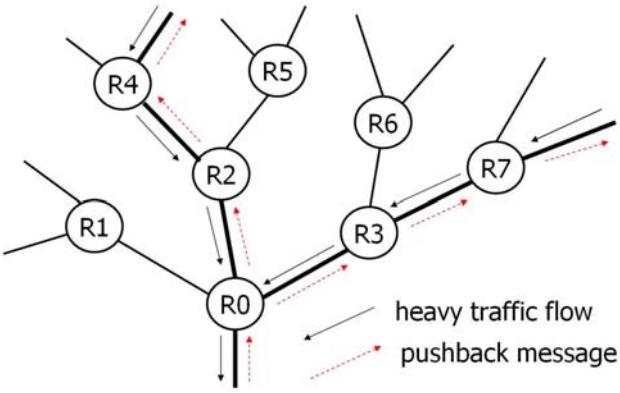


Fig.1 Diagram of pushback

### 4.1 Tiers model

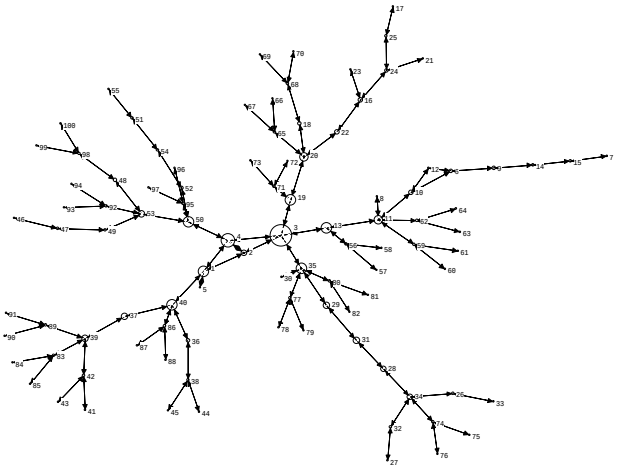

Fig.2 Tiers model

The Internet has a hierarchical structure, as shown in Fig.2. In this model, nodes are categorized into three types: edge nodes (LAN nodes), bridge, router or switch nodes (Metropolitan Area Network - MAN nodes) and gateway (WAN) nodes. Empirically, this idea is very natural [4]. For example, in the Science Information Network, which is the Internet Information Infrastructure for universities and research institutes in Japan, many universities connect to key university (MAN), which is connected to a backbone WAN. In addition, many university clients are connected to each other by a LAN.

In the simulation, the bandwidth of the link is 100 Mbps, 22 Mbps, and 10 Mbps (WAN, MAN, and LAN,

respectively), and the delay time of the link is a uniform random number (0 ms to 160 ms).

### 4.2 Transit-stub model

At present, the Internet can be viewed as a collection of interconnected routing domains, which are groups of nodes under a common administration that share routing information [5] [6]. A primary characteristic of these domains is routing locality, in which the path between any two nodes in a domain remains entirely within the domain. Thus, each routing domain in the Internet can be classified as either a stub or transit domain (see Fig.3).

A domain is a stub domain if the path connecting nodes u and v passes through that domain and if either u or v is located in that domain. Transit domains do not have this restriction. The purpose of transit domains is to interconnect stub domains efficiently. Without transit domains every pair of stub domains would need to be directly connected. Stub domains can be further classified as single- or multi-homed. Multi-homed stub domains have connections to more than one other domain. Single-homed stubs connect to only one transit domain. A transit domain is comprised of a set of backbone nodes, which are typically high-connected to each other.

In the simulation, the bandwidth of link is 100 Mbps and 10 Mbps for nodes of the transit domain and nodes of the stub domain, respectively, and the delay time of the link is a uniform random number (30 ms to 400 ms).

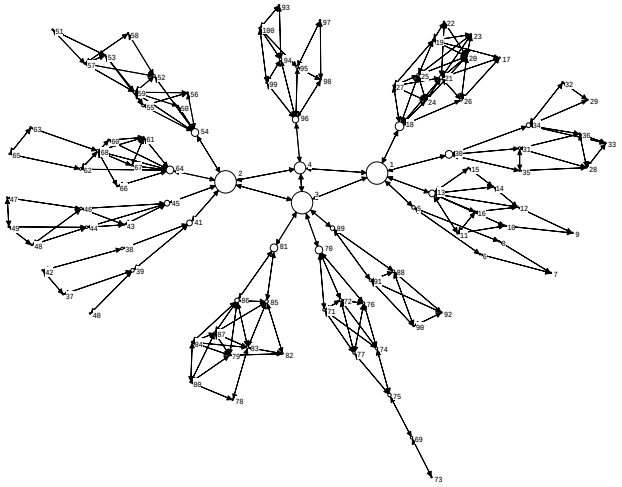

Fig.3 Transit-stub model

### 4.3 Barabasi-Albert model

The property of this model (see Fig.4) is that the degree distribution obeys the power law, which is observable in Internet AS level Topology [7].

The main features with respect to how to make Barabasi-Albert model are:

(1) Networks expand continuously by the addition of new nodes.
(2) New nodes preferentially attach to sites that are already well connected.

In the simulation, the bandwidth of the link is an exponential random number (from 10 Mbps to 100 Mbps), and the delay time of the link is a uniform random number (50 ms to 450 ms).

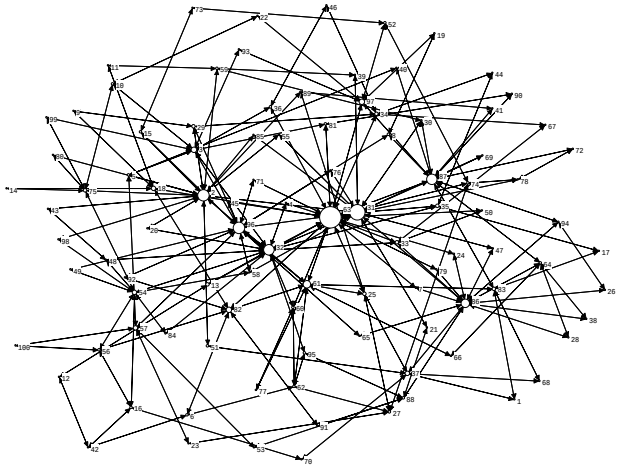

Fig.4 Barabasi-Albert model

## 5. Simulation Results

We consider the following two objects of DDoS Attacks. In the first simulation, the primary concern is to maintain the fairness between flows of equal bandwidth. There are eight UDP Constant Bit Rate flows (one good flow, seven bad flows) that have same rate. However, the router cannot distinguish between good and bad flows. Therefore, the router should deal with all types flows equally. Situations such as this usually happen in the Internet. Previously, IDSs necessarily performed misclassification, and the router was required to have the ability to maintain the fairness between flows of equal bandwidth, which appear to have equal responsibility with respect to congestion. We would like to test each congestion control method in this respect.

In the second simulation, the main concern is to protect small flows (referred as normal flows) from DDoS attack (referred as large flows). UDP flooding is one type of DDoS attack that causes the generation of large packets and congestion. Therefore, normal flow is restricted. From the viewpoint of max-min fairness, this situation is not good, and we want to validate the effect of each congestion control method with respect to this problem. All simulations were run using the NS simulator [15].

### 5.1 Simulation 1: Impose fairness of each now

Our simulation conditions of DDoS Attack (UDP flooding) are listed below.

(1) Each network (tiers, transit-stub, Barabasi-Albert) has 100 nodes.
(2) Consider eight flows (4 Mbps CBR), as shown in Fig.5.
Seven flows are Bad. (solid line)
One flow is Good. (dotted line)
(3) Source Nodes are selected from nodes of minimum degree.
(4) Destination Node is selected from nodes of minimum degree.
(5) The destination of all flows is the same.
(6) Simulation period is 10 seconds (unit time of ns2).
(7) Use droptail, RED, CHOKe, pushback method.
(8) Count packet amounts of each flow that reached the destination node.
(9) Other settings default to ns2.

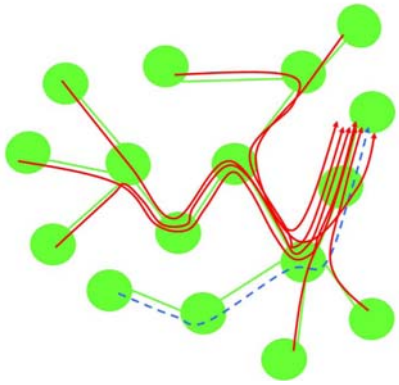

Fig.5 Simulation 1 (Main concerns: fairness)

The results are shown in Fig.6. In this simulation network, there are eight flows (seven bad flows, one good flow). The bandwidth of each flow is the same (4 Mbps CBR). Therefore, good methods maintain the difference between flows close to zero. From this point of view, CHOKe is the best method, because CHOKe can keep the standard deviation between flows in each topology small (see Fig.7).

In transit-stub topology, large packets of flows 4 and 5 can reach the destination node (see Fig.6) because the distance from the source node of flow 4 to the destination node is only 2 hops (links). Thus, flow 4 simply passes through only one node in which packet dropping occurs. In addition, the destination node has three links, and flow 5 does not use congested links to reach the destination. This is why many more packets of flows 4 and 5 can reach the destination, as compared to the packets of other flows. The reason for this situation is the architecture of the transit-stub network.

### 5.2 Simulation 2: Protect a small flow from big flows

The present simulation conditions of the DDoS Attack UDP flooding) are listed below.

(1) Each network (tiers, transit-stub, Barabasi-Albert) has 100 nodes.
(2) Consider four flows, as shown in Fig.8
6 Mbps CBR × 3: Bad (solid line)
1 Mbps CBR × 1: Good (dotted line)

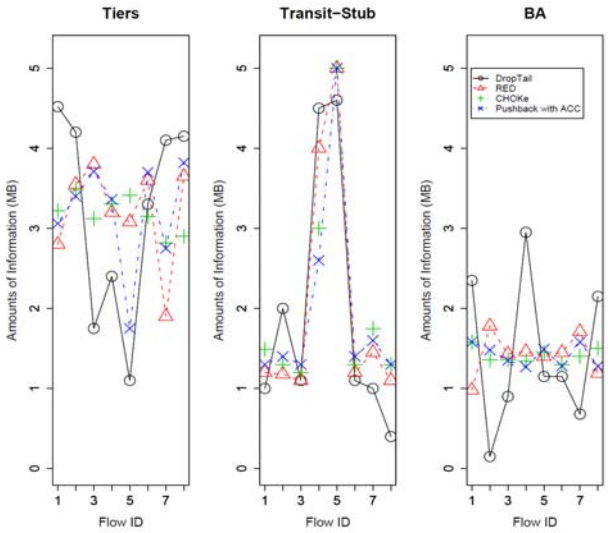


Fig.6 Information flows (Simulation 1)

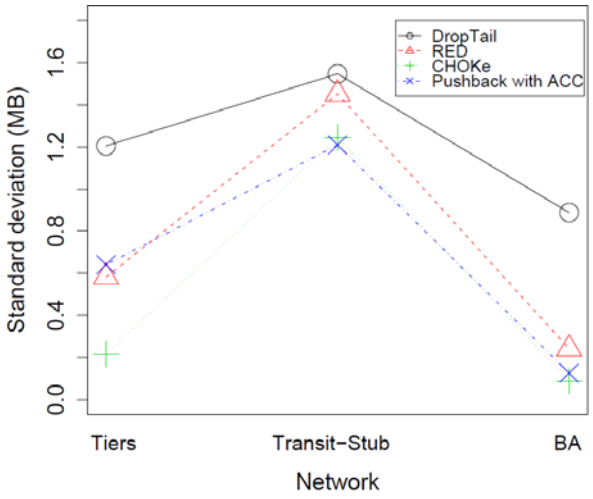


Fig.7 Standard deviation of flows on each network topology

(3) Source Nodes are selected from nodes of minimum degree.
(4) Destination Node is selected from nodes of minimum degree.
(5) The destination of all flows is same.
(6) Observe the utilization between flows at the incoming link of the destination node.
(7) Simulation period is 30 seconds (unit time of ns2).

(8) Use droptail, ACC, CHOKe, pushback method
(9) Other settings default to ns2.

The results are shown in Fig.9. And flow ID 1 represents a small flow. The capacity of the incoming link of the destination node is 10 Mbps, and the bandwidth of a good flow is just 1 Mbps. Then, the max link utilization of a small flow is 10 %. However, a small flow encounters sever congestion and is restricted by a large flow with

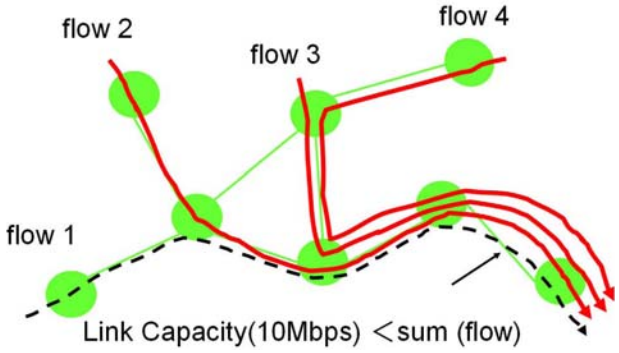


Fig.8 Simulation 2 (Main concern: protecting small flows)

droptail (link utilization of small flows is under 5% in all topologies).

The Random Early Discard (RED) method has a mechanism to drop packets randomly when the queue is not yet full. The selection probability of large flow packets in the queue is larger than in the case of small flow packets. RED seems to be able to protect small flows. However, if a packet of a small flow is selected randomly, the packet is dropped, and, unlike CHOKe, RED does not have a preferential packet drop mechanism. Therefore, RED improves small flow restriction situation slightly.

CHOKe mitigates DDoS Attacks. Small flows obtain maximum throughput in all networks. CHOKe is a stateless simple queue method. However, the results indicate that the preferential packet drop mechanism of CHOKe has a profound effect. Similarly, pushback with ACC succeeds in mitigating DDoS Attacks.

## 6. Conclusion

We investigated the effectiveness of the proposed congestion control methods for mitigating high bandwidth traffic aggregates by considering three network topologies. From the viewpoint of maintaining fairness, CHOKe is the most effective method for DDoS Attacks in three networks. From the viewpoint of protecting small flows, CHOKe and pushback with ACC can mitigate DDoS Attacks. We confirmed that a preferential drop mechanism is effective against DDoS Attack (UDP Flooding) in three different types of networks. From the viewpoint of implementation, CHOKe is easier to apply than ACC. As a result, we confirmed that bandwidth control is a good countermeasure against DDoS attacks (UDP flooding), and CHOKe is a better method than pushback with ACC.

A rate-limiting mechanism, such as CHOKe, is only one effective defense method and should be used as part of a defense architecture that includes IPSs, IDSs, firewalls, antivirus software, security-patches, and human education.

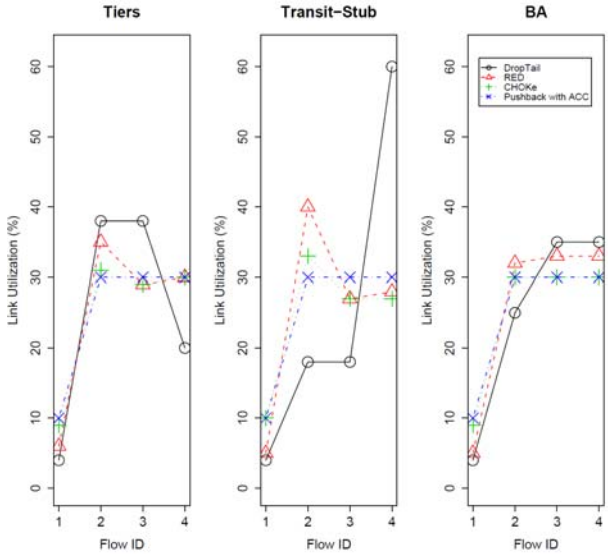


Fig.9 Information flows (Simulation 2)

The best method must be selected depending on the situation, similar to the case of an expert system in machine learning. Moreover, further research is needed to improve the rate-limiting mechanism and to validate the rate-limiting mechanism in other realistic situations. Since the results presented herein are only for UDP flow, testing using another type of flow (e.g., TCP or ICMP) and the mixture of several flow types is needed.

Scheduling algorithms can ensure the fairness between the traffic flows, but they are too expensive for broad deployment and don't scale well to a large number of users. Their technique performs tests that identifies flows that are unresponsive, TCP-friendly, or high-bandwidth and regulate them. These issues should be further investigated in the future works.

**References**

[1] J. Ioannidis and S. M. Bellovin, "Implementing pushback: router defense against DDoS attacks," presented at Network and Distributed System Security Symposium, 2002.

[2] C. L. Schuba, I. V. Krsul, M. G. Kuhn, E. H. Spafford, A. Sundaram, and D. Zamboni, "Analysis of a denial of service attack on TCP," presented at IEEE Symposium on Security and Privacy, 1997.

[3] P. Ferguson and D. Senie, "Network ingress filtering: defeating denial of service attacks which employ IP source address spoofing," IETF RFC2267 RFC 2267, January 1998.

[4] Matthew B. Doar. A better model for generating test networks. In Proceedings of Global Internet. IEEE, November 1996.

[5] Ellen W. Zegura,K. Calvert, and S.Bhattacharjee. How to model an Internet-work. In Proceedings of IEEE Infocom.IEEE,1996.

[6] Ellen W.Zegura, Kenneth L.Calvert, and Michae J. Donahoo. A quantitative comparison of graph-based models for Internet Topology. IEEE/ACM Transactions on Networking,5(6):770-783,December 1997

[7] Reka Albert and Albert-Laszlo Barabasi. Topology of evolving networks: locak events and universality. In Physical Review Letters, volume 85,pages 5234-5237, 2000.

[8] Controlling High Bandwidth Aggregates in the Network: RatulMahajan, StevenM.Bellovin, SallyFloyd, JohnIoannidis, VernPaxson, and ScottShenker

[9] Random Early Detection Gateways for Congestion Avoidance: Sally Floyd, Van Jacobson Lawrence Berkeley Laboratory University of California

[10] CHOKe A stateless active queue management scheme for approximating fair bandwidth allocation Rong Pan, Balaji Prabhakar, Konstantinos Psounis

[11] Cheng Jin, Qian Chen, and Sugih Jamin. Inet topology generator. Technical Report CSE-TR-433-00,EECS Department, University of Michigan,2000

[12] D. Schnackenberg, H. Holliday, R. Smith, K. Djahandari, and D. Sterne. (2001, June). " Cooperative Intrusion Traceback and Response Architecture (CITRA)", in Proceedings of the Second DARPA Information Survivability Conference and Exposition (DISCEX II), Anheim, California, USA.

[13] Jian Yuan, Kevin Mills. Monitoring the Macroscopic Effect of DDoS Flooding Attacks. Advanced Network Technologies Division National Institute of Standards and Technology February 4, 2004

[14] D. Sterne, K. Djahandari, B. Wilson, B. Babson, D. Schnackenberg, H. Holliday, and T.Reid. (2001, September 27). "Autonomic Response to Distributed Denial of Service Attacks", in Proceedings of Recent Advances in Intrusion Detection, 4th International Symposium, pp. 134-139. Davis, California, USA. [Online].

[15] " UCB/LBNL/VINT network simulator – ns (version 2)."available at \\http://www-mash.cs.berkeley.edu/ns/.

[16] Kihong Park, Heejo Lee: On the effectiveness of route-based packet filtering for distributed DoS attack prevention in power-law internets. SIGCOMM 2001: 15-26, 2001.

[17] C.E.R.T.(CERT).CERT Advisory CA-2000-01 Denial-of-service developments,Jan.2000. http://www.cert.org/advisories/CA-2000-01.html.

[18] Substance and measures of DdoS http://www.microsoft.com/japan/technet/security/\\secnews/columns/column060426.mspx

**Takanori Komatsu** Graduate student of Computer Science at National Defense Academy of Japan. He holds the degrees of Science in Electrical and Electronic Engineering at Tokyo University of Agriculture and Technology in Japan. His research interests include network security and Multi-agents.

**Akira Namatame** Professor of Dept. of Computer Science National Defense Academy of Japan. He holds the degrees of Engineering in Applied Physics from National Defense Academy, Master of Science in Operations Research and Ph.D. in Engineering-Economic System from Stanford University. His research interests include Multi-agents, Game Theory, Evolution and Learning, Complex Networks, He is the editor-in-chief of Journal of Economic Interaction Coordination (JEIC). He is a member of the AAAI, the IEEE, and the SICE.