Copyright @ 2011 KSII

Efficient Abnormal Traffic Detection Software Architecture for a Seamless Network

Dongcheul Lee and Byung Ho Rhee
Department of Electronics Computer Engineering, Hanyang University
Seoul, Korea
[e-mail: {jackdclee, bhrhee}@hanyang.ac.kr]
*Corresponding author: Dongcheul Lee

Received November 10, 2010; revised December 26, 2010; revised January 9, 2011; accepted February 5, 2011; published February 28, 2011

Abstract

To provide a seamless network to customers, Internet service providers must promptly detect and control abnormal traffic. One approach is to shorten the traffic information measurement cycle. However, performance degradation is inevitable if traffic measurement servers merely shorten the cycle and measure all traffic. This paper presents a software architecture that can measure traffic more frequently without degrading performance by estimating the level of abnormal traffic. The algorithm in the architecture estimates the values of the interface group objects in MIB by using the IP group objects thereby reducing the number of measurements and the size of measured data. We evaluated this architecture on part of Internet service provider's IP network. When the traffic was measured 5 times more than before, the CPU usage and TPS of the proposed scheme was 7% and 41% less than that of the original scheme while the false positive rate and false negative rate were 3.2% and 2.7% respectively.

Keywords: Traffic measurement, software architecture, abnormal traffic, monitoring, SNMP

DOI: 10.3837/tiis.2011.02.004

1. Introduction

As the Internet evolves, various types of traffic such as peer-to-peer (P2P), multi-media, and web have flown in a network. To deliver such traffic seamlessly, Internet service providers (ISP) have monitored entire network traffic to decide whether there is malicious traffic which can harm targeted servers and eventually normal traffic. When found, they have blocked malicious traffic from being transferred in. To block malicious traffic before damaging targeted servers, operators should detect abnormal traffic fast. Despite the importance of quick detection, many high-level monitoring methods such as deep packet inspection are still not being used by ISPs who still use low-level monitoring methods such as simple network management protocol (SNMP). It is because high-level methods examine raw packet data, which results in significant processing burden, and eventually late detection time to cope with high-speed network traffic. In this environment, one of the methods to detect abnormal traffic fast is shortening the traffic measurement cycle. However, because of the high network equipment requirements, the performance of traffic measurement servers deteriorates and the size of the measured data increases exponentially if the cycle is merely shortened.

There are two basic approaches to detect abnormal traffic. First approach is based on pattern matching to identify abnormal traffic. These known patterns are extracted from the known abnormal traffic. Even though it can detect anomaly with high accuracy for learned anomaly, it has low flexibility since the patterns should be updated manually. Second approach is based on a normal usage behavior profile. Analysis system examines the deviation of the current behavior profile from the normal profile. Even though it can detect new type of anomaly effectively, it can issue false alarms.

In order to use the advantages of both approaches by overcoming their limitations, many types of schemes have been proposed to monitor network traffic and to detect abnormal traffic. First one is a threshold-based abnormal traffic detection scheme [1][2]. In this scheme, operators set an upper and lower constant threshold for each equipment's interface, and if the specific value of the traffic measured from the interface is larger or smaller than an upper or lower threshold, the traffic is considered to be an anomaly. Second one is a profile-based scheme [3][4]. It defines a standard traffic model by using previous traffic information. Then if the measured value deviates from the model to a predefined extent, the traffic is considered as an anomaly. Thirdly, the subspace-based scheme [5][6] filters abnormal traffic by using origin-destination flows (OD flows) thereby avoiding measuring the entire traffic from the interfaces in a network. Finally, a wavelet-based scheme [7][8] represents a traffic signal to multiple time scale and classify them into frequency band. The traffic is deemed an anomaly if the duration for each frequency band does not show similar patterns for its network scale.

Also, there are some studies that use SNMP management information base (MIB) [9] data to detect abnormal traffic. Cabrera et al. [10] used proactive detection scheme using MIB for symptom analysis of abnormal traffic. Li and Manikopoulos [11] used statistical anomaly detection scheme using MIB traffic parameters. They converted MIB data into a probability density function to calculate statistical similarity. Puttini et al. [12] presented the associated Bayesian classification to MIB variables to detect abnormal behavior in Mobile Ad Hoc Networks. Ramah et al. [13] proposed principle component analysis based anomaly detection scheme.

However, these schemes do not resolve how abnormal traffic detection systems can shorten the traffic measurement cycle for fast detection despite it being a critical issue for most ISP's operators. [14] suggested an algorithm that can shorten the cycle, but it had poor false positive and false negative rate. We improved the algorithm in [14] to reduce the error rate and presents new efficient abnormal traffic detection software architecture that shortens the traffic measurement cycle without performance degradation by estimating the values of the interface group MIB objects. This estimate is performed by correlating the interface group MIB objects and the IP group MIB objects. Therefore, excessive performance degradation does not occur to the measurement servers. Furthermore, we present another algorithm that filters improper equipment which cannot be used in the proposed software architecture. This equipment cannot provide accurate data on their MIB, which prevents accurate estimation of the interface group objects. The experimental environment included more diverse equipments than the environment in [14], and more experimental results were added for performance evaluation.

The next section defines an abnormal traffic detection scheme that shortens the traffic measurement cycle without performance degradation. In Section 3, we demonstrate our proposed scheme showing good performance on experiments. The paper concludes with a short discussion.

2. Efficient Abnormal Traffic Detection Software Architecture

We made an abnormal traffic detection software architecture, which enables shortening the traffic measurement cycle without excessive performance degradation of the traffic measurement servers. The scheme uses both the IP group objects and the interface group objects of SNMP MIB when it measures traffic from equipment. **Table 1** shows the objects that are used in this architecture. These objects belong to layer 2 and 3 in the OSI 7 layer model [15]. Normally, ISPs only measure volume traffic of layer 2 or 3. Individual sessions are not always measured above layer 3 because they need to use special protocols like netflow [16], which lead to performance degradation of network equipment. To prevent degradation, layer 2 or 3 volume traffic is monitored all the time on all equipment, but individual session traffic is only measured on alarmed equipment when alarms are issued. In this paper, we only focus on measuring layer 2 and 3 volume traffic which is the monitoring method of most ISPs. Also, when the server measures traffic from equipment, the proposed architecture measures the IP group objects and the interface group objects alternately, whereas conventional network monitoring systems (NMSes) [17] measure the interface group objects as many times as there are interfaces. Therefore, the measurement servers' loads will not higher than other NMSes.

The functions in the proposed software architecture are divided into two steps. In the first step, it filters improper equipment which degrades the accuracy of the scheme. In the second step, it measures the values of the IP group MIB objects from equipment which were not filtered in the first step and estimates the values of the interface group objects to detect abnormal traffic.

Table 1. MIB objects used in the software architecture.

Group	Direction	MIB Objects
IP	in	*ipInReceives*
	out	*ipOutRequest, ipForwDatagrams, ipOutDiscards, ipOutNoRoutes, ipFragOKs, ipFragFails, ipFragCreates*
interface	in	*ifInUcastPkts, ifInNUcastPkts*
	out	*ifOutUcastPkts, ifOutNUcastPkts*

2.1 Filtering Improper Equipment

To describe the operation of the first step, the following terminologies are used. Let τ_j denote the time instance when a MIB object was accessed and $\tau_j < \tau_{j+1}$. At time τ_0, a traffic measurement server first tries to access the network equipment $E(n)$, where n is the total number of interfaces of the equipment. And $inIF_i(\tau_j)$ is defined as the sum of the inbound objects of ith interface group at time τ_j and can be calculated by

$$inIF_i(\tau_j) = ifInUcastPkts_i(\tau_j) + ifInNUcastPkts_i(\tau_j) \tag{1}$$

where $ifInUcastPkts_i(\tau_j)$ and $ifInNUcastPkts_i(\tau_j)$ are the values of the *ifInUcastPkts* object and the *ifInNUcastPkts* object on ith interface of $E(n)$ at time τ_j.

And $outIF_i(\tau_j)$ is defined as the sum of the outbound objects of ith interface group at time τ_j and can be computed as

$$outIF_i(\tau_j) = ifOutUcastPkts_i(\tau_j) + ifOutNUcastPkts_i(\tau_j) \tag{2}$$

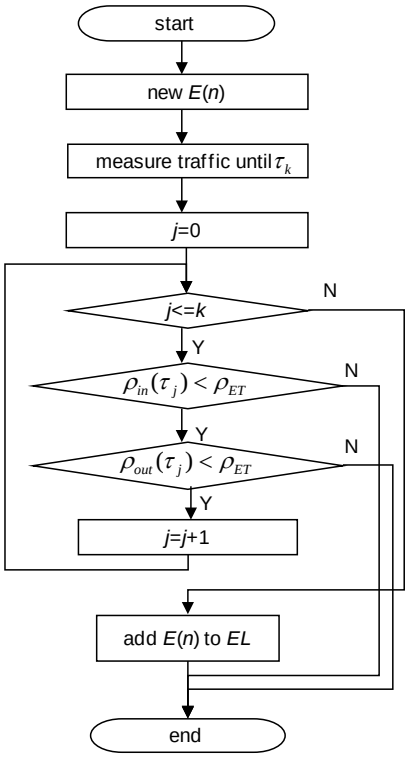


Fig. 1. The flow of the first step.

where $ifOutUcastPkts_i(\tau_j)$ and $ifOutNUcastPkts_i(\tau_j)$ are the values of the *ifOutUcastPkts* object and the *ifOutNUcastPkts* object on ith interface of $E(n)$ at time τ_j.

Also, $inIP(\tau_j)$ is defined as IP group input traffic at time τ_j and can be found as

$$inIP(\tau_j) = ipInReceives(\tau_j) \tag{3}$$

where $ipInReceives(\tau_j)$ is the value of the *ipInReceives* object on $E(n)$ at time τ_j.

Also, $outIP(\tau_j)$ is defined as IP group output traffic at time τ_j and can be computed as

$$\begin{aligned} outIP(\tau_j) &= ipOutRequest(\tau_j) + ipForwDatagrams(\tau_j) - ipOutDiscards(\tau_j) \\ &- ipOutNoRoutes(\tau_j) - ipFragOKs(\tau_j) - ipFragFails(\tau_j) + ipFragCreates(\tau_j) \end{aligned} \tag{4}$$

where $ipOutRequest(\tau_j)$, $ipForwDatagrams(\tau_j)$, $ipOutDiscards(\tau_j)$, $ipOutNoRoutes(\tau_j)$, $ipFragOKs(\tau_j)$, $ipFragFails(\tau_j)$, and $ipFragCreates(\tau_j)$ are the values of the IP group MIB objects, which are defined at **Table 1**, on $E(n)$ at time τ_j.

Fig. 1 shows the algorithm of the first step. It describes how this scheme can filter improper equipment which cannot be used in the second step of the architecture. First, when a new network equipment $E(n)$ is installed, the algorithm measures both the interface group objects and the IP group objects for predefined period τ_k, where $k>0$. After that, it checks whether the difference ratio between the IP group object and the sum of the interface group objects is greater than the predefined tolerance ratio ρ_{ET}. The difference ratio for input traffic $\rho_{in}(\tau_j)$ can be given by

$$\rho_{in}(\tau_j) = \frac{\left| inIP(\tau_j) - \sum_{i=1}^{n} inIF_i(\tau_j) \right|}{\sum_{i=1}^{n} inIF_i(\tau_j)} \times 100. \tag{5}$$

Likewise, the difference ratio for output traffic $\rho_{out}(\tau_j)$ is given by

$$\rho_{out}(\tau_j) = \frac{\left| outIP(\tau_j) - \sum_{i=1}^{n} outIF_i(\tau_j) \right|}{\sum_{i=1}^{n} outIF_i(\tau_j)} \times 100. \tag{6}$$

Finally, if there are no objects whose difference ratios are greater than ρ_{ET}, it adds $E(n)$ to *EL* which is defined as a set of network equipment that can be used at the second step. However, if there are, $E(n)$ is not added to *EL*.

2.2 Detecting Abnormal Traffic

The algorithm measures both the IP group objects and the interface group objects alternately as shown in **Fig. 2**. It measures the interface group objects every τ_{5k}, and measures the IP group objects every τ_{5k+1}, τ_{5k+2}, τ_{5k+3}, τ_{5k+4}, where k≥0.

Also, let $inPPS_i(\tau_j)$ and $outPPS_i(\tau_j)$ denote input and output number of packets per second for *i*th interface at τ_j respectively, which can be obtained from the interface group object. These variables can be found as

$$inPPS_i(\tau_j) = inIF_i(\tau_j) / \lambda_{CC} \tag{7}$$

$$outPPS_i(\tau_j) = outIF_i(\tau_j) / \lambda_{CC} \tag{8}$$

where λ_{CC} is the traffic measurement cycle.

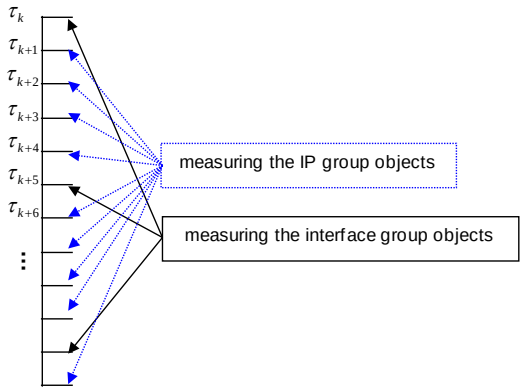


Fig. 2. Traffic measurement cycle.

Since the cycle is always uniform, it can be computed as

$$\lambda_{CC} = \tau_j - \tau_{j-1} = \tau_1 - \tau_0 . \tag{9}$$

Then $\mu_i^{in}(\tau_j)$ denotes the mean of previous weeks' input traffic from *i*th interface at τ_j and can be calculated as

$$\mu_i^{in}(\tau_j) = \frac{\sum_{k=1}^{w} inPPS_i(\tau_{j-k\cdot d\cdot h\cdot m})}{w} \tag{10}$$

where *w* is the number of weeks in which the logs of traffic stored, and *d* means 7 days, *h* means 24 hours, and *m* means 60 minutes.

Likewise, $\mu_i^{out}(\tau_j)$ denotes the mean of previous weeks' output traffic from *i*th interface at τ_j and can be calculated as

$$\mu_i^{out}(\tau_j) = \frac{\sum_{k=1}^{w} outPPS_i(\tau_{j-k\cdot d\cdot h\cdot m})}{w} . \tag{11}$$

Thresholds are used for detecting abnormal traffic. For example, if the value of the interface group object is greater than the threshold, the architecture raises an alarm. The threshold can be set by two ways: a constant threshold and a cumulative threshold. The constant threshold for each interface can be set manually by the operators. However, the cumulative threshold can be calculated automatically by using the mean of previous weeks' traffic.

Since we believe the traffic at τ_j will be similar with the traffic at τ_{j-1}, τ_{j-2}, τ_{j-3}, τ_{j-4}, a threshold is computed every 5 time instance. Let $inthr_i(\tau_j)$ and $outthr_i(\tau_j)$ denote the input and output traffic threshold of the interface group object on *i*th interface at τ_j. Therefore, $inthr_i(\tau_j)$ can be calculated by

$$\begin{aligned} inthr_i(\tau_j) &= \mu_i^{in}(\tau_j)\cdot(1+\rho_i^{IT}) \\ &\approx inthr_i(\tau_{j-1}) \\ &\approx inthr_i(\tau_{j-2}) \\ &\approx inthr_i(\tau_{j-3}) \\ &\approx inthr_i(\tau_{j-4}) \end{aligned} \tag{12}$$

where ρ_i^{IT} is the user defined tolerance ratio for *i*th interface.

Similarly, $outthr_i(\tau_j)$ can be calculated by

$$\begin{aligned} outthr_i(\tau_j) &= \mu_i^{out}(\tau_j)\cdot(1+\rho_i^{IT}) \\ &\approx outthr_i(\tau_{j-1}) \\ &\approx outthr_i(\tau_{j-2}) \\ &\approx outthr_i(\tau_{j-3}) \\ &\approx outthr_i(\tau_{j-4}) \end{aligned} \quad . \tag{13}$$

Also, $insthr(\tau_j)$ and $outsthr(\tau_j)$ denote the input and output traffic threshold of the IP group object and can be calculated by summing all $inthr_i(\tau_j)$ and $outthr_i(\tau_j)$ of $E(n) \in EL$ respectively. Therefore, $insthr(\tau_j)$ can be computed as

$$\begin{aligned} insthr(\tau_j) &= \sum_{i=1}^{n} inthr_i(\tau_j) \\ &\approx insthr(\tau_{j-1}) \\ &\approx insthr(\tau_{j-2}) \\ &\approx insthr(\tau_{j-3}) \\ &\approx insthr(\tau_{j-4}) \end{aligned} \tag{14}$$

and $outsthr(\tau_j)$ can be calculated as

$$\begin{aligned} outsthr(\tau_j) &= \sum_{i=1}^{n} outthr_i(\tau_j) \\ &\approx outsthr(\tau_{j-1}) \\ &\approx outsthr(\tau_{j-2}) \\ &\approx outsthr(\tau_{j-3}) \\ &\approx outsthr(\tau_{j-4}) \end{aligned} \quad . \tag{15}$$

Let $inSPPS(\tau_j)$ and $outSPPS(\tau_j)$ denote the input and output sum of packets per second for equipment at time τ_j respectively, which can be obtained from the IP group object. These variables can be found as

$$inSPPS(\tau_j) = inIP(\tau_j)/\lambda_{CC} \tag{16}$$

$$outSPPS(\tau_j) = outIP(\tau_j)/\lambda_{CC} \tag{17}$$

Fig. 3 shows the flow of the second step for input traffic of $E(n) \in EL$ at τ_j. It describes how this algorithm can estimate the values of the interface group objects by using the IP group objects and how abnormal traffic can be detected. Output traffic is also processed similarly. When the algorithm starts, it determines whether it will measure the IP group objects or the interface group objects. That is, if *j* mod 5 is 0, it measures the interface group objects. Otherwise, it measures the IP group objects and compares $inSPPS(\tau_j)$ with $insthr(\tau_j)$. If $inSPPS(\tau_j)$ is greater than $insthr(\tau_j)$, it measures all inbound objects of the interface group and determines whether $inPPS_i(\tau_j)$ are above $inthr_i(\tau_j)$. If they are, the algorithm raises an alarm and measures the interface group objects from the *i*th interface every τ_{j+k}, where k≥1,

until the value becomes lower than the threshold. However, if there were no values above the thresholds, the algorithm does nothing and restarts at τ_{j+1}.

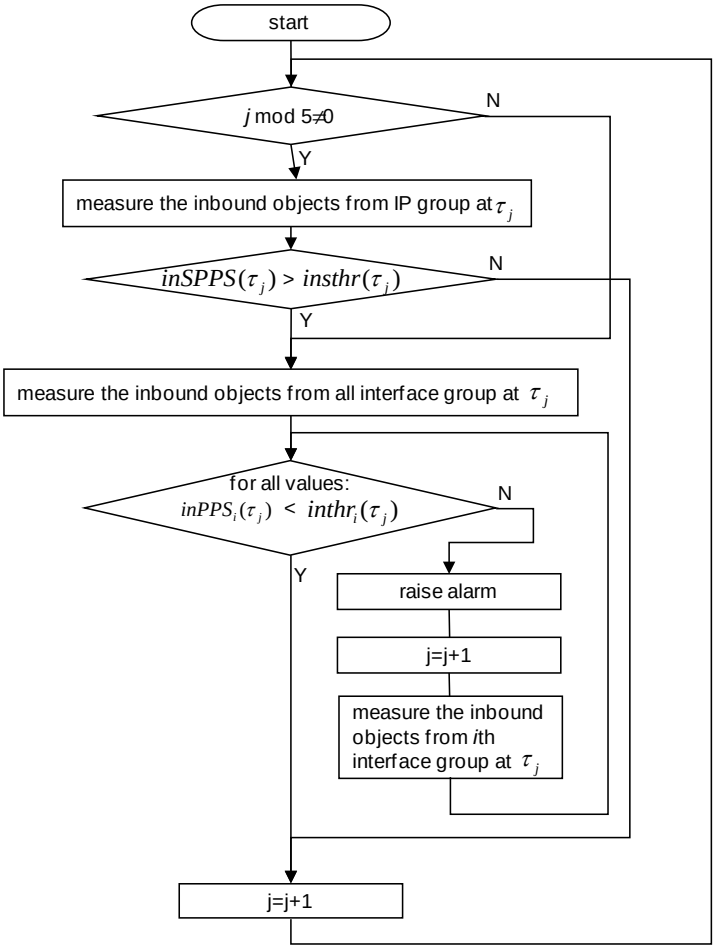


Fig. 3. The flow of the second step.

3. Experimental Results

Experiments were done to evaluate the proposed scheme on 176 IP network [18] equipment including various routers [19] and switches [20] on part of ISP's network. The network topology and the system architecture are described in **Fig. 4**. The network had hierarchical topology which consisted of center node core routers, primary node core routers, sub node core router, and aggregation switches. The traffic monitoring system consisted of application servers, database management servers (DBMS), and SNMP gateway servers. All equipment on the core network had SNMP agents. The SNMP gateway servers queried the agents for retrieving MIB data. The application servers received the data from the SNMP gateway servers and decided whether the traffic was abnormal or not. The DBMS servers stored traffic information and calculated cumulative thresholds periodically. We used cumulative thresholds for each interface and the thresholds were calculated by using MIB data from the previous 4 weeks, that is, w=4. Also, the traffic was measured every minute, λ_{CC}=1 minute.

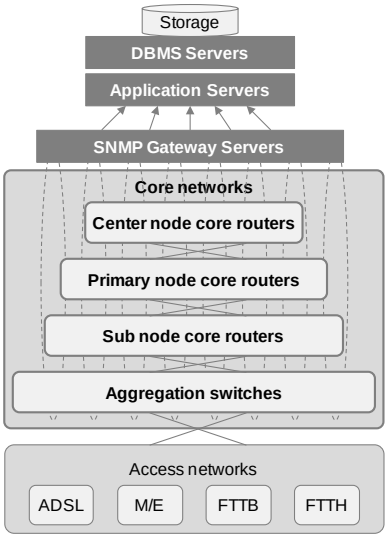


Fig. 4. System architecture and the topology of the experimental environment.

Table 2 shows all parameters and their values used in the experiments. Two experiments were performed. In the first experiment, we showed whether the first algorithm had filtered improper equipment well. Secondly, we verified whether the second algorithm had been able to predict alarms correctly by using the IP group objects.

Table 2. Parameters and their values used in the experiment.

Parameters	Values
ρ_i^{IT}	5.9%~6.1%
w	4 weeks
λ_{CC}	1 min
τ_k	1440 min
ρ_{ET}	0.1%

3.1 Experiments on Filtering Improper Equipment

In this experiment, we calculated $inIP(\tau_j)$ and $\sum_{i=1}^{n} inIF_i(\tau_j)$ and checked whether the gap between them was large or not. **Fig. 5** and **6** show the difference between the values of $inIP(\tau_j)$ and $\sum_{i=1}^{n} inIF_i(\tau_j)$ as a high-low-close graph [21]. As the height between the two coordinates on same x axis becomes taller, the difference becomes larger. The exact values of y axis were kept anonymous for reasons of security. In **Fig. 5**, the maximum difference rate of the equipment was 0.01%, which means we were able to use the equipment for the proposed architecture. However, in **Fig. 6**, the maximum difference rate was 15.2%, which means the equipment was not suitable for the estimation in the second algorithm. Since we set the tolerance ratio ρ_{ET} as 0.1%, the equipment could not be added to *EL* and were excluded from the second experiment.

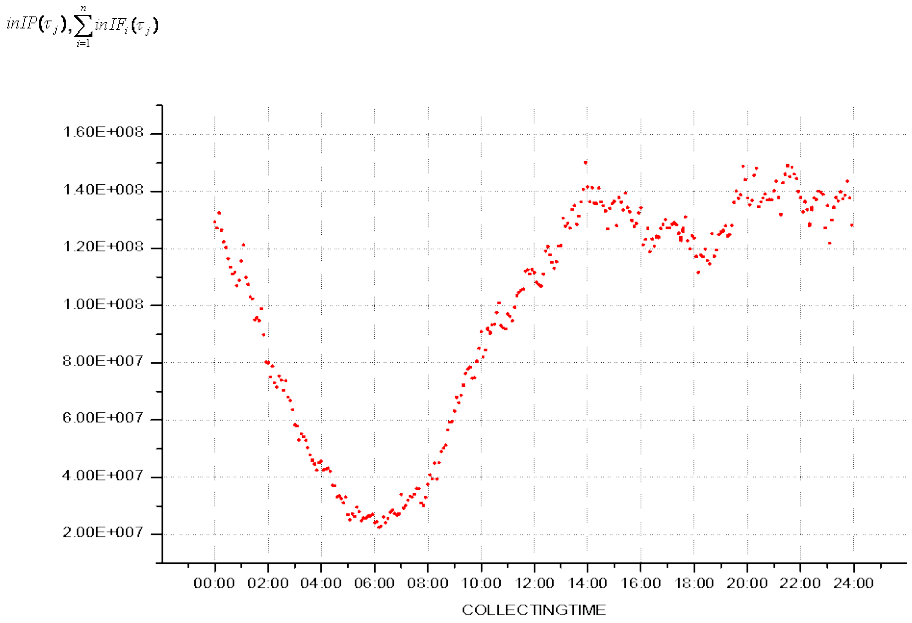


Fig. 5. Little difference between the IP group and the interface group object.

Fig. 7 shows the values of $inIP(\tau_j)$ and $\sum_{i=1}^{n} inIF_i(\tau_j)$, whose equipment had improper values of the IP group objects. The values of $inIP(\tau_j)$ on right y axis were much smaller than the values of $\sum_{i=1}^{n} inIF_i(\tau_j)$ on left y axis. Also, the tendency of the graphs between $inIP(\tau_j)$ and $\sum_{i=1}^{n} inIF_i(\tau_j)$ was quite different. This means that we cannot use $inIP(\tau_j)$ to estimate the value of $\sum_{i=1}^{n} inIF_i(\tau_j)$. Likewise, the values of $outIP(\tau_j)$ were very small compared with the values of $\sum_{i=1}^{n} outIF_i(\tau_j)$. This is mainly because some MIB objects of the equipment kept the incorrect values. Especially, in our experiments, $ipInReceives(\tau_j)$ had very small values and $ipForwDatagrams(\tau_j)$ was zero on the equipment.

Table 3 shows the equipment which was used in the first experiment. The equipment included a router series and a switch series of Cisco Systems [22], Juniper Networks [23], Hitachi [24], Extreme Networks [25], Redback Networks [26], Dasan Networks [27], and Laurel [28]. The admitted ratio means the ratio between the number of equipment in *EL* and the total number of equipment in the network. The equipment that showed 0% admitted ratio had very small or zero values for some IP group objects, which resulted in the improper values of $inIP(\tau_j)$ and $outIP(\tau_j)$. On this experiment, 6 equipments were not added to *EL*. The exact model names of equipment were kept anonymous for reasons of security. This table indicates that we can classify predictable and non-predictable equipment definitely according to their model names.

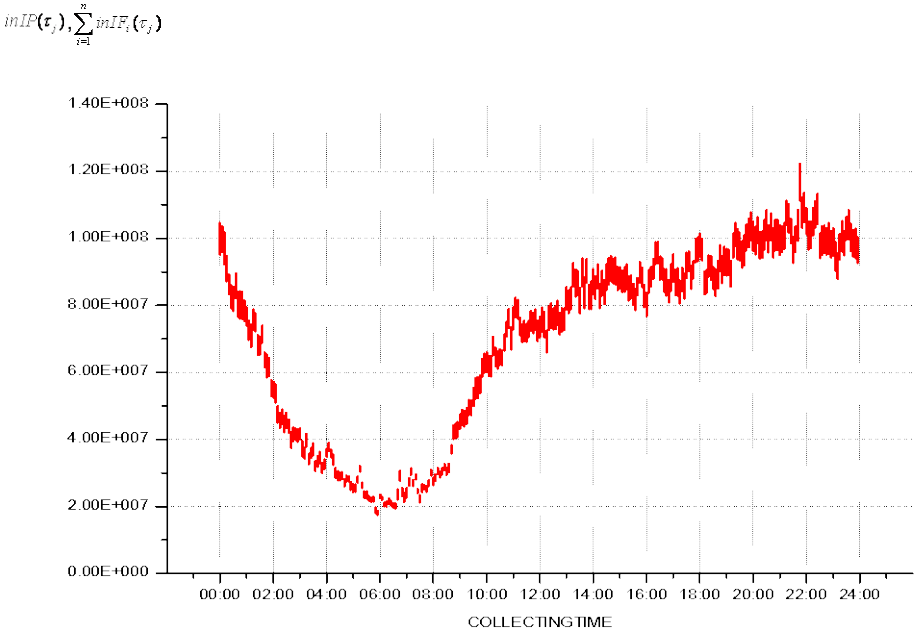


Fig. 6. Large difference between the IP group and the interface group object.

Table 3. Admitted ratios by equipment's model names.

Model name	Admitted ratio(%)	Model name	Admitted ratio(%)
Cisco A router	100	Extreme A switch	80
Cisco B router	90	Hitachi A router	100
Cisco C router	100	Junifer A router	0
Cisco D router	100	Laurel A router	100
Cisco E router	75	Redback A router	90
Dasan A switch	0		

3.2 Experiments on Estimating the Interface Group Objects

To evaluate the accuracy of the second algorithm on estimating the values of the interface group objects by using the IP group objects, we compared the result of this algorithm to that of the original interface-group-based method. The original method only uses the interface group objects when it detects abnormal traffic. **Fig. 8** shows the difference ratio between $insthr(\tau_j)$ and $inSPPS(\tau_j)$ when the original method raised an alarm under ρ_i^{IT}=0 condition. The ratio can be computed by

$$\{inSPPS(\tau_j) - insthr(\tau_j)\} / insthr(\tau_j) \times 100 . \quad (18)$$

Positive values mean that the measured values are bigger than the threshold and negative values mean the opposite case. Therefore, positive values indicate this algorithm also raised an alarm when the original method did, which means the alarm is true and the algorithm predicted correctly. However, negative values denote that this algorithm did not raise an alarm even though the original method issued an alarm, which is false negative [29].

Also, to measure the false positive rate [30], we examined whether the original method raised an alarm when our algorithm issued an alarm. As depicted in **Fig. 9**, both error ratios increased as ρ_i^{IT} was increased, which reduced the number of alarms. On the contrary, error ratios decreased as ρ_i^{IT} was decreased, which increased the number of alarms. This means that if ρ_i^{IT} was decreased to reduce the error ratio, network operators could suffer from overfull alarms.

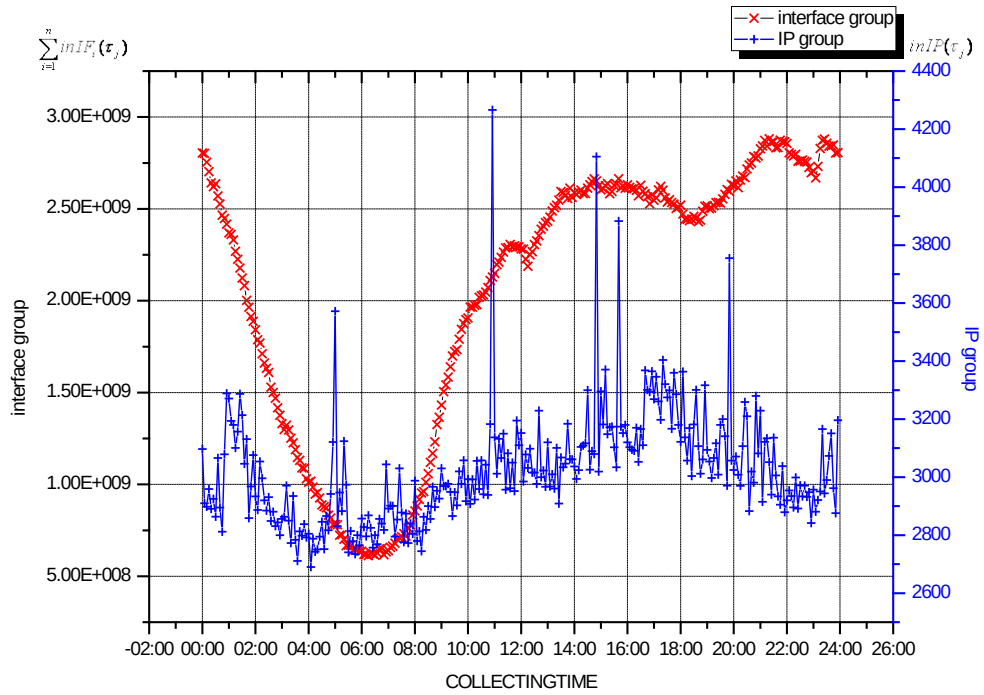


Fig. 7. Improper values of the IP group object.

Before the proposed scheme was adopted, the original method had issued approximately 10~20 alarms in a day, which was about 10% of the number of equipment in the experiment environment. In order to adjust the number of alarms between the original scheme and proposed scheme, ρ_i^{IT} was set to 5.9%~6.1%. Then false positive rate and false negative rate was 3.2% and 2,7% respectively. However, as depicted in **Fig. 9**, when we increased ρ_i^{IT} more than 6%, the error rates became too high. When we decreased it less than 6%, the system issued too many alarms that network operators could not take prompt actions.

When the filtering in the first step had not been performed, the false positive rate and the number of alarms increased by approximately 4.5% and 1% respectively. When ρ_i^{IT} was set to 5.9%~6.1%, the false positive rate was 7.8%, which resulted more measurements of interface group objects. This indicates that the first step is important to reduce the false positive rate of the proposed scheme.

Fig. 10 shows time-series CPU usage distributions on the DBMS servers and the application servers for 1 hour. The baseline graph in **Fig. 10** shows the CPU usages of the servers when the traffic data was retrieved every 5 minutes using original method. The average of the DBMS servers and the application servers was around 27% and 8% respectively. After we adapted the architecture, the servers measured traffic 5 times more frequently than before.

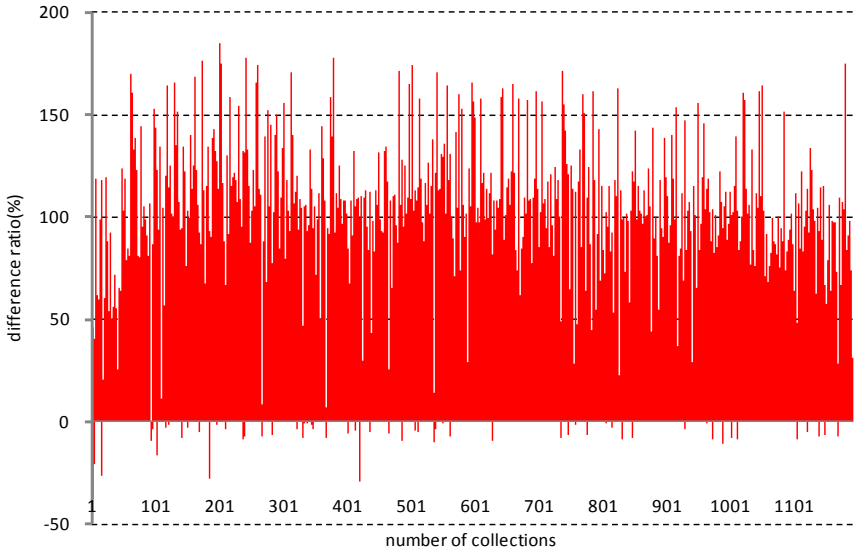
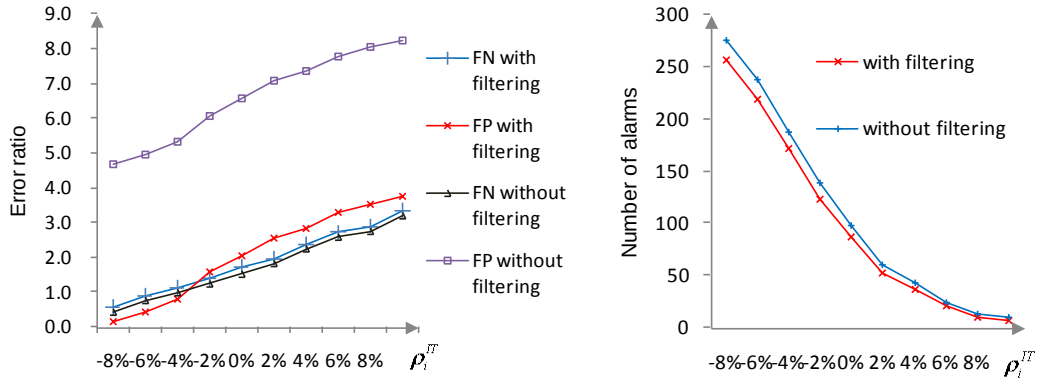


Fig. 8. Difference ratio between $inSPPS(\tau_j)$ and $insthr(\tau_j)$.



(a) Error ratio distribution **(b)** The number of alarms distribution

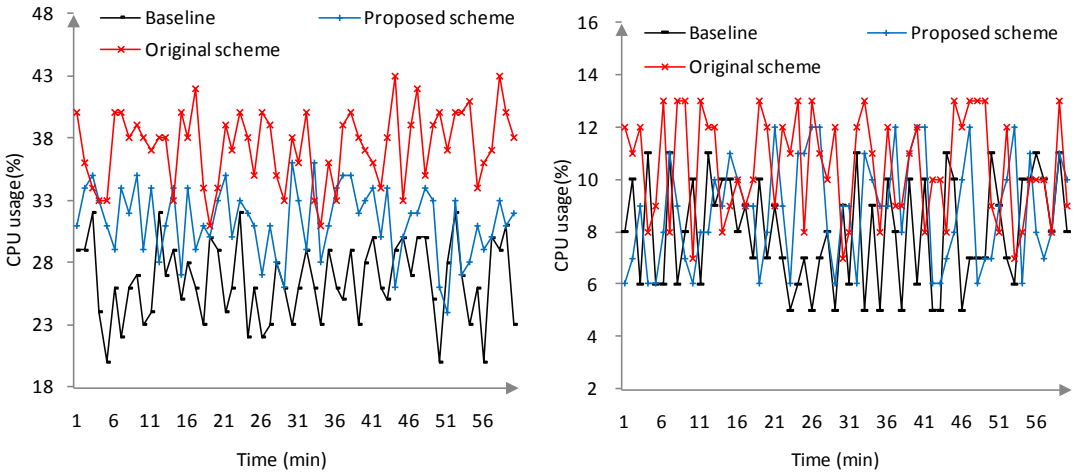
Fig. 9. Effect of tolerance ratio on error ratio and the number of alarms.

Then the average CPU usages of the DBMS servers and the application servers slightly increased by 4% and 1% respectively. Also, the average memory usage of the application servers increased by 1%. However, when we used the original method to measure traffic 5 times more frequently, the average CPU usages of the DBMS servers and the application servers increased by 11% and 2% respectively. The average memory usage of the application servers also increased by 1%. This indicates that the proposed scheme has little processing burden compared with the original scheme.

The mean number of interfaces for 176 equipments was 18. If we use the original method, the server should measure 3168 times per measurement cycle on average. However, if we use our proposed scheme, the server should measure 176 times per measurement cycle on average. **Fig. 11-(a)** shows the total number of measurements in 5 minutes when the traffic was measured every 1 minutes for 1 day. When we used original scheme, the number of measurements was 15840 and has uniform distribution. However, when we used the proposed

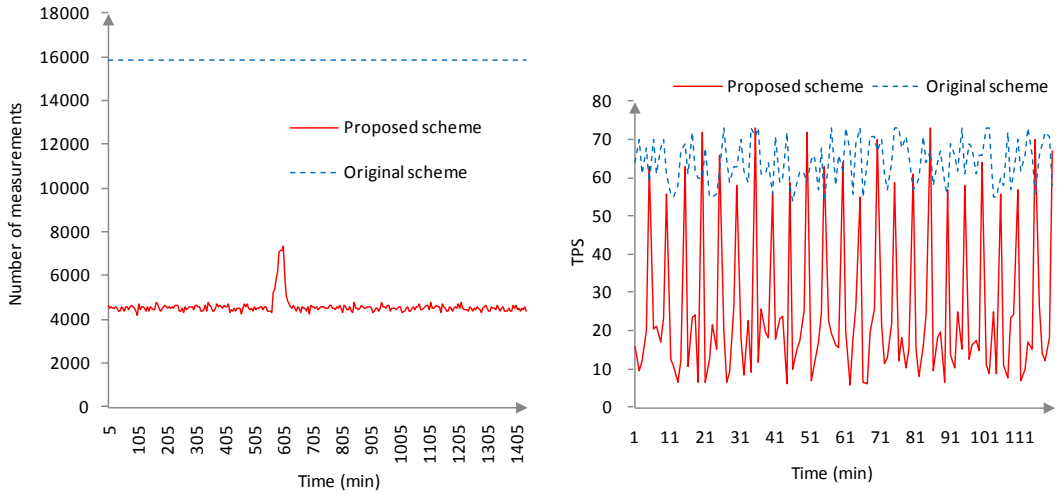
scheme, the number of measurements was reduced by approximately 65% compared with the original scheme. When there was abnormal traffic for 50 minutes, the number of measurements increased by 50%. However, even under these conditions, the number was still 54% less than the number required in the original scheme.

Fig. 11-(b) shows average transactions per second (TPS) of the application server measured for 2 hours. The overall TPS of the original scheme was 54~73, while most TPS of the proposed scheme was 5~35 and some TPS was 55~72. The average TPS of the proposed scheme was 59% less than that of the original scheme. The figure indicates that the proposed scheme requires less CPU usage, less measurements, and less TPS than the original scheme, which are critical for capacity planning of the measurement system.



(a) DBMS server **(b)** Application server

Fig. 10. Comparison of CPU usage distributions between the schemes.



(a) Number of measurement distribution **(b)** TPS distribution

Fig. 11. Performance comparisons between the original scheme and the proposed scheme.

4. Conclusions

We proposed efficient abnormal traffic detection software architecture for monitoring volume traffic of a large network by estimating the values of the MIB objects. This scheme enables shortening the traffic measurement cycle without degrading the traffic measurement servers' performance. Therefore, network operators can detect abnormal traffic more quickly before targeted servers having severe damage. For evaluation, we adopted the scheme to the ISP's network. The experimental results showed that the scheme can be used on most IP network equipment except for a few routers and switches. The accuracy of this scheme was good and could be improved more if we tuned the threshold more sensitively. Also, the scheme has little processing burden compared with the original scheme, which indicates the system will scale well to large network.

Since we evaluated our architecture on a real IP network, not on a simulation environment built from NS-2 [31] or OPNET [32], the results of our experiments give network operators confidence to adopt our architecture to their network monitoring system. As future work, we will find a way to reduce the false positive and the false negative rate of the algorithm. Also, we will investigate other MIB objects, such as TCP-MIB or UDP-MIB, for the better estimation.

References

[1] J. D. Brutlag, "Aberrant behavior detection in time series for network monitoring," in *Proc. of the 14th USENIX Conf. on System Administration*, 2000. Article (CrossRef Link)

[2] C.T. Paximadis and A.V. Vasilakos, "A two-level threshold-based traffic control scheme for ATM networks," in *Proc. of 16th Conf. on Local Computer Networks*, 1991. Article (CrossRef Link).

[3] P. Chan, M. Mahoney and M. Arshad, "Learning rules and clusters for anomaly detection in network traffic," *Managing Cyber Threats: Issues, Approaches and Challenges*, 2003. Article (CrossRef Link).

[4] K. Xu, Z. Zhang and S. Bhattacharyya, "Profiling internet backbone traffic: behavior models and applications," in *Proc. of Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pp. 169-180, 2005. Article (CrossRef Link).

[5] D. Moore, G. Voelker and S. Savage, "Inferring Internet Denial-of-Service Activity," in *Proc. of USENIX Security Symposium*, 2001. Article (CrossRef Link).

[6] H. Ringberg, A. Soule, J. Rexford and C. Diot, "Sensitivity of PCA for traffic anomaly detection," in *Proc. of ACM SIGMETRICS Int. Conf. on Measurement and Modeling of Computer Systems*, pp. 109-120, 2007. Article (CrossRef Link).

[7] P. Huang, A. Feldmann and W. Willinger, "A non-intrusive, wavelet-based approach to detecting network performance problems," in *Proc. of Internet Measurement Workshop*, 2001. Article (CrossRef Link).

[8] S.S. Kim, A. L. N. Reddy and M. Vannucci, "Detecting traffic anomalies using discrete wavelet transform," in *Proc. of Int. Conf. Information Networking*, pp. 1375-1384, 2004. Article (CrossRef Link).

[9] J. Schonwalder, "Characterization of SNMP MIB modules," in *Proc. of IFIP/IEEE Int. Symposium on Integrated Network Management(IM)*, 2005. Article (CrossRef Link).

[10] J.B.D. Cabrera, L. Lewis, X. Qin, C. Gutierrez, W. Lee and R.K. Mehra, "Proactive intrusion detection and SNMP-based security management: new experiments and validation," in *Proc. of IM*, 2003. Article (CrossRef Link).

[11] J. Li and C. Manikopoulos, "Early statistical anomaly intrusion detection of DOS attacks using MIB traffic parameters," in *Proc. of IEEE Information Assurance Workshop*, pp. 53-59, 2003. Article (CrossRef Link).

[12] R, Puttini, M. Hanashiro, F. Miziara, R.D. Sousa, L.J. García-Villalba and C.J. Barenco, "On the anomaly intrusion detection in mobile ad hoc network environments," in *Proc. of PWC, LNCS*, vol. 4217, pp. 182-193, 2006. Article (CrossRef Link).

[13] K.H. Ramah, H. Ayari and F. Kamoun, "Traffic anomaly detection and characterization in the tunisian national university network," in *Proc. of Networking, LNCS*, vol. 3979, pp. 136-147, 2006. Article (CrossRef Link).
[14] D. Lee, B Park, K. Kim and J. Lee, "Fast traffic anomalies detection using SNMP MIB correlation analysis," in *Proc. of Int. Conf. on Advanced Communication Technology*, pp. 166-170, 2009.
[15] Y. Yemini, "The OSI Network Management Model," *IEEE Communications Magazine*, pp. 20-29, May 1993. Article (CrossRef Link).
[16] T. Pao and P. Wang, "NetFlow based intrusion detection system," in *Proc. of IEEE Int. Conf. on Networking, Sensing and Control*, vol.2, 2004. Article (CrossRef Link).
[17] M. Meyer, "Decentralizing Control and Intelligence in Network Management," *Integrated Network Management IV*, Sethi et al., Eds., Chapman and Hall, 1995.
[18] S. Sengupta, V. Kumar and D. Saha, "Switched optical backbone for cost-effective scalable core IP networks," *IEEE Communications Magazine*, vol. 41, no. 6, pp. 60-70, June 2003. Article (CrossRef Link).
[19] Y. Gottlieb and L. Peterson, "A comparative study of extensible routers," in *Proc. of IEEE Open Architectures and Network Programming*, pp. 51-62, June 2002. Article (CrossRef Link).
[20] H. Fan, J. Liu, Y. Wu and C. Cheung, "On optimal hyperuniversal and rearrangeable switch box designs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 12, pp. 1637-1649, December 2003. Article (CrossRef Link).
[21] Y.P. Zhou and C.L. Tan, "Learning-based scientific chart recognition," in *Proc. of 4th IAPR Int. Workshop on Graphics Recognition*, pp. 482-492, 2001. Article (CrossRef Link)
[22] C. Lewis, "Cisco TCP/IP Routing Professional Reference (second edition)," ISBN 0070411301, McGraw-Hill Companies, 1998.
[23] T. Thomas, "Juniper Networks Reference Guide: JUNOS Routing, Configuration, and Architecture, chapter Juniper Networks Router Architecture," ISBN 0201775921, Addison Wesley Professional, January 2003.
[24] Hitachi Ltd., "Routers, Switches GS/GR Series: Hitachi," http://www.hitachi.co.jp/Prod/comp/network/index-j.htm, 2010.
[25] S. Shah and M. Yip, "Extreme Networks' Ethernet Automatic Protection Switching (EAPS) Version 1," *IETF RFC 3619*, October 2003.
[26] J. Edwards, "Building the optical-networking infrastructure," *IEEE Computer*, vol. 33, no. 3, pp. 20-23, March 2000. Article (CrossRef Link).
[27] W. Cho, S. Kim and H. Yeh, "Introduction to the 'uAuto' Project - Ubiquitous Autonomic Computing and Network," in *Proc. of the Second IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems*, 2004. Article (CrossRef Link).
[28] ECI Telecom Ltd., "Korea's largest telco offers secure, high-speed Internet Services via ECI Telecom," http://www.ecitele.com/Products/BroadbandServiceRouting/ST-series/Pages/default.aspx, 2005.
[29] S. Mane, J. Srivastava, H. Yin and J. Vayghan, "Estimation of false negatives in classification," in *Proc. of IEEE Int. Conf. on Data Mining*, pp. 475-478, November 2004. Article (CrossRef Link).
[30] M. Shimamura and K. Kono, "Using Attack Information to Reduce False Positives in Network IDS," in *Proc. of 11th IEEE Symposium on Computers and Communications*, pp. 386-393, June 2006. Article (CrossRef Link).
[31] R. Kong, "The Simulation for Network Mobility Based on NS2," in *Proc. of Int. Conf. on Computer Science and Software Engineering*, vol. 4, pp. 1070-1074, December 2008. Article (CrossRef Link).
[32] J. Dorleus, R. Holweck, Z. Ren, H. Li, H. Cui and J. Medina, "Modeling and Simulation of Fading and Pathloss in OPNET for Range Communications," in *Proc. of IEEE Radio and Wireless Symposium*, pp. 407-410, January 2007. Article (CrossRef Link).

Dongcheul Lee received the B.S. and M.S. degrees in computer science and engineering from POSTECH, Pohang, Korea in 2002 and 2004, respectively. Currently, he is a Ph.D. candidate in electronics and computer engineering, Hanyang University, Seoul, Korea. His research interests include network management, algorithm and application of mobile communications, software architecture, UMTS, and international roaming.

Byung Ho Rhee received the B.S. and M.S. degrees in electronics engineering from Hanyang University, Seoul, Korea in 1975 and 1977, respectively, and the Ph.D. degree in electric and electronics engineering from National Chiba University, Chiba, Japan in 1993. Since 1981, he has been a Professor in the College of Information and Communications at Hanyang University, Seoul, Korea. From 2006 to 2008, he was the Dean of the college. Since 2004, he has been the President and Member of Committee, IT personnel, Cultivating and Policy Council, MKE, Korea. His primary research interests include network management, software defined radio, session initiation protocol, NGN, and security in wireless network.