THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

TECHNICAL REPORT OF IEICE

On Feasibility of P2P Traffic Control through Network Performance Manipulation

HyunYong Lee[†] Masahiro Yoshida[‡] and Akihiro Nakao[‡]

[†] National Institute of Information and Communications Technology (NICT), Tokyo, Japan
[‡]The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo, 113-0033 Japan
E-mail: [†] ifjesus7@gmail.com, [‡] yoshida@nakao-lab.org, nakao@iii.u-tokyo.ac.jp

Abstract We propose a new kind of P2P traffic control technique, called Netpherd exploiting the peer selection adaptation (i.e., preferring the peers who are likely to provide better performance). Netpherd tries to enable the peers to communicate with the peers of the local domain by affecting the peer selection adaptation through manipulating network performance (i.e., adding artificial delay at network device like router). Simulation results show that Netpherd can increases (decreases) the intra-domain (inter-domain) traffic by affecting the peer selection adaptation. Netpherd also improves the content download performance including the number of download completions, which means Netpherd can satisfy both the network operator and the peers.

Keyword P2P traffic control, Peer selection adaptation, Network performance manipulation

1. Introduction

To control P2P traffic, a bilateral approach based on the collaboration between the network operator and the peers has been proposed [1]-[8]. In this approach, the peers select their partners by following a guidance provided by the network operator so that some of inter-domain traffic may be redirected within a local domain. This approach is regarded as a promising way, since it can satisfy both the network operator and the peers. However, there are several practical issues to be satisfied for successful deployment [9]. For example, the bilateral approach requires a dedicated server to provide the guidance and a modification of P2P applications to reflect the guidance, which makes a real deployment difficult.

In this paper, we try to satisfy both the network operator and the peers by exploiting the peer selection adaptation (i.e., preferring the peers who are likely to provide better networking performance than others). For example, with the peer selection adaptation, BitTorrent client [10] exchanges content preferentially with other peers who have uploaded content to itself in the past at high bandwidth. Therefore, we posit that we can turn the inter-domain traffic into the intra-domain traffic if we can degrade the network performance across network domains. For this, we propose a P2P traffic control approach through network performance manipulation, called Netpherd. Netpherd tries to enable the peers to communicate with peers of the local domain by decreasing the network performance across network domains. To manipulate the network performance, Netpherd adds an artificial delay to the inter-domain traffic at network device. Netpherd does not require the dedicated server and the modification of P2P applications.

Through simulations, we show that Netpherd reduces the inter-domain traffic while increasing the intra-domain traffic in most cases. It means that the network performance manipulation enables the peers to communicate with the peers of the local domain. By examining the change of peer selection behavior (i.e., a number of unchokings), we confirm this. Netpherd also increases the number of download completions and decreases the download completion time.

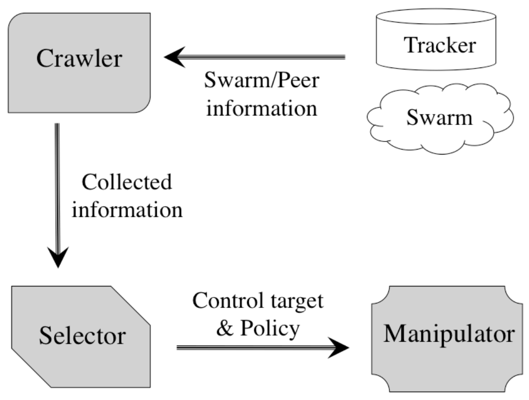


Figure 1. Conceptual illustration of Netpherd.

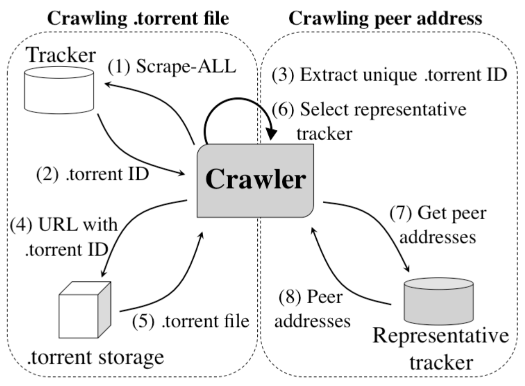


Figure 2. Scalable crawling of .torrent files and peer addresses.

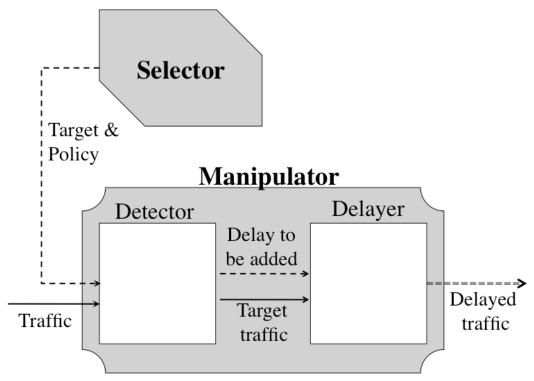


Figure 3. Manipulator: detector and delayer.

2. P2P Traffic Control through Network Performance Manipulation

In BitTorrent, with its peer selection adaptation technique called tit-for-tat, the peer allocates its upload slots (i.e., unchokes) to the peers who have uploaded content to itself in the past at high bandwidth periodically. Netpherd tries to affect the peer selection adaptation to redirect the inter-domain traffic within the local domain by adding the artificial delay to the target traffic. Netpherd consists of three components: the crawler, the selector, and the manipulator (Fig. 1). The crawler collects information of swarms and peers staying in target network domain. Then, the selector determines target swarms and peers to be controlled. The selector also generates a policy for the performance manipulation (e.g., amount of artificial delay). Finally, the manipulator detects the target traffic and adds the artificial delay to the target traffic.

2.1 Crawler

Netpherd first collects information of swarms and peers (Fig. 2).

Crawling .torrent files. To collect the swarm information, we need to collect .torrent file. Our crawler uses a .torrent storage server [16] instead of WEB site that is used by conventional approach [13]. A .torrent storage server is the server specialized in storing and redistributing .torrent files. A peer can download a specific .torrent file with a URL that includes its identifier. To collect as many identifier of .torrent files as possible, our crawler uses a Scrape-ALL [11] request. Then, the crawler downloads .torrent files from the .torrent storage server after extracting unique identifiers among the downloaded identifiers.

Crawling peer addresses. With the crawled .torrent files, the crawler collects IP address and port number of peers to know the number of peers staying in the target network domain. Unlike conventional approach accessing all trackers listed in .torrent files, our crawler contacts only one representative tracker in each swarm to efficiently obtain peer addresses, since 90% of swarms are managed by multiple trackers (4.82 on average) [12]. We define a representative tracker as the tracker that maintains the maximum number of peers in a swarm. After selecting the representative tracker, the crawler contacts the tracker periodically as the conventional approach does (i.e., the crawler repeats collecting a random subset until no more new peers can be discovered from the tracker). Additional information about our crawling is described in [12].

2.2 Selector

The selector determines the target traffic to be controlled and generates a policy for the performance manipulation. The selector describes the target traffic with the swarm ID (i.e., .torrent identifier), IP address, and port number of target peer. The selection of target swarms, peers, and corresponding artificial delay needs to enough to satisfy both parties. For example, a swarm with more than certain number of local peers can be selected. However, in this paper, we do not propose any specific approaches for the selector including the

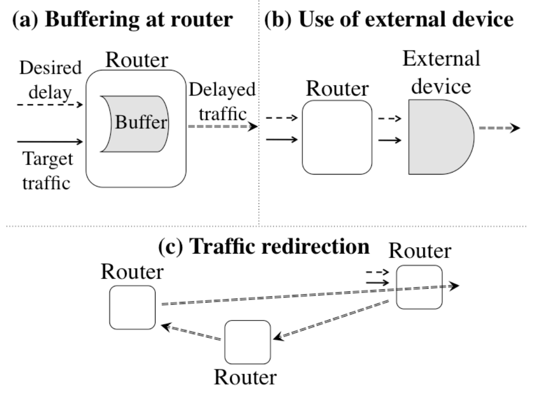


Figure 4. Implementation of delayer.

selection of swarms, since we believe that each network operator has its own selection criteria based on its traffic engineering policy.

2.3 Manipulator

The manipulator detects the target traffic and adds the artificial delay to the target traffic (Fig. 3). The manipulator consists of the traffic detector and the delayer. For the traffic detector, Netpherd can utilize existing deep packet inspection (DPI) tools [17], [18]. We believe that the detector can easily detect the target traffic, since the selector clarifies the swarm ID, IP address, and port number of the target peer.

The delayer can be implemented as follows (Fig. 4). **(a) Buffering by edge routers.** One obvious approach to add the artificial delay is to buffer packets at the edge router. This would require large buffers, making the router expensive and power inefficient [14]. The edge router also needs complex packet scheduling mechanisms.
(b) Use of external device. We may be able to utilize some external devices such as network emulator [19], [20] that can be attached to the edge router. This approach does not require the large buffer at the router and the modification of router. But, performance under real traffic load is questionable.
(c) Redirecting packet through routers. The edge router can redirect the target traffic through the routers of the local domain [15]. Edge routers monitor the end-to-end delay to other routers, so that the edge router can determine appropriate router for the traffic redirection. This approach requires one

Table I Inter-AS Delay of Target AS

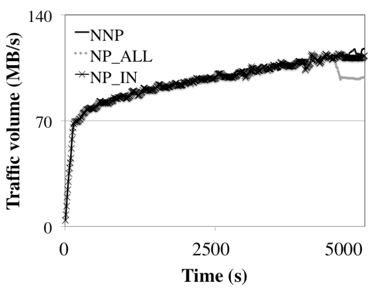
Simulation time (sec)	Delay (msec)
0 – 500	100
500 – 1000	100 + 100
1000 – 1500	100
1500 – 2000	100 + 200
2000 – 2500	100
2500 – 3000	100 + 300
3000 – 3500	100
3500 – 4000	100 + 400
4000 – 4500	100
4500 - 5000	100 + 900

centralized server selecting and managing a set of routers for the packet redirection.

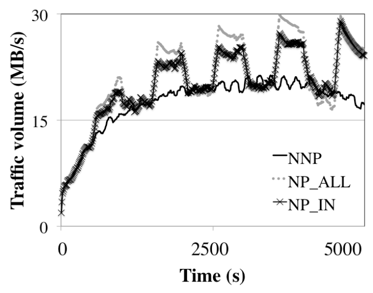
3. Performance Evaluation

3.1 Simulation Setup

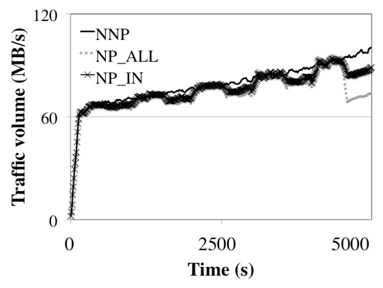
Here, we focus on how the artificial delay affects the peer selection adaptation, the traffic volume, and the content download performance. For this, we measured BitTorrent swarms through our crawler before the simulation. Among the swarms measured by the crawler, we randomly select 1 swarm including 861 seeders and 10,703 leechers. The peers are distributed over 416 ASes. And we build the simulation environment based on ns-2 [21] as follows. BitTorrent tracker returns 50 peers and 250MB-sized content is shared. Each peer has the same link capacity (i.e., 120KB/s for download and 40KB/s for upload) and 5 upload slots including one optimistic unchoking slot. The delayer is implemented at routers with a buffer of unlimited size. We set 50ms for a delay of intra-AS link and 100ms for a delay of inter-AS link. During simulations, we add the artificial delay to the inter-AS link every 500 seconds as shown in Table I. After increasing the delay for one interval, then we reset the delay to the first delay value for next interval to examine what happens when the artificial delay is eliminated. To study an effect of amount of artificial delay, different amount of delay is added. We select 7 peer-intensive ASes 283 seeders and 3,152 leechers (INT_AS) and 10 peer-scarce ASes 10 seeders and 178 leechers (SCA_AS) to examine what happens in ASes with different number of peers when the artificial delay is added. We define NON_AS as non-INT_AS including SCA_AS. For performance comparison, we add the artificial delay



(a) Total traffic

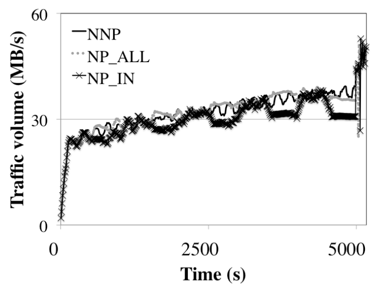


(b) Intra-domain traffic

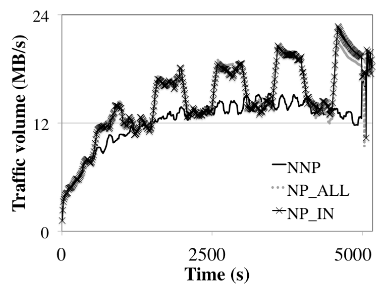


(c) Inter-domain traffic

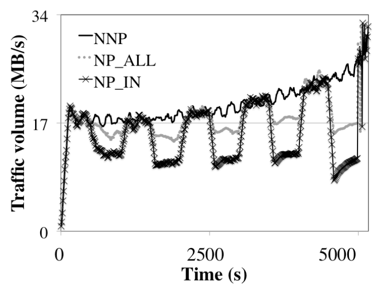
Figure 5. Traffic volume (all AS)



(a) Total traffic

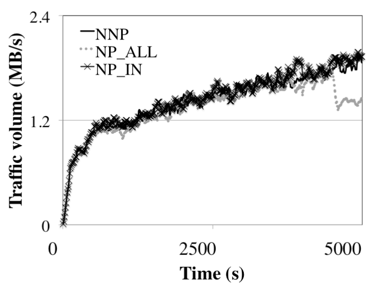


(b) Intra-domain traffic

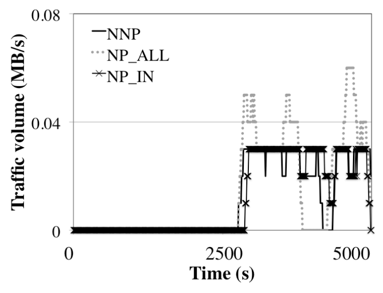


(c) Inter-domain traffic

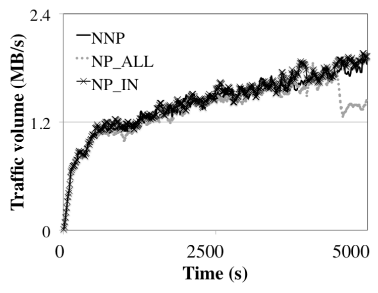
Figure 6. Traffic volume (INT_AS)



(a) Total traffic



(b) Intra-domain traffic



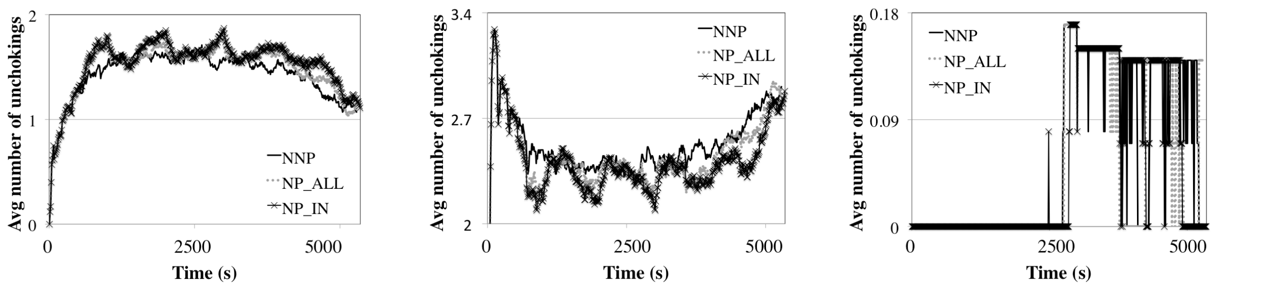
(c) Inter-domain traffic

Figure 7. Traffic volume (SCA_AS)

to all inter-AS traffic (NP_ALL) or to inter-AS traffic of INT_AS (NP_IN). We also include NNP as a base case where no artificial delay is added.

3.2 Traffic Volume

We first examine how the artificial delay affects the download throughput. In the collection of all AS, NP_ALL increases the total traffic volume slightly and NP_IN shows similar total traffic volume compared to NNP (Fig. 5(a)). NP_ALL and NP_IN increase the intra-domain traffic and decrease the inter-domain traffic over compared to NNP ((Fig. 5(b) and Fig. 5(c)). In NP_ALL case, the increased intra-domain traffic is slightly larger than the decreased inter-domain traffic and thus this results in the increased total traffic volume. On the other hand, NP_IN shows similar amount of change in the intra-domain and the inter-domain traffic and thus there is no noticeable change in the total traffic volume. However, if the artificial delay is too long (i.e., 900ms in our simulations), the decreased inter-domain traffic volume becomes larger than the increased intra-domain traffic volume, which results in the reduced total traffic volume in both NP_ALL and NP_IN. When the artificial delay is eliminated, NNP, NP_ALL, and NP_IN show similar traffic volume. It means that the peers communicate with its neighboring peers as they do before the performance manipulation when Netpherd stops adding the delay.



(a) Unchokings within the same AS (INT_AS) (b) Unchokings across ASes (INT_AS) (c) Unchokings within the same AS (SCA_AS)

Figure 9. Peer unchokings behaviors with the artificial delay.

Table II Download Performance

	Number of download completions			Average download time (sec)		
	NNP	**NP_ALL**	**NP_IN**	**NNP**	**NP_ALL**	**NP_IN**
All AS	144	245	287	4746	4700	4779
INT_AS	83	148	96	4753	4592	4730
SCA_AS	0	2	4	None	4936	4910

NP_IN shows noticeable change in the intra-domain and the inter-domain traffic (Fig. 5(b) and Fig. 5(c)). It means that we may be able to achieve the attractive results by just manipulating few ASes with many peers. INT_AS shows better performance improvement than others (Fig 6), since many peers of INT_AS know some peers of the same AS as their neighboring peers (Fig. 8). Except INT_AS, more than 50% of peers do not know any peers of the same AS as the neighboring peers. On the other hand, most peers of INT_AS have the peers of the same AS as the neighboring peers up to 15. NP_ALL increases total traffic slightly compared to NNP (Fig 6(a)). On the other hand, NP_IN decreases total traffic compared to NNP. NP_IN and NP_ALL show similar increase in the intra-domain traffic (Fig 6(b)), but NP_IN shows much decreased inter-domain traffic compared to NP_ALL (Fig 6(c)). The decreased inter-domain traffic is larger than the increased intra-domain traffic and thus this leads to the reduced total traffic in NP_IN.

On the other hand, SCA_AS decreases the total traffic volume slightly when the artificial delay is added in both NP_ALL and NP_IN (Fig 7). Actually, 97% of SCA_AS peers do not know any peers of the same AS as the neighboring peers (Fig. 8). Therefore, most peers face the decreased download performance, since there is only decrease in the inter-domain traffic.

3.3 Peer Selection Adaptation

Now, we examine how the artificial delay affects the peer selection behavior. We select one AS of INT_AS that includes 66 seeders and 800 leechers. When the artificial delay is added, the number of unchokings for peers of the same AS increases in both NP_ALL and NP_IN (Fig. 9), since most peers of INT_AS have the peers of the same AS as the neighboring peers (Fig. 8). NP_IN shows more number of unchokings within the same AS than NP_ALL. On the other hand, in SCA_AS, we do not observe any noticeable change (Fig. 9(c)).

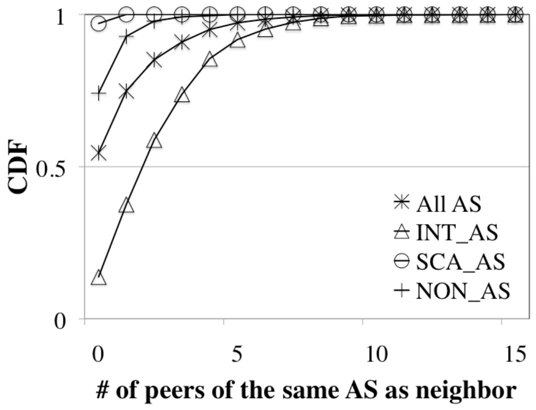


Figure 8. Distribution of number of neighboring peers of the same AS.

3.4 Download Completion Time

Table II shows the number of download completions and the average download completion time. With NP_ALL and NP_IN, the peers begin to complete the content download earlier than the peers with NNP. But, NP_IN shows longer average download completion time than NNP, since many peers complete the download later than the peers with NNP. One interesting observation is that INT_AS with NP_IN shows better download performance (i.e., the increased number of download completions) than INT_AS with NNP. We find that the network performance manipulation affects the download throughput of peers in different ways. For example, a peer who knows peers of the same AS as the neighboring peers improves the download throughput, since the increased intra-domain traffic is usually larger than the decreased inter-domain traffic. On the other hand, a peer who does not know peers of the same AS decreases the download throughput, since there is only the decreased inter-domain traffic.

4. Conclusion

The peer selection adaptation is another potential chance for P2P traffic control. Netpherd exploits the peer selection adaptation to enable the peers to communicate with the peers of the local domain by adding the artificial delay to the target traffic. The feasibility of the P2P traffic control through the network performance manipulation is verified through the simulations with the real trace of BitTorrent swarms. In particular, the simulation results show that Netpherd can really affect the peer selection adaptation by manipulating the networking performance. The results also confirm that enough number of peers within the local domain is one of the key factors for successful traffic localization. After extending our design as we mentioned, we plan to implement Netpherd in Internet environment.

References

[1] H. Xie, Y. Yang, A. Krishnamurthy, Y. Liu, and A. Silberschatz, "P4P: Portal for (P2P) applications," in Proc. of *ACM SIGCOMM*, 2008.

[2] V. Aggarwal, A. Feldmann, and C. Scheideler, "Can isps and p2p users cooperate for improved performance?," *SIGCOMM CCR,* vol. 37, no. 3, 2007.

[3] D. Saucez, B. Connet, and O. Bonaventure, "Implementation and preliminary evaluation of an isp-driven informed peer selection," in Proc. of *ACM CoNEXT*, 2007.

[4] D. Choffnes and F. Bustamante, "Taming the torrent: A practical approach to reducing cross-ISP traffic in peer-to-peer systems," in Proc. of *ACM SIGCOMM*, 2008.

[5] C. H. Hsu and M. Hefeeda, "ISP-friendly peer matching without ISP collaboration," in Proc. of *ROADS Workshop*, 2008.

[6] HyunYong Lee, Akihiro Nakao, and JongWon Kim, "BiCo: Network operator-friendly p2p traffic control through bilateral cooperation with peers," in *Computer Networks*, vol. 55, issue 9, June, 2011.

[7] F. Lehrieder, S. Oechsner, To. Hobfeld, D. Staehle, Z. Despotovic, and W. Kellerer, "Mitigating unfairness in locality-aware peer-to-peer networks," in *International Journal of Network Management*, vol. 21, issue 1, 2011.

[8] J. Seedorf, S. Niccolini, M. Stiemerling, E. Ferranti, and R. Winter, "Quantifying operational cost-savings through alto-guidance for p2p live streaming," in Proc. of *International Conference on Incentive, Overlays, and Economic Traffic Control*, 2010.

[9] S. L. Blond, A. Legout, and W. Dabbous, "Pushing bittorrent locality to the limit," in *Computer Networks*, vol. 55, issue 3, 2011.

[10] J. Seedorf and E. Burger, "Application-layer traffic optimization (ALTO) problem statement," Internet Engineering Task Force, RFC5693, 2009.

[11] B. Cohen, "Incentives build robustness in bittorrent," in Proc. of *P2PEcon*, 2003.

[12] G. Dan and N. Carlsson, "Dynamic swarm management for improved bittorrent performance," in Proc. of *IPTPS*, 2009.

[13] Masahiro Yoshida and Akihiro Nakao, "A resource-efficient method for crawling swarm information in multiple bittorrent networks," in Proc. of *AHSP*, 2011.

[14] S. Iyer, R. Zhang, and N. McKeown, "Routers with a single stage of buffering," in Proc. of *ACM SIGCOMM*, 2002.

[15] M. Yu, M. Thottan, and L. Li, "Latency equalization as a new network service primitive," *IEEE/ACM Transactions on Networking*, 2011.

[16] Torrage, Torrent storage cache, http://www.torrage.com/.

[17] IPOQUE, http://www.ipoque.com/.

[18] SANDVINE, http://www.sandvine.com/.

[19] Dummynet, http://info.iet.unipi.it/luigi/dummynet/.

[20] NIST Net, http://www.antd.nist.gov/tools/nistnet/index.html/.

[21] The Network Simulator ns-2, http://www.isi.edu/nsnam/ns/.