Tạp chí Tin học và Điều khiển học, T. 18, S. 4 (2002), 339–346

SIMULATION - BASED EVALUATION OF PERFORMANCE OF A SNOOP TCP SCHEME OVER NETWORK WITH WIRELESS LINKS

NGUYỄN ĐÌNH VIỆT

Abstract. TCP is a transport protocol designed and tuned to work well in fixed networks, however TCP performance is degraded severely in a network with wireless links. To enhance TCP performance in such networks, some proposals have been made in the last several years. Among them Snoop TCP proposal is one of the most important ones. Our paper investigates the effect of various error rates of wireless link on the performance of snoop TCP; normal TCP is also put to our investigation to make a comparison. We use the network simulator ns-2, version 2.1b9 extensively. Our simulation results show that in comparison with normal TCP, snoop TCP enhances normalized throughput of LAN and WAN with wireless link 137% and 427% respectively.

Tóm tắt. TCP là giao thức giao vận được thiết kế và tinh chỉnh để làm việc tốt trong các mạng cố định, tuy nhiên hiệu suất của TCP bị giảm trầm trọng trong mạng có đường truyền không dây. Trong những năm qua để cải thiện hiệu suất của TCP trong các mạng đó, người ta đưa ra một số kế hoạch. Một trong số các kế hoạch quan trọng nhất là Snoop TCP . Trong bài báo này, chúng tôi nghiên cứu ảnh hưởng của các tỷ suất lỗi khác nhau của đường truyền không dây lên hiệu suất của Snoop TCP; TCP thông thường đã được nghiên cứu nhằm mục đích so sánh. Chúng tôi sử dụng phần mềm mô phỏng mạng ns-2, phiên bản 2.1b9. Kết quả mô phỏng của chúng tôi cho thấy so với TCP thông thường, Snoop TCP đã cải thiện thông lượng chuẩn hóa của mạng LAN có đường truyền không dây lên 137%, với mạng WAN có đường truyền không dây, lên tới 400%.

1. INTRODUCTION

In a network with wireless links, packet losses are mostly caused by transmission errors or long delay handoff, but not network congestion. The resolutions proposed to this problem can be divided into two main categories. The first consists in hiding the lossy parts of the network so that only congestion losses are detected at the source. The second type of solution consists in enhancing TCP with some mechanisms to help it to distinguish between different types of losses.

In this paper we are interested in the snoop TCP scheme, which is one of the TCP-level solutions dealing with bit-error losses to hide the lossy parts of the networks [3], [5]. Snoop-TCP will be studied deeply in this paper by simulation.

2. SNOOP TCP

The aim of this scheme is to improve the end-to-end performance on networks with wireless links without changing existing TCP implementations at hosts in the fixed network (Fixed Host, FH) and enable seamless integration of mobile devices communicating over wireless links with the rest of the Internet. We can achieve this by modifying the standard Internet network-layer software at the base station (BS) and mobile host (MH). *The snoop modifications consist of caching packets and performing local (intelligent) retransmissions (at the link layer) across the wireless link by monitoring the acknowledgments to TCP packets generated by the receiver, and by local timers* [1], [2].

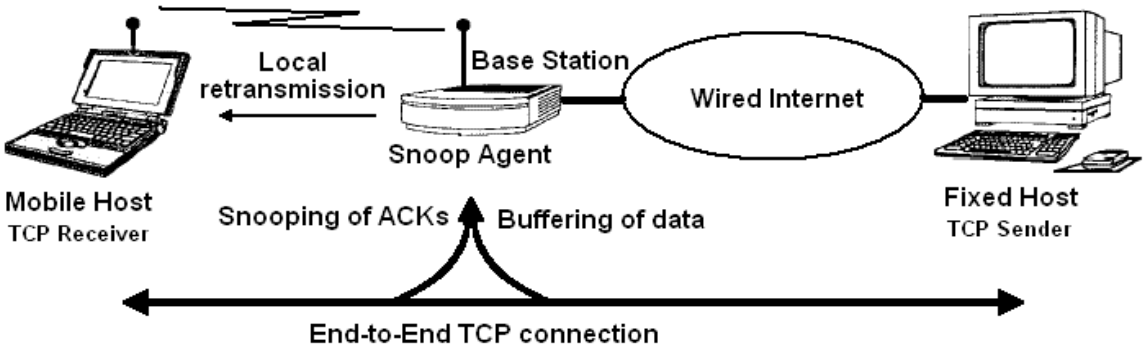

Figure 1. Snooping TCP as a transparent TCP extension

The snoop agent has two linked procedures, snoop_data() and snoop_ack(). Snoop_data() processes and caches packets intended for the MH while snoop_ack() processes acknowledgement (ACKs) coming from the MH and carries out local retransmissions from the BS to the MH. The flowcharts summarizing the algorithms for these two procedures are shown in figures 2 and 3 and will be described below.

2.1. Snoop data

A TCP packet (also called segment) is identified uniquely by the sequence number of its first byte of data and its size. At the BS, snoop agent keeps track of the last sequence number seen for the connection. One of several kinds of packets can arrive at the BS from the FH, and snoop_data() processes them in different ways:

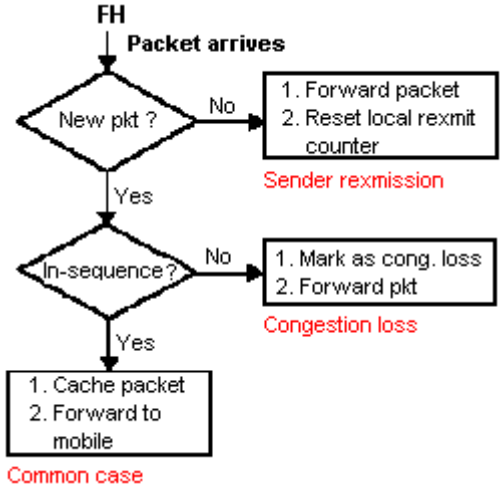

Figure 2. Flowchart for snoop_data()

1) *Common case: a new packet in the normal increasing sequence arrives at the BS.* In this case, the packet is added to the snoop cache and forwarded on to the MH. ((New pkt = Yes) and (In-sequence = Yes)).
2) *Sender retransmission: An out-of-sequence packet that has been cached earlier* (New pkt = No): This is a less common case, but it happens when dropped packets cause timeouts at the sender. It could also happen when TCP sender performs fast retransmit. Different actions are taken depending on whether this packet is greater or less than the last acknowledged packet seen so far.

If the sequence number is greater than the last acknowledgment seen, it is very likely that this packet didn't reach the MH earlier, and so it is forwarded on.
If, on the other hand, the sequence number is less than the last acknowledgment, this packet has already been received by the MH. At this point, one possibility would be to discard this packet and continue.

3) *Congestion loss: An out-of-sequence packet that has not been cached earlier* ((New pkt = Yes) and (In-sequence = No)). In this case, the packet was either lost earlier due to congestion on the wired network or has been delivered out of order by the network. For simplicity, in this paper we assume that, there are not congestion losses and out of order deliveries in wired part of the network.

2.2. Snoop ACKs

Snoop_ack() monitors and processes the acknowledgments (ACKs) sent back by the MH and performs various operations depending on the type and number of acknowledgments it receives. These ACKs fall into one of three categories:

1) *A new ACK:* This is the common case (when the connection is fairly error-free and there is little user movement), and signifies an increase in the packet sequence received at the MH. Snoop_ack() procedure will performs as follow:

   Free buffers: This ACK initiates the cleaning of all acknowledged packets in snoop cache.

   Update RTT: The round-trip time (RTT) estimate for the wireless link is also updated at this time.

   Intelligent update: This estimate is not done for every packet, but only for one packet in each window of transmission, and only if no retransmissions happened in that window. The last condition is needed because it is impossible in general to determine if the arrival of an acknowledgment for a retransmitted packet was for the original packet or for the retransmission.

   Propagate ack to sender: Finally, forward acknowledgment to the FH.

2) *A spurious ACK:* This is an acknowledgment less than the last acknowledgment seen by the snoop module and is a situation that rarely happens. It is discarded and the packet processing continues.

3) *A duplicate ACK (DUPACK):* This is an ACK identical to a previously received one. In particular, it is the same as the highest cumulative ACK seen so far. One of several actions is taken depending on the type of duplicate acknowledgment and the current state of snoop:

   First DUPACK (First one? = Yes): If we have the packet (in snoop cache) then we send it and discard the DUPACK. If the packet is not in the cache, it needs to be resent from the FH, so the DUPACK needs to be forwarded to the FH because the TCP stack there maintains state based on the number of duplicate acknowledgments it receives when it retransmits a packet.

   Successive DUPACK (First one? = No): This case occurs when an "expected" DUPACK arrives, based on the above maximum estimate. The missing packet would have already been retransmitted when the first DUPACK arrived (and the estimate was zero), so this acknowledgment is discarded. In practice, the retransmitted packet reaches the MH before most of the later packets do (because it was retransmitted at a higher priority) and the BS sees an increase in the ACK sequence before all the expected DUPACKs arrive. Retransmitting packets at a higher priority improves performance at all error rates.

   Intelligent retransmission when receiving first DUPACKs:
   In order to make the number of such DUPACKs as small as possible, the lost packet is retransmitted as soon as the loss is detected, and at a higher priority than normal packets. This is done by maintaining two queues at the link layer for high and normal priority packets. This enables retransmitted packets to reach the mobile host sooner, reducing the number of duplicate ACKs and leading to improved throughput. Snoop agent keeps track of the number of local retrans-

missions for a packet, but resets this number to zero if the packet arrives again from the sender following a timeout or a fast retransmission. In addition to retransmitting packets depending on the number and type of acknowledgments, the snoop module also performs retransmissions driven by timeouts.

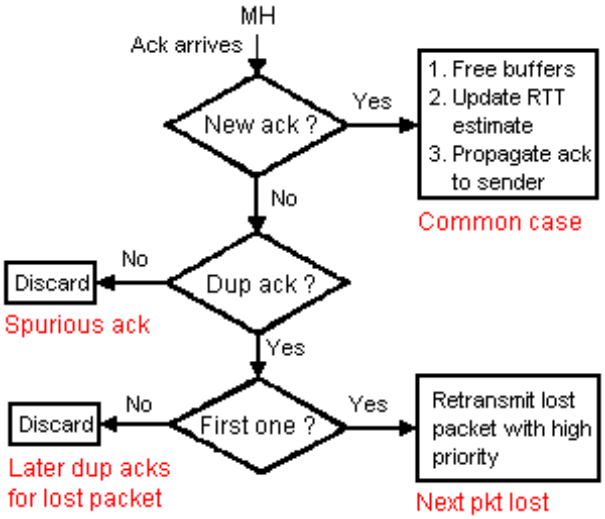

Figure 3. Flowchart for snoop_ack()

In addition, snoop agent also estimates the maximum number of duplicate acknowledgments that can arrive for this packet. This is done by counting the number of packets that were transmitted after the lost packet prior to its retransmission.

3. SIMULATION

We use ns-2 simulator [7], version 2.1b9 with integrated Snoop Agent [6] to simulate network with different parameters of Wired and Wireless link, TCP sender, and Base Station. To process simulation data in trace file, the program trace-graph version 1.60b is used extensively [8].

Our simulation network is depicted on Figure 4. A TCP sender denoted by SRC (Source), and a TCP receiver denoted by SNK (Sink). SRC is attached to FH (Fixed Host) in wired network, and SNK is attached to MH (Mobile Host). MH accesses SRC in FH via BS (Base Station) using wireless link, this means that in our simulation data packets are sent from FH to MH.

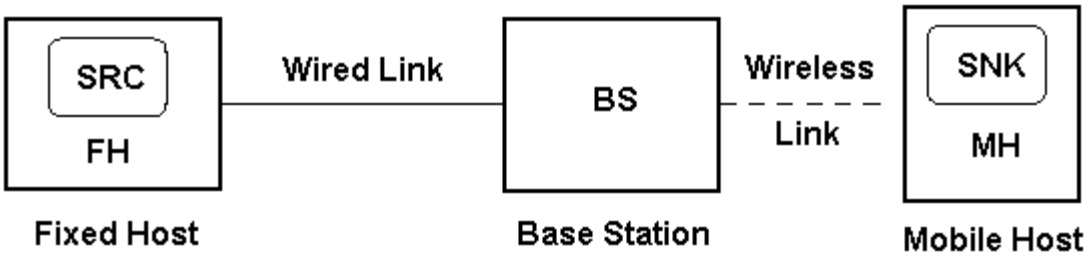

Figure 4. Simulation network

3.1. Wired Link Parameters

Different values of wired links are used:

• Duplex link, 10Mbps, 10ms in case of LAN with wireless link simulation.

• Duplex link, 10Mbps, 100ms in case of WAN with wireless link simulation.

3.2. Wireless Link Parameters

Our simulation is performed with wireless link transfer rate of 2Mbps, corresponding to a LAN 802.11, and with error characteristics described below.

Error Model

Characterizing the loss behavior is an important problem since it is one of the few key parameters affecting all levels of the network stack. The loss characteristics of wireless channels have been empirically observed to be bursty due to various fading effects. In [4] authors used trace-based approach to record, analyze and validate packet loss information. The large number of traces had been collected provide a good basic for developing and validating several wireless error models. The authors presented three models for erroneous characteristics of wireless channels, they are: Uniform error model, Two-state Markov model, and Improved Two-state Model. They also pointed out that when using these models in simulation, the obtained results differ by 21%, 13% and 5% respectively from the results of trace-driven simulation. For the simplicity, Markov model may be adequate for some applications and is frequently used in simulation study.

Implemented Error Model:

In simulations we use modified two-state Markov error model as illustrated in the Figure 5. The model contains two states: good state (Good) and bad state (Bad), each having its own exponential distributed error-packet arrival rate λ_G and λ_B respectively. T_G and T_B are lengths of Good and Bad states. The transition probabilities from Good to Bad and from Bad to Good are P_{GB} and P_{BG}, with $P_{GB} = 1/T_G$ and $P_{BG} = 1/T_B$, and probabilities of remaining in Good and Bad are $P_{GG} = 1 - P_{GB}$ and $P_{BB} = 1 - P_{BG}$.

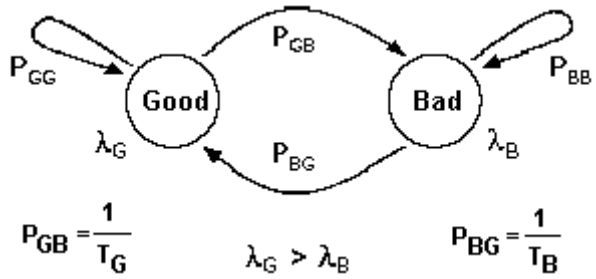

Figure 5. Modified two-state Markov Error model

Our simulations are performed with the following parameters of error model [4]:

• Good state: $\lambda_G = 166$, P_{GB}=0.006, P_{GG}=0.994, $T_G = 0.66$s.

• Bad state: $\lambda_B = 2.6$, P_{BG}=0.382, P_{BB}=0.618, $T_B = 0.01$s.

In some specific cases, we vary λ_B or T_B to investigate the effect of packet error rate or bad-state length on the performance of TCP.

3.3. TCP Connection Parameters

In our simulations TCP sender is set up with the following parameters:

• In case of WAN with wireless link simulation: Window size = 100 packets (approximates to bandwidth-delay product of wired link = 10Mbps × 100 ms = 1Mb ≈ 100 packets)

• In case of LAN with wireless link simulation: Window size = 15 packets. Other parameters use default values, for example: RTT granularity = 500 ms.

3.4. Snoop Agent Parameters

• Retransmission timeout interval for the next packet to send, rto is set:

$$\text{rto} = \max(\text{srtt\_} + 4*\text{rttvar\_}, \text{snoopTick\_})$$

• Smoothed round-trip-time (RTT) srtt is calculated as follows:

$$srtt\_ = g\_ * srtt\_ + (1\text{-}g\_) * rtt, \text{ with } 0 < g\_ < 1.$$

rtt: a RTT measurement from the most recently acked data packet.
rttvar_: a variable used to calculate standard deviation in RTT, and is used to determine rto. The algorithm for calculation of rttvar_ is as follows:
Delta = rtt - srtt_;
if (delta < 0) delta = - delta;
if (rttvar_ != 0) rttvar_ = g_ * delta + (1-g_) * rttvar_;
else rttvar_ = delta;
snoopTick_: minimum retransmission timer granularity (default = 0.1s, varied: 0.025 ÷ 0.1 s)
g_: a filter gain constant (default = 0.125) with a suggested value of 0.9 in wired networks.

• maxbufs_: maximum number of packet buffer for snooping (default = 100).

4. SIMULATION RESULTS

4.1. Needed simulation time

In term of needed simulation time, it means the duration of time that the interested simulation results converge to a steady value. Determining simulation time is a matter of great important when ns simulator is used, because the size of its output file (trace file) increases linearly with the simulation time and sometimes becomes hundreds of megabytes and it would take haft an hour to process such a big file.

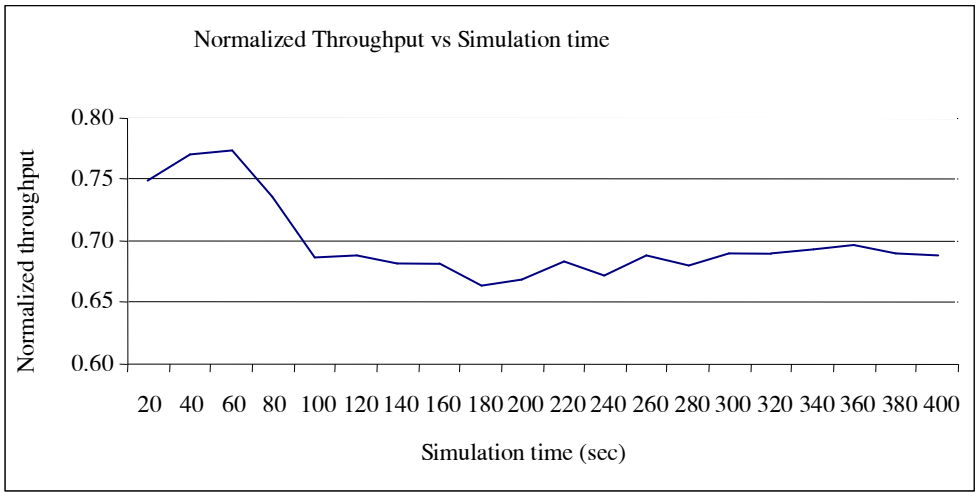

Figure 6

Firstly we carry out simulations with simulation times 20s, 40s... 400s to define needed simulation time. The simulation network is depicted on Figure 4 with parameters correspond to the case of WAN with wireless link and snoop agent is used (specific values of parameters are given in sections 3.2, 3.3 and 3.4).

Table 1. Normalized Throughput vs. Simulation time

Simulation time (sec)	20	40	60	80	100	120	140	160	180	200
Normalized Throughput	0.75	0.77	0.77	0.74	0.69	0.69	0.68	0.68	0.66	0.67
Simulation time (sec)	220	240	260	280	300	320	340	360	380	400
Normalized Throughput	0.68	0.67	0.69	0.68	0.69	0.69	0.69	0.70	0.69	0.69

The results of simulations are in Table 1 and represented in Figure 6. It is obvious that 100s is minimum needed simulation time, and results go up and down around the average value of 0.684 with the deviation of 2% when simulation time ranges from 140s to 400s. We choose 150s simulation time for all our later simulations.

4.2. Normalized Throughput vs. bad state error-packet arrival rate

We carry out 36 simulations to investigate the effect of bad-state error-packet interarrival on normalized throughput of TCP protocol. Four simulated scenarios are: WAN/LAN with/without snoop. Results are on Table 2 and represented by a graph on Figure 7. From the obtained results, it is very clear that:

- In all four cases, the normalized throughputs decreased with the increase of packet error rate.
- WAN suffers a more serious effect of a lossy wireless link than LAN does.

With good-state/bad-state error-packet inter arrival of 166pkt and 2.6pkt respectively (average values obtained by trace-based approach for modeling wireless channel behavior [4]), we obtain: In the case of WAN with wireless link, snoop-agent enhances normalized throughput approximately 427% (0.64/0.15). In the case of LAN with wireless link, snoop-agent enhances normalized throughput approximately 137% (0.89/0.65).

Table 2. Normalized Throughput vs. bad-state error-packet inter arrival

Bad-state error-packet inter arrival	128	64	32	16	8	4	2	1	2.6
WAN + snoop	0.83	0.84	0.83	0.77	0.78	0.74	0.67	0.64	0.64
WAN + no-snoop	0.26	0.25	0.25	0.20	0.18	0.18	0.18	0.14	0.15
LAN + snoop	0.93	0.93	0.92	0.92	0.92	0.90	0.89	0.85	0.89
LAN + no-snoop	0.82	0.82	0.82	0.81	0.79	0.74	0.64	0.70	0.65

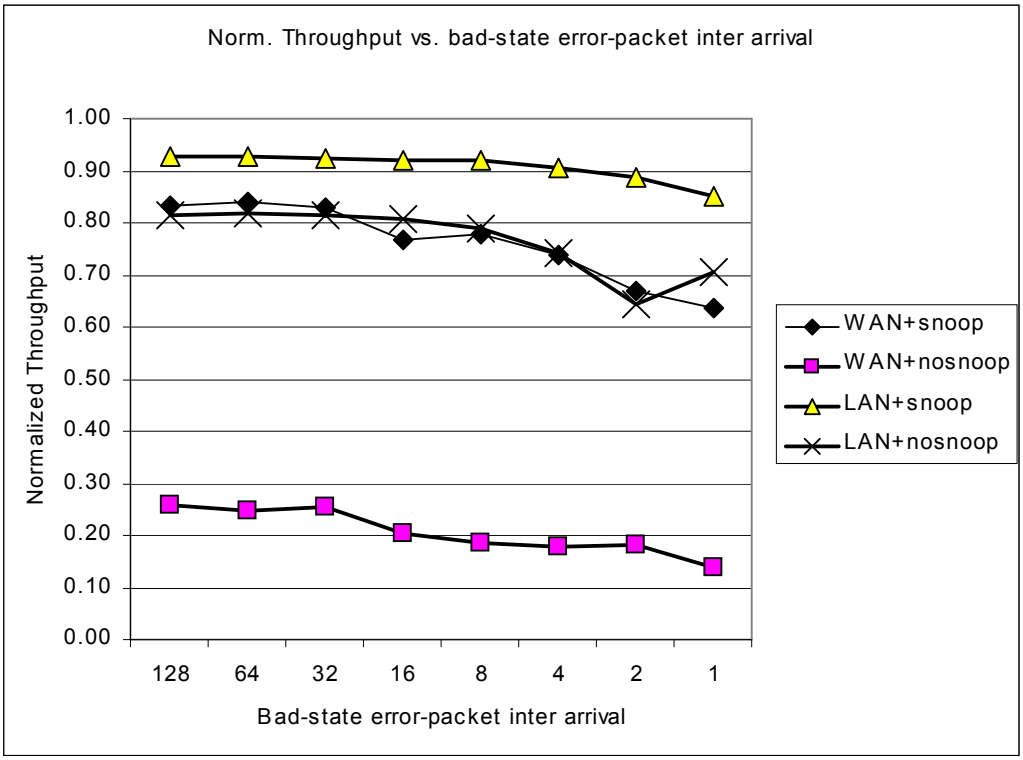

Figure 7

5. CONCLUSION

We have presented the simulation-based evaluation of performance of Snoop TCP scheme over network with wireless link. Our results have shown that snoop TCP is superior to normal TCP. In a LAN with wireless link, snoop TCP performs 137% better than normal TCP does, and in the case of WAN with wireless link, snoop TCP performs 427% better.

REFERENCES

[1] H. Balakrishnan, S. Seshan, and R. H. Katz, Improving reliable transport and handoff performance in cellular wireless networks, *ACM Wireless Networks* **1** (1995).
[2] K. Ratnam and I. Matta, Effect of Local Retransmission at Wireless Access Points on the Round Trip Time Estimation of TCP. *Proc. IEEE 31st Annual Simulation Symposium '98*, Boston, MA, April (1998).
[3] Hari Balakrishnan, Student Member, IEEE, Venkata N. Padmanabhan, Student Member, IEEE, Srinivasan Seshan, and Randy H. Katz, Fellow, IEEE, A Comparison of Mechanisms for Improving TCP Performance over Wireless Links, *IEEE/ACM Transactions on Networking* **5** (6) (1997).
[4] G. T. Nguyen, R. H. Katz, B. D. Noble, and M. Satyanarayanan, A trace-based approach for modeling wireless channel behavior, *Proc. Winter Simulation Conf.,* Dec. 1996.
[5] Nguyen Dinh Viet, TCP Enhancements and Performance Over Networks with Wireless Links, *Journal of Computer Science and Cybernetics* **18** (2) (2002).
[6] Snoop agent source codes: ...\ ns-allinone-2.1b9\ ns-2.1b9\ tcp\ ns-ll.tcl, snoop.h, snoop.cc.
[7] http://www.isi.edu/nsnam/
[8] http://www.geocities.com/tracegraph/

Received October 6 - 2002

Faculty of Technology - HaNoi National University