Journal of Southeast University (English Edition) Vol.19 No.1 Mar. 2003 ISSN 1003—7985

Research of NS dataflow mechanism and its analyzer implementation

Jin Ye Fan Jun

(Department of Computer Science and Engineering, Southeast University, Nanjing 210096, China)

Abstract: This paper analyzes the main elements in NS network simulator, makes a detailed view of dataflow management in a link, a node, and an agent, respectively, and introduces the information described by its trace file. Based on the analysis of transportation and treatment of different packets in NS, a dataflow state machine is proposed with its states exchange triggering events and a dataflow analyzer is designed and implemented according to it. As the machine state functions, the analyzer can make statistic of total transportation flux of a specified dataflow and offer a general fluctuation diagram. Finally, a concrete example is used to test its performance.

Key words: network simulation; NS simulator; dataflow analysis; state machine

Network simulator (NS) is the network simulation integration circumstance produced by U.C. Berkeley. It is characterized by the opening and extending simulation system, which can be operated on most operation system (OS) platforms. NS is also an efficient tool used to research network topology and analyze network transmission. Compared with other network simulators, NS has strong ability in dataflow simulation. In a discrete event simulation system, the generation of dataflow and its management mechanism in NS are core functions for usage. From the version ns-2.1b4 to ns-2.1b9, NS group has added more and more new dataflow into it, but the basic dataflow mechanism still functions. So, it is very useful to analyze the mechanism and trace detailed structure of dataflow management, which can give us valuable information in network simulation extension and simulation trace file analyzing.

An important step of simulation is analyzing the result and abstracting useful information. Till now NS has provided the text-based trace file and graphic analyzer: nam simulation data dynamic graphic system for analyzing. These means can offer detailed information of the whole simulation process, but cannot provide an easy way to analyze and generate statistics of unique data flow in a complex network circumstance. Specially we lack the mechanism to trace specified dataflow on a single node. The problem obstructs our simulation work. So, this paper summarizes the mechanism of a dataflow and introduces the implementation of a dataflow analyzer. Finally we give an example by network simulation.

Received 2002-09-10.

Foundation item: The Natural Science Foundation of Jiangsu Province (BK2001205).

Biography: Jin Ye (1978—), male, graduate, eddik@seu.edu.cn.

1 Analysis of NS Dataflow Management Mechanism

NS simulator supports the generation of many sorts of dataflow. Its wholesome process begins from network topology construction, and covers the network layer, transport layer and application layer of the simulation network. The simulation scripts describe the network from the network layer, gives detailed information of the relations between transport layer and network layer. It also provides a map from the application layer to the transport layer. From this structure, the core simulation program divides dataflow into several types on network layers and transport layers, such as TCP, UDP, Telnet, MPLS, rtProto, DVProto[1] (the last two are simulation routing protocols), and so on. Although dataflow requires different management policies, the main frames of the core mechanism follow a similar routine. These basic frames are formed by key structure elements in the NS simulator.

There are three key simulation elements: node, link and agent, which are all indispensable in dataflow management.

1.1 Dataflow management in a link

The mechanism is demonstrated by Fig.1[2]. A data packet sent by a node will be pushed into a link queue, to wait for transporting. Then it is popped out of the queue and sent along the simulation link, which can be represented by a timer for transporting delay.

Finally, it is pushed into the node queue of the destination node. In the graph, drophead_ represents the head of a queue. All packets that cannot be sent to the destination will be pushed into it. There are two types of such packets. The first type is a timeout packet. When such a packet is waiting in a link queue, the simulator starts a queue timer simultaneously for each packet. Once the time expires, the corresponding packet will be moved to drophead_ queue, which signifies that the current packet is lost on this link. Then according to different protocols, simulator can arrange a suitable policy to deal with such loss. If protocol requires retransmission of the packet, the simulator will signal the time when the loss happens and add a new transmission event to the event queue. If the protocol has no retransmission mechanism, the packet will still exist in the drophead_ queue, until the queue is full of the packets. In this circumstance, the head packet of the queue will be discarded.

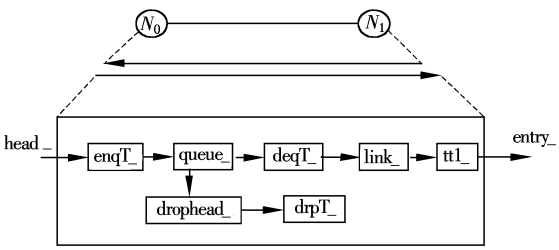


Fig.1 Dataflow management in a link

The other type of packet is an overflow packet. The link queue has its own length. If the queue reaches its maximum length, the following packets will overflow from it and be moved into the drophead _ queue directly. Management of these packets in drophead _ queue is the same as in the previous ones. During the process dataflow passes a link, the simulator signals the time when each packet enters the link queue (enqT _) and the time when each packet leaves it (deqT_). From these two variables, the simulator can offer the time during which a packet stays on the link. Then the packet will be mapped to a corresponding link (link_), which has been established at the beginning of simulation.

1.2 Dataflow management in a node

When a node is receiving or sending a packet, the simulator needs classification and filter mechanism for further transmission. Illustrated in Fig.2, the mechanism in a single node is shown[3]. It is the structure of a point-to-point transmission node. A packet enters the node from its entrance (entry_). The node uses address classifiers and port classifiers to filter each packet to the proper exit. Port classifiers are used to divide diverse network simulation service, while address classifiers enable the node to transmit packets to the correct next hop. If the node needs to multicast, there are several levels of classifiers internally. If the node can follow different routes, it needs to provide independent classifiers for each route. Especially in a link-state routing protocol simulation, routes with the same cost also need nodes to offer a load-balance policy, which increases the complexity of classifier levels. To each level of classifier, the simulator provides timers to monitor each process. The timeout, error, and exception will be written in trace files, and trigger a new event in the event queue to deal with it.

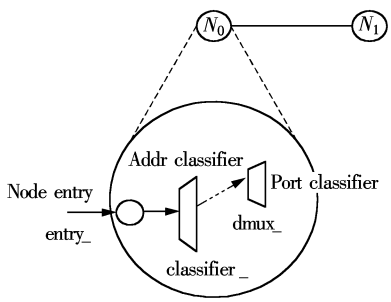


Fig.2 Dataflow management in a node

The process of dataflow in nodes and links provides the fundamental management of dataflow in the network layer. As this is a simulation, dataflow between nodes does not run for real transmission. Instead, timers and queues are the main parts. The link queue timer simulates the transmission delay, and data packets only move from one queue to another. Alternative queues between nodes and links are connected by some filters and classifiers, which impose on them.

1.3 Dataflow management in an agent

Another special simulation element of NS is the agent. In the process of dataflow, the agent functions as an object to trigger different management policies. With the definition of an agent type, elements of node and link can choose the proper way to manage current dataflow events through a series of API functions. These API functions simulate application data in a real network, and each of them is mapped to the agents' functions. So, an agent is a connection between an application layer and a transport layer. There are many

types of agents, and different packets are carried by different agents and mapped to its transmission policy. For example, the queue and timer expiration managements depend on the type of agent. From this point of view, the agent is the upper key element in dataflow management, while the other two form the lower mechanisms.

1.4 Description of dataflow in trace file

Depending on the mechanism of the three key elements described above, the NS simulator can generate a normal trace file by trace-all (a Tcl script command for NS usage)[4] command. Each line of a trace file describes information of data packets in a very short time interval. Content of a line is divided into 14 parts and they each represent different information. Here is an example of a line abstracted from a trace file.

r 1.812219 2 3 tcp 1 000 ------- 1 1.1 4.2 34 203

The first symbol represents the current state of the packet, namely pushing into a queue, popping out of a queue, retransmitting, discarding, and so on. The second number is current simulation time. The third and fourth items are the numbers of two nodes, between which the current packet is transmitting. The former is the sending node, while the latter is the receiving node. The fifth item is the type of current packet. The sixth item of the line is the size of the packet. The following 4 items represent symbol bits of trace information, which should be enabled for usage. The eleventh item is an option file defined for IPv6. Then the following 2 numbers represent the source and destination addresses of the current packets with the port numbers. The penultimate number is set for compatibility of packet sequence in ns-1[5]. And the last one is the group number of current packet.

A whole trace file consists of hundreds or thousands of such lines. We can get detailed information of a point on the time axis. But if we want to trace consecutive information of specified dataflow from the file, we have to pick several single lines from the whole file and cannot get intuitive result. Due to this shortcoming, we implement a simulation dataflow analyzer to draw the graph of specified dataflow statistic on a single node during a period of time.

2 Design and Implementation of Dataflow Analyzer

During the course of statistics and analysis of dataflow character in the simulation, the load of nodes in network and transmission information of current dataflow needs to be traced. From this request, the dataflow analyzer should fulfill the following tasks: ① To separate different dataflows and trace only one of them; ② To generate statistics of the loss and retransmitted packets, which influence the actual dataflow flux; ③ To get rid of interferential dataflow of the same type, which will not add to actual dataflow flux.

The analyzer finally offers what we need to trace during requested time intervals by outputting a dataflow time-size graph. This graph can give us a direct view of variation and trends of dataflow on a single node.

2.1 Design of analyzer

Due to the above request, we classify the operation of data packets on a node into two classes: 1-tuple and 2-tuple. The analyzer will select the proper statistic a method according to the transition of these two classes.

There are two elements of 1-tuple: $\langle - \rangle$ means the current node generates a data packet and sends it out; $\langle + \rangle$ means the current node receives a data packet.

There are four elements of 2-tuple: $\langle +, - \rangle$ means the current node transmits a data packet; $\langle -, r \rangle$ means the current node retransmits a data packet; $\langle -, d \rangle$ means the current node discards a retransmitted data packet; $\langle +, d \rangle$ means the current node discards the node directly.

Generally speaking, elements of the 1-tuple class represent the end nodes of a link, while elements of the 2-tuple class represent transfer nodes during a whole path. Fig.3 shows the relationship of these six states. To drive the transfer of states, we also need the group number of the packets, discrete time, and some other node information.

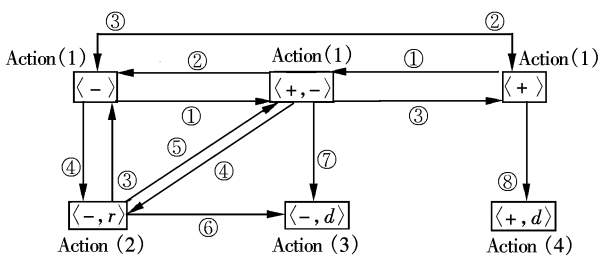


Fig.3 State machine of dataflow analysis

State actions:

Action(1) Log size and type of the data packet and increase sum of the same type dataflow by the size.

Action(2) Log current simulation time, reset timer and add the size of packet to the sum of dataflow

of the same type.

Action(3) Log discarded packet group number. If it is a retransmitted one, subtract the size of packet from the sum of the dataflow of the same type.

Action(4) Log discarded packet group number while keeping the previous sum of the same type dataflow.

Transfer conditions and actions. ① Simulation time is increased, or the node is a transfer node of current packet. ② Simulation time is increased, or the node is the destination of current packet. ③ Simulation time is increased, or the node is the source of current packet. ④ Timer expires, or the queue in the node with retransmission mechanism overflows. ⑤ Pop the head element of node queue and send out. ⑥ Retransmission timer expires, and then the node needs to discard packet. ⑦ The node has no retransmission mechanism, and then the node needs to discard packet. ⑧ Packet transmission time is out or data have errors, and then the node needs to discard packet.

Among these states, two states of 2-tuple $\langle -, d\rangle$ and $\langle +, d\rangle$ will come back to the initial state after the action. The initial state can transfer to three other states: $\langle -\rangle$, $\langle +\rangle$ and $\langle +, -\rangle$.

According to the above state machine, we can easily design the analyzer to parse specified dataflow in a period of time. Given the proper step of each statistica' time, a dataflow graph can be precisely drawn.

2.2 Application of dataflow analyzer in simulation

Firstly, we design a simple network topology to simulate the link state routing protocol in NS. The whole topology is shown in Fig.4 (a). The rtProto protocol that simulates routing protocol will run on it. During the course of path seeking and data transmission, there will be TCP, rtProto and ACK dataflow in the simulation network. The node N_2 transmits data and selects equal cost routing, and acts as the source, destination and transfer node during the whole course. Thus we can obtain multiple states on it and trace one kind of dataflow to prove our analyzer's function. We set the time interval from 0.41 s to 1.40 s, and expect the analyzer to draw TCP dataflow on N_2. Fig.4 (b) represents the result. We preset the link from 2 to 4 to be down at the point of 0.80 s. Then the original two equal cost paths have only one exit, which is 2 - 3 - 4. When such a link is broken it will cause the regeneration of simulation router tables on N_2. The Commands below will describe this circumstance[6]:

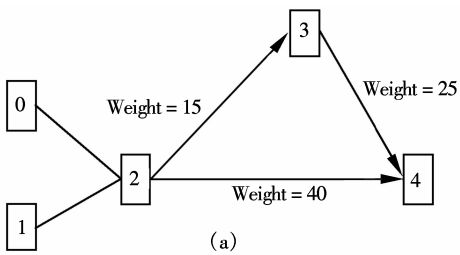
[$ self build-tcp $ n0 $ n4 0.8] set class_0; # The variation routine followed by N_0-N_4 route;

[$ self build-tcp $ n1 $ n4 1.6] set class_1; # The variation routine followed by N_1-N_4 route;

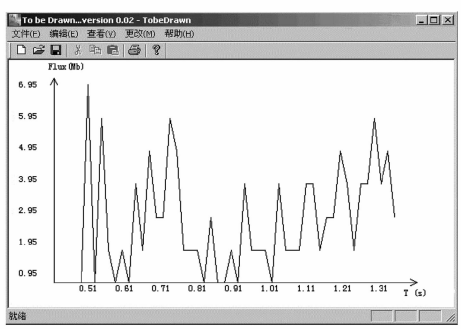
$ ns rtmodel Deterministic {.35.25} $ n2 $ n4;

[$ ns link $ n2 $ n4] trace-dynamics $ ns stdout; # Definition of link broken event from N_2 to N_4;

We can see the TCP flow reaches its nadir at about 0.8 s. The time between 0.8 s and 0.85 s is the delay for N_2 to detect the broken link. Then N_2 stops dataflows on it, and is ready to construct a new routing table. But N_2 finds the equal cost path 2 - 3 - 4 still exists, so the dataflow on it will resume soon after 0.91 s, and keep the average flow from 0.91 s to 1.2 s. At the time of 1.2 s a new routing table has already been constructed, the average TCP flow on N_2 begins increasing. But if only one path exists, the average flow is still lower than flow before 0.8 s. From this graph, we can easily find out the influence of the changes in network on TCP flow and the time when each event happens. Such analysis can help us greatly in flux statistics and control.



(a)



(b)

Fig.4 The application of dataflow analyzer. (a) Network topology and link costs; (b) Graph of TCP flow on N_2 (produced by dataflow analyzer)

3 Conclusion

With the rapid development of the Internet, network simulation will be widely used in the future. Enhancement of capability of network simulation tools and enrichment of means of simulation process and

analysis will help us greatly in network research and provide an important evaluation basis for network optimization and amendment.

References

[1] Fall K, Varadhan K. The NS manual [EB/QL]. http://www.isi.edu/nsnam/ns/ns-documentation.html. 2002-4-24/2002-8-30.

[2] Huang Polly. NS-2 tutorial [EB/QL]. http://netweb.usc.edu/huang/talk/. 1999-8-11/2002-8-30.

[3] Jin Ye. Analysis and implementation of network simulation software[D]. Nanjing: Department of Computer Science and Engineering, Southeast University, 2000.

[4] Fall K, Varadhan K. The NS(v2) simulator workshop[R]. workshop at SIGCOMM, 1997.

[5] Greis M. Tutorial for the network simulator [EB/QL]. http://www.isi.edu/nsnam/ns/tutorial/. 2000/2002-8-30.

[6] Ernest J, Friedman-Hill. Programming using Tcl/Tk [EB/QL]. http://herzberg.ca.sandia.gov/. 1996/2002-8-30.

NS 数据流机制及其解析器实现的研究

金 烨 樊 隽

(东南大学计算机科学与工程系,南京 210096)

摘 要 分析了 NS (network simulator) 仿真器的主要元素,介绍了它的结点、链路以及代理对数据流的处理方式和流程,描述了在脚本文件中识别和采集数据流的方法.基于仿真结果表示的方法,得到了解析数据流的状态,并构造了相应的转移事件和状态机,实现了针对指定流类型统计网络中任意结点流量的分析器,在一个 NS 应用中,通过分析器产生了统计结果,证明了该设计的可行性和有效性.

关键词 网络仿真; NS 仿真器; 数据流分析; 状态机

中图分类号 TP393