Journal of Network and Computer Applications (2000) **23**, 93–107
doi:10.1006/jnca.2000.0104, available online at http://www.idealibrary.com on IDEAL®

A simulated network management information base

Colin Pattinson
School of Computing, Leeds Metropolitan University, Leeds LS6 3QS, UK,
e-mail:C.Pattinson@lmu.ac.uk

(Received 24 September 1999; accepted 17 January 2000)

Initially motivated by the requirement to provide trainee network managers with realistic 'hands-on' experience of network management platform, without the problems associated with access to 'live' network management data, a simulation-based approach has been adopted, which makes use of a production-standard network management platform, interacting with processes ('model agents') representing network entities. The development of model agents which provide simulated network management data, whose values are modified according to some pre-determined pattern allows the user to see a (simulated) network whose behaviour, as represented by the variation in the values of data objects within the management information base (MIB), is virtually indistinguishable from a real network with real network entities.

This paper describes the processes by which these patterns are controlled, offers an example of how the type of network behaviour reported in published measurements can be simulated, and concludes by proposing the use of such an approach in the study of the behaviour of network managers. © 2000 Academic Press

1. Introduction

The near-universal adoption of the computer network as a means to deliver computing resource and information sharing within and across organisational boundaries has placed the network hardware and software in a position which can be described as 'mission-critical'. This widespread reliance on computer networks by organizations large and small, across a broad spectrum of users and application areas has created an awareness of the need for those networks to be managed. Without such management, the network's operation may be compromised by undetected faults, unsatisfactory levels of performance or breaches of security, whilst the organization is unaware of the deployment or usage of the network devices, and cannot determine appropriate mechanisms to recover costs from users. As a consequence of this, a number of technological solutions have been developed to assist the network manager in fulfilling some or all of the tasks required. Typically based around the structures and data definitions of the Simple Network Management Protocol (SNMP), tools are available which allow the network manager to interrogate, control and study the behaviour of network devices, with each device providing status information via a 'management agent', which works in a client-server relationship with the management platform. See, for example, the tools described by Leinwand and Conroy [1].

An example of such a network management (NM) platform, which is characteristic of many such platforms, is the tkined system, developed by the Technical University of Braunschweig, Germany [2]. This is the platform which has been utilized in the work described in this paper, although it is envisaged that the overall approach is transferable to other platforms which offer broadly the same characteristics. The consequential benefits of this, in providing a common evaluation baseline, are described fully later.

The variety of possible tools and techniques which can be encountered by the trainee computer network manager are discussed in a number of text books: Rose [3] and Stallings [4] discuss the details of the protocols and data structures used in NM, with an emphasis on SNMP; Leinwand and Conroy [1] provide practical examples, primarily of 'fault' situations, which illustrate how the information provided in an SNMP management information base (MIB) is used to study and influence network behaviour. However, the written examples and illustrations are deficient in one aspect, they lack the realism of practical exposure to situations which develop and change over time. The work described in this paper was motivated by the desire to study the behaviour of network managers when confronted by a range of operational scenarios, to include both 'correct' operation and the 'fault' situations referred to above. Ideally, this should offer a controlled, repeatable, predictable (to the designer/observer, not the user) and graduated set of tasks. This paper discusses the ways in which network performance can be described statistically and identifies the way in which a network can be modelled in a variety of simulation techniques. The paper then describes a novel approach in which the techniques of simulation* are used to control the data structures and variables of an artificial SNMP MIB, in such a way as to permit the development of 'model agents' whose behaviour and characteristics can be controlled to produce behavioural data identical to that determined by measurement of 'real' devices on 'real' networks.

2. Numerical information as a descriptor of network behaviour — related work

Statistics relating to fault and performance measurements obtained by the NM platform and other network management operations are of interest in a number of ways: they can inform and direct the provision of management/equipment more appropriately; they can act as benchmarks for the assessment of different hardware and software configurations; they can be used, in conjunction with simulation/modelling, as part of the planning process. Statistical results relating

* For the purposes of this paper, the distinction between the terms 'model' and 'simulation' is that used by [1], p. 121, 'The model is a set of rules the simulation needs to follow', thus the simulated network is defined by a set of model agents, each with its own model MIB.

to the performance of a number of network types and topologies have been published over the years [5–7], describing the characteristics of networks under a variety of loading conditions.

These studies reveal that:

- traffic flows can be expressed mathematically (e.g. the network utilization and packet size distributions quoted in [5]);
- such flows can be related to the characteristics of the systems on the network (e.g. the difference in traffic according to whether files are stored locally or remotely [5], the behaviour of file servers as compared to workstations [6] and the different traffic profiles of different end user applications [7]);
- traffic flows are different for each end system [6] and vary over time [6, 7].

These patterns, and the fact that they can be described mathematically, imply that a mathematical model can be developed which would accurately represent the flow of data on the network.

3. Network simulation — related work

Another important tool for the network manager is simulation, which allows the construction of (mathematical) representations of whole or parts of a network. This then allows questions of the 'what if' form to be answered in support of the planning process, or permits studies of the characteristics of particular network design points without the need to obtain and install component parts. Such techniques have been applied to both circuit and packet-switched networks, such as those described by Schwartz [8], which studies, among other topics, the dimensioning of flow control algorithms and the performance of different routing algorithms.

One significant recent use of these techniques has been the study of the role of the Call Admission Control (CAC) process in providing guaranteed Quality of Service (QoS) in multimedia networks. Typical of this work is the way in which Tse and Grosslauger develop a CAC process, and then model that process to evaluate its effectiveness in delivering some QoS on a particular (simulated) network [9, 10].

An example of the use of a 'whole network' simulation, in which users can vary characteristics during operation, is MacDonald's discrete event simulator, cnet [11]. In cnet, protocol implementations are tested on a simulated wide area network whose performance parameters (data loss, transfer rates, etc.) can be varied in a controlled manner. The motivation here is to provide a tool under which protocol implementations can be tested in a controlled environment. Commercial tools (e.g. [12]) are available which simulate a wide range of network components, topologies and behavioural characteristics, and are widely used in the arena of network planning.

Some authors have applied simulation methods to network management systems: Kheradpir *et al*. [13] report on the use of simulation techniques to evaluate network management architectures in circuit switched networks. Perhaps of more relevance to this paper is the work by Sidou *et al*. [14], a methodology for simulating the behaviour of managed objects within an OSI NM environment, specifically the Telecommunication Management Network (TMN). Their objectives are to validate the information model used within the TMN, and to provide a reference configuration against which to evaluate NM tools. Whilst the major motivation for the work described in this paper is to study NM practices and methods, a similar approach could be taken here to develop reference configurations for evaluating SNMP-based computer network management systems.

4. A statistical description of network behaviour

It is clear from the above that a variety of network types, applications and behavioural patterns can be fully described by the use of an appropriate set of performance measures (whether derived from measurement or simulation/modelling) in combination with other data relating to the physical and logical configuration of the network. Indeed, this is the purpose of the MIB concept, which uses just such a data set to represent a network. The work described in this paper takes this process one stage further, by developing a data set which is not related to a ‘real’ network, but instead can be manipulated to simulate the characteristics of any appropriate network type and behaviour pattern. The method also permits other authors’ published figures to be utilized as data sources for the simulation, so widening the range of data sets available.

The opportunity afforded by the use of the MIB’s data constructs allows the method described here to be used in another important area of study, the comparative evaluation of network management platforms themselves. The use of a common set of network management data permits the same simulated network to be ‘managed’ by any network management platform which conforms to the SNMP specifications. This would allow a platform to be evaluated so as to determine its effectiveness in a set of network management tasks, the evaluation being either of an absolute nature (does the platform permit management on the network under particular circumstances and requirements?) or a comparative nature (how does platform A compare with platform B in respect of certain criteria?).

5. Model MIB derivation

The relationship between the ‘model MIBs’ whose development is described in this paper, the network manager and the observer is shown diagrammatically in Figure 1. The NM tool referred to in Figure 1 is the tkined package [2], modified to communicate with our model agents; the SNMP agent is a process which

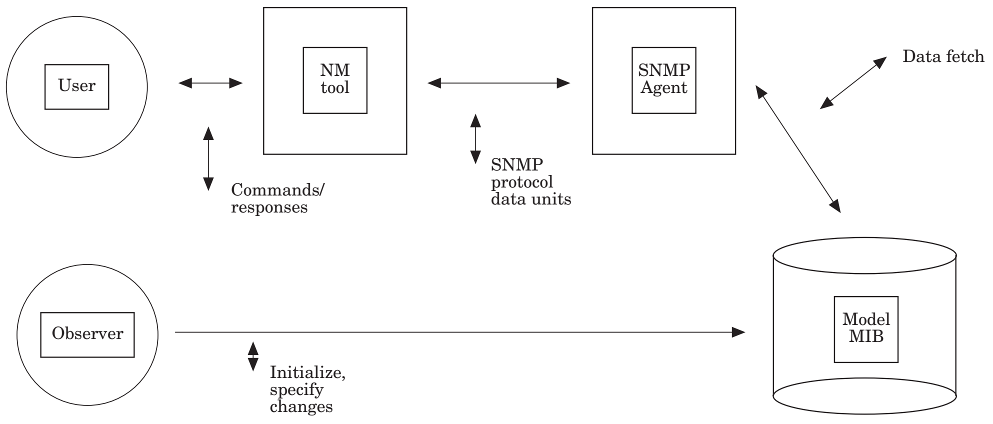

Figure 1. The user’s interface with a model MIB through the same NM tools as for a live MIB, whilst the instructor directly controls the behaviour of the model to simulate activity.

receives and responds to data requests issued by the NM tool, providing object values (data) retrieved from its model MIB, whose construction is described here.

Appropriate variation of these ‘model MIB’ object values can allow a variety of network usage patterns to be simulated, therefore allowing trainees to become familiar with the characteristics of many more scenarios that could be experienced on a ‘live’ network. Such a technique can also be used to simulate fault conditions, with the transition from ‘normal’ to ‘faulty’ behaviour taking place under controlled conditions, whilst the trainee acts in the role of network manager. Central to the method is a way in which the MIB objects’ values can be modified over time. Superficially, this is simply a matter of associating an appropriate statistical function with each MIB object, modifying each using numbers drawn in some manner from that distribution. However, the sheer number of such objects could make such an approach costly to specify; and the layered nature of most communication systems means that it is unrealistic to consider each object in isolation. Instead, advantage is taken of the relationships between objects to more effectively represent the way in which a layered protocol causes consequential effects to ‘ripple’ through a network entity, and to reduce the specification effort.

Rose [3] describes the relationships between MIB data items through the use of case diagrams. This paper shows how a ‘model MIB’ is developed, interpreting case diagrams to ensure correct relationships between data items.

6. Case diagrams

The use of case diagrams provides a pictorial illustration of the relationships between MIB objects. As Rose explains [3, p. 142], a primary purpose of such case diagrams is ‘to avoid defining objects which are derivable from other objects’. In the case of objects with a counter syntax, this means that the objects

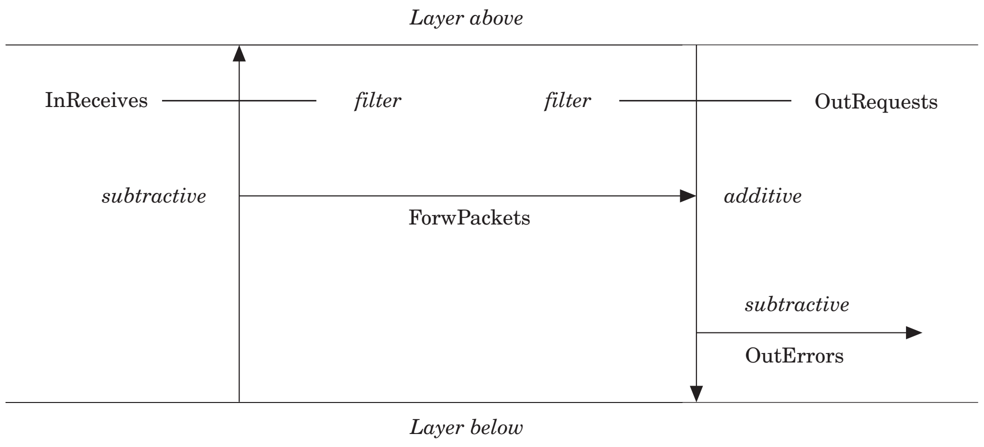

Figure 2. Example case diagram (all case diagrams after [3]).

must not be (easily) related arithmetically'. Stallings [4, p. 153] further describes how the use of case diagrams allows a developer to 'determine if all cases are accounted for—and that there are no unnecessary duplications'.

A case diagram is a visual representation of the additions and subtractions which are necessary fully to determine the value of a particular variable and defines three types of counters: additive; subtractive and filter. For example, Figure 2 shows these relationships:

$$\text{Packets from the layer below} = \mathit{forwPackets} + \mathit{inReceives}$$

$$\text{Packets sent to the layer below} = \mathit{outRequests} + \mathit{forwPackets} - \mathit{outErrors}$$

The diagram also illustrates that *inReceives* packets are passed on to the layer above, and that *outRequests* packets are received from that layer. Each 'set' of MIB counter variables is defined by the use of these diagrams, and the presentation method is effective in its simplicity and clarity. The step of connecting counter flows between layers (for example, to derive a linkage between tcp and ip packets) is usually omitted, but is easy to visualize.

7. Data representations for modelling purposes

In the development of a model MIB, which simulates the behaviour of a network entity by directly manipulating the values held by MIB objects, it is also necessary to determine the means by which objects interrelate, but now the motivation is somewhat different. Whereas in the 'real' MIB, object values are changed as a result of the network entity performing some activity (such as sending or receiving a tcp segment) and the concern of the MIB designer is to ensure that such an activity is 'accounted for' without 'unnecessary duplications', in the model MIB it is important that the consequential results of that operation are recorded in the appropriate MIB objects: for example, sending a model tcp segment would cause one or more model ip packets to be sent, with these in turn producing some model interface activity. Note this example concentrates on the situation in which the data transfer attempt was successful, some data transfer activity is, of course, unsuccessful, and results in the incrementing of error-related counters, rather than the 'data sent' variety described above.

This example implies that the manipulation of model objects will be driven by the behaviour of the user level protocols (for the purpose of this paper, 'user level protocols' include udp; tcp and the Internet Control Message Protocol icmp), such that the overall behaviour of an entity in this simulation is governed by the statistical distributions of the model user data, with appropriate modification to determine relative proportions of errors at lower levels. This allows the development of data sets which model the behaviour of a range of different behaviours, allowing model MIBs to be developed for a variety of performance patterns. It also means that a complete model MIB involves defining the statistics of the udp, tcp and icmp counters, plus the proportionalities required to determine errors, retransmissions, losses, broadcast and multicast packets and frame sizes. It is therefore appropriate to consider the process by which case diagrams can be adapted for model definition.

8. Using case diagrams to develop model MIB data

As discussed above, the emphasis in these models is on the user level behaviour patterns, therefore we directly specify statistical distributions for the protocols associated with the user level, namely udp, tcp and icmp. At this point, it is helpful to separate the incoming counters from those relating to outgoing data, and to consider each separately.

Consider first incoming data, that is received by this entity and which appears at the left hand side of Figures 3–5, all of which will be carried by of the three protocol types noted above. Therefore, the counters relating to received udp traffic (*udpInDatagrams*), tcp traffic (*tcpInSegs*) and each of the 12 *icmpIn* counters excluding icmpInMsgs (see below) are governed by some statistical distribution. However, the case diagram for the udp group (Figure 3) shows that some udp datagrams are discarded without being delivered to the user. We therefore assign statistical distributions to the two counters *udpInErrors* and *udpNoPorts*, and the <u>total</u> count of udp datagrams arriving at this entity is then the sum of the three counters. In the case of the tcp group, the definition is different, here *tcpInSegs* gives the <u>total</u> number of tcp segments received, including those in error, of which there are *tcpInErrs* (see Figure 4). In the model, a statistical distribution is associated with each of these variables. Finally, *icmpInMsgs* is a simple summation of the values of each individual *icmpIn* Counter (Figure 5).

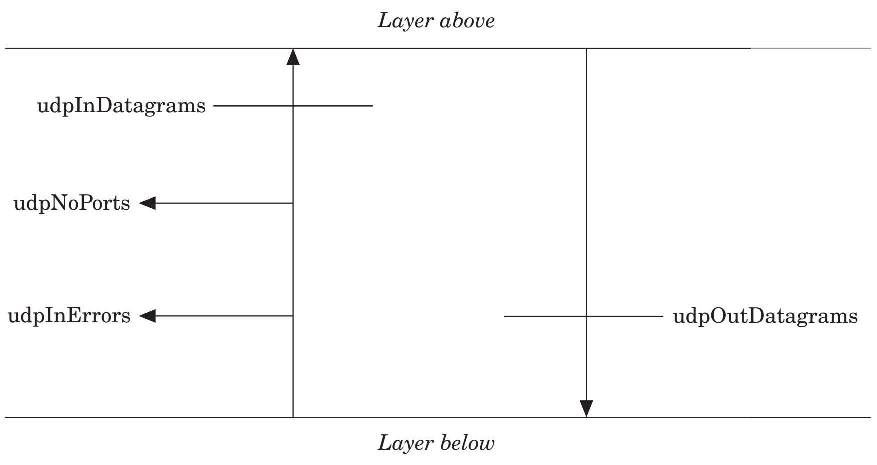

Figure 3. Case diagram for udp counters.

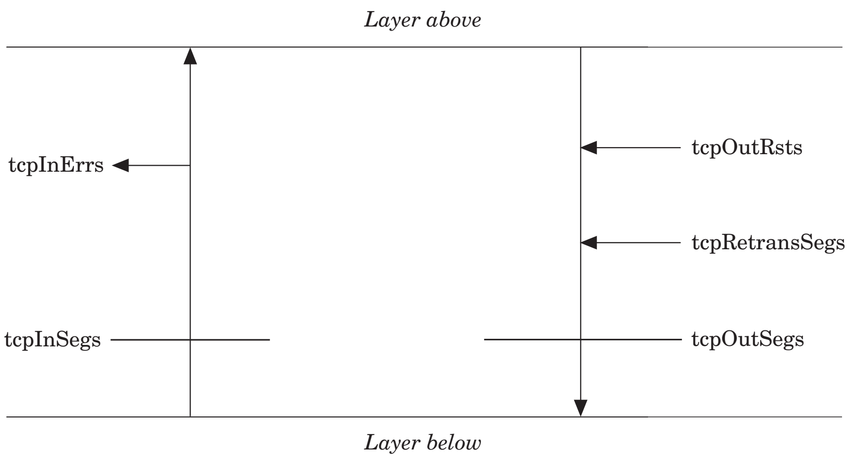

Figure 4. Case diagram for tcp counters.

Now, noting that one or more of the user level protocol data units will be carried in one or more ip packets, we can derive the value of ipInDelivers at any time to be the sum: (icmpInMsgs * *icmpSize*) + (tcpInSegs * *tcpSize*) + ((udpInDatagrams + udpNoPorts + udpInErrors) * *udpSize*), where *udpSize* (*tcpSize*, *icmpSize*) is the mean number of udp (tcp, icmp) packets carried by each ip packet—in many cases, these values are unity, i.e. one 'user-level' packet per ip packet.

Reference to the appropriate Case Diagrams for outgoing data (the right hand side of Figures 3–5) shows that udp traffic is represented solely by the counter *udpOutDatagrams*; tcp traffic is the combination of *tcpOutSegs* (the user's tcp traffic), *tcpOutRsts* and *tcpRetransSegs* and icmp traffic is the summation of the

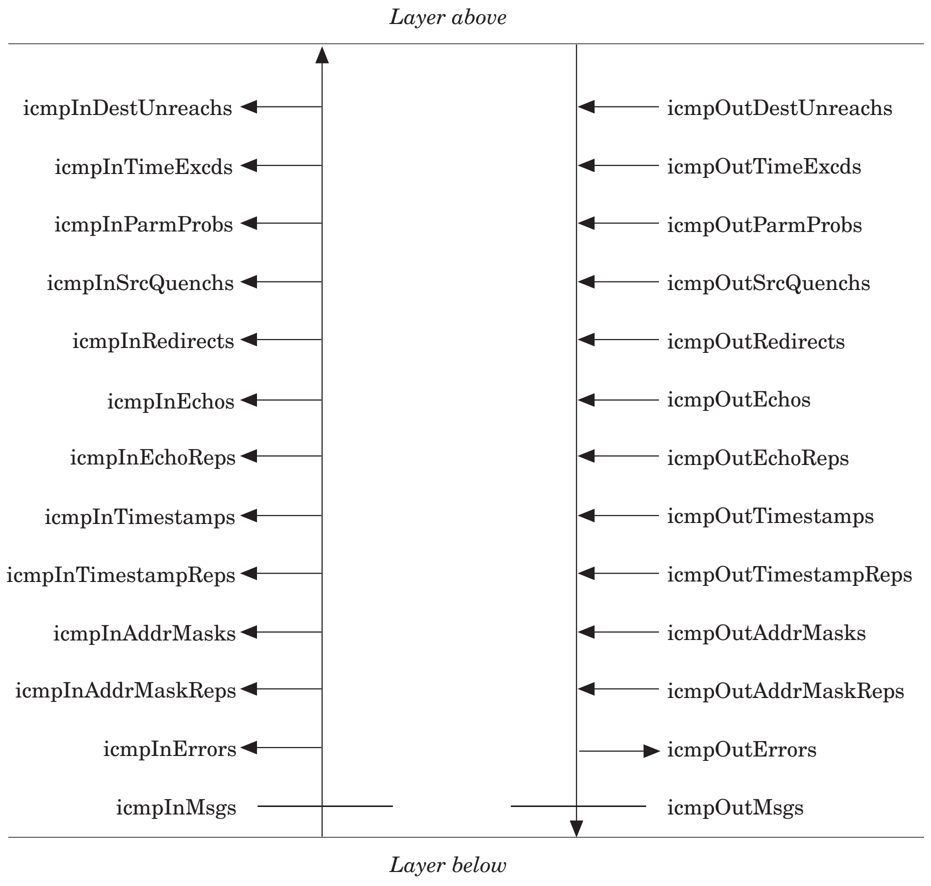

Figure 5. Case diagram for icmp counters.

counters for each *icmpOut* message type, minus a count of those transmissions which were unsuccessful (*icmpOutErrors*), this sum giving a value for the counter icmpOutMsgs. Incorporation of size factors as above produces a value for the total number of outgoing ip packets generated by these three protocols, which is the MIB value ipOutRequests.

At the cost of determining statistical information for those variables in *italics* above, we have arrived at a means of mathematically representing the number of ip packets delivered to and received from the user layers of our model network entity. The Case Diagram for the ip group within the MIB (Figure 6) then provides guidance for the next stage, in which some departing ip packets are discarded due to lack of resource or no route being available and some are fragmented, either successfully or otherwise. This requires determination of values for *ipOutDiscards, ipOutNoRoutes, ipFragOKs* and *ipFragFails* which are subtracted from ipOutRequests, and *ipFragCreates*, which is added (with

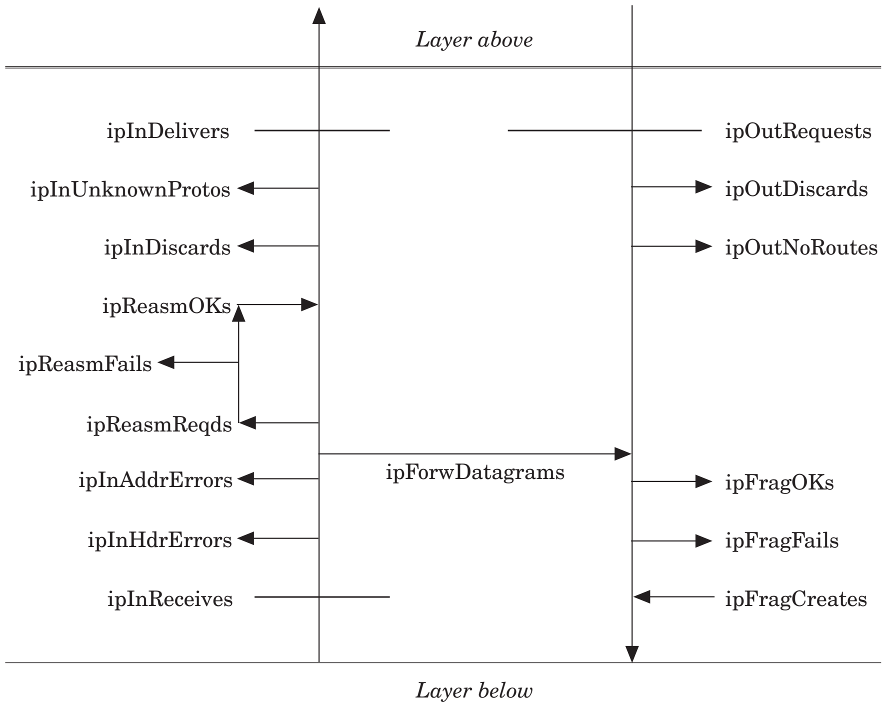

Figure 6. Case diagram for ip counters.

ipForwDatagrams—see below) to ipOutRequests to arrive at a count of the total number of ip packets delivered to the layer below. A similar operation on the arrival side involves the determination of the relevant additive and subtractive counters for arrivals, to derive a total number of received ip packets, ipInReceives. For modelling purposes the 13 values required at this stage are determined by identifying probabilities for each activity, which act as multipliers of ipInDelivers/ipOutRequests according to their presence on the incoming/outgoing side of the case diagram.

Finally, we refer to the case diagram for the interfaces group (Figure 7) which shows that outward bound ip packets are passed to an interface device, where they are delivered either as unicast or multicast packets, or discarded due to errors of lack of resource; and that incoming packets are also either unicast or multicast, with each interface discarding some packets on arrival due to error, resource limitation or unrecognized protocols. Distribution between the relevant counters is again achieved on the basis of probabilities used as multipliers as in the ip case. It is also necessary to allocate relative weightings of traffic between interfaces in those cases where more than one interface exists. Appropriate choice of the incoming and outgoing packet sizes can also influence the size of frames sent and received by this model entity.

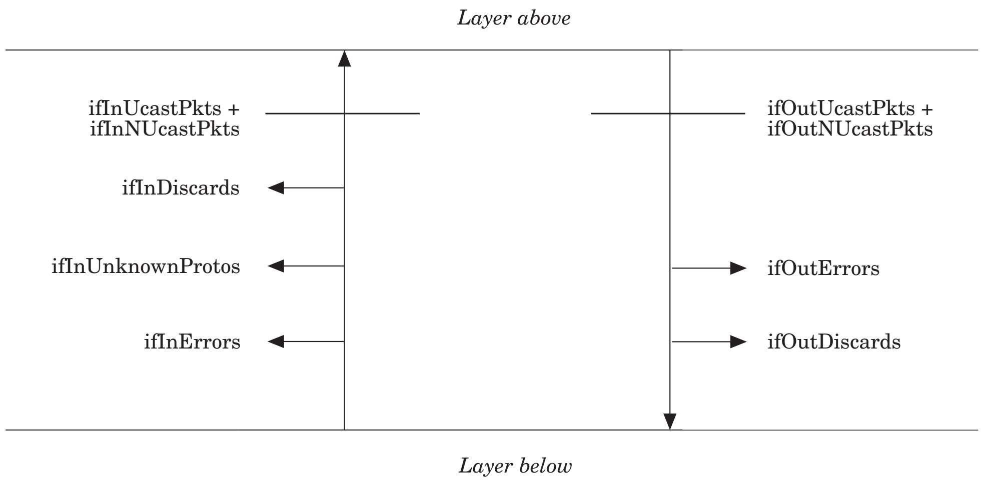

Figure 7. Case diagram for if counters.

In conclusion, then, the model comprises:

- Specification of the statistics relating to user-level protocols, (4 udp counters; 9 tcp counters and 24 icmp counters).
- Allocation of probabilities at the ip level (13 variables).
- Weightings of traffic where more than one interface is in use (n variables, where n>1=number of interfaces).
- Allocation of probabilities at the interface level (9 * n variables, n=number of interfaces).
- Selection of mean packet sizes for incoming and outbound packets (2 variables).

Each device in the model network is represented by a process which is able to communicate with the NM platform, each such process providing object instance values from its own model MIB. Updating of the model MIB then proceeds by each of the above user-level variables being incremented by some value drawn at random from the specified distribution (e.g. normal, Poisson, step increment, random [15]) at predetermined time intervals. These updated values are then used to calculate values for other variables according to the allocated probabilities and weightings, thus allowing the 'ripple' effect described above.

This paper has concentrated on the counter variables, as they are the most important for the purposes of modelling traffic flows. There are also a number of other variables in the standard MIB, providing information such as the system name and location, as well as tables relating to routing and tcp connections. The former are handled here by declaring them as static data items, the latter are supplied as text data, read at regular intervals, this allows such tables to be varied over time in a manner which reflects the behaviour of real tables. These table variables are of particular relevance in the development of fault scenarios which we have utilized in the development of NM training modules, in which trainee users are tasked with the detection and isolation of fault conditions. This application was described in an earlier paper [16].

9. An example

This approach to the definition of packet counts permits the *overall* volumes of data transmitted and received by model nodes to be governed by distributions representative of actual information exchanges, for example, the network measured by Khalil *et al*. [6] had the following characteristics:

- non-uniform distribution of traffic between nodes;
- a mix of tcp and udp protocols;
- packet size distribution which is trimodal for tcp, bimodal for udp;
- domination of total network throughput by udp packets;
- Poisson-governed inter-arrival times at busy periods.

A model with similar characteristics can be generated by the following operations:

- *Non-uniform distribution of traffic between nodes*. This is achieved by setting up each model node with different mean incremental step sizes, so that whilst the distribution *function* remains the same, the *values* drawn at random from that distribution are of different scale, and by provision of table information representing a communication matrix in which some nodes have more links than do others, therefore defining a situation in which ‘busier’ nodes send/receive more traffic to more destinations.
- *A mix of protocols, comprising tcp and udp protocols*. The model will require the initialisation and control of counters relating to that traffic (*tcpInSegs; tcpOutSegs; udpInDatagrams and udpOutDatagrams*).
- *Packet size distribution which is trimodal for tcp, bimodal for udp*. The SNMP variables which indicate packet size are *interfaces.ifTable.ifInOctets* and *interfaces.ifTable.ifOutOctets* therefore it is these variables which are controlled, with a distribution representing the composite values (with appropriate scaling) of the two components.
- *Domination of total network throughput by udp packets*. This is represented by selection of scaling values for *tcpInSegs; tcpOutSegs; udpInDatagrams* and *udpOutDatagrams* such that the aggregate value of the first pair is appropriately smaller than the second pair.
- *Poisson-governed inter-arrival times at busy periods*. By restricting the model to operation at busy periods only this requirement is met by controlling

the increment actions on the relevant packet arrival (departure) counters (*tcpInSegs; tcpOutSegs; udpInDatagrams* and *udpOutDatagrams*) to accord with an overall Poisson distribution.

In this case the *icmp* subtree is not required, though data values could be supplied for completeness, so that the actual model centres upon the six variables noted above, with the remaining settable variables acting as background data to complete the overall picture. The original report also described traffic produced by nd (network disk) a protocol used to support paging/swapping by diskless workstations across the network and not referenced by SNMP data items, the portion of traffic carried by nd in the network observed by Khalil *et al*. [6] is carried here by udp.

10. Virtually indistinguishable from a real network

We believe that the system as currently deployed is sufficiently realistic to possess the features of a real network, and therefore provides users with an experience comparable to that of monitoring an actual network of the same characteristics as that in the simulation. We wish to collect empirical evidence to fully verify this claim, however anecdotal evidence gives us cause for satisfaction. Some users (final year and postgraduate computer communications students) have initiated physical searches for machines they have ‘seen’ using this tool. Some of the more advanced fault finding tasks, which involve on screen messages from unhappy ‘network users’ after some ‘fault’ has caused them to lose their ‘service’ create a similar reaction.

Finally, this tool has been used by some six part-time students whose jobs involve the use of similar network management tools on real systems, all have commented positively on the realism of the simulation.

11. Potential and applicability

The interfacing a real NM platform with a set of model agents provides us with the opportunity to explore the behaviour of network managers, in respect of the use they make of standard NM tools, both in ‘normal operation’ and in response to some problem situation. By developing a set of model agents, presenting MIB data representing light/heavy usage; significant variations of peaks and troughs, requiring possible reallocation of resource; and fault conditions requiring diagnosis and rectification, we can monitor the ways in which a network manager interacts with their NM platform. This monitoring provides us with the opportunity:

- to permit network managers in a practice situation to review their actions after an activity, and determine whether those actions were efficient or appropriate to the task in hand. This is achieved by means of a record and playback

mechanism, where the user can replay their activities after the event. Further, the fact that the same scenario can be repeated (simply by reactivating the model agents) means that the impact of alternative actions, suggested by the review, can be investigated;
- to identify any sequences of 'commonly requested' MIB object instances, thereby offering the opportunity for requests for those sequences to be generated automatically, rather than relying on the user making each individual request;
- to provide insights into the availability, accuracy, timeliness and appropriateness of MIB objects as currently constituted, thus providing the opportunity to review agent, NM platform and MIB structures in the light of these findings;
- to undertake a comparative review of the 'usability' of NM platforms in the light of behaviour patterns which a specific network is expected to produce, for example, certain design factors within one NM platform might be more appropriate for the management of a small, relatively fault-prone network, but not for a large network in which performance or routing issues are paramount, in which case a different platform is more appropriate.

Experience to date, using this method in support of teaching on undergraduate and postgraduate courses, has shown the soundness of the overall method, and gives confidence as to the realism, applicability and usefulness of the overall approach.

12. Conclusions

A mechanism has been developed which allows the exploration of network monitoring using the principles of the SNMP MIB-II, but using simulation techniques to allow networks of model agents to be developed for the specific purpose of identifying particular behaviour patterns which can be presented to the user in a controlled fashion. This has a number of educational/training benefits, in that trainee users can be led through a series of network management tasks which can be controlled and replicated, so allowing them to be linked with a theoretical study of the methods and tools required to monitor a computer communications network. The use of such a mechanism, in conjunction with suitable instrumented NM platforms, will now be incorporated into a further project, in which we intend to study the activities of the (human) network manager.

References

1. A. Leinwand & K. Fang Conroy 1995. *Network Management: A Practical Perspective 2nd Edition*. Reading, MA: Addison Wesley.
2. Tkined 1998. At URL: *http://wwwhome.cs.utwente.nl/∼ schoenw/scotty.*
3. M.T. Rose 1994. *The Simple Book: An Introduction to Internet Management 2nd Edition*. Engelwood Cliffs, NJ: Prentice Hall.
4. W. Stallings 1996. *SNMP, SNMPv2 and RMON: Practical Network Management 2nd Edition*. Reading, MA: Addison Wesley.

5. E. Drakopoulis 1993. *Performance and traffic analysis of a network-based distributed system. Computer Communications* **16**, 155–167.
6. K.M. Khalil, K.Q. Luc & D.V. Wilson 1990. *LAN Traffic Analysis and Workload Characterization*. In *Proceedings 15th IEEE Conference on Local Computer Networks*, 112–122.
7. W.T. Marshall & S.P. Morgan 1985. *Statistics of Mixed Data Traffic on a Local Area Network. Computer Networks and ISDN Systems* **10**, 185–194.
8. M. Schwartz 1987. *Telecommunication Networks Protocols, Modeling and Analysis*. Reading, MA: Addison Wesley.
9. D. Tse & M. Grosslauger 1997. *Measurement-based Call Admission Control: Analysis and Simulation.* In *Proceedings IEEE Infocom '97, Kobe, Japan, April 1997*, Vol. 3, 981–989.
10. M. Grosslauger & D. Tse 1997. *A Framework for Robust Measurement-Based Admission Control*. In *Proceedings ACM SIGCOMM 97, Cannes, France, September 1997*, Vol. 27, 237–248.
11. C. MacDonald 1999. *The cnet networking simulator website*. Found at URL: *http://www.cs.uwa.edu/au/pls/cnet*
12. Mil3, Inc, 2000. *Opnet modeler product description website*. Found at URL: *http://www.mil3.com/products/modeler/home.html*
13. S. Kheradpir, W. Stinson, R. Chipalkati & G. Bossert 1991. *NEMACS: The NEtwork MAnagement and Control Simulator*. In *Proceedings IEEE Globecom '91* 30A.2.1 - 5.
14. D. Sidou, S. Mazziotta, R. Eberhardt 1995. *TIMS: a TMN-based Information Model Simulator, Principles and Application to a Simple Case Study*. In *Sixth International Workshop on Distributed Systems: Operations & Management*. Ottawa Canada: IFIP/IEEE.
15. N.A.J. Hastings & J.B. Peacock 1975. *Statistical Distributions.* London UK: Butterworths.
16. C. Pattinson & A. Dacre 1998. *A Network Model for Network Management Teaching*. In *Proceedings of 3rd Australasian Conference on Computer Science Education*. New York USA: ACM, 62–66.

Colin Pattinson is a senior lecturer in computer communications, and head of the Computer Communications Research Group at the School of Computing, Leeds Metropolitan University, Leeds, UK. He received his BSc degree in Computational Science from the University of Leeds in 1982, and the degree of PhD, also from the University of Leeds in 1986. His research interests include the use and development of network management systems and network performance issues. He is currently involved in teaching network management modules at BSc and MSc levels, as well as supervising three doctorate students.