Integrated Network Management, II
I. Krishnan & W. Zimmer (Editors)
Elsevier Science Publishers B.V. (North-Holland)

Combining Knowledge-based Techniques and Simulation with Applications to Communications Network Management

Padhraic Smyth*, Joseph Statman*, Gordon Oliver* and Rodney Goodman**

*Communications Systems Research, Jet Propulsion Laboratory 238-420, 4800 Oak Grove Drive, Pasadena, CA 91109, USA

**Department of Electrical Engineering, California Institute of Technology, 116–81, Pasadena, CA 91125, USA

Abstract

In this paper we discuss the idea of combining simulation and knowledge-based techniques for the purposes of designing a communications network management system. While rule-based expert systems have considerable potential in terms of network management applications, the construction of such systems is a complex and expensive task. For a *new* network where no expertise is available, the problem is particularly difficult. We describe here an approach which combines both simulation techniques and the knowledge-based paradigm to solve this problem of designing "expert" systems without an expert. In particular we outline how this approach is being applied to a GPS (Global Positioning Satellite) Packet Radio Datalink network. The main conclusion is that the benefits of building a communications network simulator can be improved significantly by the addition of knowledge-based techniques to the modeller's arsenal of tools.

1. Background and motivation

How does one design an intelligent network management system for a communications network which has not yet been built or fielded? Obviously one would rather not wait several years until operators accumulate sufficient experience that a rule-based expert system can be built using their knowledge. In particular, consider the problem of designing such a network management system for a *new* type of network, perhaps one which exhibits novel characteristics. In this scenario, the performance of the network under *normal* operating conditions may only be partially understood, let alone the problem of trying to characterize the behaviour of the network under *abnormal* conditions (such as traffic overloading, network failures, etc.).

An example of such a network is the proposed GPS (Global Positioning Satellite)

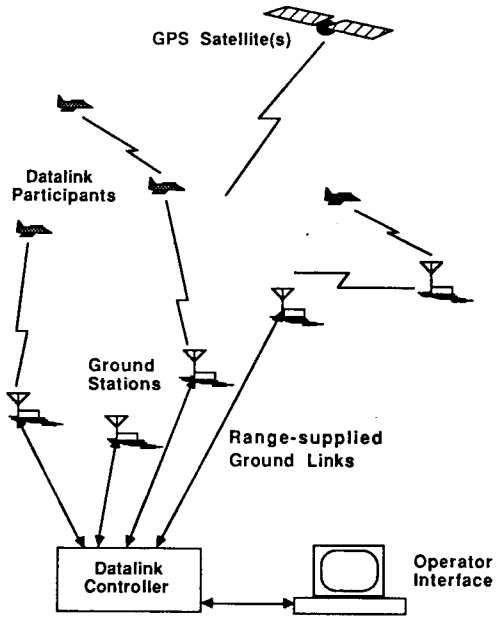

Figure 1. A simplified block diagram of the Datalink network.

Range System Datalink, currently under development for the Range Applications Joint Program Office (RAJPO) [1]. The GPS Datalink network provides two-way digital communications between range participants and a host range command and control center. A simplified block diagram of the network is shown in Figure 1.

Its primary purpose is to support GPS position data collection from range participants to the control center in real-time. The system is implemented as a packet radio network utilising a TDMA participant allocation scheme, with a distributed network routing algorithm. The network supports 200 kbits per second message throughput with data being transmitted in 736-bit packets. Participant rates range from 10 packets per second, down to 1 packet every 10 seconds. Data transmission occurs on an L-band link using 1.6Mhz of bandwidth. The network features both frequency diversity (to combat multipath interference) and spatial diversity (to combat terrain blocking of line of sight reception).

The goal of this paper is not to describe the Datalink system in detail. Rather, we

will use it as an example of a complex network whose behaviour is difficult to model in analytical form. The traditional approach to this problem of *characterizing network performance* involves modelling the network in an idealized manner and simulating its behaviour in software. We argue that it is beneficial to go one step further and combine network simulation with knowledge-based techniques. We will show that this approach is advantageous for three important reasons:

1. Knowledge-based techniques can be used to interpret and manage the large volume of data which simulations typically generate, in particular, addressing the problem of correctly classifying the context in which network performance parameters are measured.
2. Machine learning algorithms can identify both performance management rules and fault management rules.
3. The simulator can be used as a test-bed for developing a closed loop network monitor and control system.

2. The benefits of combining simulation and artificial intelligence

The marriage of simulation and knowledge-base technologies is a relatively recent phenomenon [2]. Typically this marriage works in two ways. The first involves using simulation capabilities *within* AI applications, such as a fault diagnosis system — roughly speaking, this type of approach is referred to as *model-based reasoning*. By using a causal model of the system one can in principle reason "from first principles." Given the nature of this paper we will not dwell on this particular approach here, although it obviously has considerable potential in the communications domain [3]. The second approach, and the one which we focus on in this paper, is that of using knowledge-based techniques *within* a more traditional simulation model, i.e., to extend the scope of standard simulation techniques.

To quote Widman and Loparo [2] "simulations generate much more data than they do information." In this paper we will describe the application of knowledge-based approaches to make better use of standard simulation models. The overall objective is the effective use of artificial intelligence as a means of extracting more information from the simulator. In particular, for the Datalink network, we have three primary goals to meet:

1. **Performance characterization**: In addition to answering the basic types of "what-if" questions about the network, we wish evaluate in detail the network performance by studying its transient behaviour, i.e., in a distributed network of this nature we need to evaluate how performance parameters such as bandwidth utilization and burst packet losses are affected by "environmental" factors such as the rate of participant exit and entry to and from the network, terrain blockage, etc. It is essential to transform the raw data into useful information, a function that

cannot be performed manually if we are to obtain statistically reliable amounts of data concerning the events of interest. In addition, since by any definition this abstraction process is knowledge-based (i.e., is more heuristic than algorithmic in nature), it makes sense *not* to implement this function as part of the simulator itself, but rather as an external module.

2. **Fault management and prediction**: We would also like to use the simulator if possible to generate rules for network management, i.e., one would like to be able to give the operator a set of rules of the form "if parameter X deviates by more than y units then event z will probably occur in w minutes from now." This information could either be in the form of text in an operations manual or on-line as part of the system interface. It is conceivable that one could generate these rules manually, by manipulating the simulator until one effectively becomes an expert on how the model works. Note that while the model-designer is in principle an expert regarding its operation, the actual dynamics and temporal characteristics of a model through its state-space can be very difficult to predict, even if the basic rules of behaviour at the component level are relatively simple. Hence, it makes much more sense to take advantage of the wealth of machine learning techniques currently available to *automatically* induce such rules from the data. This marriage of machine learning and simulation appears not to have been tapped, yet the potential gains are significant: real data is usually costly to obtain for learning algorithms and the opportunity to obtain new data is often not available. The limits on the quality of information obtained is in principle limited only by the quality of the simulation model itself.
3. **Real-time network monitor and control**: One might imagine that if one has characterized the performance of the system (and developed prediction rules, etc.), that building a real-time monitor and control system would be quite straightforward. Unfortunately this is not the case. The real-time nature of the problem makes it quite difficult to extrapolate the rule-based expert system technology from the "classic" passive advisory systems such as Mycin [4] to active, real-time control loops. Truly real-time expert systems are difficult to construct and successful field implementations of this approach are relatively rare to date. Hence, for a dynamic system such as the Datalink, where critical events may develop in the time-frame of a few seconds, the implementation of an efficient and robust network controller is quite challenging. The simulator is an ideal test-bed for experimentation. As with machine learning, gaining access to a real system in order to calibrate and test a network control algorithm is often neither practical nor economical. In the case of the Datalink, the system will not be in production until mid-1991 or beyond and the early field systems will be somewhat scaled down versions of the full-fledged version.

Hence, the value of the simulator to the Datalink project extends significantly beyond the traditional benefits of simply answering "what-if" questions and obtaining

performance curves. In the remainder of the paper we will describe in more detail each of the three components outlined above. First, however, we will give a brief overview of the overall functionality of the simulator and a description of the proposed architecture of the real-time network management system.

3. System architecture

3.1 Description of the Datalink simulator

Since the Datalink is a TDMA system, it lends itself to discrete event simulation at the time-slot boundaries. By choosing this level of granularity we effectively choose to ignore any sub-slot timing effects which may occur (analysis indicates that timing and synchronization should not be a factor in network performance). In addition, with the slot boundary scheme, we choose to simulate at the packet level rather than at the bit level. This is very much a practical consideration since all of the network algorithms effectively operate at the packet level (once each participant decodes the convolutionally encoded packet in hardware). The simulation model has been coded in C and implemented on a Sun Sparcstation. The most complex portion of the code involves modelling the central control facility, since this part of the algorithm must perform all the book-keeping tasks associated with TDMA slot assignment, uplink message scheduling, broadcast messages, etc.

We initially intended to use the simulator for two primary purposes. The first purpose was to answer specific detailed "what-if" questions, such as the ability of the distributed routing algorithm to equally distribute traffic between relayers in accordance with participant duty cycle limits. The process of evaluating these "what-if" questions is currently under-way. The second primary purpose of the simulator was to characterize network parameters by running large-scale simulations and collecting statistical data, e.g., plotting bandwidth utilization as a function of participant entry or exit activity. However, it quickly became apparent that this latter process was somewhat more involved than simply keeping track of various numerical parameters. In fact, in order to arrive at sensible results, it would be necessary to tag each measurement with some type of *context indicator*, since the conclusions which can be drawn tend to be very sensitive to context. For example, short bursts of packet loss can occur in a variety of situations, and in particular they depend on the prior history of events immediately preceding the burst. For the simulator itself to track, correlate, and reduce all this data, seems like an unreasonable approach. After all, the simulator is accurately reflecting the real system, which itself will generate relatively uncorrelated low-level information. Hence, it was clear that a knowledge-based real-time "context-classifier" would considerably enhance the overall information output of the simulation model, and indeed would be necessary for many of the scenarios of interest. A further requirement on this monitor was that it would need to run in real time (since the simulator

runs roughly in real-time) if it was to be useful.

At this point in the discussion it is instructive to take a closer look at the requirements for a real, field-implemented network monitor and control system for the Datalink, i.e., an operational network management system, the development of which was initially unrelated to the simulation modelling.

3.2 Designing a real-time network management system

As mentioned earlier, the design of a real-time network management system is considerably more difficult than simply implementing a rule-based expert system – rule-based systems are most typically implemented in an *advisory* configuration with no concept of temporal reasoning involved, and, despite claims to the contrary, little work has been done on the more difficult and practical problem of developing truly *autonomous, active* expert systems which can operate in real-time with minimal operator intervention. Based on our previous experience with network management problems [5], we divided the Datalink problem into three parts:

1. **Performance management:** Monitor network performance and adjust network parameters (such as relay-depth and slot assignment rates) in real-time to maximize the probability of correct packet reception from each participant.
2. **Fault management:** Detect when network behaviour deviates from normal and identify appropriate corrective action.
3. **Operator interface:** Implement an intelligent user interface to the network operator in *both* directions, i.e., both interpret and condense incoming data and present it in an informative manner, and filter operator inputs by checking for consistency and providing advisory feedback — note that this advisory component is but a small part of the overall system and is very much context-driven.

In Figure 2 we show a diagrammatic representation of this architecture. Note that this general architecture of a dedicated real-time context classifier, with modular expert sub-systems, is quite generic for real-time monitor and control applications.

4. Bridging the gap: from simulator to real-time system

Initially we intended to develop the real-time monitor and control system using a fielded Datalink system as our testbed. The simulator development was considered a relatively separate activity. However, as should be obvious to the reader at this point, both problems have much in common. Hence, using the simulator as a testbed became the approach of choice. We outline the major steps involved:

4.1 The simulation model:

We built a standard simulation model, using the C language. Given the experimental nature of this project, we chose C as the base language for the simulator rather than using a special-purpose simulation package. The two primary reasons for doing this were that first we needed a high degree of flexibility in our software architecture and wanted to maintain control over all portions of the code, and secondly, we wanted to customize the code for speed, so that the simulator would run in as near real-time as possible. This latter requirement is dictated by the fact necessity of real-time simulation if we are to use the simulator as a testbed for developing real-time network management prototypes. The simulation model was implemented in a modular fashion, with the user interface (input syntax, output displays, etc.) being as removed as possible from the actual network model. This approach facilitates both standard manual interaction with the model (via mouse-driven display windows, etc.), and also *automated* interaction, where external programs (such as the knowledge-based techniques to be described below) can interact with the model in a non-supervised manner. In this latter mode, the user-friendly interface modules can be switched off and all simulator output is written directly to a UNIX file. In this way the simulation model serves both as a useful "what-if," user-friendly tool, and as a realistic "low-level" simulation of the actual network.

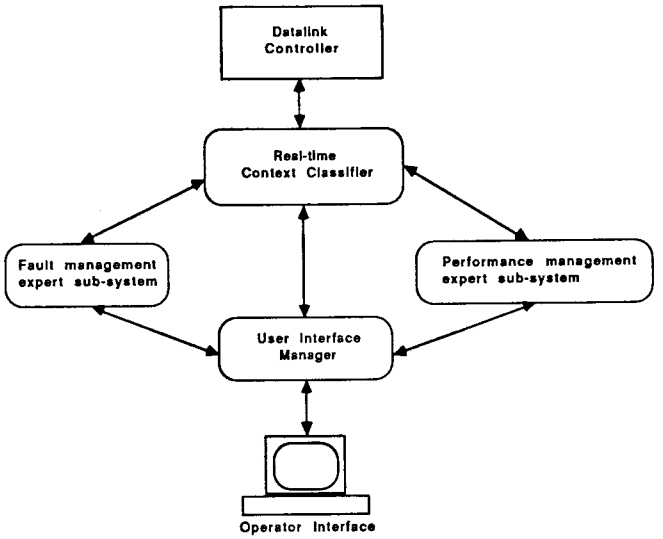

Figure 2. A block diagram of the real-time network manager prototype system.

4.2 Knowledge-based simulation analysis:

As we outlined earlier, building a complicated simulation model can result in large volumes of data being generated. In our case we simulate at the packet rate of 330 packets per second in near real-time. Because the routing algorithm is distributed, each of the N participants in the network computes cost and routing information every time a packet is received — these cost tables are computed locally by each participant based on distance, signal quality, and duty-cycle parameters. Typically the number of participants, N, ranges from 5 to 25, although in principle the network can support up to 200 participants. There are a variety of special routing features, such as the use of alternate ground relays, to achieve both spatial and frequency diversity. In addition the Data Link Controller (DLC) operates continuously, aligning downlink packets, managing the TDMA slot assignment table, scheduling uplink packets, etc. Hence, over a typical period of interest of say 30 minutes the simulation model can generate a huge amount of data. The primary characteristics of this data is that the vast majority of it is non-interesting — however occasionally a sequence of noteworthy events will occur.

The questions we are interested in answering are typically of the form "what is the average number of packets lost in succession (burst losses) when network routing changes occur?" In a more abstract sense we are interested in the impact of so-called *environmental* variables on *network* variables, where environmental variables are loosely defined as those parts of the simulation which strictly speaking are not part of the network model itself, i.e., the number of participants, the dynamics of the participants, terrain models, channel models, network entry/exit patterns, etc. The key point to note is that there are a large number of possible states or "contexts" for the network to be in. In addition, when one factors in the time-dependencies which are implicit in the operation of the network (there are unavoidable delays of up to 10 seconds involved with many network state changes), the problem of analyzing the data output from the simulator becomes formidable.

The approach we take is to effectively kill two birds with one stone: we are developing a dedicated "context classifier" as described earlier to operate in real-time on network information. This prototype serves the dual purpose of enhancing simulation analysis, *and* of providing a major component of the real-time network controller. We are developing this context classifier with a higher-level language than C using one of the more recently developed commercial C-based expert system shells. The trend in expert system development tools has been towards more general-purpose, flexible languages, typically C-based object-oriented systems. The era of special-purpose languages, such as Lisp and dedicated hardware for AI applications, has been largely replaced by a more practical, software-engineering attitude in the AI industry. The newer shells typically are quite portable and robust, and most importantly, can be integrated easily with more standard languages. These developments imply that it is now practical to combine large-scale simulation projects with knowledge-based analysis systems using the type of approach we are describing here.

4.3 Learning algorithms for simulation analysis:

Consider that the trajectory of the network through its state-space is somewhat akin to the behaviour of a Markov model. Local regions of the state space are deterministic, reflecting the finite-state machine nature of the underlying network algorithms, while the gross structure of the model is stochastic, reflecting the underlying influence of the environmental variables. In general, it is difficult to characterize the trajectories or dynamics of the model over time since the system is non-linear and quite complex. As mentioned earlier, much of this knowledge of network operations comes only through experience, either by interaction with the simulator, or by direct experience with the real system.

An alternative, and far less costly approach, is the use of automated learning algorithms which exercise the simulator, as a human would, to accumulate expertise over time. Recent work by Smyth and Goodman [6] has described an information-theoretic approach to the problem of finding the most important set of rules from empirical data — in particular their algorithm (the ITRULE algorithm) can find the most informative state transitions in a Markov model, given data samples from the model [7]. The applicability of this style of information theoretic learning to simulation shows great potential, yet these techniques are relatively new and untried. For example, most learning techniques are based on the assumption that the data is initially presented and no further data is available — the idea of an algorithm which could control the inputs to the simulator, and hence generate the type of data it requires to improve its learned knowledge of the system, is quite interesting but relatively unexplored. For the Datalink project we are using the aforementioned ITRULE algorithm for identifying both performance and fault diagnosis rules.

4.4 A network management prototype using the simulator as a test-bed:

As outlined earlier, we can develop a prototype real-time network manager, using the simulator as a test-bed for the development process. The network manager must perform three primary functions: (i) performance management, i.e., choosing network parameters such that performance is optimized, (ii) fault management, both diagnosis, and prediction based on component degradation, and (iii) operator support and user-interface, i.e., the implementation of an intelligent front end which can manage the information flow between the operator and the Datalink central controller. We will not dwell at length on this aspect of the project, except to note that the context classifier described earlier is an integral part of the design, i.e., the events list as output by the classifier is read by a prioritization mechanism which in turn produces a stack of prioritized events for each expert subsystem to deal with. In this manner the expert modules can continuously focus on the most important problems. The development of this part of the system will be quite complex as the actions of each of the expert "agents" must be coordinated such that conflicts do not occur.

5. Conclusions

We have argued in this paper that the combination of simulation modelling with knowledge-based techniques has significant practical advantages. This is not to say that all modelling problems are best pursued in this manner. Nonetheless, it seems quite likely that with the proliferation of large networks, which exhibit complex non-linear characteristics, that this approach may often be worth the extra investment. In particular, we have identified three primary areas where knowledge-based techniques can profitably be used: (i) as an information management aid for improving the simulation analysis of complex models, (ii) in the use of learning algorithms for identifying underlying causal relationships in the model which can only otherwise be discovered by expensive trial and error techniques, and (iii) as a testbed for the development of real-time network management systems.

Acknowledgements

The research described in this paper was performed at the Jet Propulsion Laboratories, California Institute of Technology, for the United States Air Force Systems Command, under a contract with the National Aeronautics and Space Administration.

References

1. M. Birnbaum, R. F. Quick, K. S. Gilhousen, J. Blanda, "Range Applications Joint Program Office GPS Range System Data Link," in *Proceedings of the ION GPS-89 — 2nd International Technical Meeting of the Satellite Division of the Institute of Navigation*, pp.103–108, Colorado Springs, CO, 1989.
2. L. E. Widman and K. A. Loparo, "Artificial Intelligence, Simulation and Modeling: A Critical Survey," in *Artificial Intelligence, Simulation and Modeling*, L. E. Widman, K. A. Loparo, N. R. Nielsen (eds.), John Wiley and Sons, New York, pp.1–44, 1989.
3. R. O. Yudkin, "On Testing Communication Networks," *IEEE Journal on Selected Areas in Communications*, vol.6, no.5, pp.805–812, June 1988.
4. R. Davis and J. J. King, 'The origins of rule-based systems in AI,' in *Rule-based Expert Systems: the MYCIN projects of the Stanford Heuristic Programming Project*, B. G. Buchanan and E. H. Shortliffe, Reading, MA: Addison-Wesley, pp.20–54, 1984.
5. R. M. Goodman, J. W. Miller, P. Smyth and H. Latin, "Real-time Autonomous Expert Systems in Network Management," in *Integrated Network Management*, B. Meandzija and J. Westcott (eds.), Elsevier Science Publishers B.V. : Amsterdam, pp.599–624, 1989.

6. P. Smyth and R. M. Goodman, 'An information-theoretic approach to rule induction from databases,' *IEEE Transactions on Knowledge and Data Engineering*, in press.
7. P. Smyth and R. Goodman, 'Rule induction using information theory,' in *Knowledge Discovery in Databases*, G. Piatetsky-Shapiro and W. Frawley (eds.), MIT Press, in press.