

Intelligent Real-Time Network Management

April, 1991

Technical Note 15

By:

Anand S. Rao
Australian Artificial Intelligence Institute

Michael P. Georgeff
Australian Artificial Intelligence Institute

This research was supported by TELECOM Australia, under Contract No. 7060. This paper appeared in the *Specialized Conference on Artificial Intelligence, Telecommunications and Computer Systems, Avignon-90*, Avignon, France, 1990.

Abstract

This paper describes research concerned with automating the monitoring and control of telecommunication networks. It describes an Interactive Real-Time Telecommunications Network Management System (IRTNMS) developed to diagnose and solve problems occurring in the telecommunications network in real time. IRTNMS builds on the Procedural Reasoning System (PRS) which, unlike traditional monitoring and control systems, is able to reason about and perform complex tasks in a very flexible and robust manner, somewhat in the manner of a human assistant. Using various problem scenarios in the telecommunications network domain, it is shown how PRS manages to combine both goal-directed reasoning and the ability to react rapidly to unanticipated changes in its environment. In conclusion, some important issues in the design of PRS are reviewed and future enhancements are indicated.

1 Introduction

There is an increased awareness in the telecommunications industry of the applications of expert systems and AI technology [1; 7]. Expert systems are being developed and installed to perform various tasks in the telecommunications domain [5] including:

- interpretation and diagnosis applications;
- monitoring and control applications; and
- network planning and design applications.

The real-time aspects of monitoring and control make it the more challenging of the different applications and will be the focus of this paper.

Most countries in the world, including Australia, are subject to widespread growth in their telecommunications networks. These networks are experiencing a transition from electromechanical sequential switching to stored program controlled (SPC) analogue switching; from SPC analogue to SPC digital switching; from SPC digital to Integrated Digital Networks (IDN); and from IDN to Integrated Services Digital Networks (ISDN). Although the networks are dimensioned to carry normal traffic, with alternative routes for average peak-hour demands, problems like congestion, overload, or failure in parts of the network occur frequently. These problems are due to events such as natural disasters (e.g., bush-fires, earthquakes, floods), phone-in polls, a high number of callers on certain days (e.g., Christmas Day and Easter), or failure of switching systems and transmission links.

All these events lead to a high level of congestion as a result of excessive unsuccessful call attempts, which quickly spread throughout the entire network. This results in poor service to the customer and loss of revenue. To solve these problems, Network Traffic Management (NTM) is being introduced in many centres around the world. NTM is the function of monitoring the status and performance of the network in real time and, when necessary, taking action to control the flow of traffic to ensure the maximum utilisation of the network capacity in all situations [8].

TELECOM Australia has embarked on an ambitious plan to introduce NTM in both their analogue and digital networks with Network Traffic Management Centres established in each state to monitor and control state traffic flows. A National Network Management Centre is to be established to monitor and coordinate traffic flow in the National network [8].

The Australian Artificial Intelligence Institute is working with Telecom Australia to develop an interactive, real-time AI system for diagnosing, controlling, and monitoring the telecommunications network. This system, called the Interactive Real-Time Telecommunications Network Management System (IRTNMS), is built on top of the Procedural Reasoning System (PRS) [2]. PRS is a multi-process, embedded, real-time reasoning system written in LISP and running on SUN and Symbolics workstations.

The organisation of the paper is as follows. In Section 2 we analyze the basic requirements of a network management system. In Section 3 we give a brief description of PRS. This is followed by the description of IRTNMS in Section 4. Section 5 provides sample interactions of the system. The real-time performance of the system is analysed in Section 6. We conclude in Section 7 by comparing IRTNMS with some of the other existing network management systems and highlight the salient features of IRTNMS.

2 Basic Requirements of a Network Management System

The diagnosis of problems in the network, corrective control actions, and monitoring of controls already taken is currently done manually by network operators. This places an enormous cognitive load on the human operators due to the rate at which the raw data are arriving (typically in three- or six-minute cycles, but some network parameters such as route alarms are displayed as soon as they occur), the dynamic nature of the data, different types of controls available for different generations of equipment, delayed feedback from the network for control actions already taken, lack of stability in the network parameters at some instances, and so on.

It is very important for any system designed to perform these tasks to be as flexible, robust, and interactive as possible. It should be capable of diagnosing and responding to a variety of problem scenarios apart from managing and monitoring the normal traffic flow of the network. The system should be capable of interfacing and communicating with other aspects of telecommunications networks such as the signalling and transmission networks. It should be capable of taking a variety of control actions, like expansive controls, protective controls,¹ and re-routing a percentage of the total offered traffic, or a percentage of the total number of overflowing bids.

Where possible, the system should solve multiple problems and suggest multiple controls for solving a single problem. While the system is solving a particular problem it should also keep monitoring the status of the control actions already taken and also be responsive to new problems that are developing in the network. The system should be flexible enough to assign priorities to various problems and dynamically change them. For example, if the system is working on a congestion problem and diagnoses a focused overload condition elsewhere, it should be capable of changing its focus and solving the more serious problem first before coming back to the original problem.

The system should be able to explain the reasons for any proposed course of action in terms that are familiar to network operators. The operators should have facilities to examine both graphical and textual information at various levels of detail. The system should be capable of operating in manual, semi-automatic and automatic modes. In the semi-automatic mode the system should ask the operator about the problems to be solved and the controls that can be activated. In the automatic mode the system diagnoses and solves all the problems without any operator intervention. Finally, the human operators should be able to override the normal reasoning of the system.

Achieving this kind of behavior is well beyond the capabilities of conventional real-time systems. It requires, in contrast, mechanisms that can reason in a “rational” way about the state of the network and the control actions that need be taken in any given situation. Moreover, the system must be both *goal directed* and *reactive*. That is, while seeking to attain specific goals, the system must also be able to react appropriately to new situations in real time. In particular, it must be able to completely alter focus and goal priorities as circumstances change. In addition, the system must be able to *reflect* on its own reasoning processes. It must be able to choose when to change goals, when to plan and when to act, and how to use effectively its deductive capabilities.

The next section describes the Procedural Reasoning System (PRS) developed over a number of years at SRI International [3; 4; 2], which has many of the real-time features mentioned above.

¹Expansive controls re-route traffic via other available spare capacity, while protective controls re-route calls to a Recorded Voice Announcement (RVA) or to a Congestion tone.

3 Procedural Reasoning System

PRS is designed to be used as an embedded, real-time reasoning system. As shown in Figure 1, PRS consists of (1) a *database* containing current *beliefs* or facts about the world; (2) a set of current *goals* to be realised; (3) a set of *plans*, called knowledge areas (KAs), describing how certain sequences of actions and tests may be performed to achieve given goals or to react to particular situations; and (4) an *intention structure* containing all KAs that have been chosen for execution. An *interpreter* (or *inference mechanism*) manipulates these components, selecting appropriate plans based on the system's beliefs and goals, placing those selected on the intention structure, and executing them.

The system interacts with its environment, including other systems, through its database (which acquires new beliefs in response to changes in the environment) and through the actions that it performs as it carries out its intentions.

Goals and Beliefs

The beliefs of PRS provide information on the state of the environment and are represented in a first-order logic. For example, the fact that the number of overflowing bids (ROFL) on a particular route, say `Batman A to Brunswick`, is 56 can be represented by the statement `(value rofl Batman Brunswick 56)`.

The goals of PRS are descriptions of desired tasks or behaviors. In the logic used by PRS, the goal to achieve a certain condition `C` is written `(! C)`; to test for the condition is written `(? C)`; to wait until the condition is true is written `(^ C)`; and to conclude that the condition is true is written `(=> C)`. For example, the goal to activate a control, say `BATX101` to 30% could be represented as `(! (activated-control BATX101 30))`, and to test for it could be represented as `(? (activated-control BATX101 30))`.

Knowledge Areas

Knowledge about how to accomplish given goals or react to certain situations is represented in PRS by declarative procedure specifications called Knowledge Areas (see for example, Figure 3). Each KA consists of a *body*, which describes the steps of the procedure, and an *invocation condition*, which specifies in what situations the KA is useful and applicable. Together, the invocation condition and body of a KA express a declarative fact about the results and utility of performing certain sequences of actions under certain conditions [3].

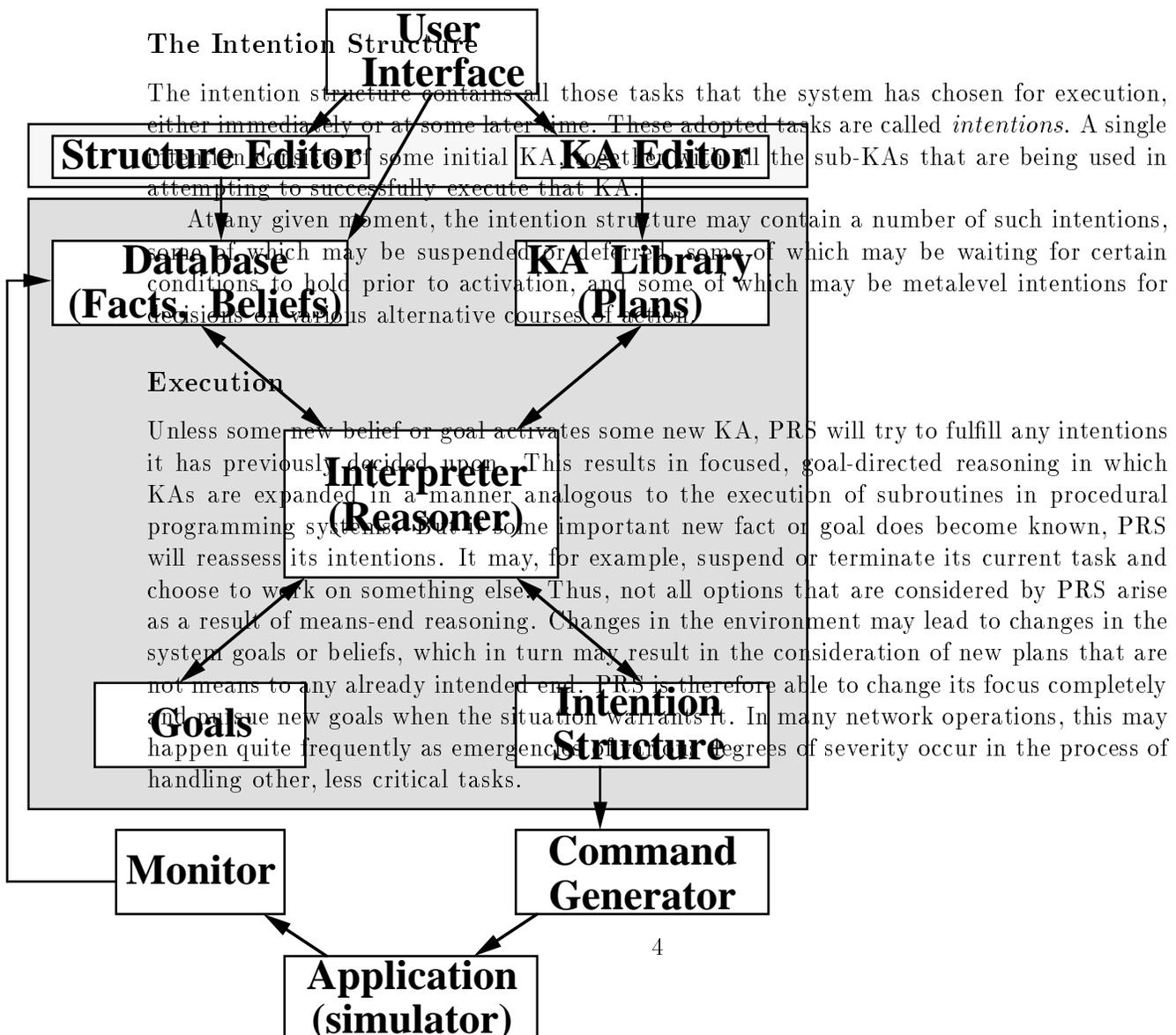
The body of a KA can be viewed as a plan or plan schema. It is represented as a graph with one distinguished start node and possibly multiple end nodes. The arcs in the graph are labeled with the subgoals to be achieved in carrying out the plan. Successful execution of a KA consists of achieving each of the subgoals labeling a path from the start node to an end node. This formalism provides a natural and familiar representation of plans involving the usual control constructs, including conditional selection, iteration, and recursion.

The invocation condition describes the *events* that must occur for the KA to be executed. Usually, these events consist of the *acquisition* of some new goals (in which case, the KA is invoked in a goal-directed fashion) or some *change* in system beliefs (resulting in data-directed or reactive invocation) and may involve both.

The set of KAs in a PRS application system not only consists of procedural knowledge about a specific domain, but also includes *metalevel* KAs, that is, information about the manipulation of the beliefs, goals, and intentions of PRS itself. For example, typical metalevel KAs encode various methods for choosing among multiple applicable KAs, modifying and

Figure 1: Structure of the Procedural Reasoning System

manipulating intentions, and computing the amount of reasoning that can be undertaken, given the real-time constraints of the problem domain.



Multiple Systems

In some applications, it is necessary to monitor and process many sources of information at the same time. To facilitate this, PRS was designed to allow several interpreters to run in parallel. Each interpreter has its own database, goals, and KAs, and operates asynchronously relative to other interpreters, communicating with them by sending messages. We shall refer to such interpreters as *processes*.

4 IRTNMS

The interactive real-time telecommunications network management system is built as an application on top of PRS. As described earlier, PRS can have multiple processes, each with its own database, goals, and plans, and operating asynchronously relative to other PRS processes, communicating with them by sending messages.

Three processes of PRS were set up to handle the IRTNMS network management task. They are: (a) DIAGNOSIS; (b) CONTROLS; and (c) MONITOR. The DIAGNOSIS process receives the route data from the network and diagnoses the different problems occurring. These problems are passed on to the CONTROLS process. The CONTROLS process suggests the different control actions that could be taken to alleviate the problems; the controls are then sent to the exchanges. The MONITOR process is continuously working in parallel with the above two processes and monitors the controls that have already been taken. It suggests suitable modifications or deactivation of controls. In the current prototype a SIMULATOR replaces the actual network. The system architecture of the various processes involved in network management are shown in Figure 2.

Simulator

The network simulator designed by Telecom Research Laboratories (Telecom Australia) simulates the normal flow of traffic in the network as well as some of the problems that occur frequently such as final-choice congestions, focused overloads and exchange failures.

The simulator sends exception data to the DIAGNOSIS process in fixed cycles of three or six minutes. The PRS processes can also request certain values like the route idle capacity or number of overflowing bids, which the simulator then sends to the process that request the values. This has been done to minimise the communication between the PRS processes and the simulator and prevent the PRS processes from being overloaded with unnecessary data.

The simulator also receives commands from IRTNMS to activate, deactivate or modify controls as well as change some of the network parameters in order to simulate certain problem scenarios.

DIAGNOSIS Process

The database of the DIAGNOSIS process consists of the exchange and link data. The link data describes the network configuration and provides the first-choice and second-choice routes of all the origin–destination pairs. The database also contains various threshold values.

The main task of the DIAGNOSIS process is to diagnose the different types of problems. It is initiated by route exceptions, exchange exceptions or route alarms received from the simulator. It requests information from the simulator and is capable of diagnosing the following problems:

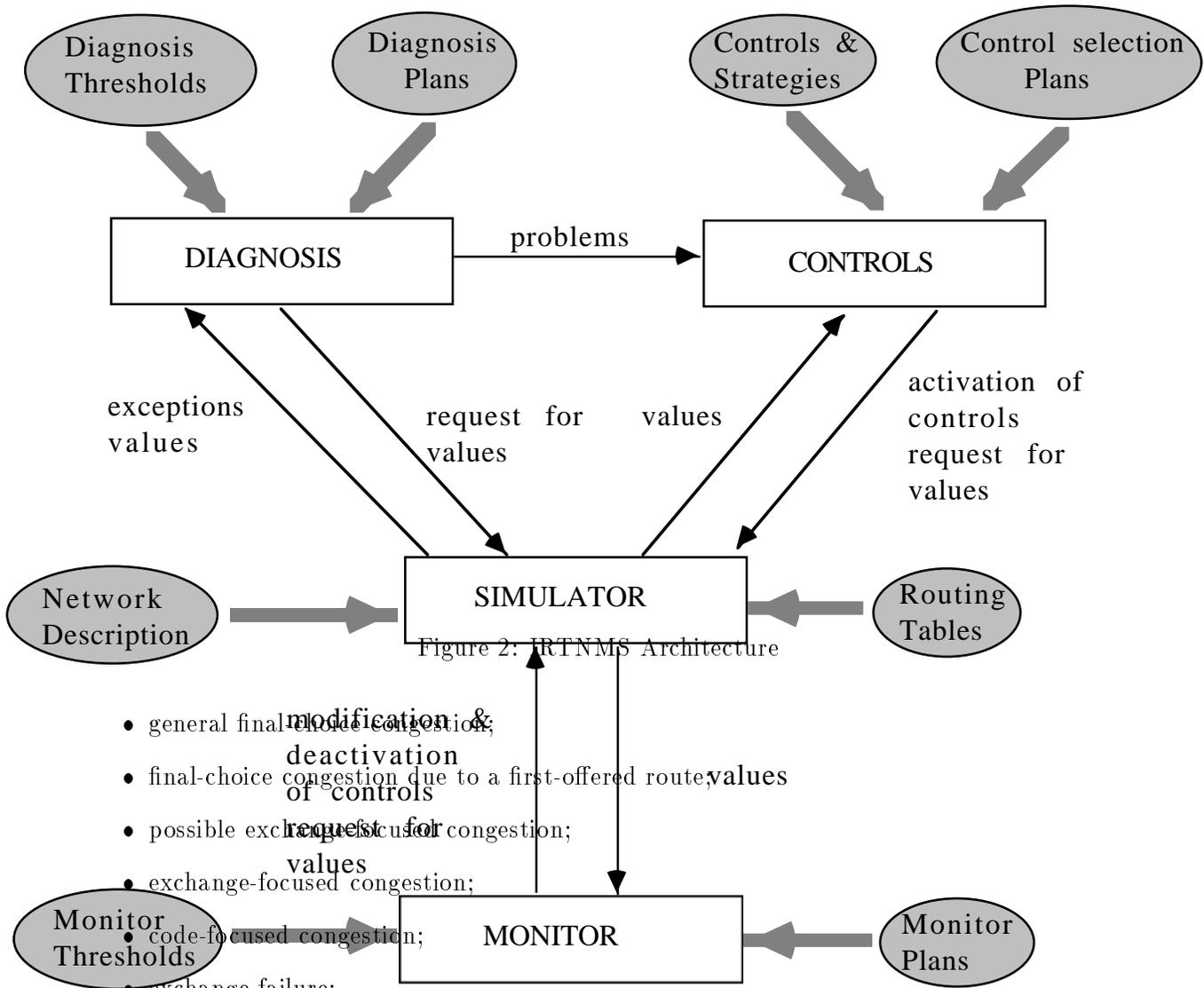


Figure 2: RTNMS Architecture

- general final-choice congestion;
- final-choice congestion due to a first-offered route;
- possible exchange-focused congestion;
- exchange-focused congestion;
- code-focused congestion;
- exchange failure;
- exchange overload; or
- a combination of the above.

A problem is diagnosed to be a general final-choice congestion problem, if the first-offered routes overflowing onto the final choice contribute more or less the same percentage of total overflowing calls. If the contribution of a single first-offered route is far greater (a limit that can be declared in the database) than the rest, then the problem is a final-choice congestion caused by this first-offered route.

The system diagnoses a problem to be a *possible exchange-focused congestion*, if all the routes trying to access the exchange have route percentage overflows greater than the preset value. If the final-choice routes of any of these routes are overflowing, then the problem is diagnosed to be an *exchange-focused congestion*.

The system diagnoses a *code-focused congestion* problem if the number of call attempts via various routes trying to access a particular code associated with an exchange overflow and exceed a preset limit. The diagnoses of code congestions occur only for those codes which are being monitored.

An *exchange failure* problem is diagnosed by the system, if the number of working circuits falls below the preset limit. An *exchange overload* occurs when the percentage overflow of an exchange exceeds a preset limit and the percentage answer seizure ratio of the destination code associated with the exchange falls below a certain limit.

The DIAGNOSIS process receives only route, exchange and code exceptions; all other values it needs are received on a demand basis from the simulator. As the system may have to wait for the simulator to send these data, all problems of a particular type are solved in parallel. This ensures that while one problem is waiting for the data to arrive from the simulator the other problems are being solved.

The diagnosed problems are sent to the CONTROLS process and the DIAGNOSIS process starts working on the next set of data, while the control is selected by the CONTROLS process. Thus problem diagnosis and control selection are done in parallel.

CONTROLS Process

The database of the CONTROLS process contains all the exchange and link data as declared in the database of the DIAGNOSIS process. In addition it also contains the strategies for solving various problems and the controls applicable under each strategy.

The controls selected by the system essentially depend on the type of problem. Normally, expansive controls are tried first, followed by protective controls. The controls at the originating node are tried first, followed by intra-state controls, and then by inter-state controls. However, there are instances when the above preference criteria are overridden by the system. For example, a potential expansive control to re-route the first-offered route overflowing onto a final-choice route, can be removed if the number of calls re-routed are insignificant compared to the total overflow.

Now we look at the different priorities given by the CONTROLS process to different control selection procedures. For a general congestion problem expansive controls for re-routing the final-choice route are tried before the expansive controls for the other first-offered routes overflowing onto it. Whereas for a congestion problem due to a first-offered route, controls for re-routing the first-offered route are tried first, followed by controls for re-routing the final-choice route, and finally the controls for the other first-offered routes.

If the CONTROLS process cannot find expansive controls to solve a particular problem, it examines protective controls. It takes a global view of all the protective controls and distributes the percentage of calls to be re-routed based on their contribution to the total overflow.

An exchange-focused congestion is solved by selecting the controls for solving the final-choice congestions associated with all the origins trying to access the problem exchange. Thus, solving exchanged-focused congestion makes the solving of some of the other final-choice congestions unnecessary. A code-focused congestion is solved by taking code controls at all the origins trying to access the code. Hence, all the controls for a code-focused congestion are intrastate or interstate protective controls.

An exchange overload problem is solved by taking controls on all the origins trying to reach the exchange. Initially these controls re-route a percentage of the overflow traffic, and on operator request re-routes a percentage of the total traffic offered. The controls for exchange failure are similar except that the controls are taken only on the congested routes.

If a control has been activated, but has not yet taken effect and the problem reappears the operators usually wait for some time for the problem to either disappear or reappear again in the next cycle. The operators determine the waiting period based on their experience. In order to model this, the system suspends a problem and waits for a pre-determined time before it asks the operator if the problem needs to be resolved. Currently the waiting duration of the system is obtained by trial and error in a simulated environment. Waiting for a pre-determined time or for some condition to become true models the notion of delayed feedback. This prevents the CONTROLS process from solving the same problem again and again.

MONITOR Process

The database of the MONITOR process contains various thresholds for deactivating and modifying controls and in addition contains all the strategies and controls as described earlier.

Each control for each problem in the network is continuously monitored by the MONITOR process. All problems are monitored in parallel and all controls of each problem are again monitored in parallel. While monitoring controls, the MONITOR process distinguishes between three different types of controls : (a) expansive control activated on a final-choice route; (b) expansive control activated on a first-offered route; and (c) protective control activated on a first-offered route. In the first case, the system increases the percentage activation of the control if the number of overflowing bids has increased and decreases them if they have decreased. The control is deactivated if there is sufficient idle capacity on the final-choice route. In the second case, the control on the first-offered route is increased or decreased depending on the final-choice overflow. In both these cases, the MONITOR process modifies the controls if the control routes themselves start overflowing, i.e., situations in which the control is complicating the problem rather than solving it. The protective controls on the first-offered route are modified based on the final-choice overflow.

The system waits for a fixed period of time for the network to stabilise before deactivating the controls. This prevents the system from continuously activating and deactivating the same control. The running of the MONITOR process in parallel with the DIAGNOSIS and CONTROLS process ensures that the controls are removed or modified as fast as possible taking into account the fluctuating nature of the network.

IRTNMS User Interface

The network operator has access to both textual and graphical information. The system displays all the problems of the network together with their time of occurrence in the *operator panel*. The display has sufficient details for the operator to identify the type of problem, the origin and destination nodes involved, the percentage overflow, and number of overflowing bids. Suspended problems are marked accordingly. The system displays all the controls needed for solving a particular problem. Again the display contains sufficient information to identify the type and time of the problem and the type and nature of the control action being taken. Appropriate messages for deactivating or modifying the controls are also displayed in the operator panel.

The colour graphical display running on the SUN workstation displays the entire network or a part of the network, highlighting the problems. The controls taken by the system are also shown. The entire user interface is menu driven and the operator can examine and/or modify the database.

Expansive Controls For Congestion due to FO Route

INVOCATION:

```
(*GOAL (? (EXP-CONTROLS $P-NUM $S $TYPE $SCOPE)))
```

CONTEXT:

```
(AND (*FACT (PROBLEM-AT $P-NUM $TIME))
      (*FACT (PROBLEM-TYPE $P-NUM FOC))
      (*FACT (PROBLEM-ORIGIN $P-NUM $O))
      (*FACT (PROBLEM-DEST $P-NUM $D))
      (*FACT (PROBLEM-HUD $P-NUM $HU-D))
      (*FACT (ORDERED-HU-ROUTES $P-NUM $O-HU-D)))
```

```
(? (rerouted-from-hu-route
    $p-num $o $d $hu-d $s $type $scope))
```

```
(? (value rofl $o $d $rofl1))
```

```
(? (== $rofl1 0))
```

```
(? (> $rofl1 0))
```

DOCUMENTATION:

"Gives the order in which expansive controls have to be tried for a congestion caused by a first-offered route"

Figure 3: Expansive Control Selection for Final Choice Congestion

5 Sample Interactions

In this section, we examine a sample run of the system and illustrate how the various modules of the system interact in diagnosing, controlling and monitoring problems. This will help illustrate the salient features of the system.

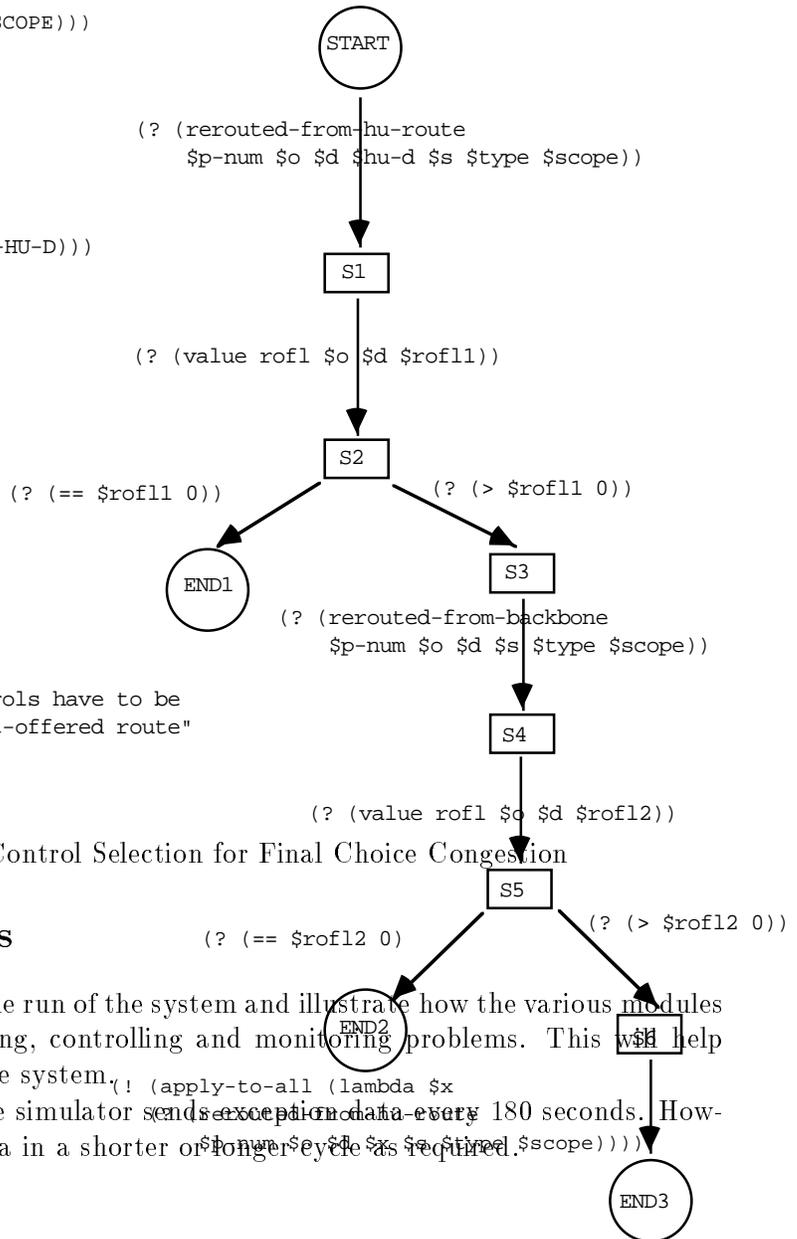
In the following sample run the simulator sends exception data every 180 seconds. However, the simulator could send data in a shorter or longer cycle as required.

Final-Choice Congestion

If the simulator starts simulating the network at time 0, then at the end of 180 seconds it sends a set of exception data to the DIAGNOSIS process. The format of this data is as follows:

```
(route-exception 180 RPOFL Brunswick Northcote 12)
(route-exception 180 RPOFL Brunswick Batman 47)
```

The first item in the above list states that there is a route-exception at time 180 from the node Brunswick (BRUX) to Northcote (NCOX) whose route percentage overflow is 12%. The DIAGNOSIS process diagnoses the problems which are then displayed on the operator panel.



These problems are passed onto the CONTROLS process. Once the DIAGNOSIS process has completed diagnosing one set of route-exception data it can start processing the next set of data.

For the particular situation under consideration the problems diagnosed by the system are given below:

- (1) First-offered congestion from Brunswick to Northcote due to Batman at time 180 (ACTIVE)

The above display states that there is a final choice congestion from Brunswick to Northcote at time 180 caused by the first-offered route Brunswick to Batman.

The CONTROLS process then solves the problem by first examining the expansive controls. It makes use of the procedure shown in Figure 3 to re-route the traffic on the first-offered route that is causing the problem. If this does not solve the problem then it tries to re-route traffic from the final-choice route and then all the other first-offered routes that are overflowing onto the final-choice route.

After going through all the procedures to solve the problem the system suggests the following solution, namely to re-route the first-offered traffic from Brunswick to Batman via Footscray.

First-offered congestion Brunswick-Northcote (180):

- (1) Activate BRUX102: Reroute Batman overflow traffic via Footscray
33 calls at 60%

The number of calls involved in the above case is 33 and constitutes 60% of calls. In the semi-automatic mode of operation the system prompts the operator to choose one or more of the controls to be activated and in the automatic mode the control action is automatically taken by the system.

Code-Focused Congestion

Allowing the system to run for a further period of 180 seconds, we have the following problems at the end of 360 seconds.

- (1) Code-focused congestion at 311 code of Footscray at 360 (ACTIVE).
- (2) First-offered congestion from Brunswick to Northcote due to Batman at time 360 (SUSPEND)

If the operator chooses to solve the code congestion problem, then the system suggests the following code controls. These controls selectively block the 311 code traffic from various originating nodes.

Code-focused congestion Footscray-311 (360):

- (1) Activate BATX501: Reroute 311 overflow traffic at Batman A to RVA
140 calls at 50%
- (2) Activate BRUX502: Reroute 311 overflow traffic at Brunswick to RVA
102 calls at 50%
- (3) Activate GBRX503: Reroute 311 overflow traffic at Greensborough to RVA
80 calls at 50%
- (4) Activate NC0X504: Reroute 311 overflow traffic at Northcote to RVA
220 calls at 50%

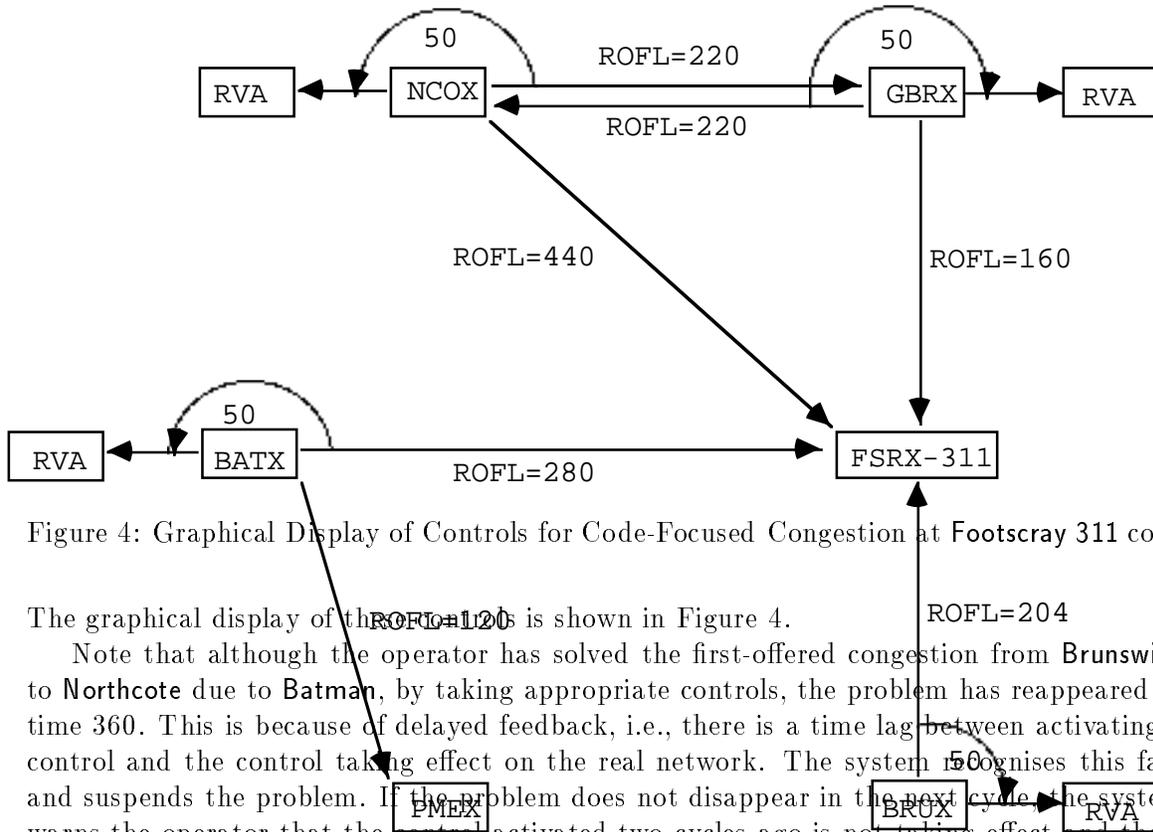


Figure 4: Graphical Display of Controls for Code-Focused Congestion at Footscray 311 code

The graphical display of the controls is shown in Figure 4.

Note that although the operator has solved the first-offered congestion from Brunswick to Northcote due to Batman, by taking appropriate controls, the problem has reappeared at time 360. This is because of delayed feedback, i.e., there is a time lag between activating a control and the control taking effect on the real network. The system recognises this fact and suspends the problem. If the problem does not disappear in the next cycle, the system warns the operator that the control activated two cycles ago is not taking effect and there might be a potential problem. Conventional expert systems are incapable of recognising and reacting to such delayed feedback in the flexible manner outlined above. PRS achieves the above capability by suspending intentions and then waiting for a specified period of time before reactivating them.

Allowing the system to run for a further period of 180 seconds, we find that at time 540 seconds, there are no more active problems in the network. Thus the CONTROLS process will be idle. However, as described earlier the MONITOR process will be monitoring all the controls activated for the final-choice congestion, and code-congestion problems. At time 540 the MONITOR process determines that there is enough idle capacity on the final-choice route Brunswick to Northcote to carry all overflowing calls. Hence, the expansive control activated at time 180 can be deactivated. However, the MONITOR process waits for a fixed period of time for the network to stabilise before suggesting that the control be deactivated. Thus if the network is stable after the wait period the following message is displayed:

First-offered congestion Brunswick-Northcote (180):

- (1) Deactivate BRUX102: Reroute Brunswick overflow traffic via Footscray.

6 Performance Analysis

In this section we measure the real-time performance of IRTNMS. The performance of real-time systems revolve around two important parameters [6]:

- *reaction time* – the time taken by the system to notice changes in the environment; and
- *response time* – the time taken by the system to respond to changes in the environment.

For example, the time difference between the time at which a problem is received by the CONTROLS process and the time it takes for the CONTROLS process to invoke a KA in reaction to this problem would be the reaction time. The response time in this case would be the time difference between the time at which the KA was activated and the time at which the KA (and all its descendants) have completed.

In Figure 5 we show the average, standard deviation, and maximum, reaction and response times for the three different processes of IRTNMS. The performance measures have been made on a SUN Sparcstation, with 20 megabytes of central memory, running Sun Common Lisp, development Environment 4.0.0 Beta-0, Sun4 Version for SunOS 4.0. The code was not optimised by the compiler, and the probing itself affects system performance.

The measurements given in Figure 5 were taken at the end of the first six minutes. This involved the system diagnosing four problems, and then taking control actions for these problems. All the values are given in sixtieths of a second. The average reaction time is less than quarter of a second and the maximum reaction time is less than six seconds. The average response time is around 100 seconds. The response times are somewhat more difficult to interpret because many of the procedures executed in the IRTNMS application wait for the simulator to return data or wait until the network stabilises. This distorts the measurement of the response times.

7 Comparison and Conclusion

There several real-time network management systems in various stages of development [1; 5]. In this section we briefly review some of them and compare them with IRTNMS.

NEMESYS [5] is designed to fight congestion in the AT & T network. It receives traffic completion data and suggests controls to alter the traffic flow. It uses rules as well as procedures and is built using a well-known expert system development tool KEE (Knowledge Engineering Environment).

Net/Advisor [5] works with the Avant-Garde, Inc. product Net/Command to monitor real-time network status and suggest actions to take to improve situations. It uses both PROLOG-based backward chaining inference mechanism and LISP code.

NETMAN [9] is a rule-based network management system implemented using the language OPS83. The rules of NETMAN are organised into nine experts and the rules invoked are based on four major control policies – basic, emergency, short-term, and long-term policies. The traffic situation determines the appropriate control policy.

The above systems have been developed either with an existing expert system development tool like KEE, or existing AI languages like LISP, PROLOG, and OPS83. IRTNMS, on the other hand has been built on top of a sophisticated embedded real-time reasoning tool, namely PRS. As PRS has many of the features needed for any real-time embedded system, the design of IRTNMS using PRS makes IRTNMS more modular, flexible and easy to modify. In other words, the real-time features of network management task are at the PRS level and the IRTNMS encodes only the knowledge of the network domain.

DIAGNOSIS				
	Number of facts	Average	Standard deviation	Maximum
<i>Reaction</i>	2786	8.59	21.29	179
<i>Response</i>	2	6535.33	4166.21	9481
	Number of goals	Average	Standard deviation	Maximum
<i>Reaction</i>	1435	6.7	15.53	171
<i>Response</i>	1231	79.07	435.46	9384
CONTROLS				
	Number of facts	Average	Standard deviation	Maximum
<i>Reaction</i>	2549	8.95	21.11	348
<i>Response</i>	9	1928.43	4015.52	12484
	Number of goals	Average	Standard deviation	Maximum
<i>Reaction</i>	1158	8.13	21.45	346
<i>Response</i>	1064	104.87	400.37	6339
MONITOR				
	Number of facts	Average	Standard deviation	Maximum
<i>Reaction</i>	175	11.62	25.72	216
<i>Response</i>	3	978.59	1316.02	2495
	Number of goals	Average	Standard deviation	Maximum
<i>Reaction</i>	66	13.66	30.27	177
<i>Response</i>	60	201.34	443.94	2212

Average, $\bar{x} = \frac{\sum_{i=1}^n x_i}{n}$, Standard deviation = $\sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n-1}}$, and Maximum = $\max(x_1, \dots, x_n)$

Figure 5: Reaction and Response times for IRTNMS

Expert systems provide a clear distinction between the *knowledge* of a particular domain and the *control* of that knowledge. Often, for embedded systems the control required is at a finer level of detail than those offered by existing languages like LISP or PROLOG. As a result most of the real-time control features have to be implemented by the user using these languages. PRS and other similar real-time systems offer a higher level of abstraction than existing expert systems with in-built real-time control features.

Some of the more important features of IRTNMS that contribute to its usefulness in network management and control are as follows:

- The parallel operation of DIAGNOSIS, CONTROLS and MONITOR; this ensures that problems will be diagnosed and solved, and controls modified or deactivated as soon as they occur without overloading the operators.
- Delayed feedback; there is a significant delay between a control action being activated and the action having effect in the real network. The CONTROLS process recognises this and suspends problems for which control actions have already been activated.
- Insensitivity to transients in the network; most network parameters are dynamically varying quantities and the MONITOR process recognises this and deactivates controls only after the network has stabilised for a certain period of time.
- Override capability; the flexible meta-level approach of PRS allows IRTNMS to dynamically reassign priorities thereby providing the operators the capability of either overriding or having precedence over the reasoning of the system.

The real-time network management system described in this paper is running under a simulated environment on SUN and Symbolics workstations. The prototype system has been under development for more than a year and currently runs on a ten-node network. The next phase of the project is to integrate the prototype system with existing network management software and conduct field trials at the Melbourne Network Management Centre.

Acknowledgments

François Félix Ingrand has played a major role in the development of PRS and has substantially extended the implementation. Andrew Hodgson, Magnus Ljungberg, and Andrew Lucas were involved in the design and implementation of IRTNMS. We would like to thank Col Butler, Bill Honey, Eugene Majsakowski, Peter Sember, and Bob Warfield of TELECOM for providing the domain expertise and Magnus Ljungberg for his valuable comments on this paper.

References

- [1] R. N. Cronk, P. H. Callahan, and L. Bernstein. Rule-based expert systems for network management and operations: An introduction. *IEEE Network*, 2(5):7–21, 1988.
- [2] M. P. Georgeff and F. F. Ingrand. Decision-making in an embedded reasoning system. In *Proceedings of the International Joint Conference on Artificial Intelligence*, Detroit, Mi., 1989.
- [3] M. P. Georgeff and A. L. Lansky. Procedural knowledge. In *Proceedings of the IEEE Special Issue on Knowledge Representation*, volume 74, pages 1383–1398, 1986.
- [4] M. P. Georgeff and A. L. Lansky. Reactive reasoning and planning. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 677–682, Seattle, Washington, 1987.
- [5] S. K. Goyal and R. W. Worrest. Expert system applications to network management. In J. Liebowitz, editor, *Expert System Applications to Telecommunications*, pages 3–44. John Wiley and Sons, New York, 1988.
- [6] F. F. Ingrand and M. P. Georgeff. Managing deliberation and reasoning in real-time AI systems. Technical Report 10, Australian AI Institute, Carlton, Australia, 1990.
- [7] J. Liebowitz, editor. *Expert System Applications to Telecommunications*. John Wiley and Sons, New York, 1988.
- [8] J. L. Seamons. Network traffic management in Telecom Australia. *Telecommunication Journal of Australia*, 38:15–24, 1988.
- [9] W. Zhan, S. Thanawastien, and L. Delcambre. Simnetman: An expert workstation for designing rule-based network management systems. *IEEE Network*, 2(5):35–42, 1988.