Application of the Recommendation Architecture to Telecommunications Network Management

L. Andrew Coward, Tamás D. Gedeon, William D. Kenworthy

***Abstract*--The recommendation architecture has been proposed as a system architecture which can enable a system to learn to perform a complex combination of interrelated functions. The capability of a system with the recommendation architecture to learn to manage complex telecommunication backbone networks has been investigated. A network model with a number of nodes and links and carrying realistic but randomly generated traffic was used as the target for the management system. Traffic data taken from the model was used as input to the recommendation architecture system. The traffic data was organized into inputs once every 5 minutes, and the management system organized these inputs into a hierarchy of repetition similarity. It was demonstrated that the outputs of this hierarchy provided information on the condition of the network. This output information was a compressed version of the inputs which correlated with major network conditions.**

***Index Terms*--telecommunications network, information compression, recommendation functional architecture.**

The Recommendation Architecture

The functionality of a system which performs a complex combination of interacting functions must be partitioned into a set of functional components which independently perform their functions but coordinate these functions by the exchange of information. In a conventional electronic system, this information is always functionally unambiguous to the receiving components, or in other words a specific input is consistently present when a set of external conditions relevant to the receiving component are present, and individual component outputs determine the state of the overall system in a consistent fashion. Components therefore must maintain an unambiguous context for all information received from other components. This requirement to maintain an unambiguous context makes parallel processing difficult and learning extremely difficult [1].

In the recommendation architecture [1], [2], information exchanged between functional components is ambiguous. Components detect the repetition of input conditions of various degrees of information complexity and indicate the presence of these repetition conditions by outputs which have enough information richness to provide a partially ambiguous but meaningful context to these outputs. However, the detected input conditions will not correlate unambiguously with functionally relevant categories of input conditions. When learning occurs, components modify the outputs they generate, but the modifications can only occur in a fashion

which allows receiving components to maintain a meaningful, although partially ambiguous, context for the outputs. The less constrictive requirement for a partial context allows heuristic definition of functionality, which means that the hierarchy of repetition similarity is heuristically organized. A system in which the hierarchy is defined by design is also possible, and heuristic organization can be guided in various ways by design, genetic or other a priori constraints [2].

In a system with the recommendation architecture the functionality of the system, including interaction between functions, is managed within a clustering separation. Within that separation, experience of an input space is heuristically organized into a hierarchy of repetition similarity which is useful for determining behaviour. Individual components select the information conditions to which they will respond if the condition repeats, and relationships between the components are such that the system selects a population of information conditions which as a whole is adequate to discriminate between different functionally relevant conditions.

The component hierarchy consists of repetitions, clusters, superclusters etc. A repetition is an information condition, a cluster is a set of repetition conditions, a supercluster is a set of cluster repetition conditions. If all information were unambiguous this would be a pattern, category, supercategory hierarchy, but the components in the clustering separation only correlate partially with such unambiguous conditions. The presence of a repetition condition on any level is indicated by outputs from the corresponding component. These outputs must have sufficient information richness to communicate a partial context to any component which receives them. The requirement to maintain a partial but meaningful context means that once an information condition has been selected as one which will be indicated by an output, an exact repetition of the condition must always in the future result in an output which includes the same output. A critical requirement in the clustering separation is therefore a management process to select the information conditions which will be recorded. This management process is described in detail in [1] and summarized in the next section.

Although the coordination of the interactions within a complex functionality occurs within the clustering subsystem, the outputs of this subsystem are ambiguous. A separate competition subsystem interprets the outputs into unambiguous system behaviours. This interpretation process uses feedback on whether a behaviour was appropriate or not to adjust the probability that such a behaviour will be selected in the future in similar information conditions. The system as

Manuscript received May 28, 2000. This work was supported in part by Nortel Networks.

L. A Coward is a Research Fellow at the School of Information Technology, Murdoch University (e-mail: landrewc@aol.com).

T. Gedeon is Professor at the School of Information Technology, Murdoch University (e-mail: tom@dijkstra.it.murdoch.edu.au).

W. Kenworthy is a student at the School of Information Technology, Murdoch University

a whole thus sorts its experience into a set of repetition conditions and associates different combinations of conditions with different behaviours. The outputs of the clustering function can thus be viewed as a set of "action recommendations" with the competition subsystem selecting the most appropriate recommendation. This view is useful for understanding the system operations, but it must be remembered that all information within the clustering subsystem is actually ambiguous, and only achieves an unambiguous functional meaning once it has been processed through a competitive subsystem. For example, no communication between components in the clustering subsystem can be unambiguously associated with the presence of an object of a particular category in the inputs, although a range of combinations of outputs will be associated with such categories by the competitive subsystem.

The separation between clustering and competition functions is a key characteristic of the recommendation architecture. It allows the maintenance of a partial context for information exchange between components withing the clustering function and the interpretation of clustering outputs to functional behaviour using feedback of consequences in the competition function. If feedback of consequences affected component outputs within the clustering function, changes to improve functional performance in one area would result in random, undesirable changes to functionality in other areas [1].

In a functional architecture, the information distributed between components must be minimized. If the components are heuristically defined, the information distribution between them must be heuristically minimized. This minimization implies that components only accept information from other components if that information is likely to be relevant. The only readily available indication of probable relevance is the frequency with which two components produce outputs which correlate in time. In a recommendation architecture an operation is therefore required in which the system is taken functionally off-line, and provisional connectivity is established between components which have frequently been active in the past at the same time or with a consistent time interval between their activity.

A clustering function takes information from an input space and generates outputs in an output spacewhich must be functionally useful but in general smaller than the input space, i.e. there must be compression of information. In practical systems there could be a series of clustering functions, with competitive functions selecting subsets of the outputs of a clustering function to pass on to the next clustering function. The result would be compression of the information required to generate behaviour in a series of stages. This compression is the process which generates signals which can be interpreted by a functionally simple competitive subsystem from the potentially huge input space.

Description of the Network Model

A network model was used as the target for the recommendation architecture management system. This network model could be configured as a set of switching nodes with links of different bandwidths between them as illustrated in figure 1. Sources of network traffic could be added which generated packet traffic in a random fashion but with a pattern of variation consistent with the variation seen in real networks, including rapid second to second variation but average traffic over several minutes varying with time of day and day of week. Traffic also reflected a mix of different types of data including email, MPEG etc.

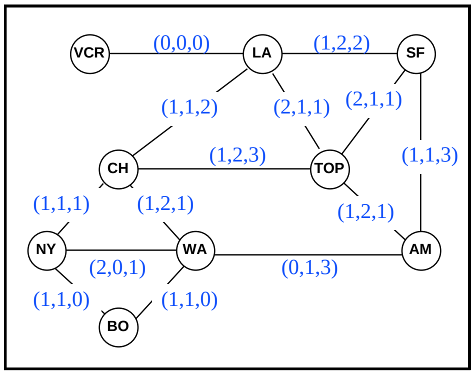

Figure 1. The network topology of the reference networks. The numbers on each link are the number of alternate paths between the two nodes via one, two and three additional nodes.

The model generated traffic information on link utilization, dropped packets, queue lengths and various statistical parameters on queue length variation. This traffic information was generated every 15 seconds for each link. An example of traffic information generated over a 15 minute period in one link is illustrated in figure 2.

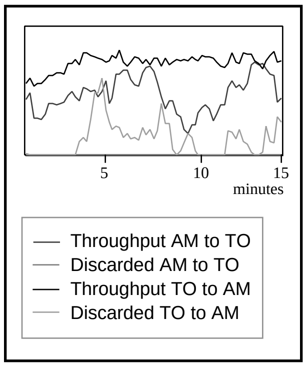

Figure 2 An example of throughput and dropped packet traffic data generated from one link in each direction over a period of 15 minutes.

The layer 3 protocol operates so that all traffic between two nodes passes directly from one node to the other if there is any bandwidth available, even if the available bandwidth is reduced to a minimal level above zero. The use of the MPLS protocol could force traffic off the direct route, allowing traffic to be divided on a percentage basis between a number of assigned paths which can be via one or more intermediate nodes.

ARCHITECTURAL CONCEPT FOR NETWORK MANAGEMENT

The architectural approach to the network management problem is illustrated in figure 3. Module I finds relatively simple combinations of traffic and network topology conditions which have a higher probability of indicating significant network changes than the raw inputs. Examples could be average traffic parameters over 15 minute periods. Module IIa heuristically organizes a sequence of module I outputs into more complex repetition conditions with the objective to make outputs from module IIa of the same order of complexity as indications of significant network changes. Module IIb takes the outputs from module IIa generated from different subnetworks such as figure 1 and heuristically defines repetition conditions of the same order of complexity as significant changes across multiple subnetworks.

The architecture illustrated in figure 3 has been implemented in Smalltalk to run in an Apple Macintosh environment. Results of extensive simulations using artificially generated data have demonstrated its capability to take inputs from spaces of from 100 to 10 thousand bits and learn to compress the input information into an output space which indicates the characteristics of the inputs in a relatively simple fashion [4]. The architecture used in [4] was used for the results described in this paper. A competitive function which could interpret the outputs has also been implemented [1] but is not described in this paper.

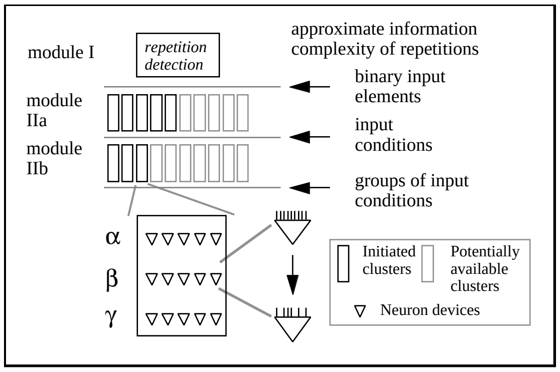

Figure 3 High level architecture of the implemented recommendation architecture system

The traffic data from the network varies rapidly and widely from moment to moment on every link as seen in figure 2. The potential input space to the management system is very large, and module I defines smaller input vectors. Module I took the traffic and topological data for the whole network in figure 1 and generated a 1246 bit binary vector every 5 minutes. This vector contained 89 bits for each of the 14 links. These 89 bits included 9 bits indicating the topological environment of the link, and 40 bits indicating the traffic in each of the two directions on the link. The 40 bits were 5 sets of 8 bits indicating average traffic load on the link during the past 15 minutes, and during 15 minute periods which are 2 hours, 6 hours, 24 hours and 7 days earlier. The most relevant information is the indication that something has changed, and the system is most sensitive to the presence of a binary bit. The meaning of binary bits which occurred in most input vectors was therefore reversed, so that the absence of the condition was indicated by the presence of the bit.

In the simulations reported here, the inputs came from only one network, and module IIb was not active. In addition, because the network topology was constant, the bits associated with topology were suppressed. The sequence of 1120 bit vectors was presented to the recommendation architecture, which then finds repetition conditions in this input vector stream, or in other words, conditions which repeat between vectors. The architecture is similar to the one used in [1] and [3]. Module IIa has an unlimited number of potential repetition clusters available. Each cluster is made up of three layers of devices, and each device has a number of inputs which can be either active or inactive, and a threshold which is the number of active inputs which will cause the device output to be active. When a cluster in initially configured, devices in layer α receive inputs which are randomly selected from the 1120 possible binary inputs. Devices in layer β receive inputs which are randomly selected outputs from layer α devices, and layer γ devices receive inputs which are randomly selected outputs from layer β. Such a cluster is inactive unless no other cluster is generating an output in response to an input vector, in which case the threshold of all devices in the cluster is lowered until the cluster produces an output from some γ devices. Any devices which has produced an output then has all its inputs except those currently active deleted, and its threshold set slightly below its resulting input count. The device will therefore fire in the future if a similar input condition occurs. Within the cluster, further device imprinting will occur in the future if there is firing of a sigificant number of β devices but no γ devices. The cluster will therefore evolve towards responding to a set of similar input conditions, and indicate the presence of those conditions by different combinations of γ outputs. Information distribution between devices was minimized by the off line "dream sleep" process as described in [1]. Proliferation of clusters was avoided by allowing one cluster to develop a spectrum of input conditions before another cluster was configured, and by other mechanisms as described in [3].

(26 5)	(24 0)	(22 2)	(22 0)	12 a.m. - 1.30 p.m.
(22 5)	(23 1)	(21 5)	(17 1)	2 p.m. - 3.30 p.m.
(0 6)	(20 16)	(17 19)	(8 18)	4 p.m. - 5.30 p.m.
(1 18)	(0 30)	(22 6)	(0 19)	6 p.m. - 7.30 p.m.
(0 21)	(0 17)	(0 13)	(0 22)	8 p.m. - 9.30 p.m.
(0 17)	(0 16)	(0 20)	(0 17)	10 p.m. - 11.30 p.m.
(0 18)	(0 20)	(0 18)	(0 17)	12 p.m. - 1.30 a.m.
(0 9)	(0 17)	(0 9)	(7 17)	2 a.m. - 3.30 a.m.
(9 3)	(8 0)	(9 8)	(24 8)	4 a.m. - 5.30 a.m.
(17 1)	(25 6)	(25 8)	(18 6)	6 a.m. - 7.30 a.m.
(25 3)	(24 0)	(25 0)	(25 0)	8 a.m. - 9.30 a.m.
(24 0)	(25 0)	(23 1)	(0 21)	10 a.m. - 11.30 a.m.
(26 9)	(23 10)	(24 10)	(22 9)	12 a.m. - 1.30 p.m.
(22 10)	(23 10)	(21 11)	(21 18)	2 p.m. - 3.30 p.m.
(0 1)	(21 20)	(10 24)	(23 11)	4 p.m. - 5.30 p.m.
(1 32)	(0 34)	(0 34)	(0 20)	6 p.m. - 7.30 p.m.
(0 14)	(0 19)	(23 11)	(0 24)	8 p.m. - 9.30 p.m.
(0 19)	(0 18)	(0 19)	(0 19)	10 p.m. - 11.30 p.m.
(0 20)	(0 18)	(0 19)	(1 15)	12 p.m. - 1.30 a.m.
(0 18)	(0 13)	(0 18)	(7 18)	2 a.m. - 3.30 a.m.
(11 10)	(8 0)	(10 9)	(24 11)	4 a.m. - 5.30 a.m.
(22 9)	(27 11)	(23 10)	(24 11)	6 a.m. - 7.30 a.m.
(25 11)	(26 11)	(24 10)	(25 8)	8 a.m. - 9.30 a.m.
(22 9)	(20 10)	(0 20)	(0 22)	10 a.m. - 11.30 a.m.

Figure 4 The heuristically generated outputs from the two clusters in response to traffic vectors. Outputs are given at 30 minute intervals over a 48 hour period.

INITIAL RESULTS OF RECOMMENDATION ARCHITECTURE SIMULATIONS WITH NETWORK DATA

The outputs generated by a run of the recommendation architecture system using data from the network model, once a set of clusters had matured through experience are illustrated in figure 4. The system created two clusters, and the total number of γ devices which were active in each cluster at 30 minute intervals over a two day period are shown.

The total number of possible outputs from the two clusters (i.e. the total number of γ devices) was 65 There was therefore an information compression of the 1120 bit space by a factor of 17. The heuristically generated cluster outputs contain enough information to show the diurnal variation in traffic, and also contain enough information to reveal the presence of MPEG traffic. Similar results were generated in response to learning from traffic data over different time periods.

DISCUSSION AND CONCLUSIONS

The results demonstrate that the recommendation architecture approach to managing a complex telecommunications network can heuristically organize a sequence of input vectors representing network traffic information into a hierarchy of repetitions, with the outputs of the repetition hierarchy being a compressed representation of the input space which contains enough information to indicate functionally significant changes in the input space.

ACKNOWLEDGMENT

Thanks to Nortel Networks for funding of this project, and to Chris Hobbs of Nortel Networks for supplying the network model.

REFERENCES

[1] L. A. Coward, "A functional architecture approach to neural systems," *J. Systems Research and Information Systems*, vol. 9, pp. 69-120., 2000

[2] L. A. Coward, *Pattern Thinking.* New York: Praeger, 1990.

[3] T. Gedeon, L. A. Coward, B Zhang, "Results of Simulations of a System with the Recommendation Architecture," in *Proceedings of the 6th International Conference on Neural Information Processing*, Volume I, pp 78-84, 1999.

[4] L. A. Coward, T. Gedeon, "Optimization of Architectural Parameters in a Simulated Recommendation Architecture," submitted for publication.